Electromagnetic Energy Harvester and Self-powered Embedded System

by

Jinyeong Moon

B.S., Korea Advanced Institute of Science and Technology (2005)
M.S., Stanford University (2007)

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2016

Signature redacted

Author .... [illegible] ....

Department of Electrical Engineering and Computer Science

May 20, 2016

Signature redacted

Certified by.. ....

Steven B. Leeb
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Signature redacted

Accepted by .... [illegible] Leslie Kolodziejski
Chair, Department Committee on Graduate Theses

Electromagnetic Energy Harvester and Self-powered Embedded System

by

Jinyeong Moon

Submitted to the
Department of Electrical Engineering and Computer Science
on May 20, 2016, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Energy harvesting offers an important design option for creating sensing and control elements without a requirement for custom wiring or batteries. The independent and care-free nature of energy harvesting enables monitoring devices to penetrate wider and deeper into our daily lives, making accommodation of fine sensing and control for condition-based maintenance ever more feasible. Finer granularity in sensing and control, which is the future of energy efficiency, alone is an immense benefit as it can reduce time and cost associated with a potential repair. Combined with condition-based maintenance, it can prevent potential down-time of a machine under monitoring.

An exciting possibility creates a "self-powered" embedded system with an integrated energy harvester for electromechanical diagnosis. This non-intrusive energy harvester is designed to extract energy from magnetic fields around a power line of a load, in the manner of a current transformer. In contrast to the conventional usage of magnetic elements, such as transformers and inductors, the analysis on this "current transformer" reveals a critical result: for any given core for any particular application, power harvest is maximized when the core is permitted to saturate at an opportune time in the line cycle. The design of this integrated energy harvester is fully explored in the thesis, including: development of new models to incorporate a fully saturating magnetic core for simulation; designs of power electronics circuits for maximizing power harvest; and integration of the harvester into the embedded system as a practical power supply.

The design of a self-powered and low-power embedded system, vibration assessment monitoring point with integrated recovery of energy (VAMPIRE), is discussed in depth in the thesis. The overall architecture of the embedded system is first presented, followed by designs of individual subsystems, the power package and the sensor package. In the power package, initialization, energy buffer, power interfaces, power regulation, and microcontroller design are explored. In the sensor package, power budget, sensors, data storages, storage management, wireless communication, and corresponding user interfaces are explored.

Finally, impedance spectroscopy for an electromechanical load is discussed. Using the electrical and vibrational data that are nonintrusively collected from electromagnetically self-powered embedded system, structural issues of the load, i.e., changes in the stiffness of mounts and the imbalance of a shaft, can be clearly identified, making it feasible for this self-powered embedded system to be used for condition-based maintenance.

Thesis Supervisor: Steven B. Leeb
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

First of all, I would like to thank my research advisor, Professor Steven B. Leeb, for his unceasing support throughout my doctoral studies. His creative visions and ideas have always motivated me and helped me accrue new perspectives that I have never thought of before. Also, a completely new set of skills that I acquired on larger scale electronics is mostly due to his continued advice and care. I would also like to thank my thesis committee members, Professor David J. Perreault and Professor Jeffrey H. Lang, for their keen insights, comments, and suggestions on my research.

The research in this thesis was supported by The Office of Naval Research, The Grainger Foundation, and The Kwanjeong Educational Foundation. I would like to acknowledge these generous funding sources.

There are many friends and colleagues I would like to express my gratitude to. I thank Sangwoon Park for his hearty ways of cheering me up whenever I need a buck-up. I thank my previous and best officemate Arthur Chang for participating together in one of the most unforgettable events in my life. I thank my excellent neighbor Arijit Banerjee for his kind consultations on every matter, being it technical or not. I thank Seungbum Lim and Al-Thaddeus Avestruz for sharing deep understandings on many topics. I also thank many people who have/had been in 10-050 and 10-061 inside the laboratory for electromagnetic and electronic systems (LEES).

I would like to emphasize that all these works would not have been possible without unwavering support from my parents. I thank my father and mother, Seok Yeoul Moon and Insook Baek, for always believing in me and taking pride in me. I will never forget to pay back their overwhelming love, which I know would take forever and still be short. I thank my sister, Jiyoon Moon, for being a patient and understanding little sister, tolerating the tyrannic older brother.

Finally, I would like to dedicate this thesis to my loving wife and my best friend, Minjee Kim, who has been my sole purpose of living since I met her. Without your unconditional love and sacrifices, I would not have reached the point I am standing now. I cannot thank you enough for making me a better person that I am now.

Thank you for enlightening me and widening my perspectives in so many ways. I would like to also dedicate this thesis to my son, Juwon Moon. Thank you for being my perfect boy. With your smile, I feel like I can conquer the world.

Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

Fine grain sensing and control underlies the future of energy efficiency. Electromechanical systems can waste energy for a variety of reasons. They may be operated poorly. For example, they may be left on when the system does not need to operate, or, when operating, they may be controlled to an inappropriate or unachievable setpoint. Many electromechanical systems operate under closed-loop feedback control, which can be a disaster for an efficient operation in pathological situations. Heating, ventilation, and air conditioning (HVAC) components, for example, operate to achieve a comfort setpoint, and, without intervention, will continue to do so regardless of a developing but not-yet-crippling failure like loss of refrigerant charge. Avoiding these types of failures are not generally easy because most of the time the collected information, e.g., temperature, does not convey additional information on subsystems' status, e.g., the level of refrigerant charge. Distributed and additional sensing at finer granularity can provide a detailed look at operations, and, with appropriate signal processing, it can provide actionable information for preserving mission-capability and operational efficiency.

More detailed information on the entire system, including its subsystems, can facilitate a wide adoption of condition-based maintenance as more potential failures can be screened early. Condition-based maintenance is extremely beneficial in many perspectives. For example, consider a big ship, such as Navy's or Coastguard's, that has a number of motors associated with various tasks, e.g., fire hydrant pump, water pump, winder, generator, etc. They are operated at different times for different durations at different speeds with different loads. Naturally, they degrade at different rates, structurally or electrically, and need servicing at different times. If these motors are managed with failure-based maintenance, i.e., a degradation of a motor is found by a failure of a motor, the overhead involved in the repair is immense: more parts need to be replaced; more time is required for the repair; and a more skilled labor is required. Moreover, all of these inconveniences can affect mission-capability of a ship. As a remedy to this problem, more manpower can be assigned to inspect on each motor more frequently. However, this still does not guarantee a complete prevention, since, unless a problem is found in the inspection, not-yet-crippling issues may go unnoticed and later on develop into a real inconvenience. A better approach is to provide motors with condition-based maintenance. By deploying a monitoring package that includes multiple sensors for each motor, potential problems can be detected before the actual break-down of a motor. Not only would it decrease time and cost associated with the repair, but it would also decrease or even prevent the potential down-time of a motor that can affect mission-capability of a ship. Therefore, condition-based maintenance with sensing and control at finer levels is the ultimate goal of a sensor system. The complication of the condition-based maintenance approach is that each monitored subsystem, e.g., a motor, requires a monitoring package and a way to power it continuously while it collects information to infer the health of the monitored subsystem.

However, continuous monitoring raises one question: how is the monitoring package powered? If the monitoring package can access the existing power grid that powers the load under monitoring, it can be as simple as purchasing a commercially available product and plugging it to the wall for desired operations. In case such a product is not available, a custom sensor node that fits the monitoring objectives should be developed, using a wide variety of small and low-power sensors and data communication technologies. This electrical monitoring package will collect data, communicate with a user, and transfer meaningful data. Users might be able to extract actionable information from the received data by performing appropriate signal processing. However, such an electrical system comes with a price: sensing, control, and communication elements all require power and data paths that can quickly create a dizzying requirement for new or additional wiring. For this reason, there is a great attraction to a "dual-use" design philosophy especially in sensor and monitoring applications. If the monitoring package can squeeze one more function out of its existing systems, it can alleviate or even eliminate some of the wiring requirement. In many situations, the grid connection might not be available for a variety of reasons, e.g., safety, security, noise, or interference. One might consider a battery as a primary energy source, but it fundamentally has the same problem as the case with failure-based maintenance: batteries require periodic maintenance, rendering it not useful in continuous monitoring for condition-based maintenance. Energy harvesting can be a great solution in this perspective. It allows adequately low-power sensors and controls to operate from power "sources" derived from parasitic or symbiotic energy flows like mechanical vibration [2, 3, 4], thermal gradients [5, 6], acoustic vibrations [7], and light [8]. Not only can it provide power to situations without a grid connection, but also simplify power wiring in general even with external power connections. It is also an action of the "dual-use" design philosophy if the monitoring package extracts both energy and information from the load under monitoring.

Harvesting sufficient energy from these energy sources can add great flexibility to a monitoring package and open up a new range of installable targets without dealing with custom power wiring and inconvenient battery issues. This thesis extends this idea of developing an energy harvester, and applies it to the field of electromechanical diagnosis. An embedded system with an energy harvester for self-sustained operations is developed for nonintrusive load monitoring (NILM), which is an extremely beneficial feature for a monitoring system as it does not interfere with the existing system [9, 10]. Nonintrusive load monitoring is also beneficial in a case where it is not always guaranteed to have a complete control over a monitored load as many load monitoring techniques involve invasive methods, e.g., exciting a motor at specific frequencies for an analysis. The entire design and development of a vibration monitoring embedded system is presented, from modeling a physical phenomenon and

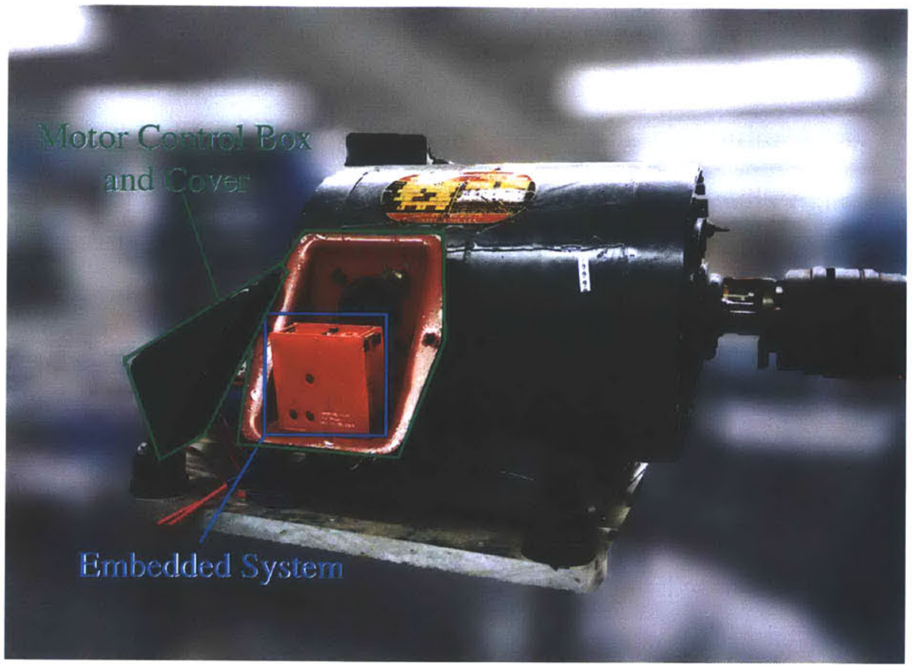


Figure 1-1: Notional Installation of the Embedded System

designing power electronic circuits to designing an embedded system and softwares for electromechanical diagnosis, specifically targeting motors inside a big ship.

Generally, requests from ship operators include the followings: first, a monitoring device should not make an ohmic contact to the existing equipment to block any potential power or data signal noise from affecting their power grid; second, it should not have external wiring visible to the outside to prevent physical hazard issues, e.g., wires going to nearby electrical outlets, solar panels on roof, or computers for power or data transfer purposes; third, it should be small that it can be neatly integrated into or installed inside existing structures without taking additional physical space; lastly, it should be able to sustain its operation for a long time. These constraints simply reconfirm the necessity of introducing an energy harvester.

Any energy harvesting scheme should be able to generate more power than what the monitoring package consumes in average power. Having a steady power influx of more than the threshold power consumption is the easiest way to solve the problem. However, considering ambient energy harvesting is not practically predictable nor controllable, with sparse and short energy peaks and long zero energy periods, an energy buffer is required, and a thorough estimation or observation has to be carried out beforehand to calculate the size of the required energy buffer to sustain the continuous average power consumption of the monitoring package. Power peaks determines the size of an energy receiving interface, and frequency of the influx of energy determines the size of an energy buffer. The combined size is a crucial factor as it determines the feasibility of the monitoring device — the entire self-powered embedded system — in the presence of the space constraint. In this thesis, the self-powered embedded system is designed to be retrofit into the control box of a motor to be safely enclosed and protected from the external environments, and at the same time not posing any hazard to the existing system as well. A notional illustration of an embedded system in a control box of a motor is presented in Fig. 1-1. Any harvesting method with predictability and controllability, therefore, gains an enormous advantage because the overall size of the energy harvester can be significantly lowered. Though most of energy harvesting schemes are based on sporadic energy extraction, the lower bound of the size of a harvester can be calculated assuming a continuous and steady influx of harvested energy — the best case with the smallest energy receiver and zero energy buffer. For example, for 100 mW extraction, temperature energy harvesting would require at least $1000\,\text{cm}^2$ [5] with the temperature gradient of $\Delta T = 10\,\text{K}$. Vibration energy harvesting would require at least $500\,\text{cm}^2$ [11]. Considering the face area of a usual control box of a motor is around $100\,\text{cm}^2$, even the most optimistic (and probably unrealistic) cases for temperature and vibration energy harvesting for 100 mW are unrealizable. Solar and wind can generally provide much higher than 100 mW ($15\,\text{mW/cm}^2$ for solar [11] and 1200 mW h/day for wind [11]), but these are not viable as well, inside a dark, small, and enclosed space.

This chapter continues with the introduction to a new perspective in an ambient energy harvesting with a much more predictable and controllable energy source that can satisfactorily address all the concerns arisen by the constraints: electromagnetic energy harvesting. A brief introduction to embedded system designs for a monitoring device then follows, and the thesis contribution and the detailed thesis flow are provided.

1.1 Electromagnetic Energy Harvesting

The energy harvester designed and presented in this thesis extracts energy from magnetic fields emanating from a wire that supplies power to a load of interest. The heart of the energy harvester is an inductive coupling with a magnetic core. A wire of the primary side, i.e., the load of interest, goes through the center of a magnetic core of the energy harvester. To minimize intrusiveness, a single winding is used for the primary. The secondary side of the core, i.e., the energy harvester, has a high number of windings in comparison. This physical structure is conventionally used for a current transformer for sensing current, but the usage of a core differs in that the goal of the harvester's core is to extract power from the electromagnetic coupling.

References [12, 13, 14, 15, 16], and [17] are examples of work exploring the problem of harvesting energy from magnetic fields. Generally these efforts focus on extracting energy from kinetic motion or vibration induced magnetic fields with permanent magnets. For example, [12] discusses a resonance-based vibration energy harvester with air-core coils and permanent magnets, and [13] and [14] perform similar analyses with magnetic cores with higher permeabilities. The references [15, 16], and [17] review the design of vibration or rotation based magnetic energy harvesters with permanent magnets. These magnetic harvesters in [15, 16], and [17] rely on parasitic inductances to operate boost converters. This thesis, on the other hand, presents a harvester extracting energy from magnetic fields solely through electromagnetic coupling without any association with permanent magnet or kinetic/vibration-based power sources. The approach does not rely on parasitic inductance and does not require a boost converter. It makes direct use of a permeable core as a nonlinear power source to harvest and transfer energy. Unlike ambient vibration or kinetic motion, magnetic fields generated by a current-carrying wire can be accurately calculated. Moreover, the current profile of a motor is much more predictable than kinetic or vibration-based ambient harvesting. According to the scheduled program of a motor throughout the day, the complete and accurate current profile might be predicted. As mentioned, this can significantly decrease the size and complexity of the harvesting circuitry.

Electromagnetic energy harvesting has several distinct characteristics. Due to its resemblance in physical construction to a current transformer for current sensing, the harvester poses very low insertion impedance on the primary side — the harvester should not interfere with the current supplied to a motor. In sensing applications, linearity is important, and core saturation is generally avoided during the design and operation of a magnetic core sensor. However, for energy harvesting, the fascinating result is found, directly in contrast to the situation for current sensing, that permitting the magnetic core of a current-driven transformer to saturate at an opportune time can in fact maximize the harvested energy. Since the transformer is current-driven, a rectifier on the secondary side is free from dead-time issues. There is additional benefit for employing electromagnetic energy harvesting in that it can also function as a current sensing transformer by control, providing electrical information of an operating electromechanical machine.

Available power density from the harvester may profoundly limit sample rate, data precision, signal processing, transmission bandwidth, and data storage capacity and accessing rates of a sensor node that would be powered by the harvester [11]. Provided with the size constraints for the harvester, it ultimately determines the feasibility of a self-powered sensor node. As briefly shown in the previous pages, ambient energy harvesting, e.g., temperature and vibration, generally has low power density such that it cannot provide an adequate power level even for a sensor node with tens of mW power consumption with the size of a control box of a motor. Electromagnetic energy harvesting, on the other hand, has a relatively higher power density. For example, the final prototype that will be presented in this thesis uses the magnetic core with the core volume of $2.9\,\mathrm{cm}^3$ (the surface area of $2.9\,\mathrm{cm}^2$ and the depth of $1.0\,\mathrm{cm}$) and can obtain approximately $12\,\mathrm{mW}$ per every ampere in the primary wire. The harvestable power is proportional to the primary current with the same harvester size because the primary side now generates stronger magnetic fields from which the energy is extracted. The direct scalability states that the electromagnetic energy harvesting can easily support a sensor system with up to hundreds of milliwatt or even a watt level power consumption with a kW or higher rated motor, which is usually the case for motors in a big ship. Furthermore, since magnetic fields from a motor are guaranteed to a certain level as long as the motor is in operation — the minimum idling current —, electromagnetic energy harvesting provides a much stable power influx, compared to other ambient harvesting methods with sparse influx. This can greatly ease the design of an energy buffer. For example, assuming the minimum idling current of a few amperes for a motor, electromagnetic energy harvesting can provide a few tens of mW continuously, and a monitoring package whose power consumption is less than the harvested amount can be self-sustained indefinitely without an energy buffer.

The detailed analysis and design of electromagnetic energy harvesting and power electronic circuits will be discussed in later chapters. Magnetic saturation, especially, will be extensively covered as it is the key to the maximization of power harvest. In the subsequent section, an embedded system that is powered by electromagnetic energy harvesting for electromechanical diagnosis will be introduced.

1.2 Vibration Monitor

Physical implementation is important for end-user applications, and should be considered from the design stages, especially when the spatial constraints are present. Some physical aspects of the self-powered embedded system that would be installed inside a motor can be determined early: one of the current-carrying power wires of a motor inside the control box would go through the center of the toroidal magnetic core, and a collection of printed circuit boards (PCBs) that contain a microcontroller, sensors, and other circuitries would be firmly attached to the control box of a motor, gathering sensor data; all these components need to be tightly assembled to fit inside a small enclosure, and maintain an electrical isolation from the motor's wiring; the packaging of the embedded system should be sufficiently resilient to withstand constant vibration of a motor for a long time. Incorporating the aforementioned aspects, several prototypes are built. The self-powered embedded system is named a vibration assessment monitoring point with integrated recovery of energy (VAMPIRE). VAMPIRE includes multiple sensors, such as an accelerometer, a temperature sensor, and a voltage sensor, multiple microcontrollers for different tasks, and multiple wireless communication devices to satisfy transmission bandwidth and power dissipation requirements.

The energy is extracted from electromagnetic coupling from a current-carrying wire and stored into a supercapacitor block through a rectifier. Since the embedded system starts from the zero energy state, all the initial control signals for harvesting and rectification are generated passively until a sufficient voltage is developed in a supercapacitor block to turn on a microcontroller. When the microcontroller becomes operational, power electronic circuits and power maximization circuits are properly controlled, significantly enhancing the amount of harvested power. The voltage of the supercapacitor will be actively regulated, and will supply power to the embedded system that contain sensors, processors, and communication devices. The sensor data will be sampled periodically and the data will be stored in a local temporary data storage. If a certain operating signature is detected, the temporary local storage will move the sensor data into a permanent storage. Users can access the embedded system through Bluetooth Low Energy (BLE) for command and real-time monitoring purposes. Users can also access the embedded system through Wi-Fi to quickly fetch the previously stored data from the permanent storage on the embedded system. Once the user is capable of accessing a desired set of sensor data, a signal processing can take place and detect a potential mechanical anomaly associated with the motor.

One thing to note is that VAMPIRE additionally collects vibration and electrical data during motor's spin-down events. Since it is essentially a step excitation, the new set of data collected during this event contains as much information in theory as a wide frequency sweep, even comparable to intrusive monitoring. However, when this event occurs, power to the motor no longer exists, and magnetic fields are no longer present. The embedded system therefore needs an energy buffer as previously discussed, and a power interface to the buffer is also required. User's requests through wireless communication can also take place after the motor stopped. This is inherently uncontrollable and unpredictable, but it has to be estimated to design the size of the buffer (or declare the maximum operation time supported with a certain size of a buffer). The detailed research on design and tradeoff will be provided in the VAMPIRE chapter in this thesis. Also, the algorithm on how to extract meaningful information or indication on the health of a motor from the spin-down data will be discussed.

Although this thesis is specifically focused on an electromechanical system, note that the design approach can be extended to any primary device that requires monitoring and that has an accessible current-carrying wire. For example, a light detecting sensor node for energy saving notification can be powered by the same electromagnetic energy harvesting scheme from a current-carrying wire of a lamp.

1.3 Thesis Contribution

The proposed energy harvesting scheme using electromagnetic coupling will be introduced in detail in Chapter 2. In contrast to a conventional usage of inductors and transformers, where magnetic saturation is avoided, the proposed energy harvesting scheme requires magnetic saturation for maximum energy extraction from the coupling. Magnetic saturation will be analyzed in depth, and a model to accurately calculate the amount of energy extraction with the presence of magnetic saturation is developed based on Maxwell's equations. A customized numerical solver is developed for actual simulation, and the numerical solver will be cross-checked against the experiment data.

In Chapter 3, power electronic circuits that can significantly enhance the energy extraction from the harvester will be proposed. Combined with the fact that the construction of a core is a current-driven transformer, several techniques will be introduced based on the concept of the "transfer window", which derives from periodic magnetic saturation. For a faster and easier circuit simulation, a new core model will be derived from the Maxwell model developed in Chapter 3. The new experiment data will be provided to validate the new model, and the enhancement on power harvest using proposed power electronic circuits will be verified. A tradeoff between the proposed power enhancing circuit techniques in case of environmental condition changes will be explored. As a reliable power supply to the subsequent sensor package, the harvester is analyzed in several other perspectives, such as power flow control, voltage regulation, and passive stage control. Finally, the electromagnetic energy harvester will be investigated in a special case of very high primary current.

The design of an embedded system will be covered in Chapter 4. The power budget and features list will be extensively discussed, and the integration of an energy buffer will be explored. Finally, VAMPIRE will be installed inside the control box of a motor, and extract a valuable vibration data from the motor that can diagnose an electromechanical problem of the motor. An algorithm and signal processing behind the diagnosis will be introduced.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Electromagnetic Energy Harvesting

2.1 Overview

An illustration of the electromagnetic energy harvester is presented in Fig. 2-1 and Fig. 2-2. Assume the primary side, i.e., an electromechanical load of interest, is operating and carrying an AC sinusoidal current in its power wiring, shown as a brown wire in Fig. 2-1. The sinusoidal AC current flowing through the wiring of the primary side naturally generates the time-varying magnetic field (H) around the wire, and the H field translates to the time-varying magnetic flux density (B) according to the B-H curve of the core material. If the magnetic core is wound with the secondary side, the time-varying B field is essentially a voltage, as the Maxwell-Faraday equation states, and the resulting voltage appears across two terminals of the secondary core windings. In fact, according to Ampere's law and the status of the physical connection of a load on the secondary side, a current may be induced on the secondary side, and it may in turn affect the H and B fields inside the core. To be more thorough about these interactions, the secondary side current should be included in the beginning together with the primary side current for the H and B fields. This leaves two unknown variables, i.e., the secondary voltage and the secondary current, in the equation obtained by the Maxwell-Faraday induction. Once the voltage-ampere (V-I) characteristic of the load on the secondary side is described, e.g., zero if the core is left disconnected on the secondary side or Ohm's law for a resistive load, the

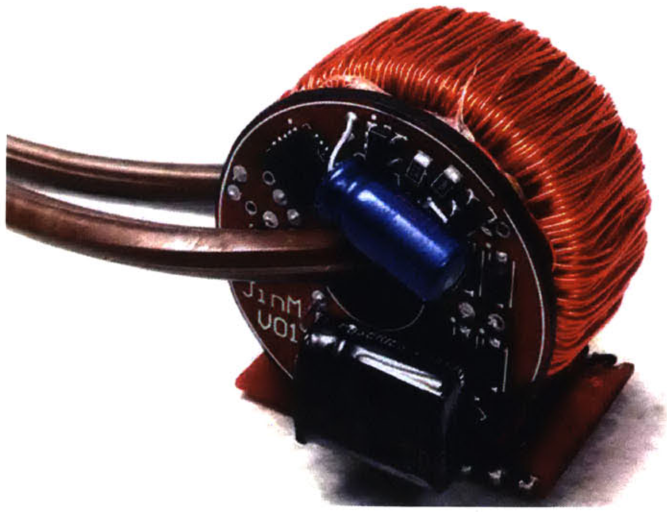


Figure 2-1: Notional Illustration of the Electromagnetic Energy Harvester

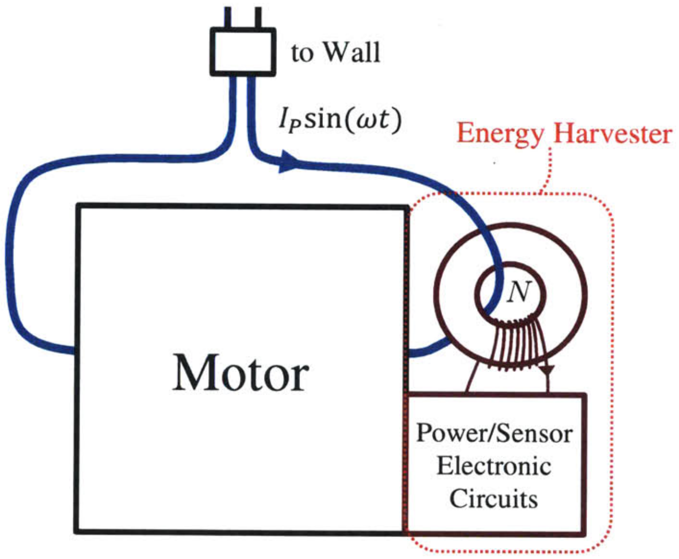


Figure 2-2: Notional Block Diagram

systems equation is now complete, and the exact systems response can be solved mathematically. By electromagnetically coupling the primary side and the secondary side, nonzero current and voltage can be induced on the secondary side at the same time. This implies that nonzero energy can be harvested from the primary side and transferred to the secondary side without an ohmic contact between the two.

However, the explanation thus far does not deviate at all from that of the principle of a current transformer. Structurally, a current transformer is almost identical. The most notable and crucial difference between a current transformer and an electromagnetic energy harvester is that for energy harvesting purposes the magnetic core is intentionally saturated to the full extent every line cycle (or twice in every line cycle considering a rectifier) for maximum power harvest. In normal applications with magnetic cores, such as inductors and transformers, the saturated region of the B-H loop is intentionally avoided because the linearity of a magnetic core is critical for correct operations. For example, a current sensor based on a current transformer, which also has a large number of windings on the secondary side, mirrors a fraction of the primary current on the secondary side. By letting the secondary side current go through a small resistor to develop a measurable voltage across it, the current on the primary side can be accurately estimated. If the core in a current transformer cannot guarantee a linear relationship between the core voltage and the primary side current, it simply cannot construct the current waveform of the primary side, rendering it useless. For inductors and transformers, their operations rely on a relatively flat constant inductance, which can be described as a single permeability around the steep slope region in the B-H curve, kept far away from saturation as well.

On the other hand, there are other applications that employ saturating magnetic cores. In magnetic amplifiers [18], magnetic switches [19, 20], or saturable reactor applications, such as [21, 22, 23], and [24], magnetic cores are intentionally saturated during the operation. However, they greatly differ in the usage that their purposes are for changing the insertion impedance to act as switches, not harvesting power.

Another thing to note is that the new energy harvesting scheme introduced in this thesis is clearly distinguishable from the existing works in magnetic energy harvesting, such as [15, 16], and [17]. As mentioned in Chapter. 1, the existing works harvest energy from vibration or kinetic motion with permanent magnets playing a vital role in generating such motions. Meanwhile, the electromagnetic energy harvester does not extract energy from vibration or kinetic motion, nor associate any permanent magnet in design. It purely harvests energy from an electromagnetic coupling.

In order to prepare for the actual harvester design, magnetic saturation will be

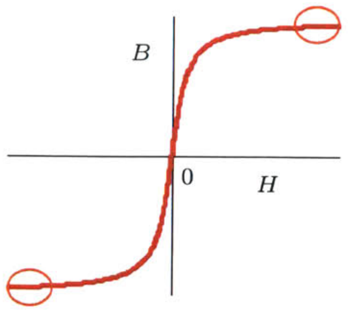


Figure 2-3: Saturated Flat Regions of the B-H Curve

addressed in the scope of energy harvesting in the next section, as it is relatively unfamiliar in power electronics circuits designs.

2.2 Magnetic Saturation

Physically, magnetic core saturation indicates that most of the magnetic moments existing in a core are aligned in the same direction already that additional magnetic field does not lead to an increase in magnetic flux density any faster than the rate at which nearby air molecules are magnetically aligning to the external field. Since the rate, which is essentially a scaled magnetic permeability, of the air is much lower than that of a magnetic material like cobalt, iron, iron powder, etc., the change in B with respect to the change in H after magnetic saturation becomes negligible. So, when a magnetic core goes into saturation, the operating point in the B-H curve of the core is in the flat tail regions where $\mathrm{d}B/\mathrm{d}H \approx 0$, as shown in Fig. 2-3. As the H field is generated by the AC sinusoidal current, the change in B with respect to the change in time is also negligible, implying the voltage developed across the terminals of the secondary side is near zero as well. From the core's standpoint, magnetic saturation is identical to an internal short since two different nodes are becoming a single voltage potential. It can be seen as the zero effective impedance across two terminals of the core as presented in Fig. 2-4. Therefore, by Kirchhoff's current law (KCL), whatever current extracted from the electromagnetic coupling cannot come out of the core once it saturates. Since the current delivered to the load on the secondary side is zero, it

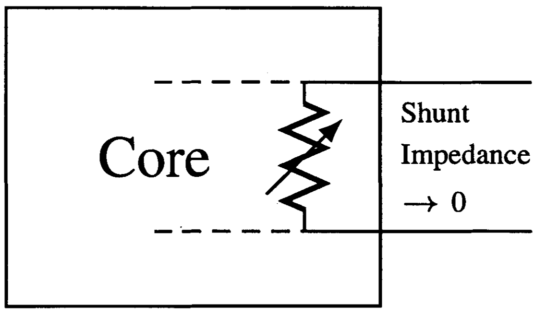


Figure 2-4: Zero Impedance across the Core under Saturation

is clear that energy harvested from electromagnetic coupling is zero when the core is saturated.

Meanwhile, when the magnetic core is not saturated, the situation is identical to a regular current transformer. The core is operated near the area where the magnetic permeability ($\mathrm{d}B/\mathrm{d}H$) is relatively higher than those flat tail regions. Since the core is not saturated, the secondary side of the current transformer appears as an ideal current source whose expression is determined only by the primary side current and the turns ratio, in parallel with shunt impedance exhibited by the magnetizing inductance. Assume a core material with an extremely high permeability, where the magnetizing inductance and shunt impedance are almost infinite such that the harvester core becomes an ideal current source with no other path than to the load on the secondary side. In this scenario, the entire full secondary side current without any loss, which is the maximum current, is extracted from the core and delivered to the load on the secondary side as if operated with a normal current transformer.

Combining the findings so far, two observations can be made: with an extremely high magnetic permeability, the entire transformer current is delivered to the load when the core is not saturated; on the other hand, zero current is delivered to the load when the core is saturated. These observations seem directly contradicting to the claim in the previous section that the core saturation maximizes the energy extraction, as no energy is harvested during magnetic saturation. However, it should be noted that the level of core saturation is not fixed at one operating point during harvesting operations. The level of core saturation is continuously and dynamically changing over

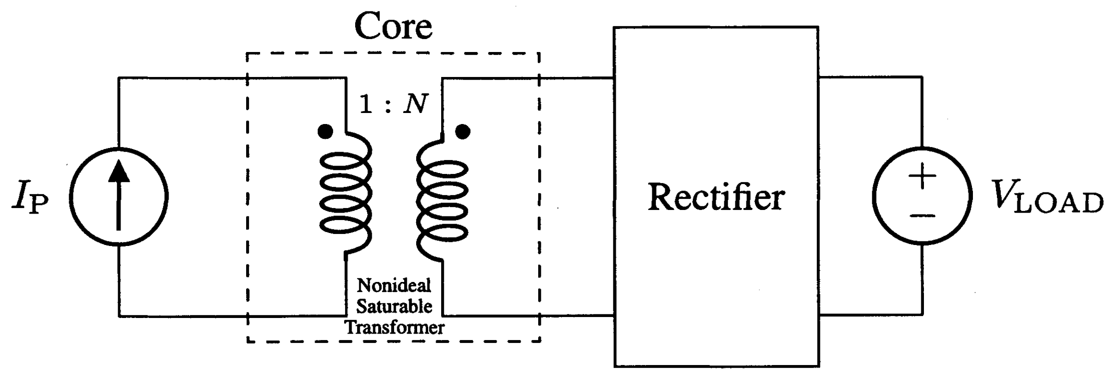


Figure 2-5: The Energy Harvester Circuit Block Diagram

the course of harvesting operations, and the core that goes in and out of magnetic saturation traverses the entire unsaturated regions of the B-H curve along the way. Due to this reason, the energy extraction over one periodic cycle should be considered for a comparison of the amounts of harvested power. Here, power extraction can be used interchangeably with energy extraction as the primary side carries a periodic sinusoid and the main interest in the analysis is the energy extraction per one periodic cycle.

The circuit block diagram of the electromagnetic energy harvester is illustrated in Fig. 2-5. The primary side AC current is denoted as I_P. The magnetic core is presented in a dashed box as a nonideal, nonlinear, and saturable transformer with a turns ratio of $1 : N$. The harvested AC current on the secondary side first goes through a rectifier and gets stored in a regulated voltage, e.g., a supercapacitor regulated at a certain voltage or an input stage of a power converter. The power harvested from the core is simply the average of the rectified harvested current multiplied by the load voltage that the harvested current is subjected to, which is denoted as V_{LOAD} in the figure. While the core is not saturated, the high magnetic permeability ensures that the harvested current going into the load is exactly identical to the rectified transformer current that is purely determined by the primary current divided by the turns ratio, independent of any other variables on the secondary side. Meanwhile, the core is accumulating volt-seconds because the load voltage is directly connected to the core while the harvested current is being transferred. Once the accumulated volt-second exceeds the physically allowed amount, which is a function of the saturation flux density level, the flux area, and a number of windings, the core goes into saturation, and energy harvesting practically halts as no current is delivered to the load. If a magnetic core does not saturate during operation, it directly implies that there is still a room for volt-seconds to grow without being saturated. Since the harvested current with an unsaturated core is determined by primary side information only, applying a higher load voltage to fill up the remaining volt-seconds will not change the average harvested current. However, the harvested power is directly increased as the load voltage is increased. Therefore, it can be stated that the core should be at least operated on the verge of magnetic saturation. The interesting part is that the maximum power is not achieved at this saturation boundary. The same approach can be used further into saturation, and the maximum power harvest point is actually at a slightly deeper saturation level, which is called a ‘soft saturation’ region.

Assume the core is on the verge of saturation, where the harvested current that goes into the load is still the entire sinusoid without any distortion. Now, the load voltage is linearly increased driving the core deeper into magnetic saturation. The core begins to enter the saturated regions of the B-H curve briefly at the end of the operation cycle, and the harvested current during this brief moment becomes zero, clipping the tail of the sinusoid. The clipping in the current waveform results in a lower average current. However, clipping starts from the zero crossing of the sinusoid, and the percentage loss in the average current — the area corresponding to the tail loss with respect to the entire area under the sinusoid curve — is initially much lower than the linear rate that the voltage is increasing at. Therefore, the amount of harvested power can still increase with a deeper saturation level. Considering extreme saturation leads to zero average current and zero harvested power, there should an optimal saturation level for power harvesting between the saturation boundary and the extreme saturation. The hand calculation and actual numerical simulation in the next sections will show that the optimal point occurs in this soft saturation region, where the clipping of the tail happens between the peak of the sinusoid and the initial zero crossing for the sinusoid. Here, the underlying assumption is that the harvester

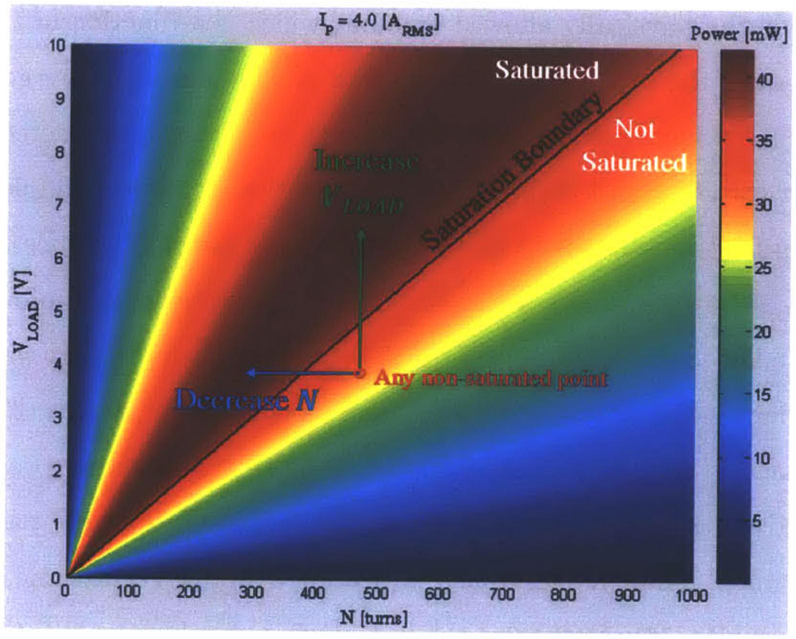


Figure 2-6: Harvestable Power vs. V_{LOAD} and N

runs without any circuit method proposed later in the thesis for power enhancement.

Here is another approach to illustrate the importance of magnetic saturation in power harvesting. Instead of filling up the remaining volt-seconds, the physically allowed amount of volt-seconds can be lowered to match the generated volt-seconds. Keeping the same core, this can be done by decreasing the turns ratio, N, which in turn increases the harvested current. If the core is still free of magnetic saturation after decreasing the turns ratio, the entire sinusoidal current waveform is still guaranteed to be delivered to the load voltage, only with the increased amplitude. Therefore, it can be stated again that the core should be at least operated on the verge of magnetic saturation. By going into deeper saturation, there is a tradeoff between the amplitude of the transformer current and the loss in the average harvested current due to saturation, identical to the previously mentioned tradeoff between the voltage increase and the loss in the average harvested current. In this case as well, the optimal power harvesting point still happens in the soft saturation region.

Figure 2-6 presents the simulation results of the harvestable power, with V_{LOAD} sweep in y-axis and N sweep in x-axis. The black line indicates the saturation boundary with the upper side being the saturated region and the lower side being the unsaturated region. Any point from the unsaturated region is guaranteed to be sub-optimal in power harvest as it can achieve a much greater amount of power by simply saturating it in either direction: by applying more volt-seconds with higher V_{LOAD}; or by lowering the physical limit with lower N. The maximum power points, portrayed as dark red, happen slightly inside the saturation boundary — the soft saturation region.

Before providing detailed mathematical derivations of electromagnetic energy harvesting to construct such simulations, the next section will briefly discuss a few important points required for a magnetic core to be used as an energy harvesting core.

2.3 Core Requirements

As previously discussed, it can be shown for any magnetic core that an unsaturated core is always suboptimal in energy harvesting and can harvest more power by saturating it to the right level. Since the core is intentionally driven to full saturation every operation cycle for the maximum efficiency, a low hysteresis loss becomes an important requirement for the core. The amount of hysteresis loss will add on to the lower bound of the power level that the energy harvester should generate for a certain load dissipation. It potentially determines the feasibility of a self-powered system design in case of extremely power- and size-constrained situations.

The second requirement for the core is that it should have high saturation flux density, B_{SAT}. Higher B_{SAT} physically implies that more magnetic energy can be crammed inside the same core volume, and, intuitively, a higher amount of harvested power is expected. It will be shown in the coming sections that the maximum power that can be extracted from the core is directly proportional to B_{SAT} in the first order. In fact, the maximally harvestable power is also proportional to the flux area, hence the size, of the core, A_{CORE}. However, the size of the core is generally not a design variable as it cannot be arbitrarily increased due to physical constraints, and it is not the material property of the core.

The last requirement is a high magnetic permeability. Considering the core serves as a current source when the core is not saturated, having a high magnetic permeability helps in that it leads to higher magnetizing inductance and shunt impedance, making the core a more ideal current source. The core as a current source should be able to instantaneously develop any arbitrary voltage that is required to sustain a current path into the load. For an easy illustration, consider a diode full-bridge rectifier connected to the core to rectify the harvested AC current and to deliver it to a supercapacitor charged at a certain voltage level. Whenever there is a nonzero — regardless of the amount — current being transferred to the load, the core is required to develop at least the load voltage plus the diode voltage drops to sustain that current. Having a high permeability can help this situation too, because the voltage across the magnetizing inductance is directly proportional to the inductance, which is again directly proportional to the magnetic permeability. So, a higher magnetic permeability leads to a wider support for developing sufficient load voltage levels, widening the design options. Also, it will be shown in the coming chapters that higher load voltage levels are desirable in several energy enhancing techniques. Therefore, an extremely high magnetic permeability is very useful in energy harvesting purposes. If the core is unable to develop a voltage that is required to enable the current path into the load, the entire current extracted from the electromagnetic coupling will internally circle through the magnetizing inductance and revert back to the primary side, and energy harvesting will never take place. Figure 2-7 illustrates the importance of a high magnetic permeability. In this simulation result, a full-bridge diode rectifier with each diode voltage drop of 0.5 V is assumed, and the supercapacitor is regulated at 5.0 V. The primary side of 90 W (120 V and 0.75 A_{RMS}) generates magnetic fields, and a turns ratio of $N = 200$ is assumed for the core. As seen in the figure, in order to provide nonzero energy harvesting, the relative permeability of at least 111,000 is required for the core. A higher permeability increases the average current into the load as shown in the figure because the core becomes a more ideal current source, but there is a diminishing return as the shunt impedance becomes sufficiently large. However, there are not many cores with the relative permeability on the order of

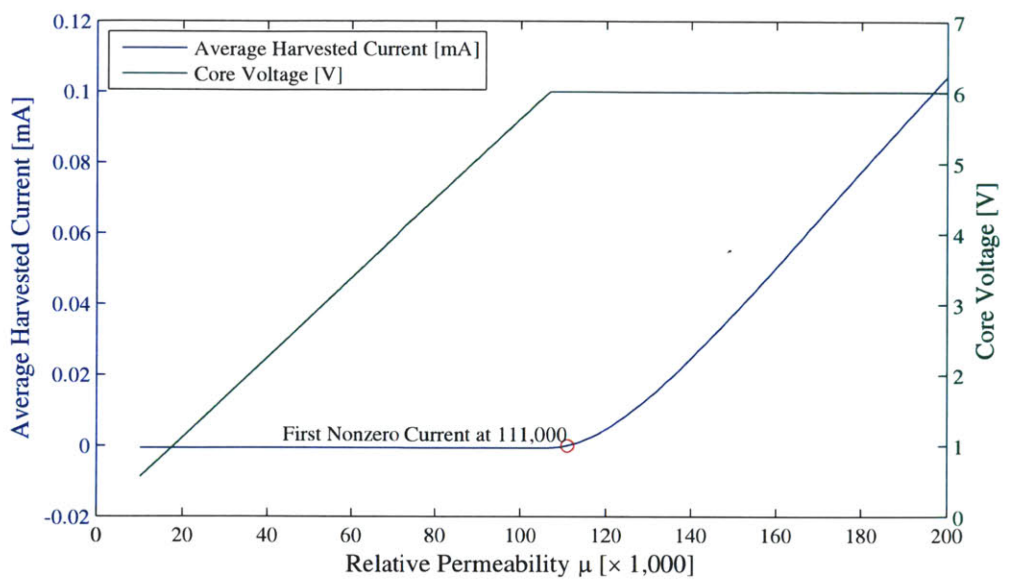


Figure 2-7: Relative Permeability and Energy Harvesting

hundreds of thousands. In this thesis, an amorphous nanocrystalline core [25] (VITROPERM 500F W380) from Vacuumschmelze (VAC) [26] is chosen after evaluating many different types of cores. It has a μW level hysteresis loss, B_{SAT} of approximately 1.2 T, and the relative permeability of approximately 274,027. Detailed information, including dimensions, is discussed in later sections.

2.4 Transfer Window

Whenever a harvested current from an electromagnetic coupling is transferred to a supercapacitor block, the core has to develop a voltage that is the supercapacitor voltage plus any loss in the rectifier and wire resistance across its terminals, hence accumulating volt-seconds. If the accumulated volt-seconds — flux — exceeds its physically allowed amount of accumulation, the core goes into saturation, forcing zero voltage across its terminals, and the core is no longer able to support any current into external blocks. With an extremely high magnetic permeability, the harvested current is almost identical to the secondary transformer current while the core is not saturated, and it rapidly drops to zero as soon as the core enters saturation. So, in the first order, it is reasonable to assume that power is harvested from electromagnetic coupling only when the core is not saturated. This relatively unsaturated time used for power extraction can be termed a 'transfer window.' In the subsequent sections, first order hand calculations based on this transfer window concept are explored. These calculations enables a quick determination of core size and load target, i.e., an easy "first cut" method for designing an electromagnetic energy harvester. The limits of the first order calculations based on the extent of the transfer window are also verified, providing a motivation for more accurate numerical analyses for fine-tuning a design, which are presented in the following sections.

2.5 First Order Hand Analysis

The analyses in this section involve two load types: resistive loads and constant voltage loads. The resistive load model is simple and illustrative of core behaviors, but not appropriate for an electromagnetic energy harvester, especially when the load is a regulated supercapacitor block or an input stage of a DC-DC converter. The reason is that the core is saturated every cycle unlike conventional transformers, outputting zero current for a considerable amount of time every cycle. For this reason, the constant voltage load model is more appropriate for these types of loads in the presence of magnetic saturation. The following subsections discuss the first order analyses with these two load types that are suitable for quick calculations. Note that the core saturation in these analyses is described by behavioral traits based on the concept of the transfer window, and do not include the detailed *B*-*H* curve of a core material nor Maxwell's equations.

2.5.1 Resistive Load (R_{LOAD}) Case

If the primary side current is a sinusoid with frequency of $\omega/2\pi$, the power delivered to the resistor pulsates at twice this frequency. The average power delivered to the load in each half cycle of the primary waveform can be computed given various assumptions about core saturation. The primary side period, the length of the transfer window (unsaturated time duration) in each half cycle, and the time point beginning a half cycle are denoted as T, t_{SAT}, and t_0, respectively. The average power harvest is:

$$
\begin{aligned}
P_{\mathrm{LOAD}} &= \frac{2}{T}\int_{t_0}^{t_0+t_{\mathrm{SAT}}} \left[\frac{I_{\mathrm{P}}}{N}\sin(\omega\, t)\right]^2 \cdot R_{\mathrm{LOAD}}\ \mathrm{d}t \\
&= \frac{I_{\mathrm{P}}^2 \cdot R_{\mathrm{LOAD}}}{\pi N^2}\left[\frac{\omega\, t_{\mathrm{SAT}}}{2} - \frac{\sin(2\,\omega\, t_{\mathrm{SAT}})}{4}\right]
\end{aligned}
\tag{2.1}
$$

When the core does not saturate ($t_{\mathrm{SAT}} = T/2$), the average power harvest simply becomes

$$
P_{\mathrm{LOAD,\,nonsat}} = \frac{I_{\mathrm{P}}^2 \cdot R_{\mathrm{LOAD}}}{2\,N^2} \tag{2.2}
$$

More generally, the core will saturate. To figure out t_{SAT}, a flux equality between the maximally allowed flux set by B_{SAT} for the core and the applied voltage integrated over $T/2$ can be solved. To align the zero crossings of the primary current and the power calculation, the initial time point (t_0) is set to 0. Then, the voltage integration is from 0 to t_{SAT}.

$$
\begin{aligned}
2\,B_{\mathrm{SAT}}\,A_{\mathrm{CORE}}\,N &= \int_0^{t_{\mathrm{SAT}}} V_{\mathrm{CORE}}(t)\ \mathrm{d}t \\
&= \int_0^{t_{\mathrm{SAT}}} \frac{I_{\mathrm{P}}}{N}\sin(\omega\, t)\cdot R_{\mathrm{LOAD}}\ \mathrm{d}t
\end{aligned}
\tag{2.3}
$$

The coefficient '2' before B_{SAT} on the left hand side comes from the fact that the core goes from one end of the B-H loop to the other end of the B-H loop in a half cycle, which results in a net change of $2\,B_{\mathrm{SAT}}$. Solving (2.3) gives

$$
t_{\mathrm{SAT}} = \min\left[\frac{1}{\omega}\cos^{-1}\left(1 - \frac{2\,\omega\, B_{\mathrm{SAT}}\,A_{\mathrm{CORE}}\,N^2}{I_{\mathrm{P}}\,R_{\mathrm{LOAD}}}\right),\ \frac{T}{2}\right] \tag{2.4}
$$

Since t_{SAT} is bounded by $T/2$, the minimum R_{LOAD} that saturates the core can be estimated, given I_{P} and N.

$$
R_{\mathrm{LOAD,min,sat}} = \frac{\omega\, B_{\mathrm{SAT}}\,A_{\mathrm{CORE}}\,N^2}{I_{\mathrm{P}}} \tag{2.5}
$$

In another special case where the core is heavily saturated, relatively early in the half cycle ($t_{\mathrm{SAT}} \ll T/2$), the sinusoidal current in (2.3) can be approximated as a linear function around zero.

$$2\,B_{\mathrm{SAT}}\,A_{\mathrm{CORE}}\,N \approx \int_0^{t_{\mathrm{SAT}}} \frac{I_{\mathrm{P}}}{N}\,\omega\,t \cdot R_{\mathrm{LOAD}}\,\mathrm{d}t \tag{2.6}$$

In this hard saturation regime, the expression for t_{SAT} is easily obtained without an inverse cosine function.

$$t_{\mathrm{SAT,\,hardsat}} \approx \sqrt{\frac{4\,B_{\mathrm{SAT}}\,A_{\mathrm{CORE}}\,N^2}{\omega\,I_{\mathrm{P}}\,R_{\mathrm{LOAD}}}} \tag{2.7}$$

Then, the average power harvest becomes

$$\begin{aligned} P_{\mathrm{LOAD,\,hardsat}} &\approx \frac{2}{T}\int_0^{t_{\mathrm{SAT}}} \left[\frac{I_{\mathrm{P}}}{N}\,\omega\,t\right]^2 \cdot R_{\mathrm{LOAD}}\,\mathrm{d}t \\ &= \frac{8}{3\,\pi}\,\frac{\omega^{1.5}\,I_{\mathrm{P}}^{0.5}\,B_{\mathrm{SAT}}^{1.5}\,A_{\mathrm{CORE}}^{1.5}\,N}{R_{\mathrm{LOAD}}^{0.5}} \end{aligned} \tag{2.8}$$

Comparing (2.2) to (2.8), the two expressions have opposite dependencies on R_{LOAD}. The harvested power increases with R_{LOAD} in the nonsaturation regime, and decreases with R_{LOAD} in the hard saturation regime. Therefore, it is expected that the peak will occur between two extremes.

Determining the exact R_{LOAD} value for maximum power harvest is challenging because the expression for P_{LOAD} is a complex function of t_{SAT}, and t_{SAT} also contains R_{LOAD} inside an inverse cosine function. By finding an extrema using (2.1) and (2.4), the following equation set is obtained, which can be solved numerically to find a value for R_{LOAD} that yields maximum power harvest:

$$\left\{ \begin{aligned} R_{\mathrm{LOAD}} &= \frac{2\,\omega\,t_{\mathrm{SAT}} - \sin(2\,\omega\,t_{\mathrm{SAT}})}{1-\cos(2\,\omega\,t_{\mathrm{SAT}})} \\ &\quad \times \frac{\omega\,B_{\mathrm{SAT}}\,A_{\mathrm{CORE}}\,N^2\,\sin(\omega\,t_{\mathrm{SAT}})}{I_{\mathrm{P}}\,[1-\cos(\omega\,t_{\mathrm{SAT}})]^2} \\ t_{\mathrm{SAT}} &= \frac{1}{\omega}\cos^{-1}\left(1 - \frac{2\,\omega\,B_{\mathrm{SAT}}\,A_{\mathrm{CORE}}\,N^2}{I_{\mathrm{P}}\,R_{\mathrm{LOAD}}}\right) \end{aligned} \right. \tag{2.9}$$

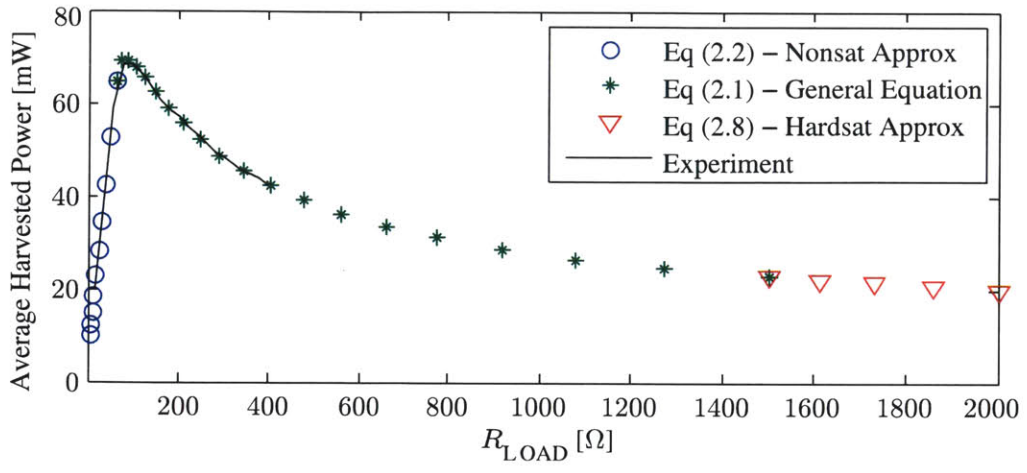


Figure 2-8: P_{LOAD} Response with a Resistive Load

Fig. 2-8 illustrates an example with $I_{\mathrm{P}} = 6.27\,\mathrm{A_{RMS}}$, $N = 200$, and freq $= 60\,\mathrm{Hz}$. In this figure, (2.8) is verified as a very close approximation to (2.1) when R_{LOAD} is relatively large, i.e., where the core enters hard saturation. More importantly, the maximum power harvest happens in the soft saturation regime, the region in between the two extremes. The solid black line indicates the experimental result.

2.5.2 Constant Voltage Load (V_{LOAD}) Case

For powering sensors and signal processing hardware, the energy harvester will likely provide power to a conversion or storage stage, not just a resistor, after a rectification. If a DC-DC converter with a switching frequency much higher than the line frequency is connected to the core as a load, the voltage across the core is effectively the cycle average of the input voltage of the converter, and it can be considered as a constant DC value to the core. Similarly, if a supercapacitor is used, due to its extremely high capacitance and the efforts of a post-regulator, the harvester again sees essentially constant DC voltage. In these cases, the important parameter for determining power transfer is the load voltage V_{LOAD}.

The calculation of the flux equality with a constant voltage load is much simpler due to the time independent load voltage.

$$2\,B_{\mathrm{SAT}}\,A_{\mathrm{CORE}}\,N = \int_0^{t_{\mathrm{SAT}}} V_{\mathrm{LOAD}}\,\mathrm{d}t \tag{2.10}$$

Therefore,

$$t_{\mathrm{SAT}} = \min\left[\frac{2\,B_{\mathrm{SAT}}\,A_{\mathrm{CORE}}\,N}{V_{\mathrm{LOAD}}},\ \frac{T}{2}\right] \tag{2.11}$$

The average power harvest can be generally expressed as

$$\begin{aligned} P_{\mathrm{LOAD}} &= \frac{2}{T}\int_0^{t_{\mathrm{SAT}}} \left[\frac{I_{\mathrm{P}}}{N}\sin(\omega\,t)\right]\cdot V_{\mathrm{LOAD}}\,\mathrm{d}t \\ &= \frac{I_{\mathrm{P}}\,V_{\mathrm{LOAD}}}{\pi\,N}\left[1-\cos(\omega\,t_{\mathrm{SAT}})\right] \end{aligned} \tag{2.12}$$

Equation (2.12) can be used to calculate the average power harvest for the special case where the core does not saturate ($t_{\mathrm{SAT}} = T/2$).

$$P_{\mathrm{LOAD,\,nonsat}} = \frac{2\,I_{\mathrm{P}}\,V_{\mathrm{LOAD}}}{\pi\,N} \tag{2.13}$$

The average power harvest in hard saturation ($t_{\mathrm{SAT}} \ll T/2$) is:

$$\begin{aligned} P_{\mathrm{LOAD,\,hardsat}} &\approx \frac{2}{T}\int_0^{t_{\mathrm{SAT}}} \left[\frac{I_{\mathrm{P}}}{N}\,\omega\,t\right]\cdot V_{\mathrm{LOAD}}\,\mathrm{d}t \\ &= \frac{2\,\omega^2\,I_{\mathrm{P}}\,B_{\mathrm{SAT}}^2\,A_{\mathrm{CORE}}^2\,N}{\pi\,V_{\mathrm{LOAD}}} \end{aligned} \tag{2.14}$$

Similar to the resistive load case, the expressions for the average power harvest in the two extremes have opposite dependencies on the main variable, V_{LOAD}. The peak will occur in between unsaturated and hard saturated operations. The maximum power harvest point is relatively easily found for the case of a voltage load by differentiating

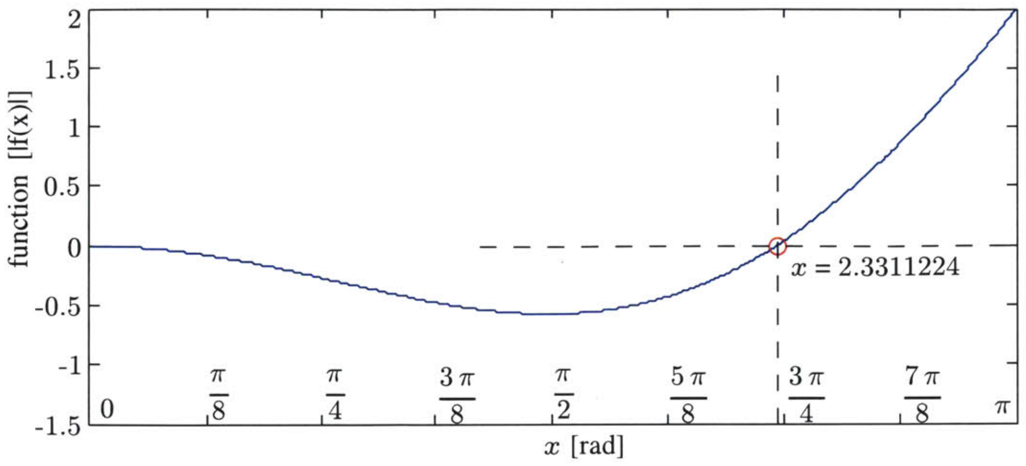


Figure 2-9: Trace of $f(x) = 1 - \cos(x) - x \cdot \sin(x)$

P_{LOAD} with respect to V_{LOAD} and solving for extrema:

$$\begin{aligned} 0 &= 1 - \cos\left(\frac{2\,\omega\, B_{\text{SAT}}\, A_{\text{CORE}}\, N}{V_{\text{LOAD}}}\right) \\ &\quad - \frac{2\,\omega\, B_{\text{SAT}}\, A_{\text{CORE}}\, N}{V_{\text{LOAD}}} \sin\left(\frac{2\,\omega\, B_{\text{SAT}}\, A_{\text{CORE}}\, N}{V_{\text{LOAD}}}\right) \end{aligned} \tag{2.15}$$

Equation (2.15) is in the form:

$$1 - \cos(x) - x \cdot \sin(x) = 0 \tag{2.16}$$

This equation has an obvious but impractical solution at $x = 0$. To avoid zero, an additional condition on x can be inferred:

$$x = \frac{2\,\omega\, B_{\text{SAT}}\, A_{\text{CORE}}\, N}{V_{\text{LOAD}}} = \omega\, t_{\text{SAT}} \leq \omega\, \frac{T}{2} = \pi \tag{2.17}$$

Based on (2.17), the trace of $f(x) = 1 - \cos(x) - x \cdot \sin(x)$ is drawn on Fig. 2-9 up to $x = \pi$. As shown in the figure, it has a single nonzero solution at $x = 2.33$. Using $3\pi/4$ as an approximate solution, the optimum V_{LOAD} can be expressed as

$$V_{\text{LOAD,pmax}} \approx \frac{8}{3\pi} \cdot \omega\, B_{\text{SAT}}\, A_{\text{CORE}}\, N \tag{2.18}$$

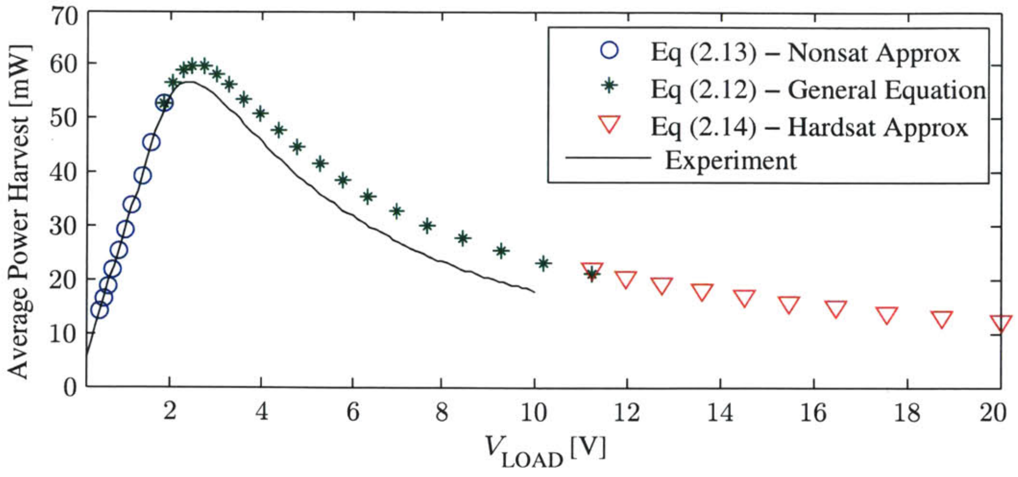


Figure 2-10: P_{LOAD} Response with a Constant Voltage Load

And the corresponding maximum power harvest is

$$P_{\mathrm{LOAD,max}} \approx \frac{8\,(2+\sqrt{2})}{6\,\pi^2} \cdot \omega\, I_{\mathrm{P}}\, B_{\mathrm{SAT}}\, A_{\mathrm{CORE}} \tag{2.19}$$

Figure 2-10 illustrates an example with $I_{\mathrm{P}} = 6.27\,\mathrm{A_{RMS}}$, $N = 200$, and freq $=$ 60 Hz. The maximum power harvest happens in between the unsaturation region and the hard saturation region. The expression of $P_{\mathrm{LOAD,\,max}}$ is independent of the magnetic permeability (μ) of the core. However, as discussed in Chapter. 1, μ needs to be high to be able to develop a sufficient voltage across the core to sustain a current path. Also, high μ improves coupling to the primary current, allowing the core to be a better current source.

The first order estimate in Fig. 2-10 deviates from the experimental observations as the core goes into deeper saturation. This error derives from the fact that the beginning or initiation of the transfer window does not perfectly align with the zero crossing of the primary side current. In (2.1) and (2.12), the initial time point of zero in the integral range and the zero phase angle of the sine function assume a perfect alignment between the beginning of the transfer window and the zero crossing of the primary side current. In practice, the transfer window "opens" earlier than the primary current zero crossing because the core comes out of the saturation ahead of the zero crossing. As soon as the core gets out of the flat tail region of the B-H

loop before the zero crossing, a non-zero slope in the B-H loop restores magnetizing inductance. Once the magnetizing inductance is restored, the magnetizing current cannot rapidly track the transformer current, and the current difference between them must be flown into the load, opening up the transfer window earlier than before. This nonideality brings significantly different results for the resistive load illustrated in Fig. 2-8 and the voltage load illustrated in Fig. 2-10.

For the resistive load, the voltage developed across the core is always proportional to the load current. Therefore, even with a modeling error in estimating the start time of the transfer window, the flux accumulation estimate does not differ greatly, as the voltage and flux accumulation are relatively negligible at the beginning of the transfer window. Therefore, the end point of the transfer window in the estimate does not deviate much from the experiment, showing a similar average power harvest, as in Fig. 2-8.

On the other hand, for the voltage load case, as soon as the load current exists, the fixed load voltage is applied across the core, and develops core flux at a fixed rate regardless of the level of the transformer current. Therefore, any mis-estimate in the start time of the transfer window will shift the end time of the transfer window by the same amount. For example, if the transfer window opens early by Δt, the transfer window closes early by the same amount Δt. Since power transfer is relatively low in the vicinity of the zero crossing of the primary sinusoidal current, early opening of the transfer window does little to affect the power harvest. However, early closing of the transfer window will significantly lower the power harvest as substantially higher primary current is flowing after the zero crossing. The worst case loss happens when the transfer window ends at the peak of the transformer current. This issue tends to be worse in the hard saturation regime because the transfer window is shorter.

Since the constant voltage load model, which is a more realistic load representation than the resistive load model, does not accurately model the physical behavior of core operations, a mathematically more thorough approach is taken in the following section to develop an accurate core model. The nonlinearity and nonidealities of a magnetic core in the presence of magnetic core saturation are fully included in the model.

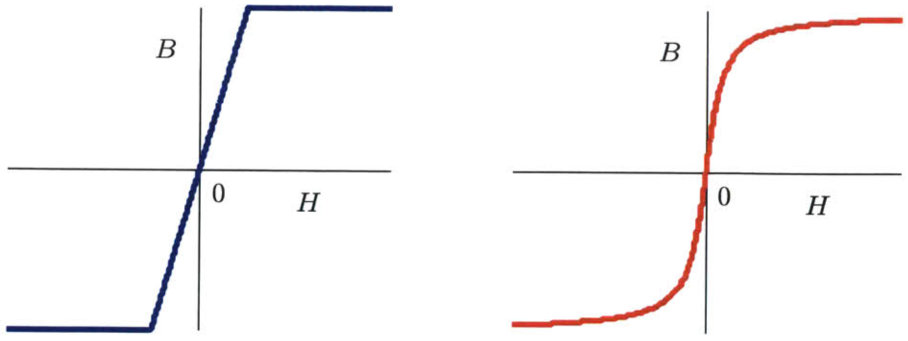


Figure 2-11: Saturating B-H Curve Models (Left: Piecewise Linear / Right: arctan)

2.6 Maxwell Model

This section presents a core model that becomes the basis of a numerical solver to predict behaviors of the harvester with excellent accuracy. The model, derived from Maxwell's equations, is especially useful for refining a target design. Nonidealities such as hysteresis core loss and wire losses are considered.

2.6.1 Derivation

Denoting the secondary side current as $I_S(t)$, the net ampere-turn seen by the core is

$$\mathrm{AT}_{\mathrm{CORE}} = I_P \sin(\omega\, t) - N \times I_S(t) \tag{2.20}$$

The primary side current is assumed sinusoidal AC with an amplitude of I_P. The magnetic field $H(r,t)$ in the core is

$$H(r,t) = \frac{\mathrm{AT}_{\mathrm{CORE}}(t)}{2\,\pi\, r} = \frac{I_P \sin(\omega\, t) - N\, I_S(t)}{2\,\pi\, r} \tag{2.21}$$

The magnetic flux density, B, is determined by the B-H curve of the core. The preliminary analysis below considers a saturating but non-hysteretic core, followed by an expansion to include hysteresis. The B-H curve of the non-hysteretic core including saturation can be modeled in various mathematical functions. An example is a piecewise linear waveform as in the left of Fig. 2-11. A more refined model uses the 'arctan' function as shown on the right of Fig. 2-11. With the 'arctan' function, the magnetic flux density can be modeled as

$$\begin{aligned} B(r,t) &= B_{\mathrm{SAT}} \cdot \frac{2}{\pi}\arctan\left(\frac{H(r,t)}{\alpha}\right) \\ &= B_{\mathrm{SAT}} \cdot \frac{2}{\pi}\arctan\left(\frac{I_{\mathrm{P}}\sin(\omega\, t) - N\, I_{\mathrm{S}}(t)}{2\,\pi\, r\, \alpha}\right) \end{aligned} \tag{2.22}$$

A scale factor of $2/\pi$ is used to normalize the arctan function to 1 when saturated. The reciprocal of α describes the sensitivity in the nonsaturated region, essentially representing the initial permeability in conventional models.

Denote the height, outer radius, and inner radius of the toroidal core as h, r_{OD}, and r_{ID}, respectively. Voltage across the core is:

$$V_{\mathrm{CORE}}(t) = \int_{r_{\mathrm{ID}}}^{r_{\mathrm{OD}}} N \cdot h \cdot \frac{\partial B(r,t)}{\partial t}\, \mathrm{d}r \tag{2.23}$$

The time derivative of the magnetic flux density can be calculated by differentiating (2.22).

$$\frac{\partial B(r,t)}{\partial t} = \frac{B_{\mathrm{SAT}}}{\pi^2\, r\, \alpha} \cdot \frac{\omega\, I_{\mathrm{P}}\cos(\omega\, t) - N\dfrac{\partial I_{\mathrm{S}}(t)}{\partial t}}{1 + \dfrac{\left[I_{\mathrm{P}}\sin(\omega\, t) - N\, I_{\mathrm{S}}(t)\right]^2}{4\,\pi^2\, r^2\, \alpha^2}} \tag{2.24}$$

Integrating (2.24) over r from r_{ID} to r_{OD} , equation (2.23) can be evaluated:

$$\begin{aligned} V_{\mathrm{CORE}}(t) =& \frac{N \cdot h \cdot B_{\mathrm{SAT}}}{2\pi^2\alpha} \times \left[\omega\, I_{\mathrm{P}}\cos(\omega t) - N\frac{\partial I_{\mathrm{S}}(t)}{\partial t}\right] \\ & \times \ln\left(\frac{r_{\mathrm{OD}}^2 + \dfrac{\left[I_{\mathrm{P}}\sin(\omega t) - N\, I_{\mathrm{S}}(t)\right]^2}{4\,\pi^2\alpha^2}}{r_{\mathrm{ID}}^2 + \dfrac{\left[I_{\mathrm{P}}\sin(\omega t) - N\, I_{\mathrm{S}}(t)\right]^2}{4\,\pi^2\alpha^2}}\right) \end{aligned} \tag{2.25}$$

This is the *V-I* characteristic of the core in the presence of magnetic saturation. Note that it is a transcendental nonlinear differential equation, which is not intuitively straightforward. In order to predict the accurate behavior of the core operation, a numerical solver based on this *V-I* characteristic is developed. First, lossy elements that should be added to this core model are discussed in the following subsections.

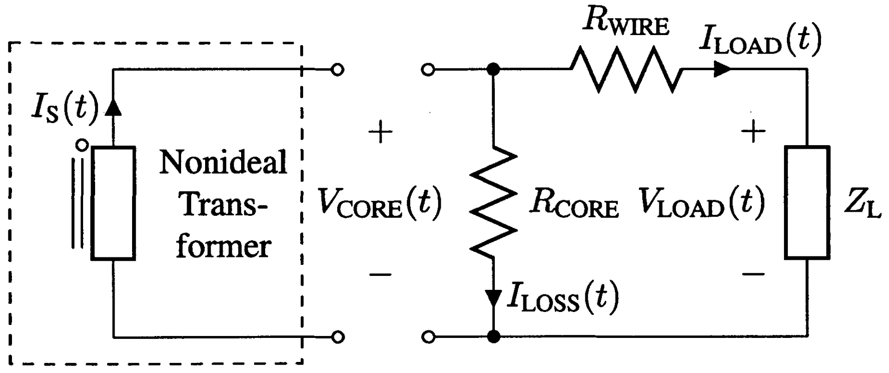


Figure 2-12: Circuit representation of Maxwell method

2.6.2 Loss Modeling

A hybrid circuit representation of (2.25) is given as a two-port box in a dashed line in Fig. 2-12. The leakage inductance is relatively small for the experimental core and is ignored in this figure. Wire loss is modeled with R_{WIRE}, which is in series with the load. The effect of hysteresis on the harvester is included with a resistance R_{CORE} in parallel with the core.

Given a load model relating $V_{\mathrm{CORE}}(t)$ and $I_{\mathrm{S}}(t)$, a complete set of differential equations that describes the system can be developed. Assuming the core is connected to an external circuit, a set of general system equations can be obtained by combining (2.25) and the following equations set:

$$\begin{cases} I_{\mathrm{S}}(t) &= I_{\mathrm{LOAD}}(t) + I_{\mathrm{LOSS}}(t) \\ V_{\mathrm{CORE}}(t) &= I_{\mathrm{LOSS}}(t) \cdot R_{\mathrm{CORE}} \\ V_{\mathrm{CORE}}(t) &= I_{\mathrm{LOAD}}(t) \cdot R_{\mathrm{WIRE}} + V_{\mathrm{LOAD}}(t) \end{cases} \tag{2.26}$$

Given an accurate load model, e.g., an equation relating $V_{\mathrm{LOAD}}(t)$ and $I_{\mathrm{LOAD}}(t)$, a systems response with any circuit configuration can be accurately calculated. The load can be any type, e.g., a resistive load model, i.e., $V_{\mathrm{LOAD}}(t) = I_{\mathrm{LOAD}}(t) \cdot R_{\mathrm{LOAD}}$, or a constant voltage load model with a rectifier or custom switching patterns. Except for this load characteristic, the remaining unknowns in the systems equations are the lossy elements, R_{WIRE} and R_{CORE}.

Wire Loss Modeling

Primary currents for the harvester are typically (but not necessarily) from the power line of monitored equipment, e.g., a motor, operating at line frequency. At 60 Hz, the skin depth of the copper wire is,

$$\delta = \sqrt{\frac{2\,\rho}{\omega\mu}} = \sqrt{\frac{2 \cdot 1.68 \times 10^{-8}}{2\,\pi\,60 \cdot 4\,\pi \times 10^{-7}}} = 8.42\,\text{mm} \tag{2.27}$$

The wire diameter for the secondary windings is usually much smaller than 8.42 mm. For example, in the prototype design, AWG 30 with a diameter of 0.255 mm is used [27]. Because δ is much larger than the wire diameter, the skin effect can be ignored, although this could be important in other applications. Following the analysis discussed in [28] and [29], the proximity effect is also negligible at the line frequency using this wire gauge. As an example, assume AWG 30 with 500 turns and 3 layers. Each turn is roughly 42.5 mm long, and $R_{\text{DC}} = 7\,\Omega$. To convert the round-wire windings into equivalent foil conductors, each conductor is modeled with $N/3$ windings, and effective height of $2\,\pi\,r_{\text{ID}}$. The effective width of each "foil" is,

$$w_{eff} = \frac{\pi\,r_{\text{AWG30}}^2 \times N/3}{2\,\pi\,r_{\text{ID}}} = 0.164\,\text{mm} \tag{2.28}$$

And,

$$\Delta = \frac{w_{eff}}{\delta} = 0.0195 \tag{2.29}$$

With $M = 3$,

$$\begin{aligned} F_R = \frac{R_{\text{AC}}}{R_{\text{DC}}} = \Delta \cdot &\left[\frac{\sinh(2\,\Delta) + \sin(2\,\Delta)}{\cosh(2\,\Delta) - \cos(2\,\Delta)}\right. \\ &\left.+ \frac{2\,(M^2-1)}{3} \cdot \frac{\sinh(\Delta) - \sin(\Delta)}{\cosh(\Delta) + \cos(\Delta)}\right] \qquad (2.30) \\ = \;& 1.00000 \end{aligned}$$

Therefore, the proximity effect is negligible, and only DC resistance is required for wire loss modeling.

Core Loss Modeling

As will be shown later, hysteresis loss for the VAC core is so low that it can be ignored if the output power is larger than several mW. Here, core loss is discussed to provide a general model for cores with higher losses. Among many techniques to estimate core loss at given frequency and B_{PEAK} level [30, 31, 32, 33, 34, 35], the core loss model in this thesis is based on a simple expression of [30] due to zero DC-bias and symmetric B-H loop operations centered at zero. The loss analysis here is proportional to $f^1\,\hat{B}^2$ in principle, though the actual implementation is based on the cyclic measurements of B_{PEAK} and I_{PEAK}.

To model core loss, a rectangular approximation of the B-H curve is made since the core has very high initial permeability. The peak load current (I_{PEAK}), the peak core voltage (hence B_{PEAK}), and the RMS voltage of the core (V_{RMS}) are tracked in every cycle. These values can be used to estimate a B-H loop area. In a numerical simulation, this power loss can be calculated and used to update the resulting core loss resistance for the next cycle. Assuming the core maintains its high permeability until saturation, where Volume is the volume of the core and H_C is the coercivity of the core, the full hysteresis loss is modeled as

$$P_{\text{LOSS-MAX}} = 2\,H_C \cdot 2\,B_{\text{SAT}} \cdot \text{Volume} \cdot \text{freq} \tag{2.31}$$

The B-H loop loss for any particular operating cycle can be calculated as a fraction of the maximum loss:

$$P_{\text{LOSS}} = P_{\text{LOSS-MAX}} \cdot \frac{I_{\text{PEAK}}}{I_{\text{SAT}}} \cdot \frac{B_{\text{PEAK}}}{B_{\text{SAT}}} \tag{2.32}$$

Finally, the loss resistance is

$$R_{\text{CORE}} = \frac{V_{\text{RMS}}^2}{P_{\text{LOSS}}} \tag{2.33}$$

This model calculation must be performed over a full cycle in order to permit calculation of the RMS values. The modeled resistor is supposed to dissipate the required amount of power over a cycle (or, if using a rectifier, a half cycle). However,

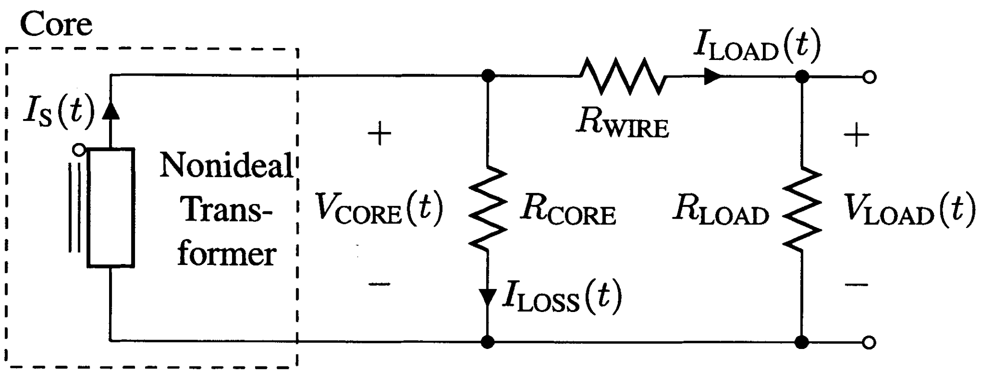


Figure 2-13: Test Circuit for the Parameter Estimation

adding such a resistor (or changing the loss resistance) will change the current divider formed with the magnetizing inductance, the core loss resistor, and the load. Since the entire circuit is continuously affected by cyclic update of the loss resistance, an additional numerical solver is used as an "outer loop" to provide a convergence to a correct core loss resistance, operating to provide correct values to the time-domain circuit solver.

2.6.3 Parameter Estimation

All the required circuit elements for numerical simulation have been discussed. However, three parameters in the *B*-*H* curve model and the core loss model are still unknown. In order to numerically solve the systems equations, three parameters must be experimentally estimated first: B_{SAT}, α, and $P_{\text{LOSS-MAX}}$.

B_{SAT} Estimation

The core can be characterized experimentally as shown in Fig. 2-13, with a resistive load connected to the core. A high load resistance helps characterize the core by forcing the core into hard saturation. In hard saturation, the analysis simplifies because hysteresis loss is at maximum, which ensures that the core is driven to $\pm B_{\text{SAT}}$, and that a distinct "cat-ear" shape in secondary voltage makes it easy to identify the temporal extent of the transfer window.

In a half cycle, $\Delta B = 2\,B_{\mathrm{SAT}}$, hence $\Delta\Lambda = 2\,B_{\mathrm{SAT}}\,A_{\mathrm{CORE}}\,N$. This must be equal to the core voltage integrated over a half cycle. With a measurement of the load voltage over a half cycle:

$$\frac{R_{\mathrm{WIRE}} + R_{\mathrm{LOAD}}}{R_{\mathrm{LOAD}}} \cdot V_{\mathrm{LOAD,\,AVG}} \cdot \frac{T}{2} = 2\,B_{\mathrm{SAT}}\,A_{\mathrm{CORE}}\,N \tag{2.34}$$

Therefore,

$$B_{\mathrm{SAT}} = \frac{T\,(R_{\mathrm{WIRE}} + R_{\mathrm{LOAD}})}{4\,A_{\mathrm{CORE}}\,N\,R_{\mathrm{LOAD}}} \cdot V_{\mathrm{LOAD,\,AVG}} \tag{2.35}$$

$P_{\mathrm{LOSS-MAX}}$ and α Estimation

The same circuit can be used for estimating B_{SAT} to determine core loss, and operate the core in the saturation regime to ensure that the hysteresis loss is at $P_{\mathrm{LOSS-MAX}}$.

With a resistive load, the general systems equation set, (2.25) along with (2.26), can be simplified to a single equation about $I_{\mathrm{LOAD}}(t)$:

$$\begin{aligned} &\frac{N\,h\,B_{\mathrm{SAT}}}{2\,\pi^2\,\alpha} \times \left[\omega\,I_{\mathrm{P}}\cos(\omega\,t) - \gamma\,N\frac{\partial I_{\mathrm{LOAD}}(t)}{\partial t}\right] \\ &\times \ln\left(\frac{r_{\mathrm{OD}}^2 + \dfrac{[I_{\mathrm{P}}\sin(\omega t) - \gamma\,N\,I_{\mathrm{LOAD}}(t)]^2}{4\,\pi^2\alpha^2}}{r_{\mathrm{ID}}^2 + \dfrac{[I_{\mathrm{P}}\sin(\omega t) - \gamma\,N\,I_{\mathrm{LOAD}}(t)]^2}{4\,\pi^2\alpha^2}}\right) \\ = \;& I_{\mathrm{LOAD}}(t) \cdot (R_{\mathrm{WIRE}} + R_{\mathrm{LOAD}}) \end{aligned} \tag{2.36}$$

where γ is defined as

$$\gamma = \frac{R_{\mathrm{CORE}} + R_{\mathrm{WIRE}} + R_{\mathrm{LOAD}}}{R_{\mathrm{CORE}}} \tag{2.37}$$

Two equations about α and γ can be generated by evaluating (2.36) at two different time points. For the evaluations, a full cycle of the primary side current and $V_{\mathrm{LOAD}}(t)$ is captured experimentally. The cosine and sine terms can be directly calculated with the selected time points, and $I_{\mathrm{LOAD}}(t)$ can be calculated by $V_{\mathrm{LOAD}}(t)/R_{\mathrm{LOAD}}$. An important point to note is that the zero crossings of the primary side current and $V_{\mathrm{LOAD}}(t)$ do not align in general. Therefore, during evaluations of $V_{\mathrm{LOAD}}(t)$ at two time points, the phase shift of $V_{\mathrm{LOAD}}(t)$ with respect to the zero crossing of the primary side should be considered. The variable α, whose reciprocal is practically a scaled initial permeability in conventional models, can be directly obtained by solving the two resulting equations. The other solved variable γ (with $V_{\text{LOAD,RMS}}$ that can be computed with the extracted cycle data) leads to:

$$P_{\text{LOAD-MAX}} = \frac{(\gamma - 1)\, V_{\text{LOAD,RMS}}^2}{R_{\text{WIRE}} + R_{\text{LOAD}}} \tag{2.38}$$

Magnetic Permeability (μ_r) Estimation

Differentiating both sides of (2.22) with $H(r,t)$,

$$\frac{\partial B}{\partial H} = B_{\text{SAT}} \cdot \frac{2}{\pi} \cdot \frac{1}{\alpha} \cdot \frac{1}{1 + \dfrac{H^2}{\alpha^2}} \tag{2.39}$$

This equation is maximized when $H(r,t) = 0$, and this is by definition the initial magnetic permeability, $\mu_0\, \mu_r$, at the zero field condition, where μ_0 is the permeability of free space ($= 4\pi \times 10^{-7}\ \text{H} \cdot \text{m}^{-1}$) and μ_r is the relative permeability. Therefore,

$$\mu_r = \frac{2\, B_{\text{SAT}}}{\pi\, \alpha\, \mu_0} \tag{2.40}$$

Here, a reciprocal of α is officially proved to be a scaled initial permeability. The nonlinear magnetizing inductance arises from the fact that μ_r is an estimate of the initial permeability, and will not be maintained constant throughout the period.

2.6.4 Numerical Simulation

The numerical circuit simulator is implemented in MATLAB. The multi-dimensional Jacobian-free numerical solvers employed in the simulator are based on the Newton and the Generalized Conjugate Residual (GCR) [36, 37] methods, and find solutions of equations generated from the Maxwell core model, the core loss model, and the load model. The Newton method is used to find a zero crossing, i.e., solution, of a function, and the GCR is used to rapidly construct the search direction for the next Newton. At each iteration of the GCR, a vector orthogonal to those obtained from earlier iterations is added to the search direction. The progression in time, i.e., estimating the next time point for transient simulation, is modeled by trapezoidal integration.

Consider the resistive load case as an example. Rearranging (2.36) to get the first time derivative of $I_{\mathrm{LOAD}}(t)$ yields:

$$\begin{aligned}\frac{\partial I_{\mathrm{LOAD}}(t)}{\partial t} = \frac{\omega\, I_{\mathrm{P}}}{\gamma\, N}\cos(\omega t) - \frac{2\pi^2\alpha}{\gamma\, N^2\, h\, B_{\mathrm{SAT}}}\times \\ \frac{(R_{\mathrm{WIRE}} + R_{\mathrm{LOAD}})\cdot I_{\mathrm{LOAD}}(t)}{\ln\left(\dfrac{r_{\mathrm{OD}}^2 + \dfrac{\left[I_{\mathrm{P}}\sin(\omega t) - \gamma\, N\, I_{\mathrm{LOAD}}(t)\right]^2}{4\,\pi^2\alpha^2}}{r_{\mathrm{ID}}^2 + \dfrac{\left[I_{\mathrm{P}}\sin(\omega t) - \gamma\, N\, I_{\mathrm{LOAD}}(t)\right]^2}{4\,\pi^2\alpha^2}}\right)}\end{aligned} \tag{2.41}$$

Trapezoidal integration applied to $I_{\mathrm{LOAD}}(t)$ results in a discrete-time approximation:

$$\begin{aligned}&I_{\mathrm{LOAD}}(t_{n+1}) - I_{\mathrm{LOAD}}(t_n) \\ &\quad = \frac{1}{2}\Delta t \left(\left.\frac{\partial I_{\mathrm{LOAD}}(t)}{\partial t}\right|_{t=t_n} + \left.\frac{\partial I_{\mathrm{LOAD}}(t)}{\partial t}\right|_{t=t_{n+1}} \right)\end{aligned} \tag{2.42}$$

The equations (2.41) and (2.42) can be combined to move all the terms to the same side, which is equated to G. The equation $G = 0$ can be numerically solved using Newton with GCR for each time point. The first order differential equation requires one initial condition. Since the primary side is a sine wave, its value is zero at $t = 0$, and if the system is fully de-energized, $I_{\mathrm{LOAD}}(t)$ is zero as well. This initial condition produces a turn-on transient. After obtaining $I_{\mathrm{LOAD}}(t)$ for the entire cycle, I_{PEAK}, B_{PEAK}, and $V_{\mathrm{CORE,RMS}}$ are calculated. Using the same steps discussed in (2.32) and (2.33), P_{LOSS} and R_{CORE} can be obtained.

Two challenges for predicting power harvest remain: first, we seek power harvesting capability in steady-state, not during the initial transient; second, the convergence of the values for R_{CORE} and $V_{\mathrm{CORE,RMS}}$ is not yet characterized. In order to guarantee convergence to a steady-state solution in the presence of core loss, a second solver

```
Until($|F_{\text{SHOOTING}}| \leq$ Tolerance for $F$) {
  for each time point $\tau$ {
    Until($|G| \leq$ Tolerance for $G$) {
      Newton_GCR on G; (residual update)
    }
    $I_{\text{LOAD}}(\tau)$ = Solved Value;
  }
  $V_{\text{CORE,RMS}}$ Calculation;
  $P_{\text{LOSS}}$ Calculation;
  Newton_GCR on F; (residual update)
}
```

Figure 2-14: Pseudo Code of the Numerical Simulator

with a shooting function F_{SHOOTING} is required; this solver returns a 2×1 vector. The added solver also employs Newton with GCR, and wraps around the time range solver. The first element of the shooting function calculates the difference of two function values that are separated by a period in time, and returns zero if the solution is in steady-state:

$$F_{\text{SHOOTING},[1]} = I_{\text{LOAD}}(t_0 + T) - I_{\text{LOAD}}(t_0) \tag{2.43}$$

The second element of F_{SHOOTING} calculates the difference between two core loss resistance values in two consecutive iterations. If the difference converges to zero, R_{CORE}, P_{LOSS}, and $V_{\text{CORE,RMS}}$ are in a correct relationship. $R_{\text{CORE-PREV}}$ denotes the core loss resistance calculated in the previous iteration.

$$F_{\text{SHOOTING},[2]} = \frac{V^2_{\text{CORE,RMS}}}{P_{\text{LOSS}}} - R_{\text{CORE-PREV}} \tag{2.44}$$

If $|F_{\text{SHOOTING}}|$ is within the tolerance, the system is in the steady-state, and the core loss is correctly estimated. Figure 2-14 describes the pseudo-code including two solvers in separate layers and the shooting function.

Table 2.1: Magnetic Core Parameters

B_{SAT}	1.190 T
$P_{\text{LOSS-MAX}}$	0.125 mW
α	2.2

2.7 Experiments and Simulation Results

The estimated parameters, B_{SAT}, $P_{\text{LOSS-MAX}}$, and α, of the amorphous nanocrystalline core, VAC VITROPERM 500F W380, are listed in Table 2.1. As discussed in Chapter. 2.3, this core exhibits an extremely low hysteresis loss on the order of µW level and high B_{SAT}. The value of 2.2 for the parameter α corresponds to an initial permeability of approximately 274,027, which is considerably higher than most of the commercially available cores.

In Fig. 3-3, four plots are presented to summarize the close agreement between the experiments and the simulation results of the Maxwell modeling method described in the previous sections.

The first two plots of Fig. 3-3 illustrate the resistive load case in two different combinations of I_{P} and N. The remaining plots illustrate the constant voltage load case in two different combinations of I_{P} and N. The experimental results (red) and simulation results (blue) are almost identical in all configurations, indicating that the modeling accurately represents physical responses of the core under various settings of load type, I_{P}, and N. The experiments also demonstrate that every power delivery peak happens after the linear —unsaturated— region. Therefore, power harvest is maximized when the core is loaded sufficiently to cause saturation. In other words, controlling the level and timing of saturation of the core is crucial to extract the maximum magnetic energy from a sinusoidal primary current.

The results of the wire and core loss simulation are also overlaid in the plots. The ratio of wire loss to the power harvest can be substantial in nonsaturated operation where the transformer current is delivered into the load for the entire cycle. However, the amount of power harvest in the nonsaturated region is sub-optimal, rendering

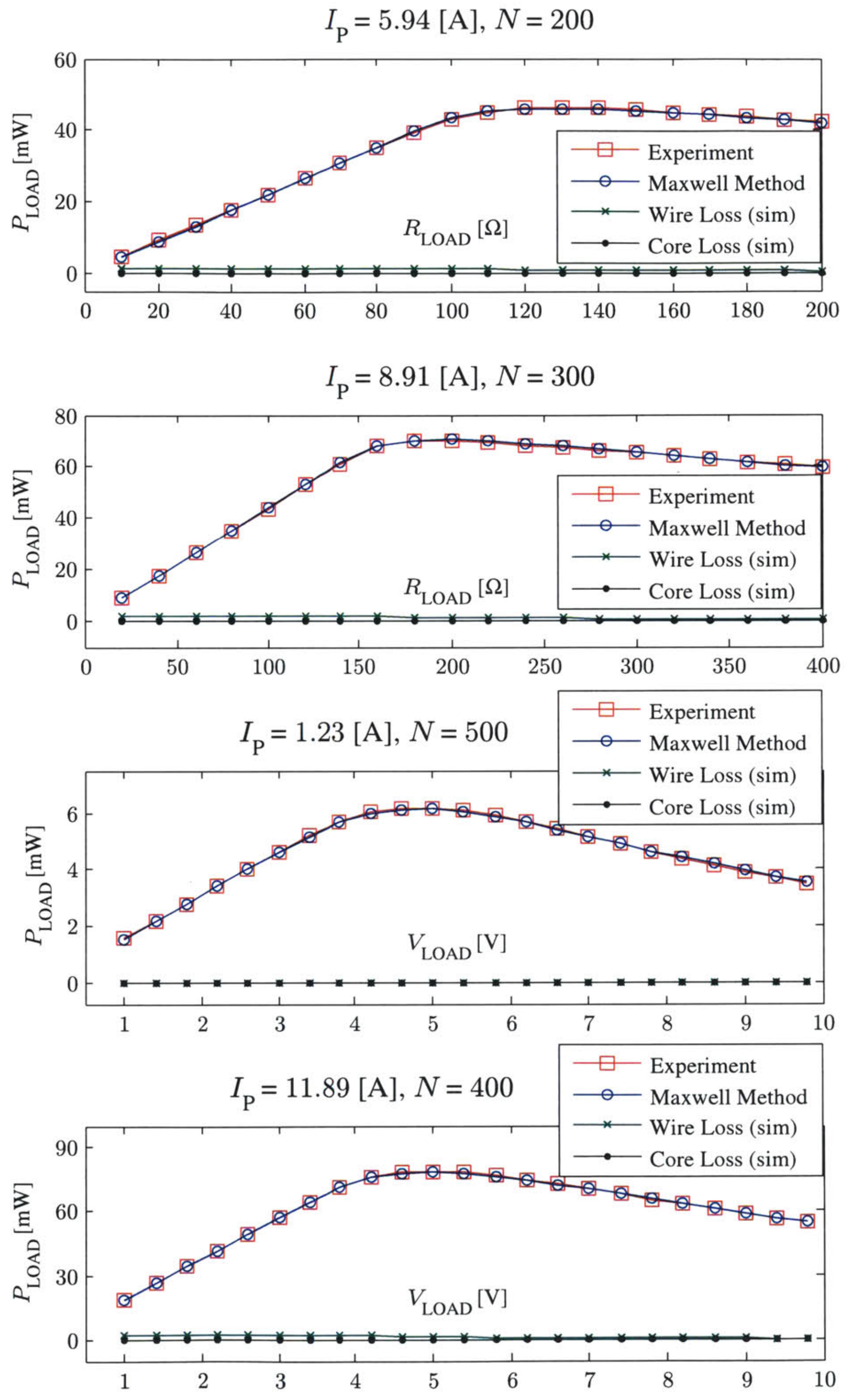


Figure 2-15: Experimental verification of the Maxwell modeling method

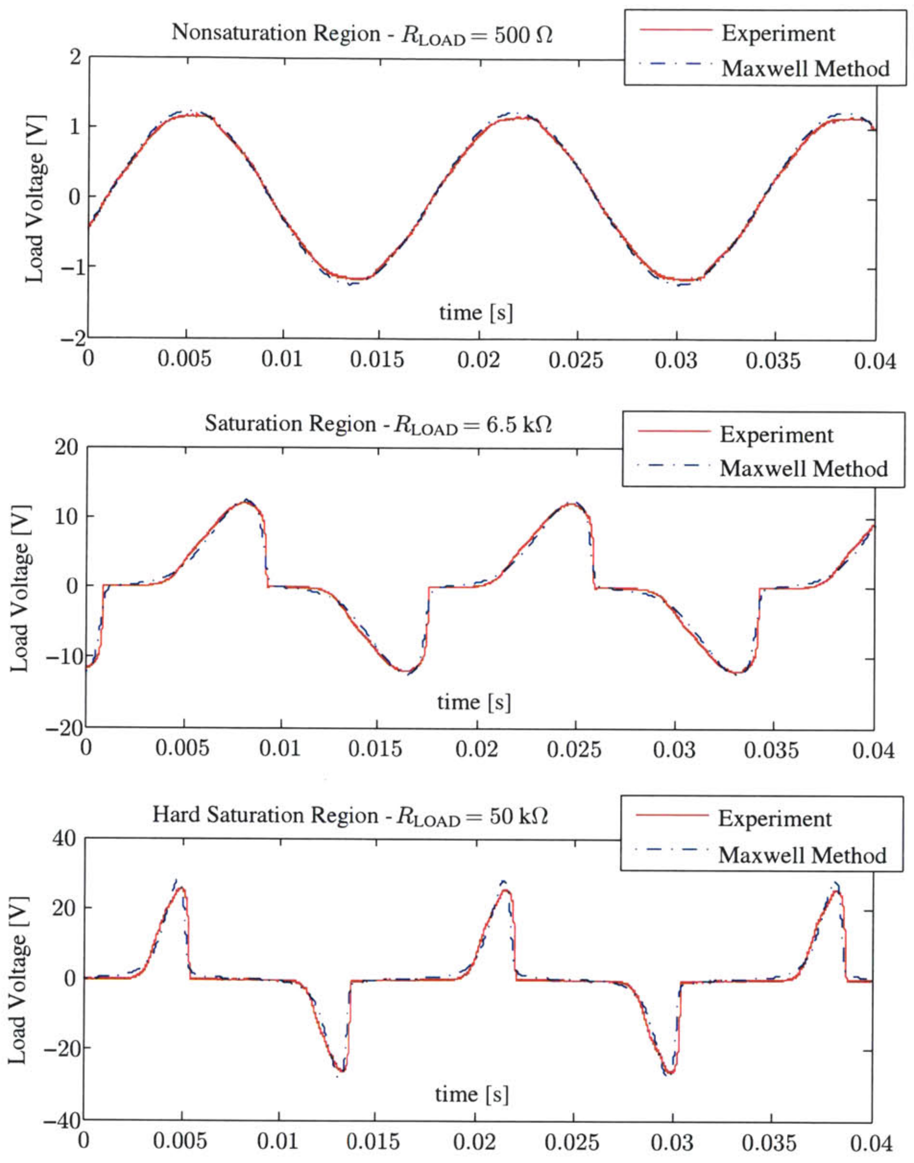


Figure 2-16: Load voltage waveforms in experiments ($I_{\mathrm{P}} = 1.23\,\mathrm{A}$, $N = 500$)

the power harvest lower than what could have been achieved by allowing the core to saturate. Therefore, by taking the maximum power harvest point, the ratio of wire loss to the power harvest can be greatly reduced. Furthermore, at the maximum power harvest point, the core goes into saturation. The transfer window is shortened, and the RMS current is lowered, further reducing wire loss. Therefore, wire loss can be considered negligible when operating at the maximum power harvest point. For example, at the maximum power harvest point in the fourth plot of Fig. 3-3, the power harvest is 78.62 mW, and the wire loss is 1.38 mW (1.76%). The core loss is negligible because the amorphous nanocrystalline core that is employed in this thesis has very low hysteresis loss at line frequency ($P_{\text{LOSS-MAX}} = 0.125\,\text{mW}$).

Figure 2-16 presents time domain comparisons between the experiments and the simulations of Maxwell method where the waveforms of the load voltage in the resistive load case are depicted for three different saturation and load conditions.

2.8 Chapter Summary

This chapter demonstrates that quick approximations can identify the transfer window for power transfer in a magnetically saturating current-driven core. This quick analysis can be used for reasonably accurate decisions on core sizing and load targets for the magnetic energy harvester. A more accurate core model based on Maxwell's equations has been presented for precise numerical design simulation, where the *B*-*H* curve of the core material is modeled and nonideal losses are considered. Parameter estimation techniques for the magnetic core are demonstrated, and the numerical solver using Newton with GCR is presented. The solver has two layers: the first being the time range solver; and the second being the steady-state and core loss consistency solver. Experimental results demonstrate that the proposed modeling method is accurate both in predictions of amount of power harvest and in time domain waveforms, across various operating points as well as different loading conditions.

Operating the harvester at typical utility line frequency, the winding loss from the proximity effect and the skin effect can be ignored, and the wire loss is solely caused by the DC resistance, which also can be ignored if the core is operating near the maximum power harvest point. Using an amorphous nanocrystalline core with low hystersis loss, the core loss also can be omitted from the analysis. Finally, an exciting and counter-intuitive point can be drawn from the results: the power harvest is maximized when the core is placed in saturation. Therefore, controlling the level and the timing of saturation of the core is crucial to achieve the maximum power harvest from an inductively-coupled current-driven magnetic core.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Power Electronic Circuits

The core in the magnetic energy harvester is energized by a primary winding that is effectively acting as a current source. The insertion impedance of the harvester transformer is generally extremely low, therefore, the load under observation, for example, a motor, sets or commands the current flowing through the primary of the harvester. Saturation of the harvester core is determined by the interaction of this primary current with the time and level of voltage applied to the secondary winding of the harvester.

When a magnetic core is saturated, the secondary side voltage across the core becomes essentially zero, because the time derivative of the magnetic flux density in the core (B) is near zero in saturation. The magnetizing inductance (referred to the secondary side) is therefore extremely low during this period, and little power is coupled from the primary winding. With a high permeability core, when the core is not saturated, the impedance of the magnetizing inductance is relatively high, and significant power transfer can occur to the load in the secondary. The "transfer window" is introduced in Chapter 2 to describe this distinct region when significant power flow occurs through the core across the transformer. In this chapter, circuit design methods are developed based on this concept of the transfer window in order to permit control and maximization of harvested energy.

A more practical model for circuit simulation is first developed based on the Maxwell model from Chapter 2. Two methods to enhance energy extraction from the electromagnetic coupling are presented. The first method is the flux shaping capacitor (FSC). The second is the transfer window alignment method (TWA). The control methods and regulation circuits for the TWA method are also explored. Finally, the limitation of the first order flux equality that has been used for all the analyses is revisited for the case of a very high primary side current.

3.1 Circuit Model

The Maxwell model discussed in Chapter 2 is complete for simulating a circuit if all the components are described in a numerical solver. The model provides little intuitive insight, and requires a custom-designed numerical solver that quickly becomes undesirable when combined with power electronic circuits on the secondary that consist of nonlinear switching devices. Here, a model for a magnetically saturating core that can be directly employed in SPICE is developed. This model will be used to design practical circuits and validate experimental results for the electromagnetic energy harvester.

3.1.1 Circuit Model

The complexity in the V-I characteristic of a magnetic core in the presence of magnetic saturation, (2.25), comes from the time differentiation and spatial integration of B with the arctan function. Before the application of these mathematical operations, H and B are simulated using the finite element method. The H and B inside the core, calculated by the finite element method magnetic (FEMM), are presented in Fig. 3-1. Between the inner and the outer radius of the core, H does not seem to change much, and B is almost constant across the range. Based on this observation, a dimensional approximation is made: the flux density, B, is considered independent of the spatial location in radial axis; and the variable, r, is replaced with the median value and the integration is eliminated. The time differentiation is not taken yet for a better understanding with a simpler expression. The FEMM simulation assumes B_{SAT} of 1.20 T, μ_r of 198,944, I_{P} of 1 A, and 60 Hz for the core and the environments.

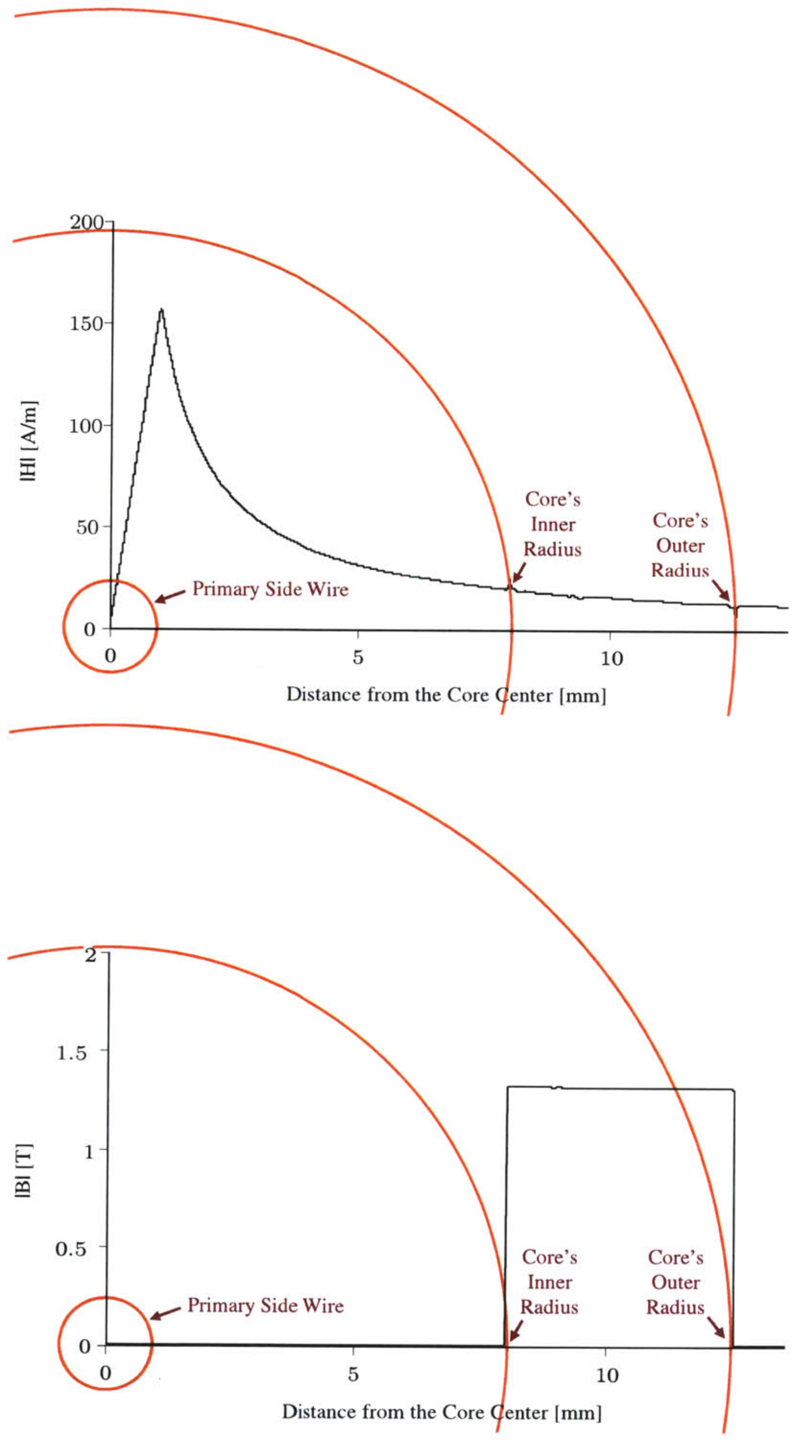


Figure 3-1: FEMM Results of the H and B Fields inside the Core

Denoting a number of windings on the secondary side, the saturation flux density, outer radius, inner radius, and height of a toroidal core by N, A_{CORE}, B_{SAT}, r_{OD},

r_{ID}, and h, respectively, the flux linkage in the toroidal core of the magnetic energy harvester can be described as:

$$\Lambda(t) = N \cdot h \cdot \int_{r_{\mathrm{ID}}}^{r_{\mathrm{OD}}} B(r,t)\ \mathrm{d}r \tag{3.1}$$

Here, the spatial integration is approximated by modeling the core flux density at every radius as equal to the core flux density at $r_{\mathrm{mid}} = (r_{\mathrm{OD}} + r_{\mathrm{ID}})/2$. Then, using the same saturation function as in (2.22), the expression for the flux can be approximated as:

$$\begin{aligned}\Lambda(t) \approx\ & B_{\mathrm{SAT}}\,(r_{\mathrm{OD}} - r_{\mathrm{ID}})\,h\,N \\ & \times \frac{2}{\pi}\arctan\left(\frac{I_{\mathrm{P}}\sin(\omega\,t) - N \cdot I_{\mathrm{S}}(t)}{2\,\pi\,r_{\mathrm{mid}}\,\alpha}\right)\end{aligned} \tag{3.2}$$

This equation can be written compactly as:

$$\Lambda(t) = \Lambda_{\mathrm{MAX}} \cdot \frac{2}{\pi}\arctan\left(\frac{N}{\beta}\,I_{\mu}(t)\right) \tag{3.3}$$

using the parameters defined below:

$$\Lambda_{\mathrm{MAX}} = B_{\mathrm{SAT}}\,(r_{\mathrm{OD}} - r_{\mathrm{ID}})\,h\,N = B_{\mathrm{SAT}}\,A_{\mathrm{CORE}}\,N \tag{3.4}$$

$$\beta = 2\,\pi\,r_{\mathrm{mid}}\,\alpha = l_{\mathrm{FLUX}}\,\alpha \tag{3.5}$$

$$I_{\mu}(t) = \frac{I_{\mathrm{P}}}{N}\sin(\omega\,t) - I_{\mathrm{S}}(t) \tag{3.6}$$

By interpreting (3.6) as a KCL constraint, a node with three branches can be virtually created for the representation of the magnetic core. The first branch is $I_{\mathrm{P}}/N \cdot \sin(\omega\,t)$, supplying the current into the node. This can be easily represented by an ideal current transformer with the winding ratio of $1 : N$ and primary current of $I_{\mathrm{P}}\sin(\omega\,t)$. The second branch is $I_{\mathrm{S}}(t)$, taking the current out of the node. It can be treated as the load current supplied from the core. The third branch is the

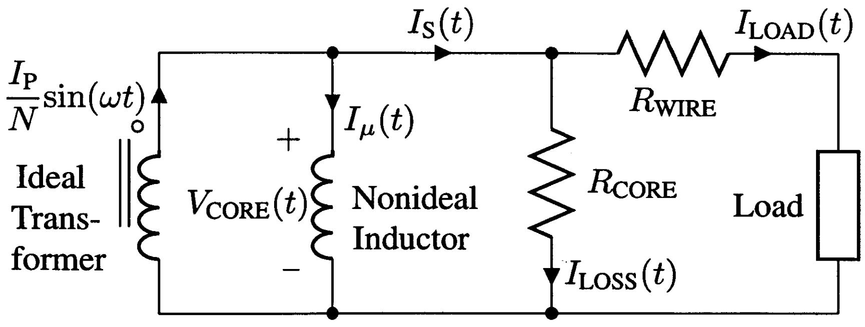


Figure 3-2: Circuit Model

current remainder, $I_\mu(t)$, as defined, and the equation (3.3) describes the relationship between current and flux of the third branch. The relationship between voltage and current of this branch can be obtained by time differentiation of the flux:

$$V_{CORE}(t) = \frac{\partial}{\partial t}\left[\Lambda_{MAX} \cdot \frac{2}{\pi}\arctan\left(\frac{N}{\beta}\, I_\mu(t)\right)\right] \tag{3.7}$$

Considering the derivative of $\arctan(x)$ is $1/(1+x^2)$, differentiating the flux equation (3.3) with $I_\mu(t)$ generates a positive value. Since $\mathrm{d}\Lambda/\mathrm{d}I$ is positive, the third branch can be understood as an inductor. The inductance is nonlinear and time-varying in an application with an AC current in the primary winding. This is consistent with intuition about the core, as, for an unsaturated core with constant permeability, the core is expected to appear from winding terminals as an inductance.

Summarizing branch descriptions, the circuit configuration can be presented as in Fig. 3-2. The circuit model builds from an ideal transformer and adds a nonideal inductor in parallel with it, while the Maxwell method has a lumped nonideal transformer that includes the effects of both. Using either a nonideal inductor element that supports a flux description, or a behavioral voltage element, the equation (3.7) can be directly used in SPICE with ease. The circuit model is attractive because it is easier to mix with power electronic components such as MOSFETs and diodes in SPICE. A concern remains in determining core loss, which may affect the overall performance of the harvester. Though core loss is negligible with the amorphous nanocrystalline core, the following is presented for a complete analysis in case of a

Table 3.1: Detailed Magnetic Core Parameters

B_{SAT}	1.190 T
$P_{\mathrm{LOSS-MAX}}$	0.125 mW
α	2.2
β	0.142
Outer Radius (r_{OD})	12.25 mm
Inner Radius (r_{ID})	8.25 mm
Height (h)	9 mm
Flux Area (A_{CORE})	3.6×10^{-5} m^2
Flux Length (l_{FLUX})	6.44×10^{-2} m

lossy core. A SPICE simulation can track the voltage across and current through the nonideal magnetizing inductance and use these waveforms to calculate the area under a B-H curve as a scaled ratio of the maximum loss presented by the full hysteresis curve for the materials as described in Chapter 2.6.2:

$$P_{\mathrm{LOSS}} = P_{\mathrm{LOSS-MAX}} \cdot \frac{I_{\mathrm{PEAK}}}{I_{\mathrm{SAT}}} \cdot \frac{B_{\mathrm{PEAK}}}{B_{\mathrm{SAT}}} \tag{3.8}$$

A circuit simulator can therefore also accurately model core loss by updating the value of R_{CORE} with a numerical estimate. Additional loss mechanisms like conduction loss can also be modeled, e.g., by R_{WIRE}, as shown in Fig. 3-2.

In the following sections, the accuracy of the circuit model is verified, and circuit design techniques based on this circuit model are introduced to enhance power harvest of an electromagnetic energy harvester.

3.1.2 Model Accuracy

The accuracy of this circuit model can be verified by comparison to experimental data and to a full Maxwell model of the core described in the previous chapter. Assuming the same core, VAC VITROPERM 500F W380, the estimated parameter values, B_{SAT}, $P_{\mathrm{LOSS-MAX}}$, α, β, and dimensional properties of the core are listed in Table 3.1.

In Fig. 3-3, four plots are presented to compare the accuracy of the circuit model to the Maxwell model and the experiments. Model accuracy is considered for two load types. The top two plots represent cases with a resistive load, and the bottom two plots represent cases with a constant voltage load. For the resistive load cases, a rectifier is not required to measure the power delivered to the load. Direct measurements of the RMS voltage across the load resistor were used to quantify the power delivered. For the constant voltage load cases, a rectifier is required to measure the power harvest. To prevent or minimize any changes to the core flux (and power loss calculation), an active rectifier with 95% utilization of active devices is used. Each diode in a conventional full-bridge diode rectifier is replaced with a Schottky diode (PMEG2010ER [38]) and a FET (PMV30UN [39] or DMG3415U [40]) in parallel to mimic an ideal rectifier. The next section discusses the experimental configuration in greater detail.

For the tests, different I_P and N configurations are used. As illustrated in the figure, regardless of the load type, I_P, and N, the circuit model is as accurate as the Maxwell model, closely tracking the experiments. Approximations made in the modeling, e.g., uniformity of the magnetic flux density throughout the core and approximating the saturation characteristic with an arctan function provide excellent accuracy for designing an electromagnetic energy harvester.

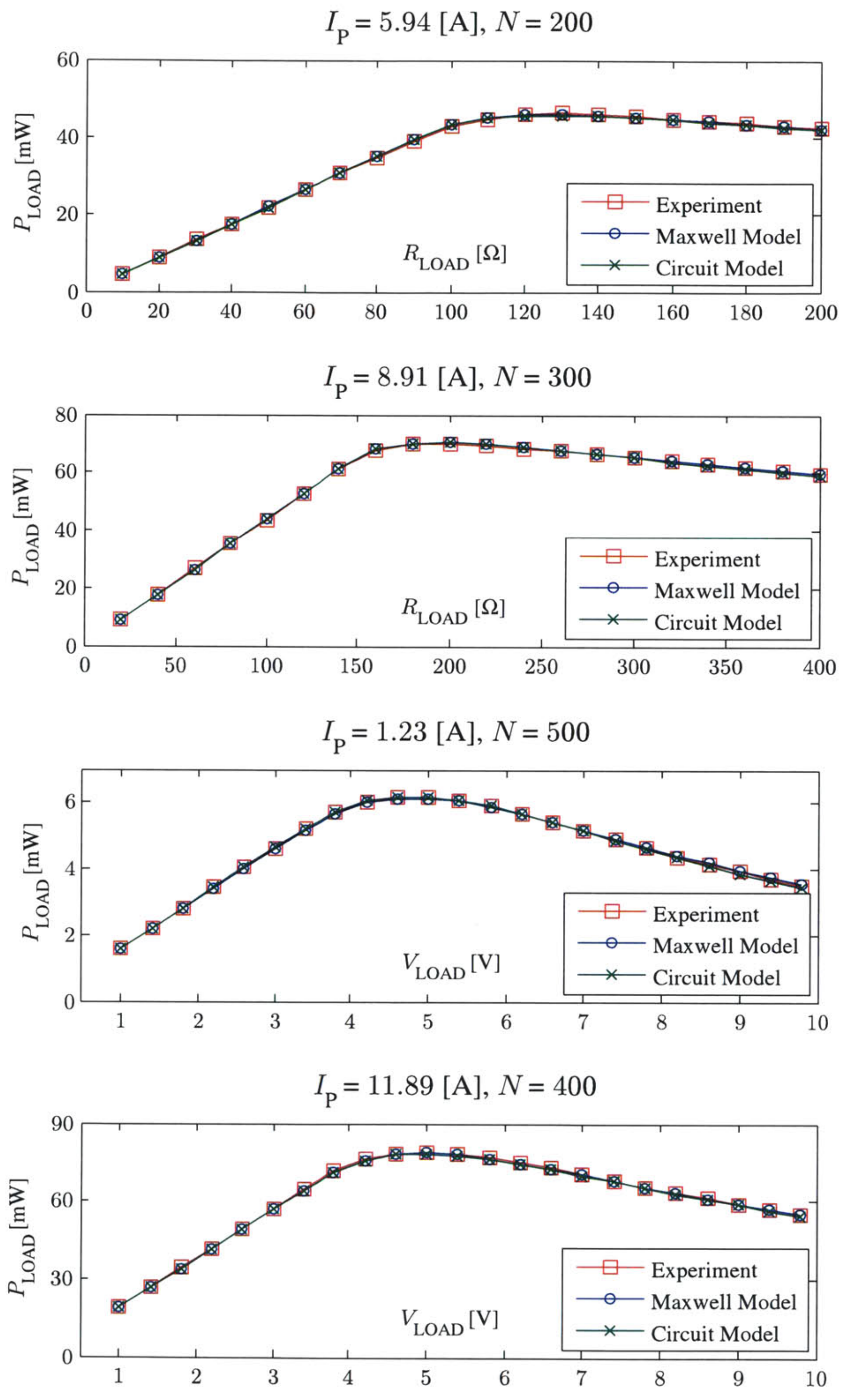


Figure 3-3: Experimental Verification of the Circuit Model

3.2 Maximizing the Power Harvest

The duration and level of the voltage applied to the secondary winding by the harvester circuitry determines when the flux in the core will build to saturation levels. At a high level, maximizing the power harvest boils down to the problem of keeping the core out of saturation for a "best" period of time. This window or period of time would ideally allow the fixed secondary current to flow for the longest period of time with the highest secondary voltage. This time period when current flows through the transformer action (while the core is not fully saturated) is the transfer window. Of course, the core could always be operated so that it never saturates. Figure 3-3 demonstrates that the unsaturated core, which is in a strictly linear region in lower R_{LOAD} or V_{LOAD}, always corresponds to a less-than-maximal power harvest. Permitting the core to eventually saturate over the course of a primary waveform cycle provides maximum power harvest. The trick is to time this saturation to maximize the energy extraction.

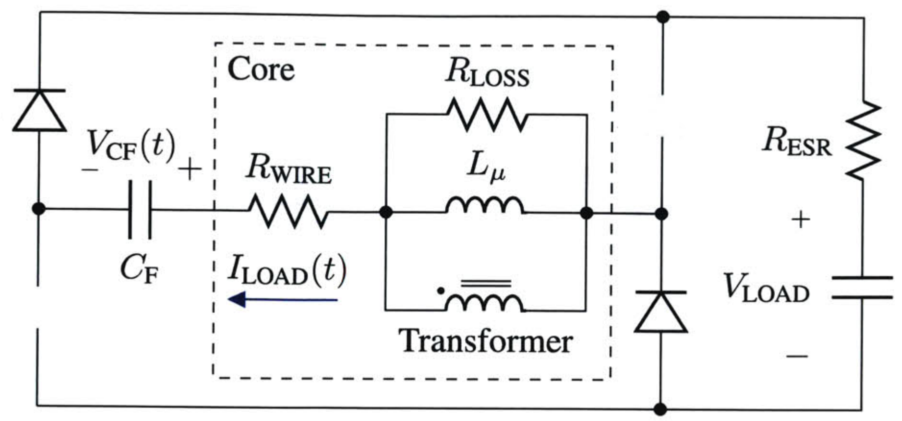
Two different circuit techniques can be used to maximize the extent and precise timing of the power transfer window. The first involves placing a capacitance in series with the core before a rectifier stage. The capacitor shapes the flux developed across the core, and lengthens the transfer window. The second is to connect the load to the core such that the center of the transfer window is aligned with the peak of the transformer current. At other times, the load is disconnected from the core, and the core is externally shorted to prevent itself from accumulating flux. The two techniques cannot be used at the same time; however each technique is capable of substantially enhancing the power harvest compared to an unmodified current transformer. Also discussed is to use a rectifier with active gate control instead of passive rectifiers. If the gate control is available at relatively low cost, e.g., from a microcontroller already serving in the sensor system, active rectification can be used with either one of the first two techniques.

3.2.1 Flux-shaping Capacitor (FSC)

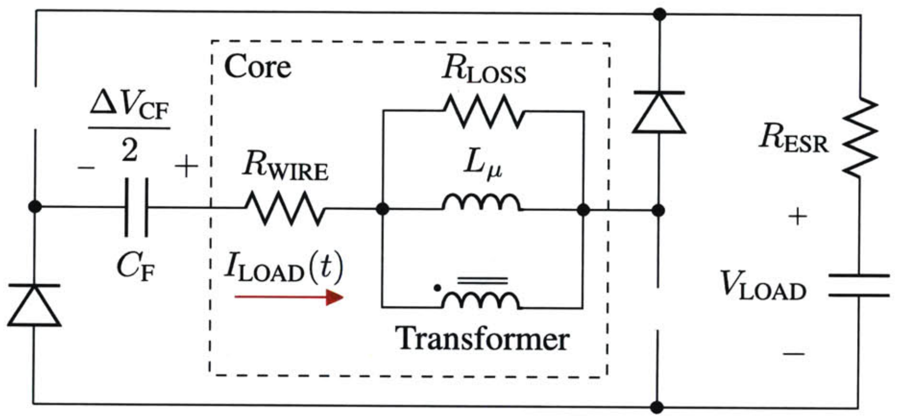
Since the magnetizing inductor is not an ideal inductor, the load receives power only when the core is not saturated. This means that the transfer window is active when the magnetizing inductance appears as a large shunt impedance across the secondary. To lengthen the transfer window, a way to manipulate the volt-seconds applied to this inductor is sought. The net voltage applied to the magnetizing inductance can be reduced by placing a capacitor C_{F} in series with the load as illustrated in Fig. 3-4. This capacitor charges during any particular half cycle. During the next half-cycle, the capacitor is charged with a voltage polarity opposite to that of the constant voltage load. By reducing this net voltage applied to the magnetizing inductance, flux accumulation becomes slower, lengthening the power transfer window and increasing the power delivered to the load.

When C_{F} is added to the circuitry, it is no longer possible to obtain a simple expression for the maximum power harvest point, as presented in (2.19). Instead of finding a numerical solution through a simulator, it is practical to develop an insight by analyzing the core behavior in a qualitative way. In Fig. 3-4 (a), where the current flows to the left, the current charges up the flux-shaping capacitor, increasing $V_{\mathrm{CF}}(t)$ until the core is saturated. When the core is saturated, the voltage across the magnetizing inductance becomes zero, preventing the transformer current from going into the load. At this instant, depending on the voltage across C_{F}, the diodes may either all turn-off or briefly switch to the alternate conduction path. If $V_{\mathrm{CF}}(t)$ is higher than V_{LOAD}, then the current path briefly switches and supplies current from C_{F} into the load until $V_{\mathrm{CF}}(t)$ is equal to V_{LOAD}, at which point the diodes turn off. This brief period continues power delivery to the load. If $V_{\mathrm{CF}}(t)$ is not higher than V_{LOAD}, the diodes are all disconnected, keeping $V_{\mathrm{CF}}(t)$ constant until the core recovers from the saturation and reverses the current direction. In the steady-state, net charge into the flux-shaping capacitor is zero. Therefore, both positive and negative voltage peaks will have the same magnitude of $\Delta V_{\mathrm{CF}}/2$.

Right after the reversal of the main current path, as illustrated in Fig. 3-4 (b), the



(a) Before the current path reversal



(b) Right after the current path reversal

Figure 3-4: The Circuit Example with a Flux-Shaping Capacitor

voltage across the core starts from $V_{\mathrm{LOAD}} - \Delta V_{\mathrm{CF}}/2$, whereas it is always V_{LOAD} if a flux-shaping capacitor is not used. For a sinusoidal primary current, $V_{\mathrm{CF}}(t)$ is a cosine wave, and, therefore, $V_{\mathrm{LOAD}} - V_{\mathrm{CF}}(t)$ becomes convex. Starting from a lower voltage and developing it in a convex manner due to the flux-shaping actions of the capacitor indicate that the transfer window is lengthened in the tail. Lengthening the transfer window by a small amount can result in a great boost in power harvest. Figure 3-5 illustrates a simulation example, where the transfer window is visibly lengthened by a flux-shaping capacitor. To illustrate the effect, Fig. 3-5 shows simulated results with an operating point of $I_{\mathrm{P}} = 50\,\mathrm{A_{RMS}}$, $N = 200$, $V_{\mathrm{LOAD}} = 18\,\mathrm{V}$, and $C_{\mathrm{F}} = 2.9\,\mu\mathrm{F}$. The power harvest is increased by 107.12% compared to the case with no flux-shaping

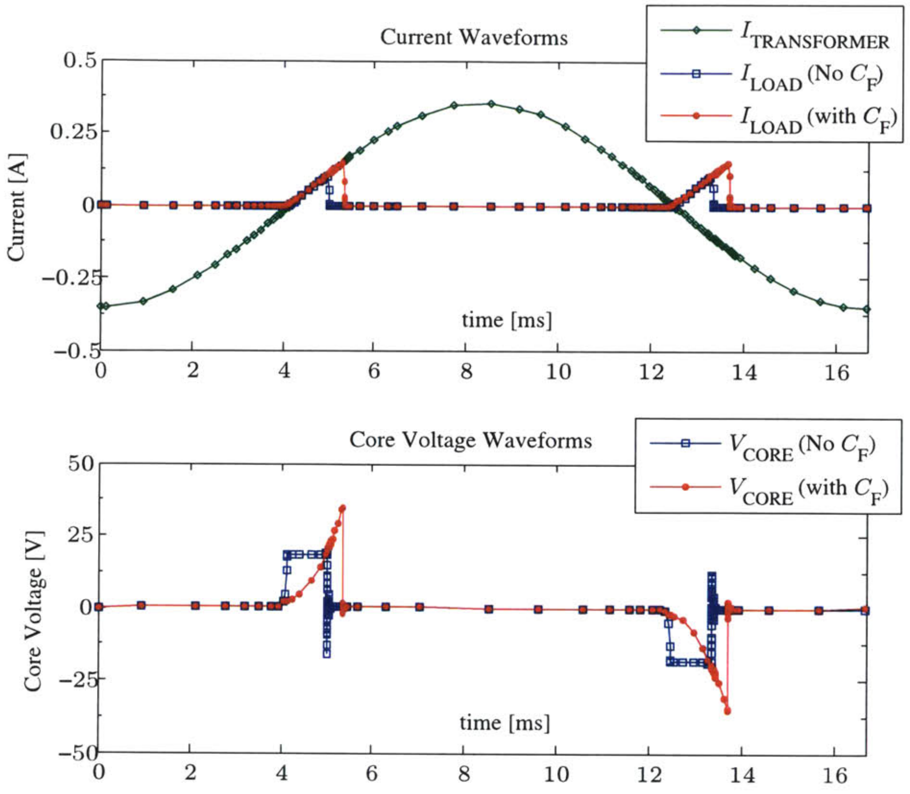


Figure 3-5: Flux-Shaping Capacitor Simulation Example in Hard Saturation ($I_P = 50\,\text{A}_{\text{RMS}}$, $N = 200$, $V_{\text{LOAD}} = 18\,\text{V}$, $C_F = 2.9\,\mu\text{F}$)

capacitor.

In a real design, the optimal V_{LOAD} and C_F must be found, subject to a given I_P and N configuration. Since it is not analytically possible to predict the peak, both parameters were swept to find the maximum power harvest as in Fig. 3-6. The lower plot represents the corresponding C_F that yields the maximum power harvest for each V_{LOAD}. In this example, $I_P = 6.27\,\text{A}_{\text{RMS}}$, $N = 200$, and a nonideal diode rectifier with a manufacturer provided SPICE model is used. The diode rectifier in the simulation consists of four Schottky diodes with an approximate forward voltage drop of 0.19 V in a mA range. The maximum harvestable power of 48.6 mW without the flux-shaping capacitor now reaches 66.6 mW, which is a net increase of 37.04%. However, there is a major issue in this method that if the RMS of the primary current continuously changes, for example, in case of periodic motor speed control, optimal V_{LOAD} and C_F for the maximum power harvest also continuously change. An example

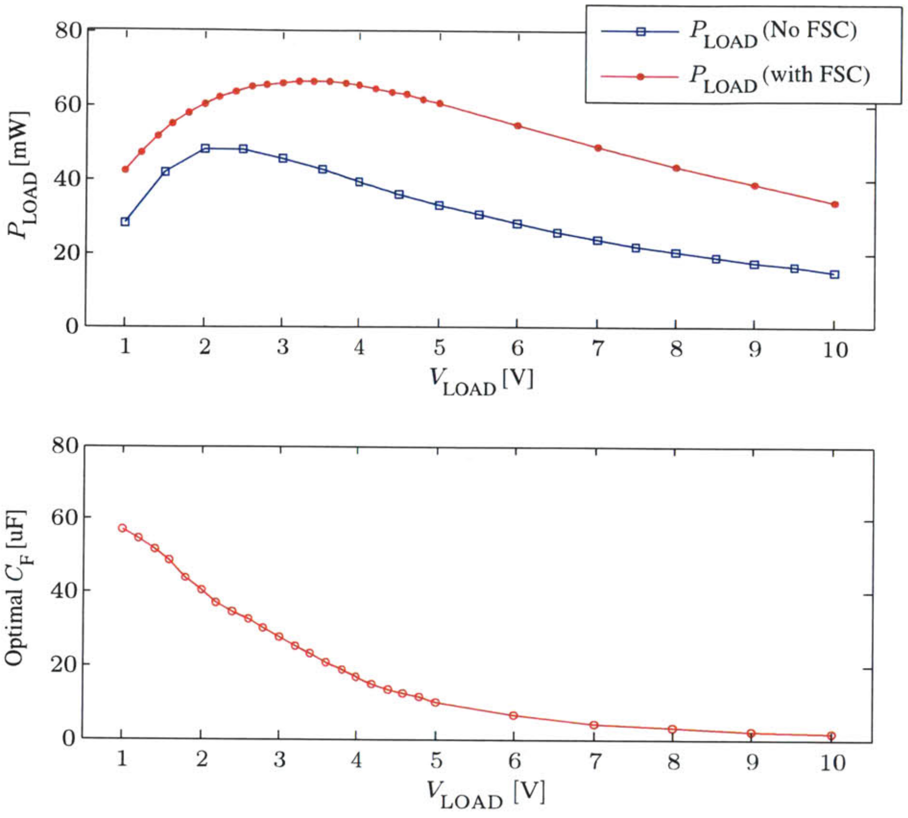


Figure 3-6: Simulation of the Flux-Shaping Capacitor Method ($I_{\mathrm{P}} = 6.27\,\mathrm{A_{RMS}}$, $N = 200$)

is illustrated in Fig. 3-7 with $N = 200$. Changing V_{LOAD} can be relatively easily accomplished by connecting a power converter with programmable duty cycle control as a load. However, two concerns associated with changing capacitance remain. First, the optimal capacitance spans a wide range as illustrated in Fig. 3-7. Second, an attempt to tune the capacitance, e.g., by selecting a capacitor from a bank of choices, requires a circuit that can effectively operate with the capacitor in a "floating" position in the circuit, complicating switch implementation.

If the primary current is extremely low such that the core never saturates during operation, adding a capacitor in series with the core, which maintains constant inductance in this case, can be viewed as a power factor corrector. This makes the waveform of the load current more rectangular, and results in higher average current. Due to this effect, the amount of power harvest is greatly enhanced even in the

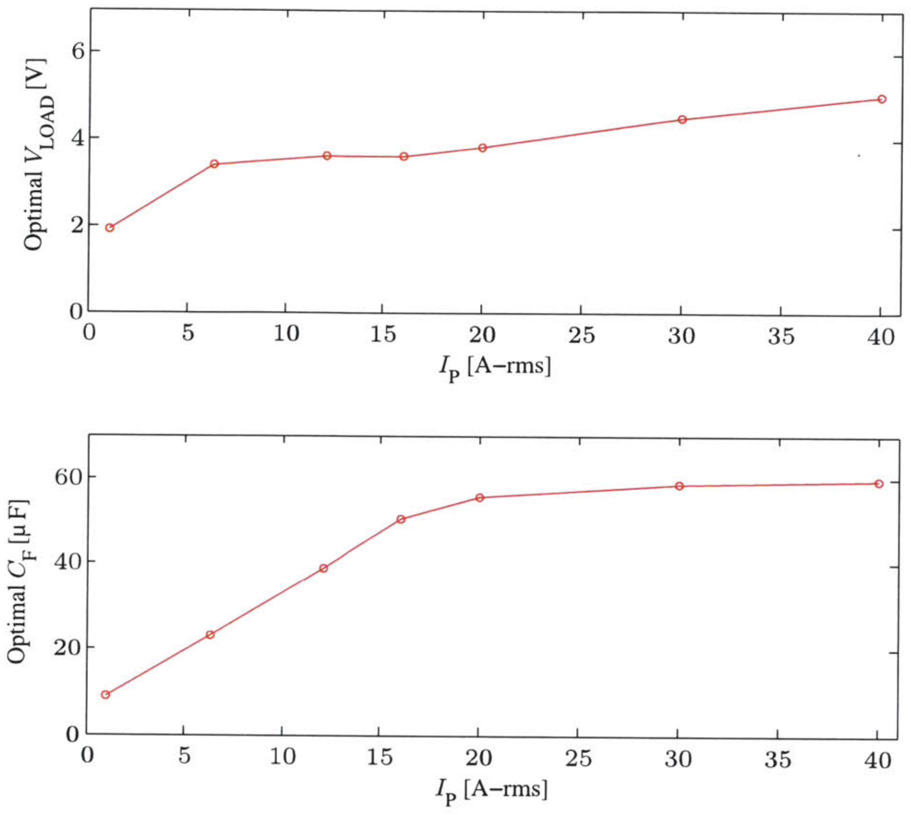


Figure 3-7: Simulation of Optimal V_{LOAD} and C_{F} vs. I_{P}, with $N = 200$

nonsaturated region with the flux-shaping capacitor, as shown near the initial linear region in Fig. 3-6. Since the load is a constant voltage source, the average power harvest is directly proportional to the average current, not the RMS current.

3.2.2 Transfer Window Alignment (TWA)

Alternatively, it is possible to actively control the connection between the magnetic core and the load, and manipulate the starting point of the transfer window relative to the zero crossing of the primary current. This manipulation is possible because the load voltage is independent of the primary current for the case of a constant voltage load. Assuming ideal rectifiers, the duration of the transfer window in the first order approximation is:

$$t_{\mathrm{SAT}} = \frac{2\, B_{\mathrm{SAT}}\, A_{\mathrm{CORE}}\, N}{V_{\mathrm{LOAD}}} \tag{3.9}$$

That is, the core will be saturated if V_{LOAD} is connected to the core for t_{SAT} seconds, regardless of when the transfer window begins relative to the zero crossing of the primary current. Given this understanding, the transfer window which permits the greatest energy harvest corresponds with a window of time when the average primary current is as large as possible. This is achieved when the middle of the transfer window is aligned with the peak of the transformer current, and the core is "shorted out", reducing the voltage on the secondary winding to zero, during times outside the transfer window in order to prevent the core from developing unnecessary flux. Without a detailed nonlinear description of the magnetic core, an estimate for the average power harvest using this transfer window alignment method is,

$$\begin{aligned} P_{\mathrm{LOAD,avg}} &= V_{\mathrm{LOAD}} \times \frac{2}{T} \int_{\frac{T}{4}-\frac{t_{\mathrm{SAT}}}{2}}^{\frac{T}{4}+\frac{t_{\mathrm{SAT}}}{2}} \frac{I_{\mathrm{P}}}{N} \sin(\omega\, t)\; \mathrm{d}t \\ &= P_{\mathrm{TWA}} \cdot \frac{\sin(J)}{J} \end{aligned} \tag{3.10}$$

where

$$\begin{aligned} P_{\mathrm{TWA}} &= \frac{2\, I_{\mathrm{P}}\, \omega\, B_{\mathrm{SAT}}\, A_{\mathrm{CORE}}}{\pi} \\ J &= \omega\, \frac{B_{\mathrm{SAT}}\, A_{\mathrm{CORE}}\, N}{V_{\mathrm{LOAD}}} = \omega\, \frac{t_{\mathrm{SAT}}}{2} \end{aligned} \tag{3.11}$$

Since $\sin(J)/J$ has an asymptote of 1 as $J \to 0$, P_{TWA} is the maximum value of $P_{\mathrm{LOAD,avg}}$. This indicates that J must be minimized to maximize the power harvest.

However, J cannot be arbitrarily small, as t_{SAT} would become proportionally small as well. As t_{SAT} becomes smaller, the portion of the cycle that is mainly described by (3.9) —the first order approximation— becomes smaller. In this case, the power harvest deviates from the first order predictions, and in fact becomes smaller. Non-ideal effects cause this reduction in power. For example, the magnetizing inductance, which has been already reset once the primary current became sufficiently low in the previous cycle, carries a small remnant current from the previous cycle in the opposite direction to the load current at the beginning of the transfer window. This results in a slight boost in the load current. On the other hand, before full saturation, the magnetizing inductance gradually drops to zero, causing a smoother roll-off of the load current at the end of the transfer window, resulting in a lower load current than the transformer current (these two non-ideal effects can be seen in Fig. 3-9). These two deviations are not perfectly balanced —the loss is higher—, and reduce the power harvest from the first order estimate. Furthermore, with smaller t_{SAT}, a finer time resolution is required for generation of control signals, and at a certain limit, the gate control is no longer completely accurate because of the fixed timing margins for the edge transitions of the TWA switches to avoid a short-circuit path from the load to the ground. A practical rule of thumb for design is to select $t_{\mathrm{SAT}} \geq T/10$, in which case $J \geq 0.314$, based on our experimentation with high permeability cores at utility line frequencies.

Given a core material with known magnetic properties and size, there are essentially two parameters available to the designer, N and V_{LOAD}, for minimizing J. Decreasing N increases the secondary side current and amplifies losses in the switches and diodes. The voltage V_{LOAD} cannot be arbitrarily high due to the voltage ratings of the switching devices in the rectifier or the voltage ratings of the supercapacitors. Raising V_{LOAD} may also incur unnecessary losses in the digital circuits that would typically form the sensor and communication load if they are directly powered by V_{LOAD}. Therefore, N and V_{LOAD} must be carefully chosen to minimize J without incurring unnecessary losses in the rest of the harvester circuit. One aspect that can aid to loosen the design constraint is that, since minimizing J has a diminishing return on

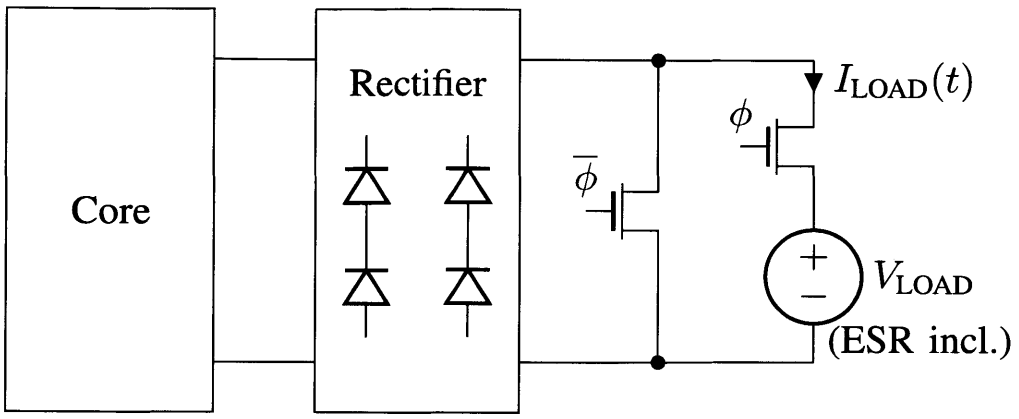


Figure 3-8: Conceptual Circuit for the Transfer Window Alignment Method

the amount of power harvest, the value of J can actually settle at a more reasonable level. It may be tolerable to slightly increase J in order to optimize overall system losses. For example, a circuit designer can pick $J = 0.6$, providing approximately 94% of the maximally achievable power harvest, P_{TWA}, while reducing the control power and the switching losses.

The conceptual TWA method, as shown in Fig. 3-8, requires two additional switches after the rectifier, and does not require any on-the-fly component change as was needed for the flux-shaping capacitor method. In Fig. 3-8, ϕ indicates a phase where the core is connected to the load, and $\overline{\phi}$ indicates the opposite phase. The simulation, presented in Fig. 3-9, clearly illustrates the shifted transfer window. Note both plots in Fig. 3-9 are generated with the same time reference. Using the same $I_{\mathrm{P}} = 6.27\mathrm{A}_{\mathrm{RMS}}$ and $N = 200$, Fig. 3-10 demonstrates the enhancement of power harvest with the TWA method. The maximum power harvest is much higher with TWA. For Fig. 3-10, a diode full-bridge rectifier based on the manufacturer-provided SPICE model of the same Schottky diode mentioned in Section 3.1.2 is used. Effects of nonideal diodes are discussed in the following section.

The TWA method drives the core into saturation more definitively than the FSC method. As J is minimized for enhancing the power harvest, t_{SAT} becomes shorter, which means the core goes deeper into the saturation regime.

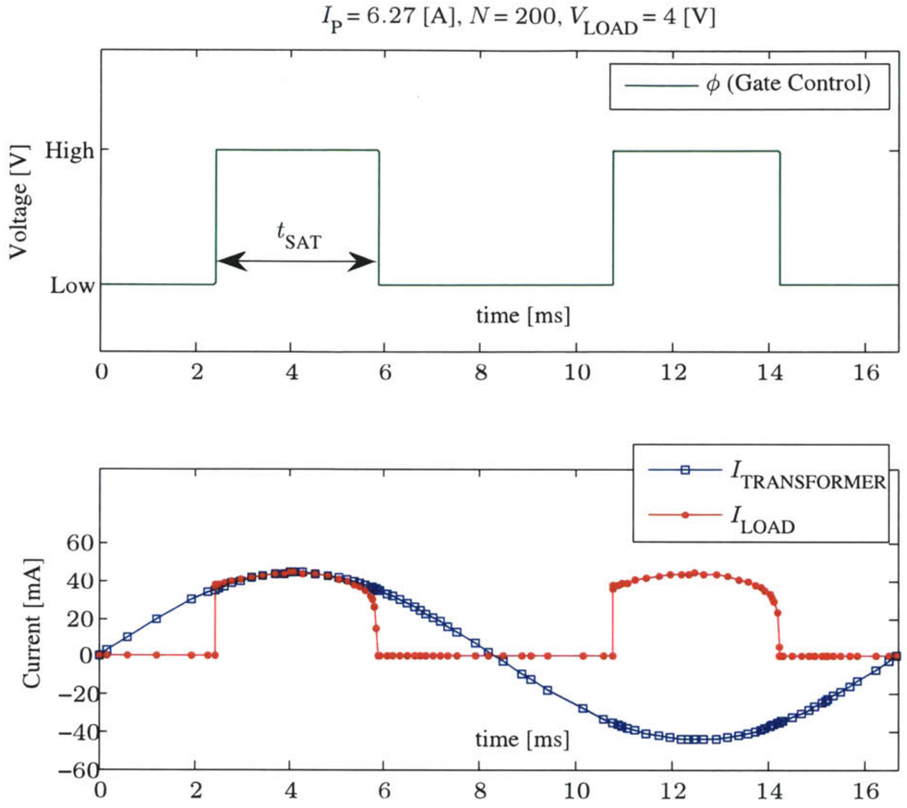


Figure 3-9: Time Domain Waveform with the Transfer Window Alignment

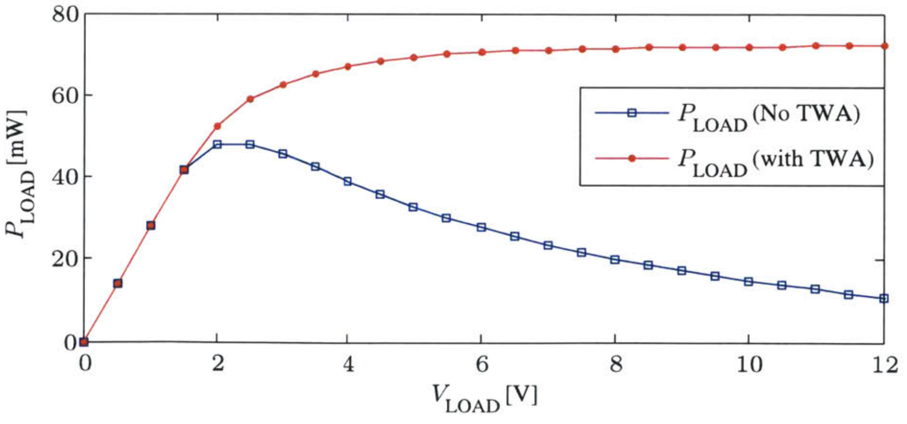


Figure 3-10: Power Harvest Enhancement with the Transfer Window Alignment ($I_P = 6.27A_{RMS}$, $N = 200$)

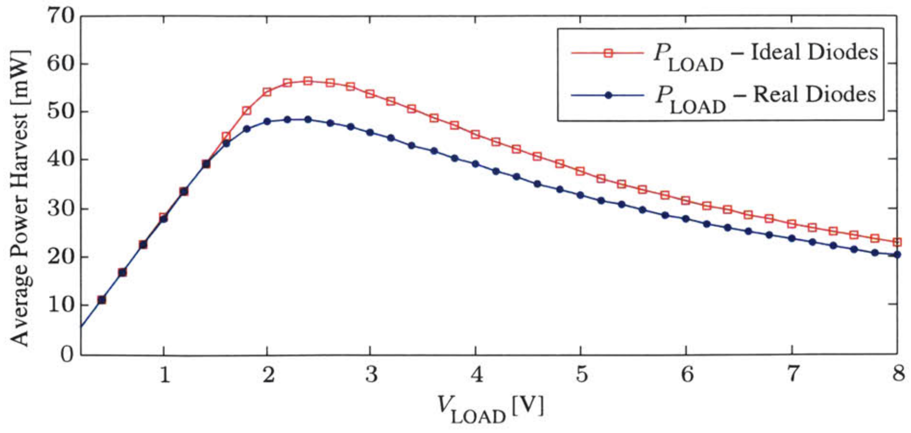


Figure 3-11: Effect of Real Diodes in a Rectifier on Harvestable Power ($I_{\mathrm{P}} = 6.27\mathrm{A}_{\mathrm{RMS}}$, $N = 200$)

3.2.3 Rectifier with Active Gate Control

Using either of the above methods, to extract power from the core, the switching devices in a rectifier must be operated in accordance with the polarity of the transformer current so that the current flows only from the core into the load. Diodes naturally enforce this condition. However, a rectifier implemented with diodes actually has an unavoidable diode voltage drop, V_{DIODE}. Because the core is a current-driven transformer, the load current always sees two diode voltage drops with no dead time, except for saturation where the load current is zero. Therefore, the power dissipation in these diodes must be accounted:

$$P_{\mathrm{SWITCH}} = 2\, I_{\mathrm{LOAD,avg}} \cdot V_{\mathrm{DIODE}} \tag{3.12}$$

The core sees $V_{\mathrm{LOAD}} + 2\, V_{\mathrm{DIODE}}$ instead of V_{LOAD} alone. The rectification contributes to the effectively higher core voltage, so t_{SAT} from the first order estimate is at least slightly overestimated. Figure 3-11 illustrates the effect of real diodes in a rectifier. In this example with $I_{\mathrm{P}} = 6.27\mathrm{A}_{\mathrm{RMS}}$ and $N = 200$, neither the FSC method nor the TWA method is applied. The P_{LOAD} vs. V_{LOAD} response with nonideal diodes shifts to the left due to faster saturation, and is lowered by P_{SWITCH} due to diode loss. Note that P_{SWITCH} is generally different in every point along the V_{LOAD} axis,

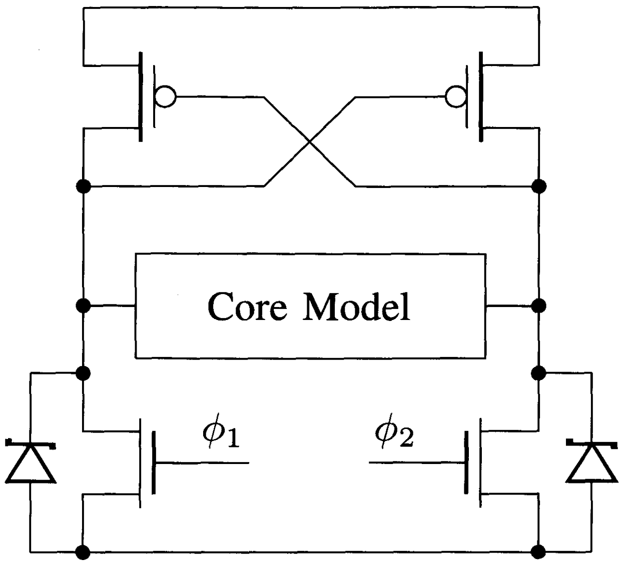


Figure 3-12: Rectifier with active gate control

since $I_{\mathrm{LOAD,avg}}$ is affected by the level of core saturation.

In order to minimize this loss, active rectification can be used as shown in Fig. 3-12, which uses transistors instead of diodes for lower on-resistance. Once the appropriate switching devices are selected for the correct current path, the cross-coupled PFET pair reduces the switching loss by one diode stage loss without any burden on gate control efforts, because the load voltage connected to the core is also powering the PFETs consistently with the selected current path. However, since a FET is a bi-directional device, it cannot block backward conduction, and automatic switching of the current paths based on the cross-coupled pair itself is not possible. That is, unless externally controlled (by diodes, for example), the PFET turns off when $|V_{\mathrm{GS,p}}|$ becomes lower than $|V_{\mathrm{TH,p}}|$ regardless of the direction of the transformer current. Therefore, the voltage of the rectifier output will fall until the FET is in the cut-off region. This leads to power flow from the load into the core, and significantly undermines the power harvest. For this reason, cross-coupled pairs in both top and bottom cannot be used to completely eliminate gate control. (however, if the core is a voltage-driven transformer, complete elimination of the gate control is possible by using cross-coupled pairs in both positions, as shown in [41] and [42]).

This issue can be avoided by applying gate control, ϕ_1 and ϕ_2, to the lower NFET pair. Alternatively, PFETs can be placed as a top pair with accordingly adjusted gate controls, and NFETs as a cross-coupled pair in the bottom. To avoid short-circuiting the output of the rectifier to ground through the FETs, it is required to ensure a finite nonoverlap period between ϕ_1 and ϕ_2. During this period, the polarity crossover of the transformer current happens, and the switching of the current paths is automatically handled by the diodes. Denoting the diode operated duration, the FET operated duration, and the on-resistance of the FET as t_{DIODE}, t_{FET}, and $R_{\text{ds-on}}$, respectively, the switching loss can be written as:

$$\begin{aligned} P_{\text{SWITCH}} &= V_{\text{DIODE}} \cdot \frac{2}{T} \int_0^{t_{\text{DIODE}}/2} I_{\text{LOAD}}(t)\ dt \\ &+ R_{\text{ds-on}} \cdot \frac{2}{T} \int_{t_{\text{DIODE}}/2}^{t_{\text{DIODE}}/2+t_{\text{FET}}} I^2_{\text{LOAD}}(t)\ dt \\ &+ V_{\text{DIODE}} \cdot \frac{2}{T} \int_{t_{\text{DIODE}}/2+t_{\text{FET}}}^{t_{\text{DIODE}}+t_{\text{FET}}} I_{\text{LOAD}}(t)\ dt \\ &+ R_{\text{ds-on}} \cdot \frac{2}{T} \int_0^{T/2} I^2_{\text{LOAD}}(t)\ dt \end{aligned} \tag{3.13}$$

where

$$t_{\text{FET}} = \min\left[t_{\text{SAT}} - \frac{t_{\text{DIODE}}}{2}, \frac{T}{2} - t_{\text{DIODE}}\right] \tag{3.14}$$

If the proposed active gate control method is applied to the peak point of the graph with dot marks in Fig. 3-11, where the nominal diode voltage drop (V_{DIODE}) is 0.19 V in mA range using the same Schottky diode, and $I_{\text{LOAD,avg}} = 22.07$ mA, the switching loss is decreased from 8.387 mW to 0.176 mW, which is 97.9% in reduction. Here assumed are $t_{\text{DIODE}}/T = 5\%$ and $R_{\text{ds-on}} = 0.1\,\Omega$. Compared to the power harvest with a diode rectifier providing 48.5 mW, the active gate control increases the power harvest by 16.9%.

Figure 3-13 illustrates two possible cases of ϕ_1 and ϕ_2 control based on the circuit of Fig. 3-12. The upper case illustrates a situation with sufficiently low V_{LOAD} that the core does not saturate during the entire cycle. The lower case exhibits saturation. In this case, the transformer current is going through the left PFET and the right NFET (ϕ_2) in the first half cycle, and then the right PFET and the left NFET (ϕ_1)

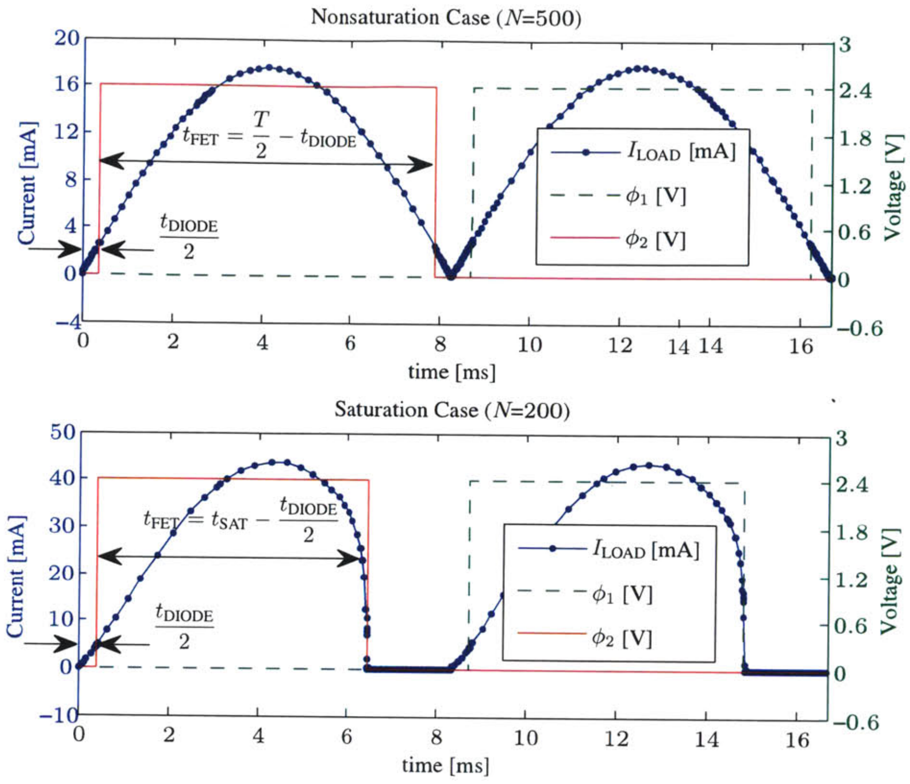


Figure 3-13: ϕ_1 and ϕ_2 Controls in Two Possible Cases

in the second half cycle. When the core goes into saturation, both NFETs need to be turned off slightly before the actual saturation. Otherwise, one of the PFETs will be turned on due to the NFET providing a ground, and the saturated core would complete a short circuit path between the output and the ground by forcing zero voltage between two drain terminals of the turned-on transistors. Since this severely harms the power harvest, t_{FET} has a limit value of $t_{\text{SAT}} - t_{\text{DIODE}}/2$ in saturation (moreover, a small timing margin of several µs before the actual saturation needs to be allocated to avoid the short-circuit path in practice). To incorporate this with a nonsaturating case, the min function is used in (3.14).

3.2.4 Experimental Verification of the Proposed Methods

A prototype was constructed as shown in Fig. 3-14 and Fig. 3-15 to verify analytical predictions. By choosing appropriate devices along the current path and correctly ad-

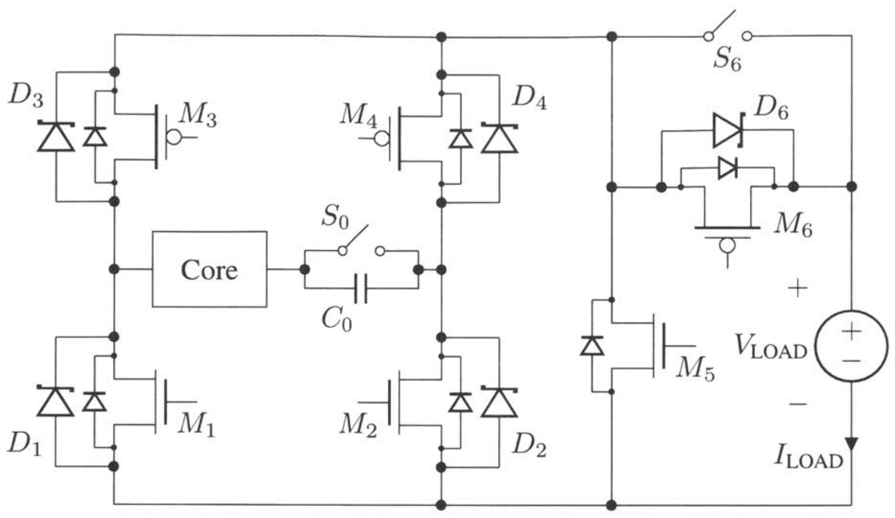


Figure 3-14: Test Board Schematic

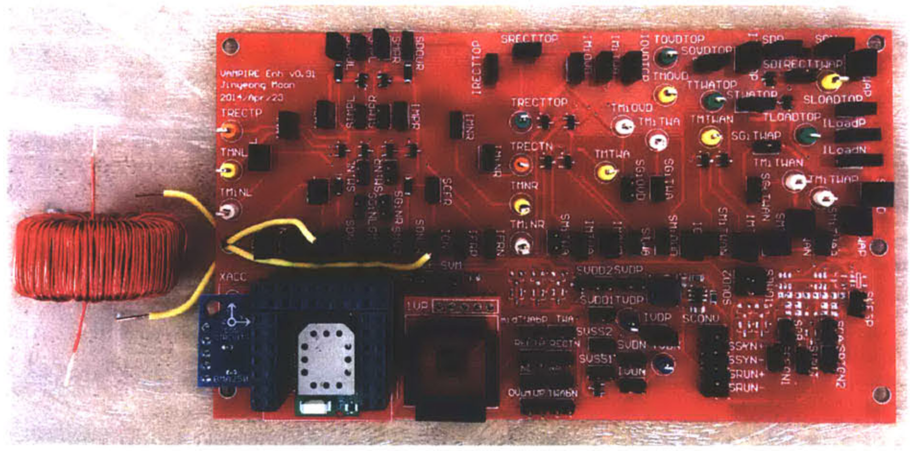


Figure 3-15: Test Board and the Harvesting Core (VAC W380)

justing necessary gate controls, single and combinatory operations of three methods are tested. Applying a constant voltage source as a load, measurements of the voltage across the load (V_{LOAD}) and the current through the load (I_{LOAD}) can be used to calculate power harvest. Note that PFET devices in the rectifier are generally represented as gate-controllable devices, as the cross-coupled mode can be thought as a special kind of active gate control. Small diodes next to FETs represent body diodes of the devices, and large diodes next to FETs indicate explicit Schottky diodes. Switches S_0 and S_6 are mechanical switches to bypass the flux-shaping capacitor (C_0)

Table 3.2: Operation Modes of the Test Board

Mode	Associated Devices in Each Mode						
Base (Diode Full-bridge)	S_0	D_1	D_2	D_3	D_4	-	S_6
Base + AR (Active Rectification)	S_0	M_1	M_2	M_3	M_4	-	S_6
Base + FSC (Flux-shaping Capacitor)	C_0	D_1	D_2	D_3	D_4	-	S_6
Base + TWA (Transfer Window Alignment)	S_0	D_1	D_2	D_3	D_4	M_5	M_6
Base + AR + FSC	C_0	M_1	M_2	M_3	M_4	-	S_6
Base + AR + TWA	S_0	M_1	M_2	M_3	M_4	M_5	M_6

or the devices used in the TWA method (D_6 and M_6) in which case M_5 is also turned off by gate control. Table 3.2 describes associated devices in each mode. Switches S_0 or S_6 described in a mode in Table 3.2 provide a short for bypass operation. For experimentation, the test board includes components that might not be needed in a final implementation. The test board includes a full set of components to explore experimental options for comparison to analytical predictions.

The test board also includes interfaces to digital sensor packages and a wireless radio package in addition to the power electronic circuits. Features added for testing increase the volume of the experimental prototype, but, of course, the practical dimensions of a real harvester application can be very small, close to the dimensions of the core itself as illustrated in Fig. 2-2 and Fig. 3-23, which is introduced shortly.

Unselected FETs in a mode are turned off by appropriate gate signals, for example, M_1 and M_2 in "Base" mode will have logic-0 signals as gate inputs, while M_3 and M_4 will have logic-1 signals as gate inputs. Also, M_5 will be gated with logic-0 as the base case does not employ the TWA method.

Experimental results associated with different operating modes are illustrated in Fig. 3-16. The RMS current of the primary sinusoid is $I_{\mathrm{P}} = 4.0\,\mathrm{A_{RMS}}$, and the number of secondary windings is $N = 200$. Active rectification, if applicable, is performed with 95% utilization of active devices, and is performed even for the voltage range where V_{LOAD} is less than the minimum supply voltage of the microcontroller by using an external power rail for the microcontroller in such cases, even though these operating regimes may not be practical in real harvester applications. The results are included

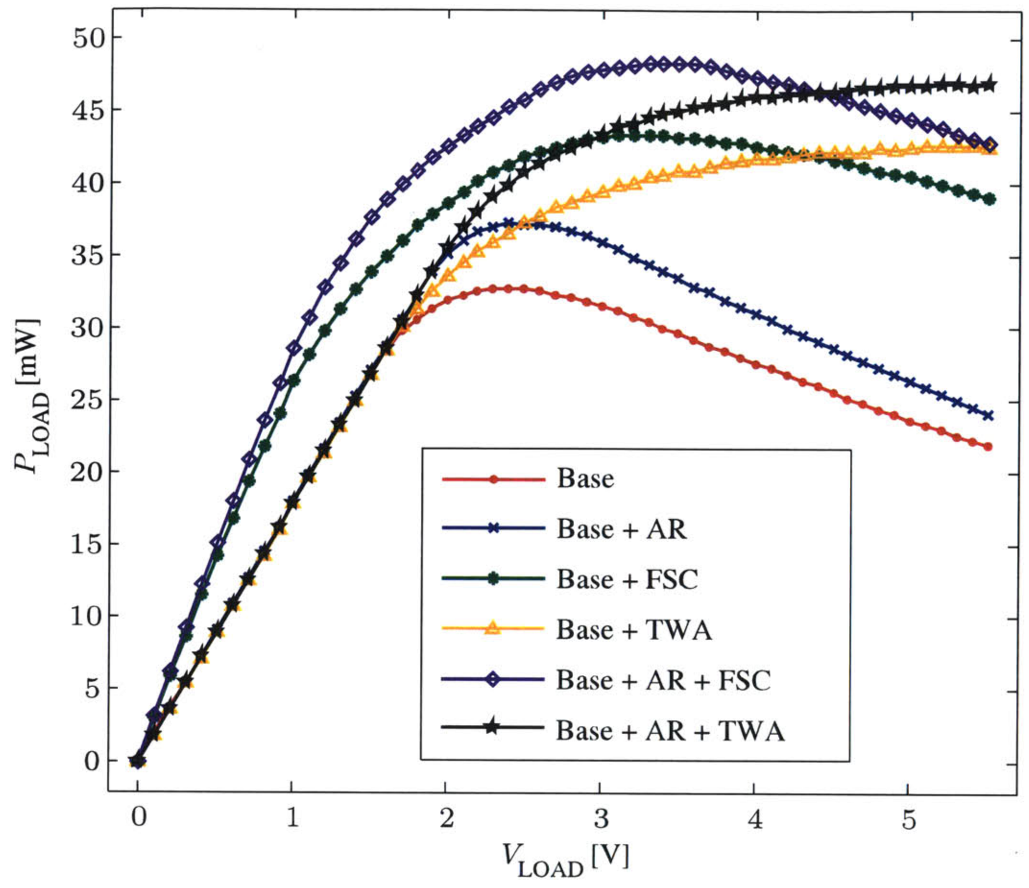


Figure 3-16: Experimental Results ($I_P = 4.0\,\mathrm{A_{RMS}}$, $N = 200$)

to show the full range of predictions. Microchip's PIC18F14K50 microcontroller [43] is used here, with a minimum supply voltage of 1.8 V. The Schottky diodes are PMEG2010ER by NXP Semiconductors [38], the PFETs are DMG3415U by Diodes Inc. [40], and the NFETs are PMV30UN by NXP Semiconductors [39].

Active rectification alone boosts the maximum power harvest from that of the "Base" case by 13.72%, from 32.71 mW to 37.19 mW. The peak harvest occurs at a slightly higher load voltage due to reduction in core voltage from elimination of diode voltage drops. Individually, both the FSC and the TWA methods greatly boost the maximum power harvest from the "Base" case by 32.47% and 30.57%, yielding maximum power harvests of 43.33 mW and 42.70 mW, respectively. As mentioned, the FSC method increases the power harvest in the non-saturated region by acting as a power factor corrector. And, as can be inferred from the figure, the maximum

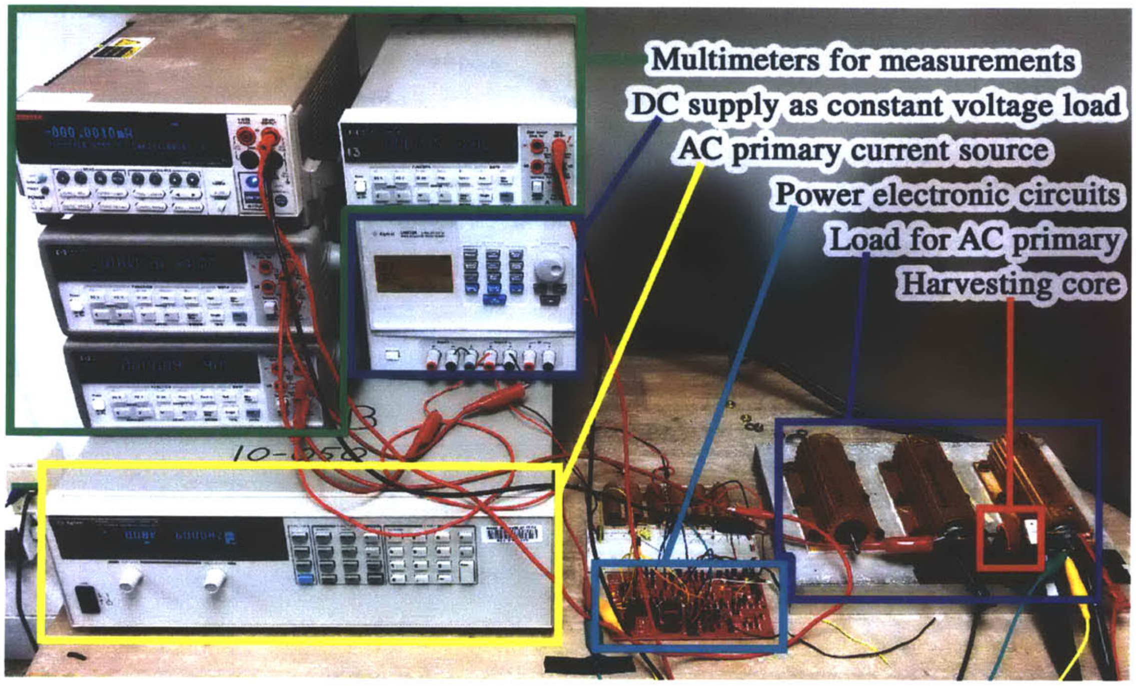

Multimeters for measurements
DC supply as constant voltage load
AC primary current source
Power electronic circuits
Load for AC primary
Harvesting core

Figure 3-17: Testing Environment for Power Efficiency Measurement

power harvest of the TWA method will be higher at higher V_{LOAD}. Finally, when combined with active rectification, the FSC and the TWA methods result in 47.81% and 43.60% increase to the maximum power harvest from the "Base" case, with actual maximum power harvests of 48.34 mW and 46.97 mW, respectively.

To demonstrate circuit performance, the most complex combination of the proposed methods (active rectification with the TWA method) was tested, and the performance of the experimental hardware was experimentally evaluated for primary currents ranging up to $I_{\mathrm{P}} = 30.0\,\mathrm{A_{RMS}}$. The testing environment is depicted in Fig. 3-17, and the results are illustrated in Fig. 3-18. In Fig. 3-18, the y-axis on the right side of the plot represents the actual amount of power harvested, and the y-axis on the left side indicates power efficiency defined as a ratio of harvested power in the secondary to real power dissipated in the primary side due to the harvester. In the figure, the same amorphous nanocrystalline core was used with $N = 200$ and $V_{\mathrm{LOAD}} = 5.0\,\mathrm{V}$. The wire gauge for secondary windings is AWG28.

Equation (3.11) predicts that the power harvest will be directly proportional to I_{P}, and this is essentially true, staying linear up to a certain point ($I_{\mathrm{P}} = 10\,\mathrm{A_{RMS}}$)

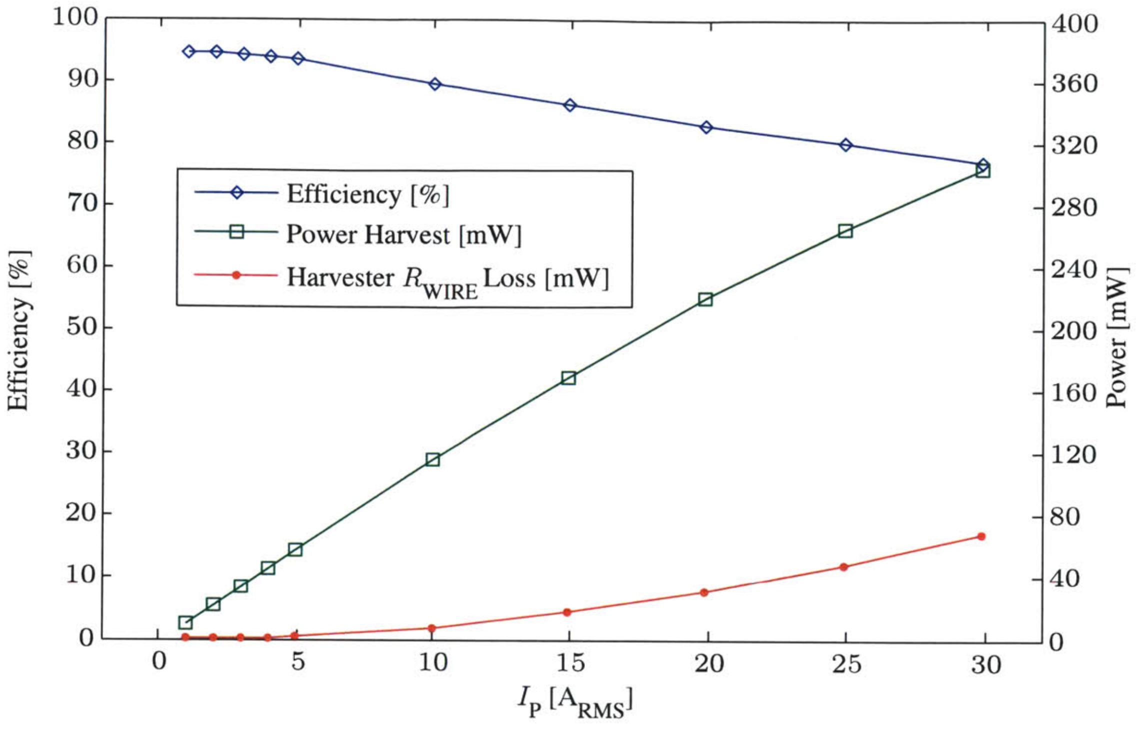


Figure 3-18: Efficiency of the harvester across I_P (V_{LOAD}=5.0 V, N = 200, active rectification + TWA)

as shown in the figure. However, as I_P further increases, the power harvest clearly starts to develop a curvature, making the amount of power harvested less than ideal, due to a second order effect that has been neglected so far: the DC resistance of the secondary windings. The gradual reduction in efficiency for increasing levels of primary current occurs because of t_{SAT}'s dependency on the primary side current, I_P, as opposed to the assumption of total independence presented in the TWA analysis. Even with active rectification, t_{SAT} is shorter than ideal because there is always a voltage drop due to nonzero DC wire resistance. Normally the wire resistance is at most several ohms; therefore, the secondary core current of tens of mA does not cause severe deviation from the ideal linear graph. However, in cases of very high primary current, flux accumulation due to the voltage drop across the DC wire resistance can be significant compared to the main flux accumulation due to the load voltage, causing shorter t_{SAT}. An even more important point is that during the time where the core is intentionally shorted to prepare for the next transfer window, the voltage drop across the wire resistance is still seen by the core, and accelerates flux accumulation, directly reducing a major portion of t_{SAT}. This can be easily mitigated by using a thicker wire for the secondary windings. However, since the thickness of the windings impacts cost and the area of the free window in the center of the core, a careful choice is needed during design. Clearly, however, practically useful levels of energy can be harvested with wires as thin as AWG28. In the next section, a more detailed analysis on this nonideality that stems from the simple flux equality is explored. Before concluding this section, however, there is one more topic to discuss: the practical implementation of the TWA method in the perspectives of switches and control.

3.2.5 Practical Implementation of the TWA method

The conceptual circuit for the TWA method illustrated in Fig. 3-8 requires two additional FETs outside the rectifier. The operation of Fig. 3-8 is simple. It is either shorting out the rectifier to delay the transfer window or providing a regular current path as a normal rectifier does when energy extraction is supposed to occur. When the rectifier is shorted, the voltage across the core is forced to be zero, rendering zero flux accumulation for the core toward magnetic saturation. On the other hand, when the transfer window is initiated, the core sees a constant voltage of the supercapacitor, and steadily accumulates flux, eventually leading to magnetic saturation. It was mentioned that since the positive terminal of the supercapacitor cannot be shorted to ground at any point, the gate controls (ϕ and $\overline{\phi}$) for two added FETs must be carefully timed so that both FETs cannot conduct at the same time. Even so, however, a realistic circuit should be improved from Fig. 3-8 because shorting out the rectifier does not always guarantee close-to-zero impedance (or voltage) across the magnetic core. For example, if the rectifier is a full bridge diode rectifier, shorting out the rectifier still exhibits $2\,V_{\mathrm{DIODE}}$ across the core whenever the secondary current is present, as discussed in Chapter 3.2.3, where V_{DIODE} refers to diode voltage drop. So, the core continuously accumulates volt-second (flux) even when it is supposed to be shorted. It obviously results in faster magnetic saturation, lowering power harvest. As an example, the amount of power harvest with the TWA method is estimated

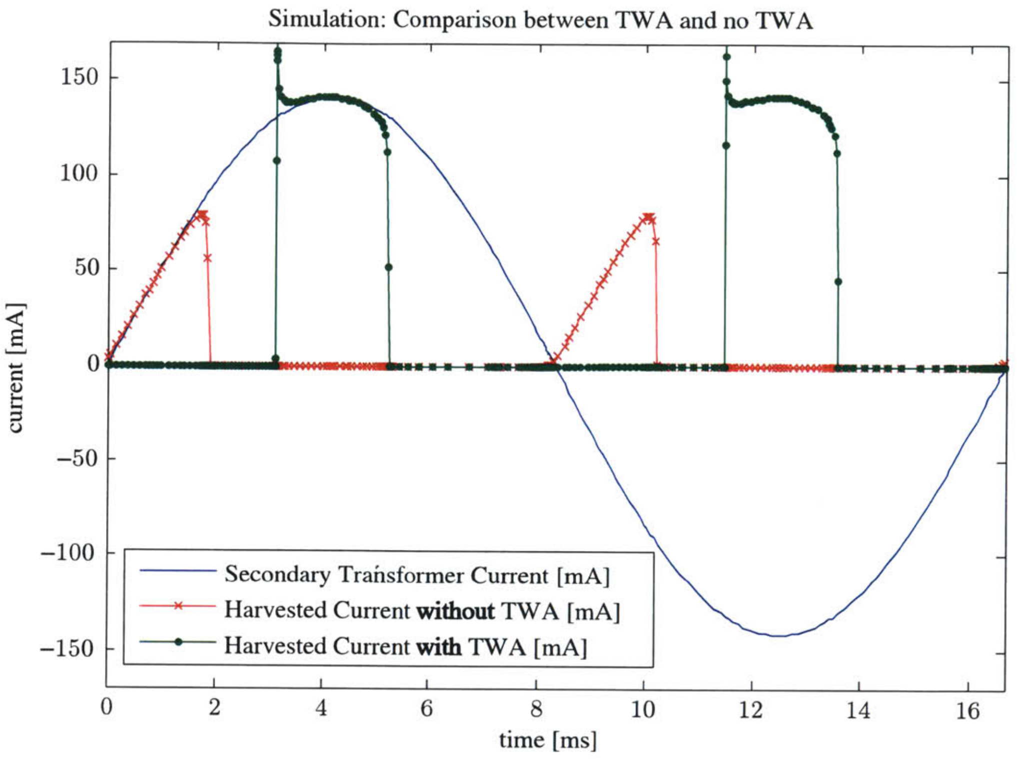


Figure 3-19: The TWA Method vs. Passive Rectification

with $I_{\mathrm{P}} = 10\,\mathrm{A}_{\mathrm{RMS}}$, $N = 100$, $V_{\mathrm{LOAD}} = 4\,\mathrm{V}$, and $V_{\mathrm{DIODE}} = 0.3\,\mathrm{V}$ (assuming Schottky diodes). The resulting current waveforms are presented in Fig. 3-19. The passive rectification case, in red, gives only 41.3 mW even with perfect diodes. The TWA method, in green, gives 139.1 mW assuming a perfectly shorted core using perfect diodes during the transfer window. If the TWA method uses realistic diodes instead of perfect diodes while delaying the transfer window, the harvested power is significantly degraded: 85.9 mW. Even though the TWA method with realistic diodes is superior to the passive rectification, a substantial amount of energy is lost due to nonideal flux accumulation during when the core is supposed to be shorted. In order to completely eliminate this issue, the TWA method requires full knowledge of the rectifier performance.

A better implementation is presented in Fig. 3-20. This circuitry is identical to the general active rectifier, with the additional benefit that two additional FETs for TWA operations in Fig. 3-8 have been eliminated. Note that a rectifier in an elec-

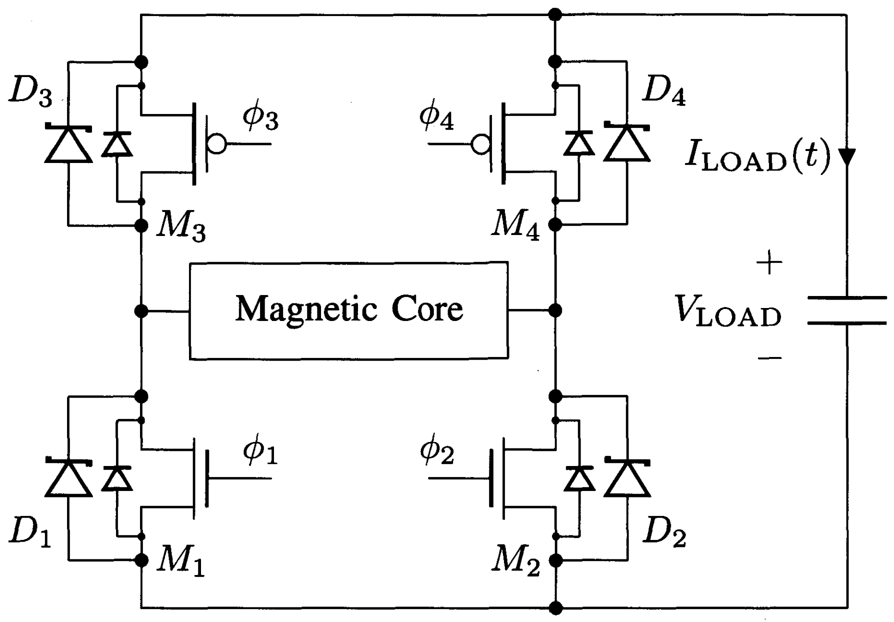


Figure 3-20: Actual Implementation of the TWA Method

tromagnetic energy harvester, which is essentially a nonlinear current transformer, does not suffer from a dead time near zero crossings of a sinusoid unlike a rectifier in a voltage transformer does. This stays true regardless of a type of a rectifier, for example, a diode rectifier. The operations of the TWA method is now implied by the control method, and incorporated with active rectification. The control signals and important time points that separate the operation regions are labeled in Fig. 3-21.

Region 1 ($t_0 \leq t < t_1$): this region is used to detect a zero crossing of a transformer current, using diodes only. Since a magnetic energy harvester works as a current transformer, as soon as the current conducts, either the left or the right node of the magnetic core will exhibit the full load voltage. A microcontroller will monitor the voltage of both sides to detect a zero crossing of the current. In this phase, ϕ_1 and ϕ_2 are driven low to switch off NFETs, and ϕ_3 and ϕ_4 are driven high to switch off PFETs. A zero crossing detection is performed every cycle to continuously synchronize a microcontroller with the line frequency so that the location of the transfer window can be accurately estimated every cycle without dedicated training cycles

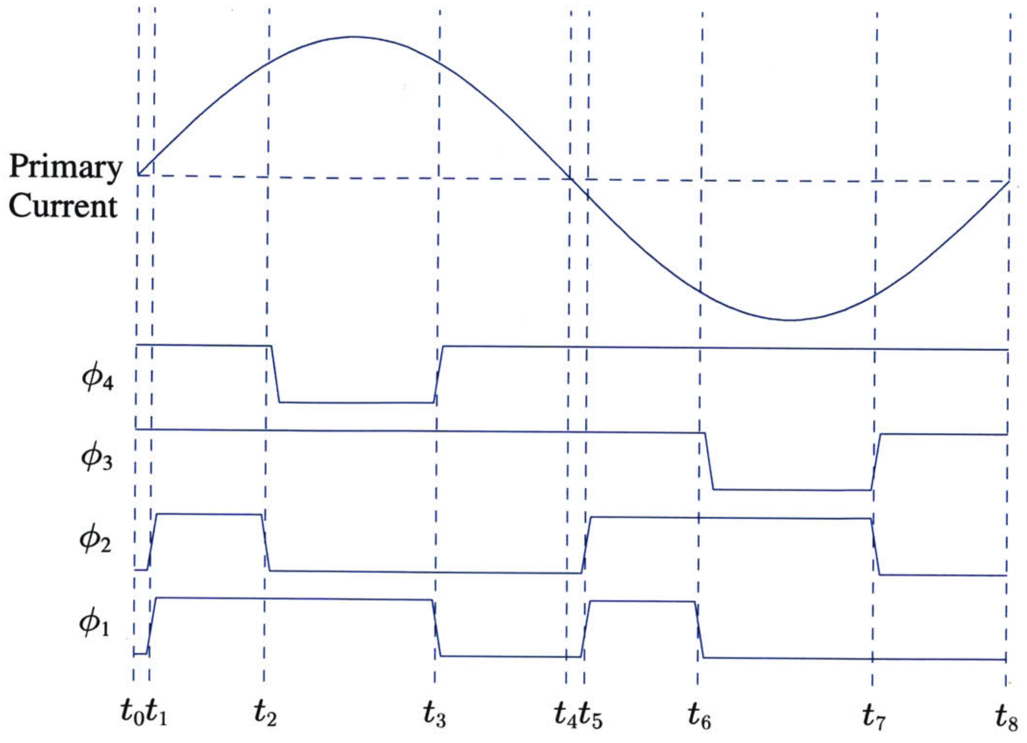


Figure 3-21: Control Signals and Operation Regions of the TWA Method

that will periodically stall the microcontroller. The diagram exaggerates the length of this region for a clearer illustration.

Region 2 ($t_1 \leq t < t_2$): as soon as the zero crossing is detected, the length of the previous cycle period can be obtained by comparing the timestamp of the current zero crossing to the timestamp of the previous zero crossing. The microcontroller uses the information of the most recent zero crossing, the length of the previous cycle period, and the length of the previous transfer window for estimating the location of the current transfer window. As soon as all the required timing calculations for the upcoming transfer window are done, both NFETs are driven high to short out the core. A negligible amount of flux will be accumulated for the magnetic core during this phase. PFETs are still driven high to make sure that the supercapacitor is isolated from the grounded core. The core is intentionally shorted to delay the transfer window.

Region 3 ($t_2 \leq t < t_3$): one of the NFETs is now turned off to move the core out of the shorted state. For example, ϕ_2 is driven low in Fig. 3-21, intending a current flow with M_1 and M_4. However, the control for M_4 that will drive ϕ_4 low cannot take place at the same time as ϕ_2 transitions to low to avoid a short-circuiting path of the supercapacitor, similar to the ϕ and $\overline{\phi}$ control in Fig. 3-8. In order to reflect this, ϕ_4 in Fig. 3-21 is driven low slightly after ϕ_2 makes the transition. This phase is where the most of the energy extraction occurs. The transfer window is initiated, shifted from the zero crossing of the sinusoid, and the maximum current over a window that is t_{SAT} long is extracted from the core through the active rectifier. This phase ends when the core goes into saturation, and the microcontroller records such an event so that the information can be used to estimate the location of the transfer window in the next cycle.

Region 4 ($t_3 \leq t < t_4$): as soon as the core goes into saturation, all the FETs are switched off. Only diodes are used to commutate the direction of the current in the rectifier. This prepares the detection of the next zero crossing of the transformer current.

Region 5 ($t_4 \leq t < t_5$): this serves the same purpose as the region 1, except for that the current direction is now reversed. Therefore, a zero crossing will be detected on the other side of the core.

Region 6 ($t_5 \leq t < t_6$): this serves the same purpose as the region 2.

Region 7 ($t_6 \leq t < t_7$): this serves the same purpose as the region 3, except for that the current is supported by M_2 and M_3. Note that ϕ_3 also transitions slightly after ϕ_1 is driven low to avoid a short-circuiting path of the supercapacitor.

Region 8 ($t_7 \leq t < t_8$): this serves the same purpose as the region 4.

With this implementation, not only the number of switches are lower, but also

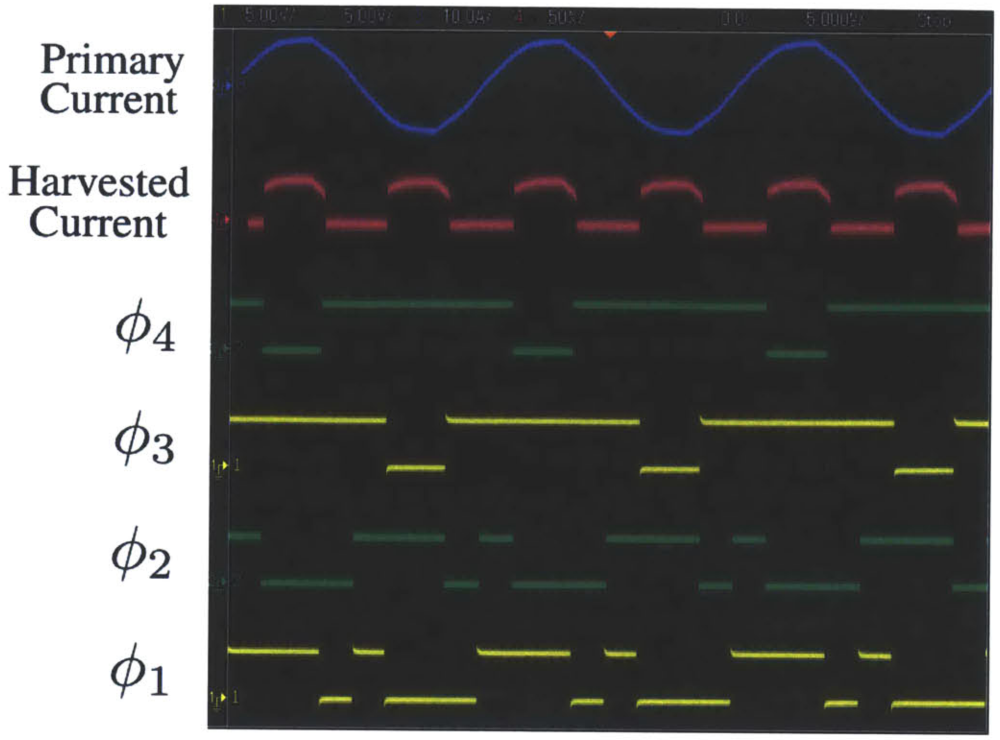


Figure 3-22: Oscilloscope Screenshot of the TWA Method and Active Rectification

the core never accumulates unwanted flux when it is intentionally shorted, except for volt-second developed by the intrinsic on-resistance of the FETs and DC wire resistance. The cross-coupled FETs, discussed in Chapter 3.2.3 to reduce the control efforts, unfortunately cannot be used in one of the top or bottom slots in the rectifier anymore, because it presents a short-circuiting path for the supercapacitor during the intentional short-out period of the core.

The test board presented in the previous chapter, in Fig. 3-15, was not constructed with this TWA implementation in mind. However, the experiment for this practical TWA implementation can be done with the same test board with S_0 and S_6 closed, M_5 disconnected by asserting its gate with logic-0, and generating gate control signals according to Fig. 3-21. However, it is not easy to see the real size of the actual power electronic circuits for the electromagnetic energy harvester. A new harvester prototype is built and illustrated in Fig. 3-23 for that purpose. All the circuit components of the test board are neatly assembled in a tight space without testing terminals. The main bottlenecks for the size of the harvester are a magnetic core and supercapacitors.

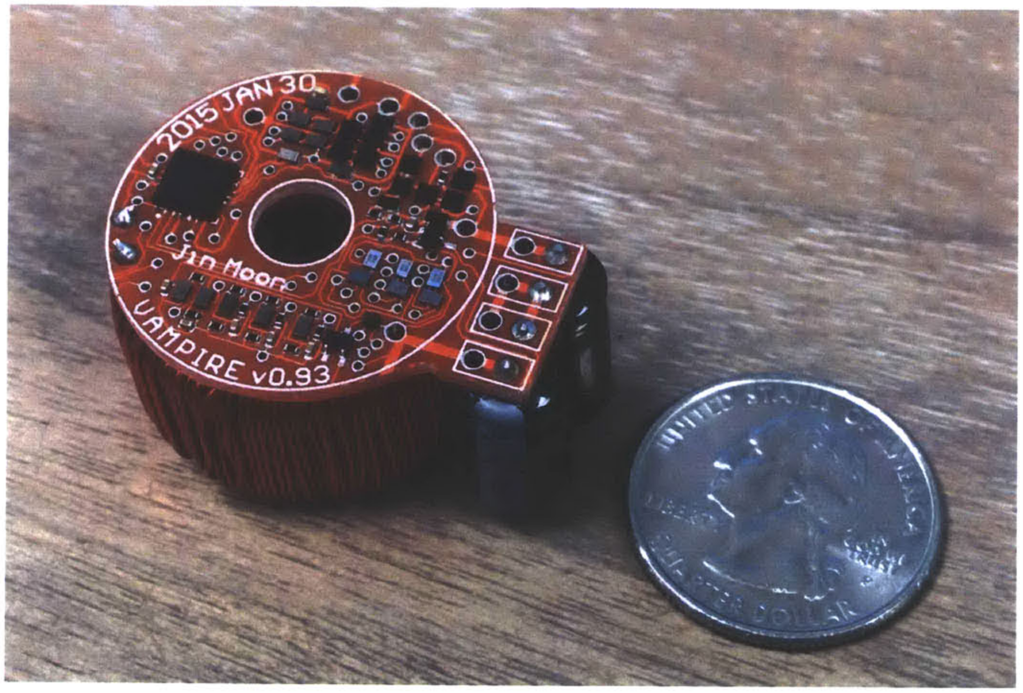


Figure 3-23: Electromagnetic Energy Harvester Prototype with the TWA Method

All other circuit components on the test board can be easily placed on top of the flat surface of the core. Considering that the circuit board of Fig. 3-23 also contains the initialization circuits and voltage regulation circuits that were not discussed in this chapter, the actual size of an energy harvesting circuitry is even smaller than shown.

The scopeshot of the TWA method and active rectification is given in Fig. 3-22. The first signals set, ϕ_1 and ϕ_2, are separately captured from the second signals set, ϕ_3 and ϕ_4. Two sets are overlaid for illustration purposes, and the zero crossings of the primary currents in both sets are aligned for providing the accurate illustration. A heater, which consumes $5.5\,\mathrm{A_{RMS}}$, is used as a primary side. The harvested power is $65.9\,\mathrm{mW}$ ($I_{\mathrm{LOAD,avg}} = 16.30\,\mathrm{mA}$ at $V_{\mathrm{LOAD}} = 4.04\,\mathrm{V}$) with $N = 200$. Using identical settings except for the rectifier, passive rectification harvests $38.6\,\mathrm{mW}$ ($I_{\mathrm{LOAD,avg}} = 9.65\,\mathrm{mA}$ at $V_{\mathrm{LOAD}} = 4.00\,\mathrm{V}$). The harvested current in Fig. 3-22 shows that the transfer window is clearly shifted to contain the peak of the primary current at its center. All the controls in Fig. 3-22 are working as intended, identical to Fig. 3-21. However, as mentioned earlier, the region 1 and 4 are barely visible as they are

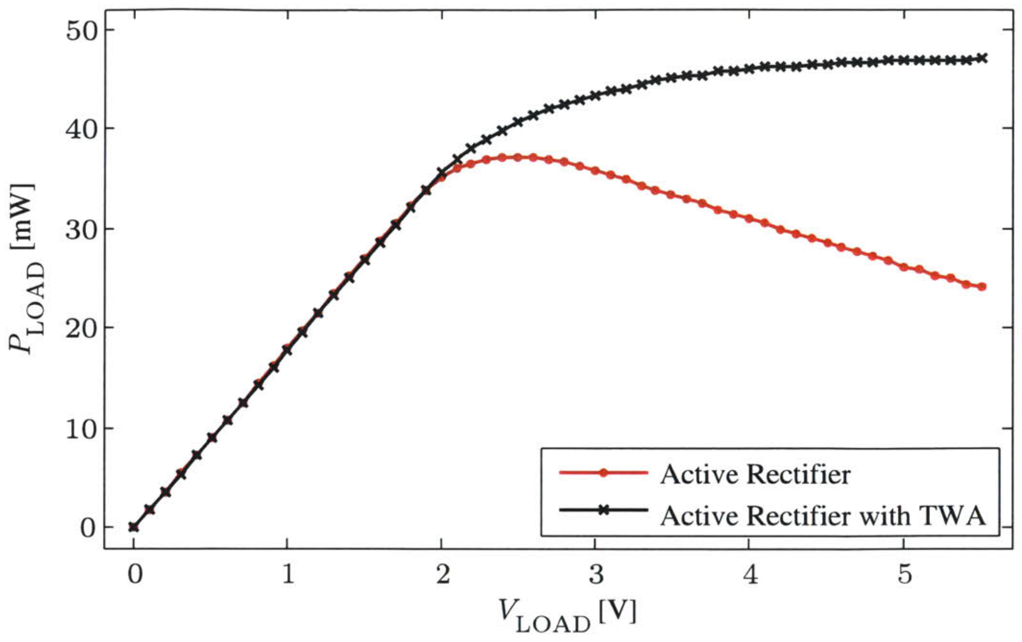


Figure 3-24: Power Harvest vs. Level of Core Saturation with the TWA Method

extremely brief compared to other phases.

For verification of the new prototype, a similar experiment is carried out. As discussed in the previous section, the TWA method increases energy extraction as the core goes into deeper saturation, but with a diminishing return. Fig. 3-24 illustrates such a behavior. In order to apply different levels of core saturation for the experiment, J is directly controlled by changing the level of the load voltage. Remember that the core goes into deeper saturation as J approaches to 0 since the transfer window is directly shrinking. A constant voltage source with a proper shunt impedance that can take in all the harvested current from the core is used as a replacement of supercapacitors. The level of the voltage source is varied from 0 V to 5.5 V. A current limited AC power source drives the primary side, with the current limited at 4.0 $\mathrm{A_{RMS}}$ (60 Hz). Using the AC power source is also benficial in generating a purer sine wave than the slightly distorted sine wave shown in Fig. 3-22 due to nonidealities coming from the load and the utility grid. The number of windings in the harvester core for Fig. 3-24 is $N = 200$.

The implementation of the TWA method introduced in this subsection is excellent in enhancing power harvest from an electromagnetic coupling. What makes it better is that it trains the important time points such that the length and the location of the transfer window are automatically adjusted each cycle. No information from the primary side needs to be measured.

3.2.6 Control Power

One microcontroller is dedicated for controlling power electronic circuits. In experiments conducted here, the power consumption of the microcontroller running at 8 MHz ranged from 2.63 mW (at 1.8 V) to 10.51 mW (at 5.5 V). Since the supply voltage of the digital circuits, including the microcontroller and sensor suites, is usually set to the lowest supply voltage to minimize the power consumption in real applications, the overhead of the entire microcontroller operations tends to be closer to the lower boundary, 2.63 mW. In this case, harvesting energy from the primary current of as low as 0.3 A_{RMS} is sufficient for sustaining the microcontroller.

3.3 Very High Primary Side Current Scenarios

In this section, the limit of the hand analyses based on the most fundamental flux equality is explored. This problem does not surface with actual numerical or SPICE simulations as the entire core operations are modeled with the *B*-*H* curve of the material along with Maxwell's equations. Under certain circumstances, the quick hand analyses on the extent of the transfer window for intuitive design fails to align with the actual simulation. The root cause is discussed in detail to provide a remedy to close the gap between the hand calculation and the numerical results.

3.3.1 Limitation of the Simplest Flux Model

The calculation of the length of the transfer window has been based on the most fundamental flux equality until saturation. Rewriting the most fundamental flux equality in case of a constant voltage load, (2.10), yields:

$$V_{\text{LOAD}} \cdot t_{\text{SAT}} = 2\, B_{\text{SAT}} \cdot A_{\text{CORE}} \cdot N \tag{3.15}$$

The esimation of t_{SAT} is:

$$t_{\text{SAT}} = \frac{2\, B_{\text{SAT}} \cdot A_{\text{CORE}} \cdot N}{V_{\text{LOAD}}} \tag{3.16}$$

This first order linear equation can be deemed true as long as there is no other significant voltage drops across two terminals of the magnetic core except for the constant voltage load seen by the core. There are many nonideal elements that can cause deviations from this relationship with different levels of impact. An example is on-resistance of a switch. Whenever there exists a current, a voltage is developed across the on-resistance, stressing the core with additional flux. Assuming 100 mΩ for simplicity, the on-resistance of a switch develops a peak voltage of 70.71 mV with a secondary current surge of 0.5 A_{RMS}. Compared to the nominal constant voltage load of 4 V for instance, 70.71 mV from the on-resistance of a switch is fairly negligible. Furthermore, the secondary current of 0.5 A_{RMS} implies a 12 kW (120 V_{RMS}-100 A_{RMS})

device in the primary side with the windings of $N = 200$, which is a reasonable number of windings for an electromagnetic energy harvester. Therefore, many applications with similar or lower power ratings can neglect the effects of the on-resistance from the switches.

The example of on-resistance might sound assuring, but in essence, it is also clearly hinting at cases that this flux balance relationship definitely fails to explain. One troublesome case is when there is a faulty component, for example, a failing diode with a high voltage drop. Also, a primary device that dissipates much higher than 12 kW definitely causes a severe deviation from the given flux equality. Similarly, any resistance that is much higher than the on-resistance of a switch, for example, extremely thin wire, causes a significant deviation even at the same or lower primary power levels. All of these cases accumulate additional flux that is comparable in magnitude to the flux development of the load voltage, raising a need for a more accurate model for the flux balance. In fact, besides faulty circuit components which cannot be predicted at the design phase, the main issue due to a high primary side current arises from DC wire resistance of the windings. This is soon explored in the next subsection, and the flux balance model is expanded to include this secondary effect.

Note that such a deviation is caused by significant flux accumulation due to various elements, not only limited to higher primary currents. However, any significant flux accumulation in parasitic elements can be explained with the approach taken in this section with higher primary power along with a scaled load voltage. Therefore, the analysis onward considers the higher primary power case only for a simpler illustration.

3.3.2 New Flux Model

Assume secondary windings of 200 with a wire gauge of AWG30 and the same core parameters as described in Table. 3.1. If it is a copper wire, the resistance of the wire is approximately 14 mΩ per each turn that is approximately 4 cm for this magnetic core. This yields 2.8 Ω for 200 turns. This is an order of magnitude larger than on-resistance of a switch. When combined with the secondary current of 0.5 $\mathrm{A_{RMS}}$, voltage developed due to the DC resistance of the wire reaches 2 V, which is directly comparable to the load voltage. Therefore, the flux accumulation from the wire resistance and the load voltage must be accounted together. A realistic modification to the flux equality leads to the following general equation:

$$V_{\mathrm{LOAD}} \cdot (t_2 - t_1) + R_{\mathrm{DC}} \cdot \int_{t_0}^{t_2} \frac{I_{\mathrm{P}}}{N} \sin(\omega\,(t - t_0))\;\mathrm{d}t = \; 2\,B_{\mathrm{SAT}} \cdot A_{\mathrm{CORE}} \cdot N \tag{3.17}$$

Denoted as t_0 is the most recent zero crossing point of the primary current. The start and the end time points of the transfer window are defined as t_1 and t_2. The angular frequency of the primary sinusoid is denoted as ω, and the amplitude of the primary side current is denoted as I_{P}. The DC resistance of the windings is denoted as R_{DC}. Essentially, the connection between the core and the constant voltage load will be made from $t = t_1$ to $t = t_2$, which is the transfer window. The magnetic core will enter saturation after $t = t_2$. Even though the transfer window starts from $t = t_1$, the voltage developed by the wire resistance must be accounted from the zero crossing of the transformer current to saturation, resulting in undesirable flux development. In practice, any other resistive component in flux development can be absorbed into R_{DC}. Similarly, any voltage component that contributes to faster saturation can be absorbed into V_{LOAD} with a proper scaling as well. In the next subsections, implications of the new flux balance model in existing harvesting methods will be discussed.

New Flux Model with Passive Rectification

Passive rectification can be represented by (3.17) if t_1 is set to t_0, which means the transfer window is initiated at the zero crossing of the primary side current. In this case, the flux equality can be simplified with the notion of the transfer window:

$$V_{\mathrm{LOAD}} \cdot t_{\mathrm{SAT}} + \frac{R_{\mathrm{DC}}\,I_{\mathrm{P}}}{\omega\,N}\,[1 - \cos\,(\omega\,t_{\mathrm{SAT}})] = \; 2\,B_{\mathrm{SAT}} \cdot A_{\mathrm{CORE}} \cdot N \tag{3.18}$$

Here, t_{SAT} denotes the length of the transfer window. Since the right hand side of (3.18) is a physical constant once the core parameters are decided, any amount that the R_{DC} part develops results in a smaller amount for the V_{LOAD} part, hence a shorter t_{SAT}. In the same sense, if the R_{DC} part is negligible, for example, very low wire resistance or very low operating current, this equation approaches (3.15). As an explicit solution is not available due to the form of the equation, t_{SAT} is numerically obtained from (3.18). The actual harvested power in the case of passive rectification can be estimated by:

$$\begin{aligned} P_{\text{passive}} &= \frac{\omega}{\pi} \int_0^{t_{\mathrm{SAT}}} V_{\mathrm{LOAD}} \cdot \frac{I_{\mathrm{P}}}{N} \sin(\omega\, t)\ \mathrm{d}t \\ &= \frac{V_{\mathrm{LOAD}} \cdot I_{\mathrm{P}}}{\pi N} \cdot [1 - \cos(\omega\, t_{\mathrm{SAT}})] \end{aligned} \tag{3.19}$$

In passive rectification, the modified flux equality affects the calculation of t_{SAT}, and t_{SAT} is always lower than the ideal prediction. Since $[1 - \cos(\omega\, t_{\mathrm{SAT}})]$ also decreases as t_{SAT} decreases, lower power harvest is expected.

New Flux Model with the TWA method

Using the general equation (3.17), the TWA method can be represented by placing t_1 and t_2 with a symmetry around the peak of the primary sinusoid, i.e., $t_1 = T/4 - t_{\mathrm{SAT}}/2$ and $t_2 = T/4 + t_{\mathrm{SAT}}/2$, where T denotes the period of the primary sinusoid. The zero crossing of the primary side current (t_0) is no longer the same as t_1. In this case, with the notion of the transfer window, the general flux equation simplifies to:

$$V_{\mathrm{LOAD}} \cdot t_{\mathrm{SAT}} + \frac{R_{\mathrm{DC}}\, I_{\mathrm{P}}}{\omega N} \left[1 + \sin\left(\frac{\omega\, t_{\mathrm{SAT}}}{2}\right)\right] = 2\, B_{\mathrm{SAT}} \cdot A_{\mathrm{CORE}} \cdot N \tag{3.20}$$

The form of the equation (3.20) is similar to that of (3.18) in that the equality also approaches (3.15) if the flux accumulation from the resistive part is negligible, and that any parasitic amount that the R_{DC} part develops directly decreases t_{SAT}. Specifically, if the primary side current increases, both (3.18) and (3.20) result in shorter transfer windows as the weight toward the wire resistance increases. However, the behavior of (3.20) much differs from passive rectification if the parasitic term dominates the flux equality, where t_{SAT} is relatively short. The fundamental difference is that the equation of passive rectification does not allow t_{SAT} to be zero while the equation (3.20) allows t_{SAT} to be zero in the TWA method. Zero t_{SAT} in the TWA method physically implies an extreme case where the core gets saturated even before the transfer window begins, resulting in zero power harvest. The DC resistance in the TWA case generates the entirety of the allowed amount of flux accumulation during when the core is supposed to be shorted to delay the transfer window. The core gets saturated without ever being connected to the load voltage for power harvest. For the case of passive rectification, t_{SAT} physically cannot be zero because as soon as the core is recovered from magnetic saturation the wire and the load receive the same secondary transformer current and develop the flux at the same time for the same duration. Whether the window is extremely short or not, this ensures nonzero t_{SAT} and nonzero power flow into the load before the core is again saturated.

With the advanced model equation comes a very important change in the analysis. Unlike the analysis done in the TWA section, t_{SAT} is no longer independent of the primary side current. The TWA method originally maximizes energy harvesting by placing the transfer window at its optimal position, using the independence of the transfer window from the primary side, as shown in (3.15). However, according to the more realistic flux equality, (3.20), the position of the transfer window actually affects the length of the transfer window. Environment variables, e.g., frequency, now have different impacts on the length of the transfer window coupled with their dependency on the primary side as well. As a secondary effect, the modified equation also complicates the calculation since it cannot provide an explicit solution like (3.16). Note that the R_{DC} parts in the new flux equations of both passive rectification and the TWA method are actually independent of N due to the fact that R_{DC} itself is also proportional to N (multiplied by the unit resistance per turn).

In the original flux equality (3.15), doubling N means doubling t_{SAT}. In a physical sense, the saturation flux linkage is doubled, and the increased amount is solely consumed by the load voltage. Therefore, the core stays out of saturation exactly twice as longer. However, it does not mean that the power harvest will double, because the winding ratio is changed to $1 : 2N$ (the transformer current in the secondary side is halved). In fact, in the TWA method, J, which is a direct knob to control the level of power harvest and should be lowered to provide higher power harvest, is proportional to N/V_{LOAD}. Therefore, increasing only N actually results in lower energy extraction. A realistic design will always match V_{LOAD} such that J remains constant at least. A simulation result to validate these points will be presented shortly.

With the modified flux equality (3.20), however, doubling only N results in more than doubling t_{SAT}. This is because the right hand side – saturation flux linkage – is exactly doubled whereas the coefficients on the left hand side do not follow fast enough. To be specific, in passive rectification with (3.18), it can be shown that $1 - \cos(\omega \cdot 2\,t_{\text{SAT}})$ is always smaller than $2\,[1 - \cos(\omega\, t_{\text{SAT}})]$. Therefore, doubling t_{SAT} on the left hand side still falls short compared to the right hand side with $2\,N$. Similarly in the TWA case with (3.20), it can be shown that $1+\sin\left(\frac{\omega \cdot 2\,t_{\text{SAT}}}{2}\right)$ is always smaller than $2 \cdot \left[1 + \sin\left(\frac{\omega\, t_{\text{SAT}}}{2}\right)\right]$. So in both scenarios, the left hand sides must be compensated with longer t_{SAT} to maintain the equality.

In practical situations where V_{LOAD} is also doubled to match J, the V_{LOAD} part on the left hand side is doubled and the total flux linkage on the right hand side is also doubled. However, the R_{DC} part on the left hand side remains rather constant regardless of changes in environmental variables. Therefore, the left hand side is not still sufficient to match the right hand side. Again, t_{SAT} has to be increased for the equality. As a result, the same J with higher N and higher V_{LOAD} performs better in energy extraction. A physical interpretation is simple: distributing the allowable saturation flux with more weight on the load voltage than the wire loss leads to higher power harvest. Since the same J with higher N yields a better result, high power primary cases should employ this approach for their designs. Fig. 3-25 illustrates the simulation results with different J and N combinations with the TWA method. Simulation results are obtained by LTSpice with the circuit model introduced in Chapter 3.1 and the core parameters given in Table. 3.1. This is to illustrate realistic impacts of J, N, and V_{LOAD} and their differences on power harvest in detail in high

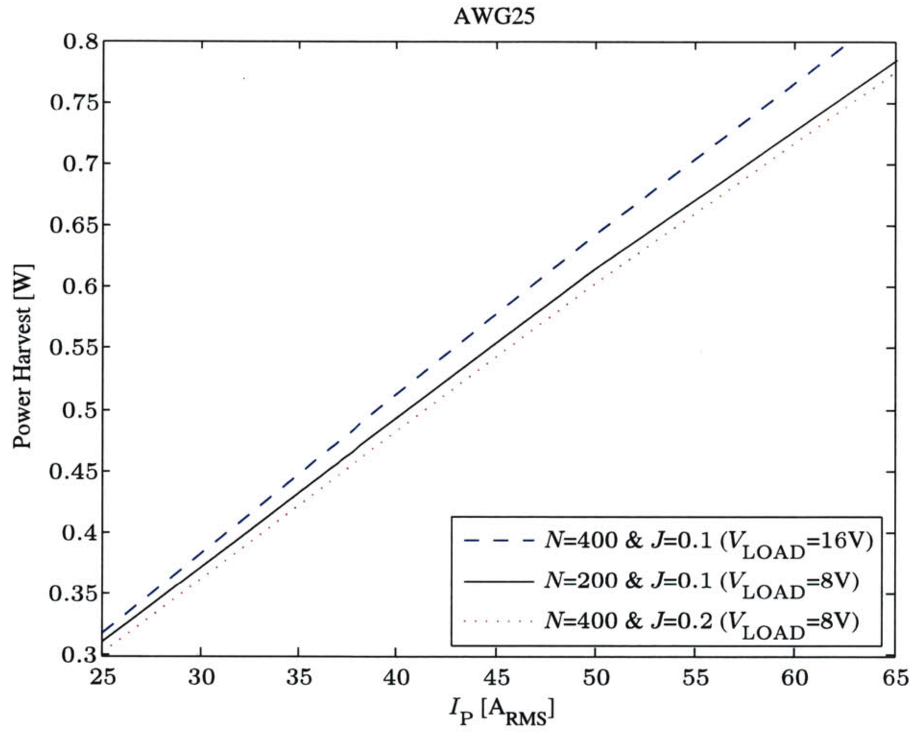


Figure 3-25: Power Harvest Comparison with the Same N or the Same J under High Power Primary Condition

power primary cases. It can be verified that increasing only N (with the same V_{LOAD}) is actually worse in power harvest (solid line vs. dotted line). Also, with the same J, the case with higher N and higher V_{LOAD} harvests more power according to the modified flux equality (dashed line vs. solid line). For the simulations, the wire gauge of AWG25 is used. Note that the difference between the dashed line and the solid line (the same J with different Ns) becomes larger as higher primary power levels put more weight toward the R_{DC} part in the flux balance. With the proposed flux balance model, these impacts can be hand-calculated as well, though it will be less accurate than numerical simulations. The accuracy of the modified flux model will be presented in the next subsection.

To summarize, the new flux equality affects the calculation of t_{SAT} in the TWA method as in passive rectification, but it includes the possibility of zero t_{SAT} unlike the case of passive rectification. Further delving into the TWA method, the new model presents that the same Js in the TWA method can yield different levels of power

Table 3.3: Estimations of t_{SAT} with Different Flux Equality Models

Flux Eq. Model	Passive Rect.	TWA Method
Simple Linear Model (3.15)	4.284 ms	4.284 ms
Detailed Model (3.20)	3.926 ms	3.641 ms
Simulation	3.836 ms	3.433 ms

harvest due to different Ns, which change the level of the secondary transformer current and the amount of flux accumulation in parasitic elements.

3.3.3 Performance of the New Flux Model

The new flux balance model for passive rectification and the TWA method is tested against the simplest flux equality given in (3.15). The testing condition is set such that the parasitic flux accumulation takes up a significant portion in the flux balance. Table. 3.3 shows the improved accuracy on the estimations of t_{SAT} with the new flux equality model. Since the magnetic core with saturation is entwined with differential equations and nonlinear material properties, the proposed estimations as in (3.18) or (3.20) cannot be as accurate as the actual numerical circuit simulation. However, the modified flux balance model gives notably better estimations of t_{SAT} than the original flux equality. The most important part is that the crucial nonideality that drives zero t_{SAT} in the TWA method with high power primary cases can be properly considered during a design process to calculate the harvested power. Also can be verified is that in this high power primary cases passive rectification has a longer transfer window than the TWA method since the TWA method loses more flux to the resistive element before the load voltage is actually connected to the core. This is consistent with the zero t_{SAT} analysis. For the generation of Table. 3.3, assumed are freq $= 60\,\text{Hz}$, $V_{\text{LOAD}} = 4.0\,\text{V}$, $N = 200$, $I_{\text{P}} = 30\,\text{A}_{\text{RMS}}$, and the wire gauge of AWG30.

In the next subsection, based on the new flux balance model, the practicality and efficiency of the TWA method are explored, revealing an interesting point: any shifting of the transfer window from passive rectification results in suboptimal power harvest in high primary current cases.

3.3.4 The Effect of Higher Primary Current

Intuitively, a higher current in the primary side generates stronger magnetic fields, increasing the amount of energy harvested from them. For passive rectification, this is true because the harvested power is proportional to $V_{\mathrm{LOAD}} \cdot I_{\mathrm{P}} \cdot [1 - \cos(\omega\, t_{\mathrm{SAT}})]$, as given in (3.19). The rate at which I_{P} is increasing always overcomes the rate at which $[1 - \cos(\omega\, t_{\mathrm{SAT}})]$ is decreasing. On the contrary for the TWA case, a higher current in the primary side does not necessarily mean higher power. It can greatly enhance power harvest for a certain range of I_{P}, but it has the clear extreme case where the entire allowable saturation flux is consumed by the wire resistance, which ends up with zero power harvest.

Initially from the zero primary current, energy extraction in the TWA method is almost linear with the rise of I_{P} due to negligible voltage across the wire resistance. As the current in the primary side further increases, the wire resistance develops higher voltage across it and a deviation from the ideal case grows larger. A thicker wire will sustain the near-ideal behavior longer than a thinner wire since the thicker wire will have assigned more flux accumulation toward the load voltage. The simulation results are provided in Fig. 3-26, illustrating an effect of wire thickness on energy extraction across a range of primary current levels. They all start as a linear function when the primary side carries a lower current. As the current goes up, the voltage developed across the wire resistance causes deviation from the ideal wire case. Clearly, the thicker wire is closer to the ideal case as it can ensure a longer t_{SAT}. If the current range of the primary side is pre-determined, the wire gauge can be calculated based on how much deviation is allowed for the maximum power case, which is described in the next section. Note that each curve in the figure is drawn up to the maximum power harvest point. After the maximum power harvest points, the losses in the transfer windows due to the wire resistances surpass the gains in power harvest with higher primary currents. The curves will eventually converge to zero as t_{SAT} becomes zero.

The TWA method enhances the amount of power harvested from the core com-

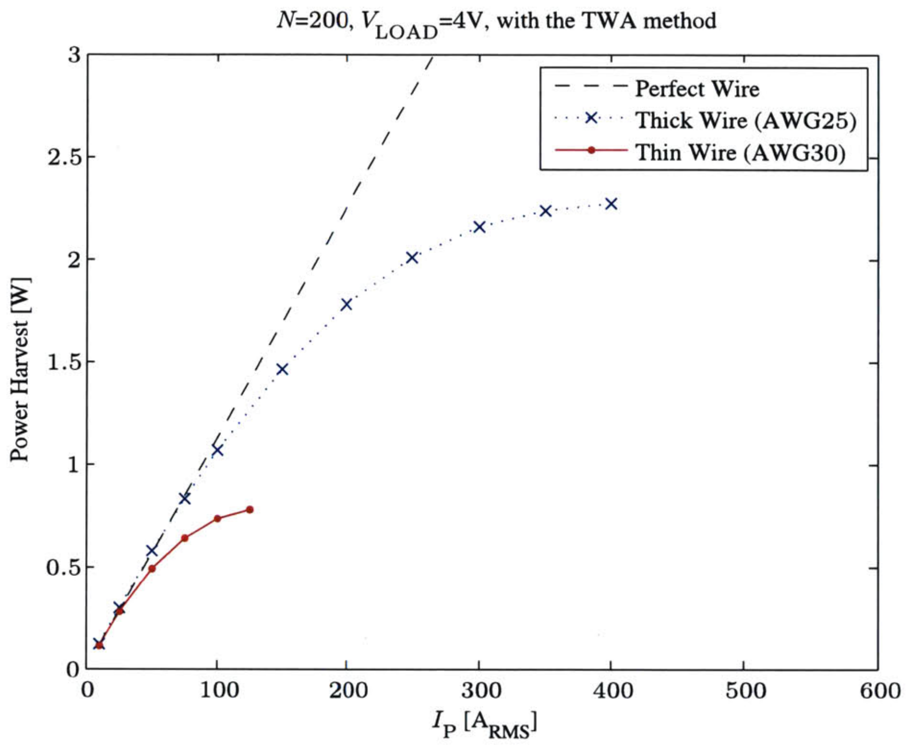


Figure 3-26: Power Harvest Comparison with Different DC Wire Loss under High Power Primary Condition

pared to the base case with passive rectification, assuming low primary current levels. However, power harvest with the TWA method can decrease to zero with higher primary currents while passive rectification guarantees higher power harvest with higher primary currents. Therefore, the curves of harvested power using two methods will intersect. Using a much wider I_{P} range, the levels of energy extraction for both passive rectification and the TWA method are depicted in Fig. 3-27. This simulation result shows the aforementioned point where the passive rectification surpasses the TWA method, which occurs at $I_{\text{P-critical}} = 425.4\,\text{A}_{\text{RMS}}$. Denoting the lengths of the transfer windows in passive rectification and the TWA method as t_{SAT1} and t_{SAT2}, respectively, the point can be estimated by combining (3.18), (3.20), and the following equation that forces the same power level at the intersection:

$$\frac{2}{T}\int_{0}^{t_{\text{SAT1}}} V_{\text{LOAD}} \cdot \frac{I_{\text{P}}}{N}\sin(\omega\, t)\ \text{d}t = \frac{2}{T}\int_{\frac{T}{4}-\frac{t_{\text{SAT2}}}{2}}^{\frac{T}{4}+\frac{t_{\text{SAT2}}}{2}} V_{\text{LOAD}} \cdot \frac{I_{\text{P}}}{N}\sin(\omega\, t)\ \text{d}t \tag{3.21}$$

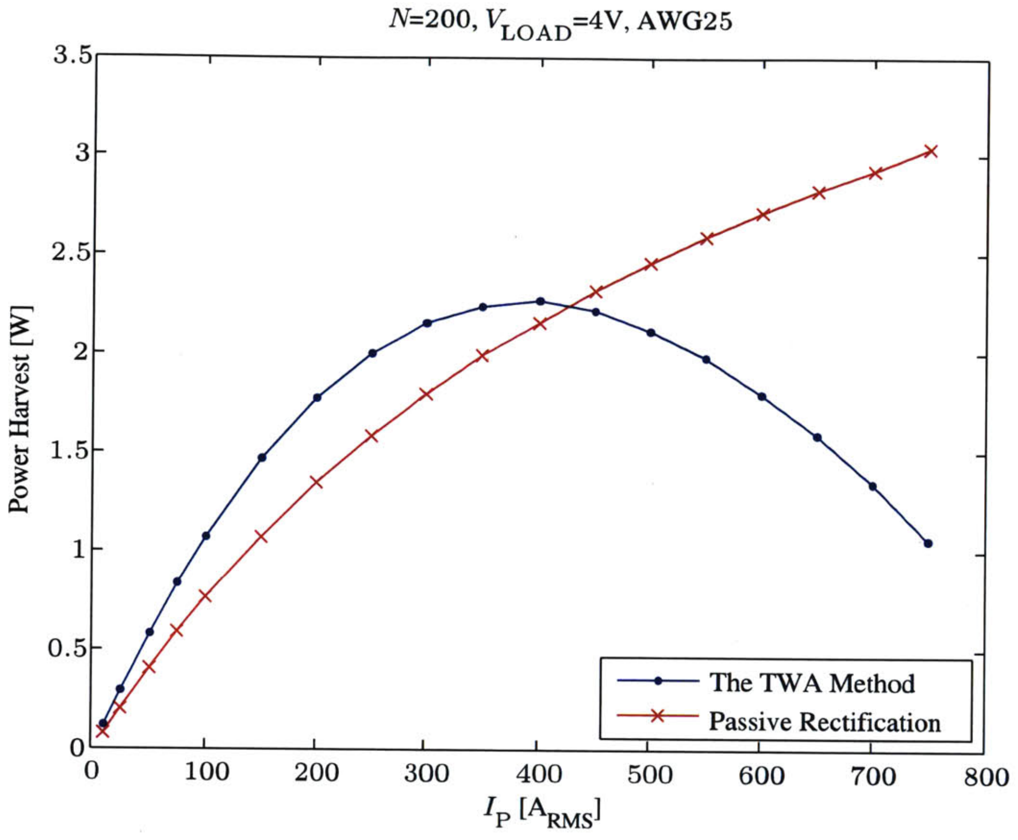


Figure 3-27: Nonideal effects on the TWA method under high power primary condition

When simplified, (3.21) becomes:

$$1 - \cos(\omega\, t_{\text{SAT1}}) = 2\sin\left(\frac{\omega\, t_{\text{SAT2}}}{2}\right) \tag{3.22}$$

Note that since the solution of (3.18), (3.20), and (3.22) does not come in an explicit form, it has to be numerically solved. The intersection of the two curves of Fig. 3-27 is $I_{\text{P-critical}} = 478.29\,\text{A}_{\text{RMS}}$ by solving three equations. Once again, this differs from the accurate circuit simulation of $425.4\,\text{A}_{\text{RMS}}$, but gives a fairly good estimate for quick design targets as a first order hand calculation. After this critical level, there is no gain by using the TWA method anymore, and the best energy extraction will be again by setting zero shift for the transfer window from passive rectification because it minimizes the flux loss in other elements than the load voltage.

Wire Considerations for Lower Power Loss

For a given primary current profile, the thickness of the windings can be increased to lower the power loss due to the resistive element in the flux balance. A tradeoff exists as the thickness of the windings can reduce the resistance, but also reduce the center window area of the core that the primary wire goes through. To provide a way to calculate wire thickness, the following two equations can be set up:

$$p = \frac{\frac{R_{\mathrm{DC}}\, I_{\mathrm{P}}}{\omega N}\left[1 + \sin\left(\frac{\omega\, t_{\mathrm{SAT}}}{2}\right)\right]}{2\, B_{\mathrm{SAT}} \cdot A_{\mathrm{CORE}} \cdot N} \tag{3.23}$$

and

$$(1-p) = \frac{V_{\mathrm{LOAD}} \cdot t_{\mathrm{SAT}}}{2\, B_{\mathrm{SAT}} \cdot A_{\mathrm{CORE}} \cdot N} \tag{3.24}$$

Here, p represents the ratio of the flux accumulation due to the R_{DC} part in the flux balance to the total allowable flux accumulation. The rest of the allowed flux, $(1-p)$, is assigned to the V_{LOAD} part. Combining two equations can eliminate t_{SAT} and yield the following equation for the R_{DC} for a given I_{P}:

$$\frac{R_{\mathrm{DC}}}{N} = \frac{p \cdot 2\, \omega\, B_{\mathrm{SAT}}\, A_{\mathrm{CORE}}\, N}{I_{\mathrm{P}} \cdot \left[1 + \sin\left(\frac{(1-p)\, \omega\, B_{\mathrm{SAT}}\, A_{\mathrm{CORE}}\, N}{V_{\mathrm{LOAD}}}\right)\right]} \tag{3.25}$$

For example, allowing 10% for parasitic flux accumulation ($p = 0.1$), R_{DC}/N should be lower than $2.744\,\mathrm{m\Omega}$ (which gives $0.549\,\Omega$ for 200 turns), using the same core parameters as in Table. 3.1, $V_{\mathrm{LOAD}} = 4\,\mathrm{V}$, and $I_{\mathrm{P}} = 100\,\mathrm{A_{RMS}}$. The per-turn resistance value and the size of the core indicate that the cross-sectional area of the wire must be larger than $0.244\,\mathrm{mm}^2$ as each turn is approximately 4 cm long in this example. The design decision will be then AWG23 for the wire. The remaining question is how to select p. One aspect is the allowable heat dissipation for the wire or the temperature limit of the harvester system as p is directly coupled with power loss in the wire. In our experiences, heat or temperature issue results in less conservative p, compared to the efforts to keep the level of power harvest and t_{SAT} not deviating too much from the ideal expectations. Our suggestion is to use $p = 10\%$ if the heat and temperature issues are verified to be secondary constraints.

If the core is already wound with a certain wire gauge without knowledge of the primary side, the maximum I_{P} that the constructed core can operate without being inefficient or incapable can be also calculated by swapping the positions of R_{DC}/N and I_{P} in (3.25).

Suggested Control Strategy

Assume a case where the level of the primary side current changes much over time, for example, periodic speed control of a motor. The TWA method and the FSC method can be useful for enhancing energy extraction, but the TWA method can be more efficient in terms of physical space and easier control with varying primary currents. One limitation is that the TWA method is only efficient when the primary side carries a relatively lower current, for example, $I_{\mathrm{P}} \leq 400\,\mathrm{A_{RMS}}$ in Fig. 3-27. The power electronic circuits must switch back to the passive mode after I_{P} reaches $I_{\mathrm{P-critical}}$ since the amount of harvested power gets smaller than passive rectification. Furthermore, more power is dissipated in the wire in the TWA method compared to the case with passive rectification, leading to unnecessary heat generation. Figure 3-28 illustrates the simulation results of wire losses in two cases. Assumed are $N = 200$, $V_{\mathrm{LOAD}} = 4\,\mathrm{V}$, and the wire gauge of AWG25. Two curves marked with solid dots illustrate the total power drawn from the primary side. The TWA method clearly extracts more power from the primary side. However, the wire loss curves, marked with hollow circles, show that the TWA method loses a significant portion of the extracted power from the primary side as I_{P} increases whereas the wire loss in passive rectification increases at a slower rate. To give a perspective, two curves from Fig. 3-27, the actual power harvested in the secondary side, are overlaid in Fig. 3-28, marked with 'x'. The wire loss in the TWA method surpasses the actual power harvest from the load even before $I_{\mathrm{P-critical}}$ occurs. It can be expected that as the harvested power with the TWA method approaches zero, the wire loss will be indistinguishable from the power extracted from the primary side.

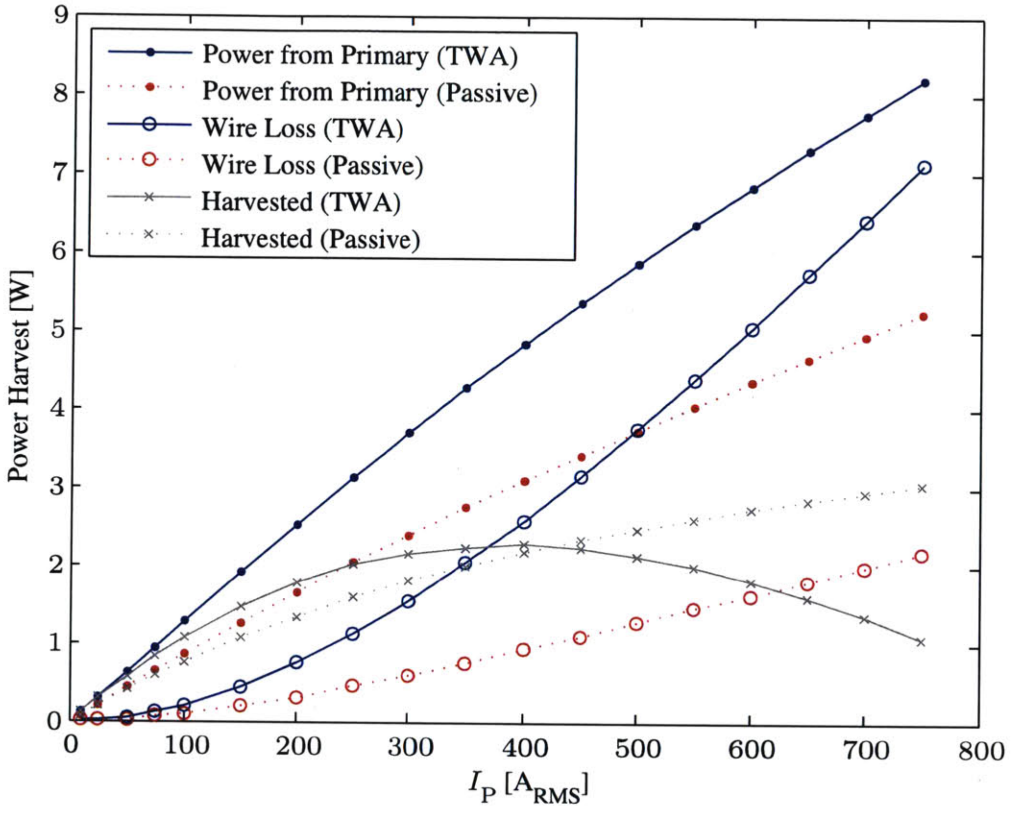


Figure 3-28: Wire Loss Comparison between the TWA Method and Passive Rectification

Due to the drawbacks that the TWA method has, it might be realistic to switch the circuit mode before $I_{P-\text{critical}}$ occurs or before the wire loss dominates. The required condition is that the harvester operating with passive rectification can extract more than the load power consumption of the sensor node, usually hundreds of mW for most of the sensor applications. It will benefit both the sensor node and the primary side by decreasing the heat generation and eliminating unnecessary power loss. The suggested control scheme is illustrated in Fig. 3-29 as a solid line with hollow circles. Here, the load power consumption with some margin, depicted as the dashed line, is set at 1.5 W. The circuit that originally operates with the TWA method changes the mode to passive rectification when the level of the primary current is higher than the threshold, which is 233.07 A_{RMS} for this simulation. More dynamically, the control for the mode switch can be autonomous with the microcontroller. By detecting a positive change in voltage of the supercapacitor over a multiple of operating cycles, it can be

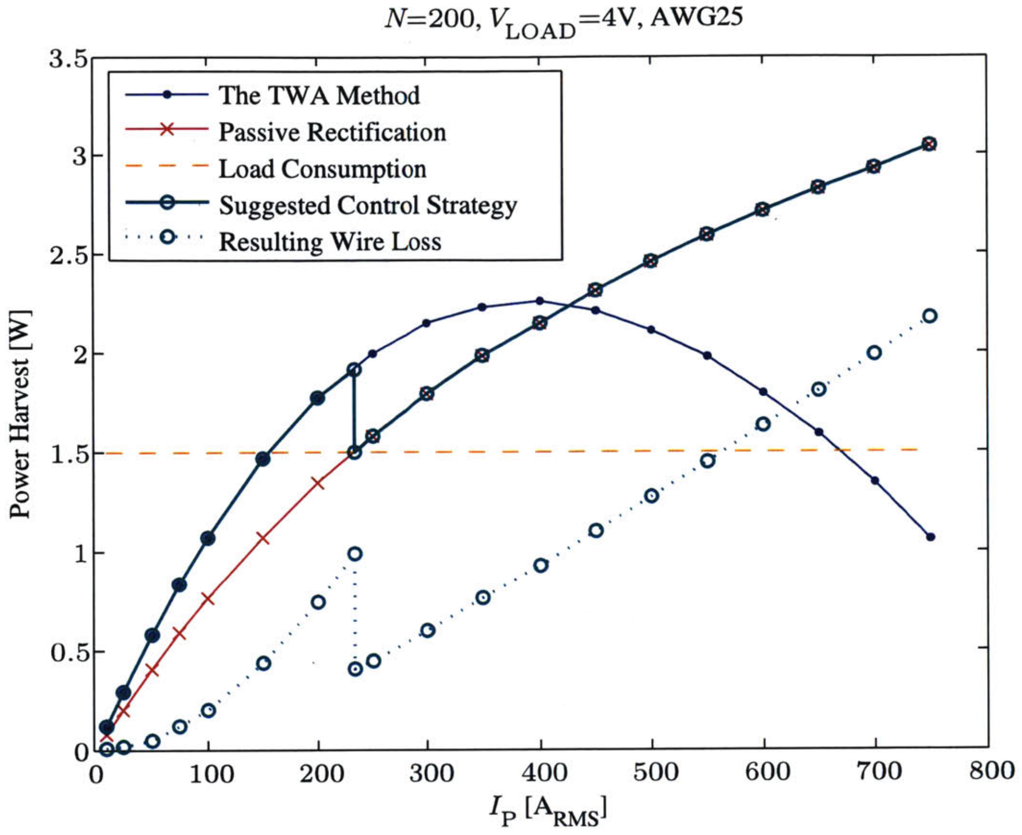


Figure 3-29: Suggested Control Strategy and the Resulting Wire Loss

verified that the TWA method generates more than what the load consumes. Once this is detected, passive rectification can be sparsely tried until a positive change in voltage of the supercapacitor is detected with passive rectification as well. With this dynamic approach, no preset of the threshold level in the primary current is required.

3.4 Chapter Summary

This chapter has presented another core model for accounting for saturation behaviors in electromagnetic energy harvesters. This model is different from the Maxwell model from Chapter 2.6 in that it does not require a custom numerical solver and is suitable for direct use in circuit simulators like SPICE. Compared to the Maxwell model, appropriate model simplifications ease computational burden, and the model continues to show excellent accuracy across various I_P and N configurations, and on two different load types. This remains true even for cores with a large ratio of outer radius to inner radius.

Two new techniques to enhance the extraction of energy from a magnetic core are introduced and verified through analyses, simulation, and experiments. The first involves a flux-shaping capacitor. This capacitor, in series with the core, shapes the flux accumulation of the core in a convex manner with a lower starting point, lengthening the transfer window. The second approach uses switches to align the transfer window with the peak transformer current. By exploiting the core's indifference to the starting point of the transfer window, the load can be connected to the core when the load can receive the maximum power during the transfer time segment. Practical control and circuit implementation for the TWA method is also presented to reduce the number of switches and minimize conduction losses that reduces the length of the transfer window. A discussion on an active rectifier is also presented to minimize losses where appropriate for the harvester application. This drastically reduces the switching loss, and realizes the circuit environment as if ideal diodes are used, as done in the transfer window analysis.

In practice, the harvester is likely to face one of three cases for the primary current. First is when the RMS current in the primary side is well-defined, and does not change much. In this case, a fixed flux-shaping capacitor can be used reliably, eliminating control efforts. Second, the primary current may exhibit substantial change over time, and a microcontroller is likely to be available, either as standalone or as part of the sensor package powered by the harvester. In this case, the TWA method permits adaptive control of the transfer window to optimize power transfer, much like finding the maximum power point for a solar panel. Of course, if a controller cannot be deployed, a basic rectifier with two diodes and two cross-coupled FETs can be used without enhancement to harvest some power. Combining the active rectification with any of the two methods, FSC or TWA, may also be desirable for higher power harvest. Moreover, for high power applications, the thickness of the core windings must be carefully considered so that the width of the transfer window is not much affected by the level of the primary current.

Finally, a new flux balance model has been introduced to incorporate the issue that occurs in the most fundamental flux equality when there is a significant flux loss due to parasitic elements. As a representative example of such a parasitic flux loss, DC wire resistance in a high power primary case was analyzed. This chapter demonstrated why the original flux equality is insufficient for circuits with high primary current cases, and presented a new set of equations for more accurate hand analyses. The performance of the new flux balance model is proven to be more accurate than the simple flux balance model. Deviations in the levels of power harvest from ideal expectations are analyzed and the implications in the design optimization of the TWA method are explored with the new model. The most important points with the new flux model are that t_{SAT} is no longer independent of the primary side and that zero t_{SAT} is possible with the TWA method. This indicates that a higher primary current can actually result in lower power harvest under certain circumstances, and in extreme cases the core can never extract energy at all. This chapter also expands this phenomenon to explain why any shifting of the transfer window from passive rectification results in suboptimal power harvest with high primary current levels. This chapter presented a way to calculate the critical level in the primary current where the TWA method performs inferior to passive rectification. It also provided an equation to determine wire thickness given the primary current profile. The control strategy with a varying current level in the primary side is presented for better power harvest and lower power loss.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Self-powered Embedded System for Electromechanical Diagnosis

This chapter describes the construction of a self-powered embedded system based on the electromagnetic energy harvester. As an exemplary application to a self-powered embedded system, a vibration monitoring point with integrated recovery of energy (VAMPIRE) is constructed for electromechanical failure detection, specifically for rotational machines, e.g., motors. This chapter covers many topics, such as digital circuit designs, power circuit designs, software designs, and signal processing. The main goal of the self-powered embedded system illustrated here is to provide a capability of condition based maintenance for machines instead of failure based maintenance. VAMPIRE is supported by the electromagnetic energy harvester that has been discussed so far, and powers the sensor package to collect the vibration data from an accelerometer and rotor speed data from a back-EMF voltage sensor during steady-state and spin-down events. The collected data can be fetched through Bluetooth and WIFI connections from VAMPIRE, and signal processing on the data provides actionable information on the machine's health and necessity for maintenance.

4.1 Architecture

The overall architecture of VAMPIRE is presented in Fig. 4-1.

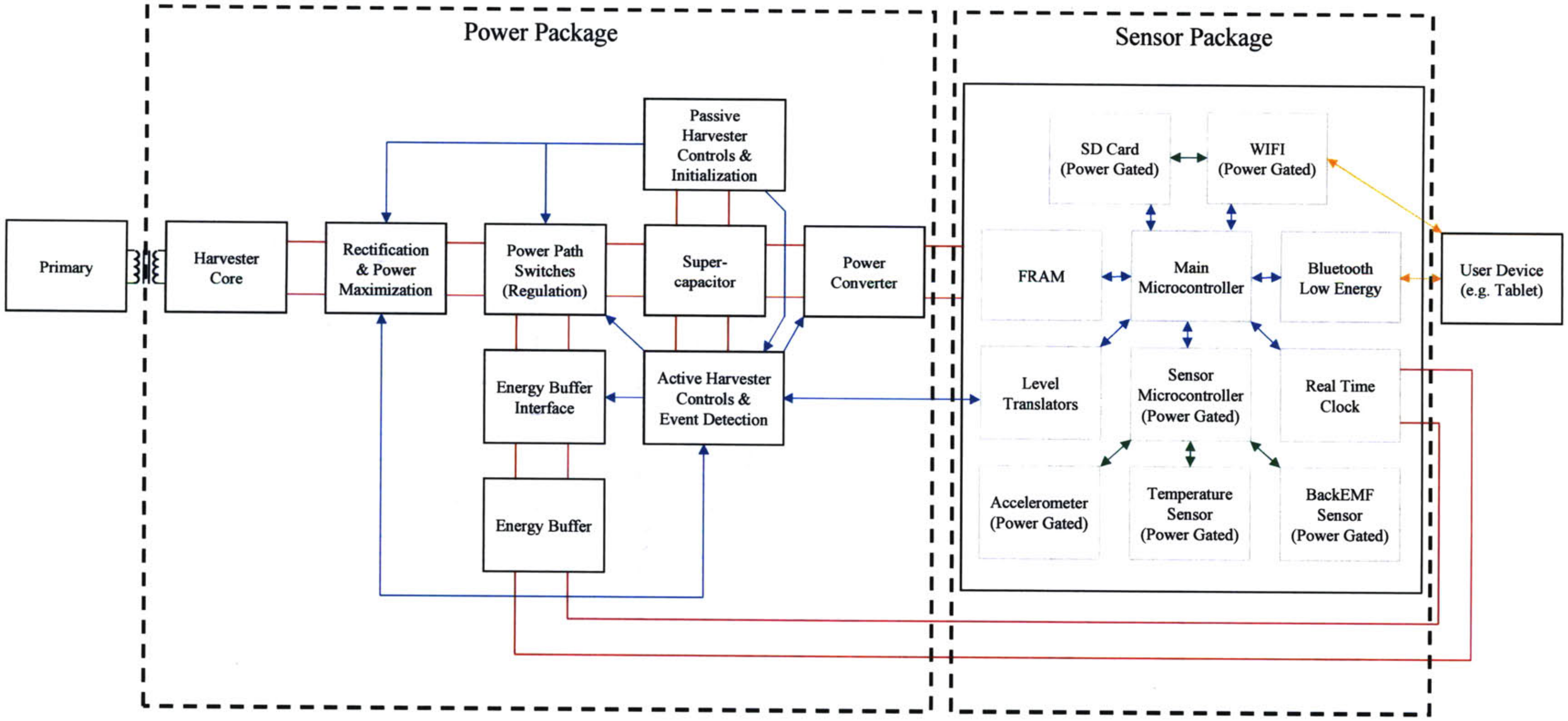

Power Package
Sensor Package
Primary
Harvester Core
Rectification & Power Maximization
Power Path Switches (Regulation)
Passive Harvester Controls & Initialization
Super-capacitor
Power Converter
Energy Buffer Interface
Active Harvester Controls & Event Detection
Energy Buffer
SD Card (Power Gated)
WIFI (Power Gated)
FRAM
Main Microcontroller
Bluetooth Low Energy
User Device (e.g. Tablet)
Level Translators
Sensor Microcontroller (Power Gated)
Real Time Clock
Accelerometer (Power Gated)
Temperature Sensor (Power Gated)
BackEMF Sensor (Power Gated)

Figure 4-1: Overall Architecture of VAMPIRE

The red lines between two blocks without arrows indicate the power rails. The blue lines with arrows indicate control and data channels. The green arrows also indicate control and data channels, but designate autonomous data management wrapped by an additional layer of a microcontroller. The yellow lines indicate wireless channels. Largely, there are two packages: the power package and the sensor package, represented by dashed boxes on the left and the right. In the subsequent sections, the operations of each package is explored.

4.2 Energy Harvester and Power Processing Package

4.2.1 Electromagnetic Energy Harvester

The primary side that carries the AC current, i.e., a rotating machine, provides the harvester with magnetic fields to extract energy from electromagnetic coupling. The harvested current from the core is generally a distorted nonlinear AC current, considering the core experiences timed magnetic saturation every cycle. This distorted AC current goes through alternating sets of switches for power processing, which is illustrated as 'Rectification & Power Maximization' in Fig. 4-1. These switches and the core connection are presented in Fig. 3-20. The harvested AC current gets rectified and at the same time the power extraction from the core is maximized by performing the TWA technique. The average harvested current is maximized by aligning the center of the transfer window with the peak of the primary sinusoid.

In this embedded system design, the same amorphous nanocrystalline core, VAC VITROPERM 500F W380, is used. The FSC method is not considered as VAMPIRE is intended for a wide range of motor currents. The harvester core is wound 200 turns with AWG23 wire, and expects a single primary winding through the center of the magnetic core for electromagnetic energy extraction. The AWG23 wire provides a relatively ideal response in terms of DC wire loss, as calculated in Chapter. 3.3.4, up to at least $I_{\mathrm{P}} = 100\,\mathrm{A}_{\mathrm{RMS}}$, which is the maximum design target of VAMPIRE.

This maximally harvested current is primarily stored in two supercapacitors connected in series, where the voltage rating of each supercapacitor (PAS0815 by Taiyo Yuden [44]) is 2.5 V and the capacitance is 1 F each. The nominal operating voltage, V_{LOAD}, for the power package is set for 4.0 V. This achieves $J = 0.8$ for the TWA method, and it guarantees approximately 90.0% of the asymptotic maximum power. Note that electromagnetic energy harvest happens only when the primary side carries current, e.g., a motor is running. If a motor is in a spin-down process by turning it off, all the ongoing operations, such as sensing, control, or signal processing, cannot be sustained by the electromagnetic energy harvester directly. It brings up the necessity for an energy buffer as introduced in Chapter. 1.

4.2.2 Power Flow

The maximally harvested current from an electromagnetic coupling using the TWA method and active rectification first goes through the power path switches, and determines whether to charge up the supercapacitors that power the digital package, shown as 'Supercapacitor' in Fig. 4-1, or the energy buffer, shown as 'Energy Buffer' in the figure, for sustenance during spin-down events. However, since this is a self-powered embedded system that can start from the zero energy state, the passive stage must be considered first.

Passive Control Phase

All the active paths that require active switchings, e.g., power maximization, regulation, and path controls, are disabled when VAMPIRE boots up for the first time. In this passive stage, the unoptimized harvested current is steered to the 'Supercapacitor' block through a full bridge diode rectifier, which can be body diodes of the FETs. The voltage of the 'Supercapacitor' block increases at a slower rate due to lack of active switching and the TWA method.

The voltage of the supercapacitor is passively monitored all the time, using a hysteresis amplifier. Once the voltage of the supercapacitor reaches the minimum voltage plus some margin for a microcontroller inside the harvester, the microcon-

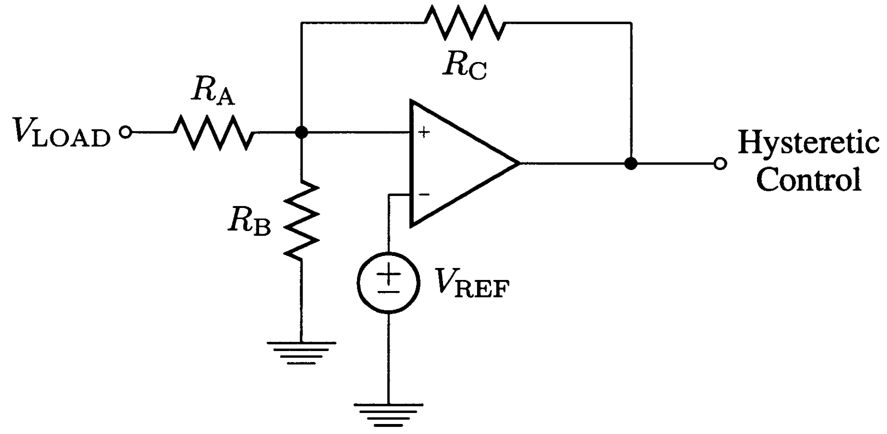


Figure 4-2: Hysteresis Amplifier for Passive Controls

troller is finally powered on. The passive hysteresis amplifier is used to provide a higher turn-on voltage and a lower turn-off voltage for the microcontroller. If this margin between the turn-on and off voltages is not sufficient, the initial surge current of the microcontroller causes the voltage of the supercapacitor to momentarily drop below the turn-off voltage, and the microcontroller will be stuck in this turn-on and reset cycle. As an extreme example, consider an operational amplifier (opamp) with a 1.8 V reference and zero hysteresis band. As soon as the voltage of the supercapacitor reaches 1.8 V, the opamp outputs high, and the microcontroller is turned on, drawing current from the supercapacitor. The wire resistance and the effective series resistance (ESR) of the supercapacitor combined with the surge current instantly lowers the voltage of the supercapacitor below 1.8 V, and the opamp outputs zero, turning off the microcontroller. As soon as the microcontroller is turned off, there is no current drawn from the supercapacitor. It instantly recovers 1.8 V, and the opamp again outputs high, turning on the microcontroller again. The cycle repeats, never moving past 1.8 V. An example of a hysteresis amplifier is presented in Fig. 4-2. The turn-on and turn-off voltages of the hysteresis band are:

$$\begin{aligned} V_{\text{Turn-On}} &= V_{\text{REF}} \cdot \left(\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}\right) \cdot R_1 \\ V_{\text{Turn-Off}} &= V_{\text{REF}} \cdot \left(\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}\right) \cdot (R_1 // R_3) \end{aligned} \tag{4.1}$$

In VAMPIRE, the hysteresis band for the microcontroller inside the power package is set to 80 mV. This assumes the maximum surge current of 40 mA for 1 s during the microcontroller initialization in the power package for the worst case. If the hysteresis band is small enough, e.g., 50 mV, many opamps with internal hysteresis can be used without an explicit hysteresis amplifiers as above. For the real circuits, a small capacitor on the order of nF can be added in parallel with R_B for glitch filtering. Another useful tip is that by powering the comparator with a device resetter or an IC supervisor, the lower hysteresis band can be forced, easing the circuit constraints.

While in the passive stage, all the gates of the active switches must be firmly held externally such that the switches remain completely turned off. It is not desirable to expect the outputs of the microcontroller to be able to firmly hold the gates of the switches before the microcontroller is powered on. To guarantee the complete turn-off of active switches, there are another set of switches to initially bias the gates of the active switches with supply rails. They could have been resistors with very large resistance, however, this approach is taken to reduce the quiescent current. To avoid floating nodes during the change of the controller (from the initial biasing switches to the microcontroller), the initial biasing switches still maintain the biasing even after the microcontroller comes alive at 1.98 V. The microcontroller does not provide active switchings immediately. It initializes all the active switching control outputs to the same value as the initial biasing levels to avoid short-circuiting current. The initial biasing switches are released from the circuit by the second hysteresis amplifier that detects a little higher voltage level, e.g., 2.07 V in this design. Hereinafter, the microcontroller provides active switchings, drastically improving energy extraction.

Active Control Phase

After the passive control phase, the microcontroller in the power package actively controls the switches. However, if VAMPIRE is booting up for the first time, V_{LOAD} around 2.07 V is still far from the nominal operating voltage of 4.0 V. Until that point, the regulation and the power path controls are not relevant, and the first priority is to charge up the 'Supercapacitor' block. With the TWA method and active rectification in place, the voltage of the 'Supercapacitor' block increases much faster. Once it goes beyond the nominal operating voltage, the digital sensor package is powered on. For the same reason discussed before, there is a small hysteresis band for turning on the digital sensor package for the first time. VAMPIRE sets the turn-on voltage of the sensor package to 4.1 V for the sensor package. After the sensor package is operational, the voltage of the 'Supercapacitor' block is regulated at 4.0 V with the maximum peak to peak ripple of 10 mV. When the supercapacitor is regulated, the harvested current can be routed to the energy buffer. The stored energy can be used later for spin-down events. When the energy buffer is also fully energized so that it cannot take any more power influx, the harvested current should be able to go back to the primary side without being sacrificed.

Active Control Phase - Energy Buffer and Supercapacitor

When the supercapacitor reaches its nominal operating voltage of 4.0 V, the harvested current is routed to other paths. One of them is the energy buffer. In this embedded system, the energy buffer is a 220 mAh Lithium polymer (LiPo) battery. LiPo batteries have much higher energy density than supercapacitors, much more suited for the size constrained designs like VAMPIRE. For example, sustaining 200 mA for 30 s requires 60 F if allowing the voltage drop of 100 mV. This implies 120 supercapacitors, identical to the ones used in the 'Supercapacitor' block, are required, which is simply too large in volume for VAMPIRE to be integrated inside motor's control box. As will be explained later, 200 mA is the current approximately required for sustaining the WIFI module inside the sensor package.

The energy buffer that comes with a huge amount of prepacked energy, like LiPo batteries, must be used with a strict rule: the net energy into the battery must be nonnegative between operations as this is a self-powered embedded system. As long as the nonnegative energy flow can be guaranteed, higher energy capacity is preferred for the battery as it can increase the operating time of the embedded system after a spin-down event. However, the energy capacity and the size of the battery are in a trade-off relationship. The main energy storage, 'Supercapacitor', on the other hand, cannot be swapped with a LiPo battery for a longer operation time, because the time to charge up the main energy storage to boot up the system takes a significantly longer time. For example, charging a fully discharged 0.5 F supercapacitor to 4.2 V requires 105 seconds with 20 mA current whereas charging a fully discharged 220 mAh LiPO battery to 4.2 V requires 11 hours with the same current. For the main energy storage, the voltage ripple due to charge loss during the operation is traded off with a fast boot time. However, the voltage ripple for 'Supercapacitor' is also very low already. The effective capacitance of two 1 F supercapacitors connected in series is 0.5 F. The effective series resistance of two supercapacitors in series is 140 mΩ. Assuming the nominal current consumption of 10 mA for the sensor package, the voltage ripple during one half sine wave of the line cycle is:

$$\Delta V = \frac{I_{\text{NOMINAL}}}{C_{\text{SUPERCAP}} \cdot 2\,\text{freq}} + I_{\text{NOMINAL}} \cdot R_{\text{ESR}} = 0.157\,\text{mV} \tag{4.2}$$

Power Path Controls

The power path control circuit for the harvested current is presented in Fig. 4-3. When VAMPIRE boots up for the first time, passive controls are generated by the aforementioned hysteresis amplifiers such that $A = \phi_1 = \phi_2 = 0$ and $\overline{A} = \phi_3 = \phi_4 = \overline{\mu B} = \overline{\mu C} = \overline{\mu E} = 1$. The harvested current is handled by the diode full bridge rectifier and routed to the supercapacitors. Small diodes right next to FETs in the figure represent the body diodes of the FETs, and larger diodes represent explicit Schottky diodes. Signal A (and $\overline{A}$) implies 'Active' switching mode, which is asserted high a little after the microcontroller is operational.

When the microcontroller comes alive and the path control is handed over to the microcontroller (after asserting A high), active switchings are available. Signals ϕ_1, ϕ_2, ϕ_3, and ϕ_4 are controlled according to the TWA method, and $\overline{\mu C}$ is driven low for better energy extraction whereas $\overline{\mu B}$ and $\overline{\mu E}$ are driven high to prevent unwanted current paths. When the supercapacitor is sufficiently charged, the sensor package is turned on, and the active power path controls are initiated.

In every line cycle, the voltage of the supercapacitor is first checked. When it goes

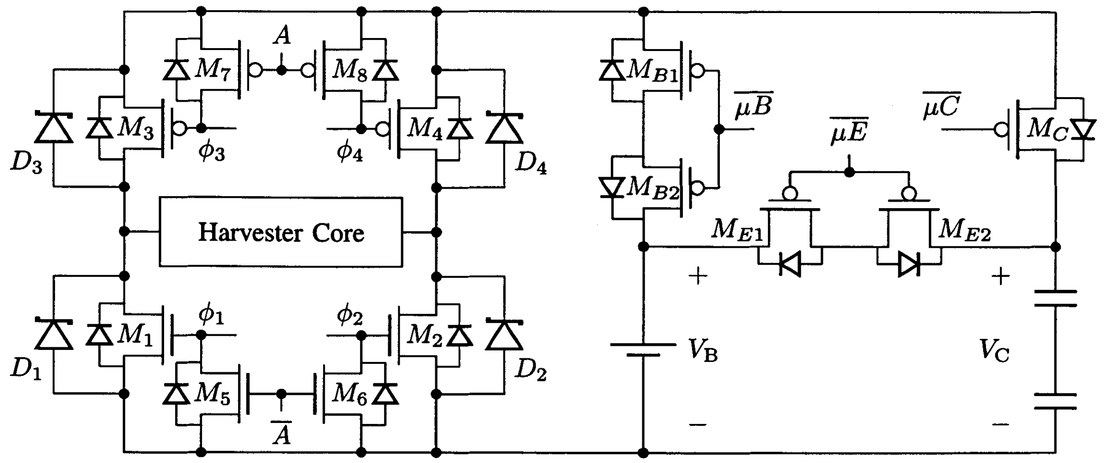


Figure 4-3: Power Path Control Circuit

beyond 4.0 V, the harvester tries to regulate the voltage by rerouting the harvested current into other paths. The next decision is made by checking the voltage of the energy buffer, LiPo battery in VAMPIRE. If the voltage of the energy buffer is less than its predefined maximum voltage, for example, 4.20 V for VAMPIRE's LiPo battery, the energy buffer receives the harvested current by setting $\overline{\mu C} = 1, \overline{\mu B} = 0$, and $\overline{\mu E} = 1$. The LiPo battery is constantly charged up.

When both the supercapacitor block and the energy buffer are at their maximum voltage, the harvested current is rerouted back to the primary side by shorting the core. The lower NFETs, M_1 and M_2, are both turned on to provide a short-circuit path for the core. At the same time, the upper PFETs, M_3 and M_4 are turned off at the same time to isolate the energy storages. Signal $\overline{\mu E}$ is to provide a current path between the battery and the supercapacitor block through M_{E1} and M_{E2} in case of a spin-down event where the electromagnetic energy harvester no longer generates energy. The battery releases the newly acquired nonnegative energy into the rest of the circuit to sustain required operations during the spin-down event. This is automatically enabled once a spin-down event is detected. This can be also controlled by user input. Since there are two different energy sources with potentially different voltage levels in general, the 'Supercapacitor' block and the 'Energy Buffer' block, the FETs, voltage-controlled gate switches, should be controlled by the highest voltage or

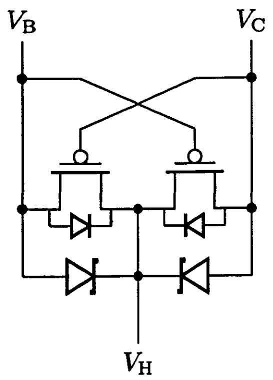


Figure 4-4: High Voltage Selection Circuit

the lowest voltage existing in the circuit to prevent unwanted current paths, depending on the type of the FETs. Since two sources share the same ground, the remaining consideration is PFET controls. Due to this reason, a circuit for selecting the highest voltage is used and illustrated in Fig. 4-4, where V_B and V_C are the inputs and V_H is the selected high voltage. The cross-coupled PFETs ensure the final voltage error is close to zero, and two Schottky diodes ensure fast tracking when the difference between V_B and V_C is small. For a space constrained PCB design, omission of the cross-coupled PFETs is preferred as Schottky diodes generally perform better in terms of reliability over wide voltage difference.

In the electromagnetic energy harvester, the battery charging circuit can be as simple as two switches with $\overline{\mu B}$. This is because the core serves as a current source. The number of windings on the core and the maximum current on the primary side determines the peak current that the battery will be subject to. Since the maximum charging rate of the battery is proportional to its capacity, the maximum current from the harvester must be considered before determining the battery capacity. For VAMPIRE, a 220 mAh LiPo battery with the maximum charging rate of 2C is chosen. It is small enough to be easily enclosed inside the VAMPIRE structure, and it can withstand the primary side current upto 97.7 $\mathrm{A_{RMS}}$ to be under its 2C rating with $N = 200$. The maximum primary current for a battery capacity is calculated by:

$$I_{\mathrm{P-RMS}} = \frac{2\,\mathrm{Capacity} \times N}{\sqrt{2}} \cdot \frac{\pi}{2} \tag{4.3}$$

The constraint 97.7 $\mathrm{A_{RMS}}$ also closely match the 100 $\mathrm{A_{RMS}}$ limitation for the wire gauage calculation.

Spin-down Event Detection

The magnetic core in the harvester is essentially a current source while it is not saturated, developing any voltage that is required by the environment to sustain a current path into the load. Whenever there is a nonzero current going into the load, whether the load being the supercapacitor block or the energy buffer, the core voltage is instantly developed to the full load voltage. Therefore, by monitoring the terminal voltages of the core as digital signals, the harvester microcontroller can easily determine the current direction, as well as the location of the zero crossings. Using passive rectification, a zero crossing coincides with a rising edge in either terminal voltages. The TWA method can potentially obscure the location of a zero crossing, however, since VAMPIRE employs a zero crossing training every cycle, the location of the zero crossing with the TWA method is already available at the expense of more control efforts. The detailed operation is discussed in Chapter. 3.2.5. As soon as the harvester microcontroller obtains the location of the most recent zero crossing, it estimates the location of the transfer window for power maximization as well as the location of the next zero crossing based on the previous history. If the next zero crossing is not detected at this estimated time point within some time margin, the harvester learns that the primary side has lost its electrical power, and asserts a signal to the sensor package to indicate the initiation of the spin-down event.

The power rail of the power package is regulated at 4.0 V whereas the sensor package, as will be explained shortly, employs 3.3 V. The indication signal does not require a considerable amount of power, so a simple resistive divider is used for level translation in this case. This also indicates that the power converter is required for the sensor package. Since the supercapacitor block is well regulated within a mV range, a simple buck converter with the fixed-duty cycle of 0.825 suffices.

4.3 Sensor and Wireless Communication Package

4.3.1 Sensors

When the digital sensor package is turned on and fully operational, the sensors acquire data at a predefined sample rate upto 2 kHz. In the VAMPIRE prototype, the accelerometer and the back-EMF sensor are sampled at the full 2 kHz, and the temperature is sampled at 20 Hz. They are all connected in Serial Peripheral Interface (SPI) as slaves to the sensor microcontroller. The sensor microcontroller and the sensors are power gated for lower power consumption, but their default state is on. The accelerometer measures the vibration of a motor. A relative vibrational energy can be inferred from this data. The back-EMF sensor measures the back-EMF voltage appearing across the phase wires of the motor by a capacitive coupling during a motor spin-down. The back-EMF voltage data is used to infer the rotational speed of the motor. More detailed illustration of the sensor package is presented in Fig. 4-5.

Data Acquisition

The accelerometer, BMA250 by Bosch [45], reports 3-axis acceleration data, each axis in 10-bit resolution. The back-EMF sensor generates an analog voltage, and is sampled with a 10-bit ADC inside the sensor microcontroller. The temperature sensor reports 8-bit data. Each fast sample packet (2 kHz) is 5 bytes (40 bits), and 1 byte (8-bit) of temperature data is added every 100 fast sample packets. The sensor microcontroller generates an interrupt once it accumulated 501 bytes, which occurs every 20 ms. The sensor microcontroller has two SPI buses. In one SPI bus, it acts as a master and controls sensors in a slave mode. In the second SPI bus, it acts as a slave and it is controlled by a SPI master, the main microcontroller. Two microcontrollers are layered so that the main microcontroller does not have to stop its state machine every 0.5 ms to sample sensors' data at 2 kHz. The separation of sampling layer effectively transforms a low bandwidth with constant data flow to a higher bandwidth with sparse input. The main microcontroller detects the interrupt and imports the 501-byte long data every 20 ms.

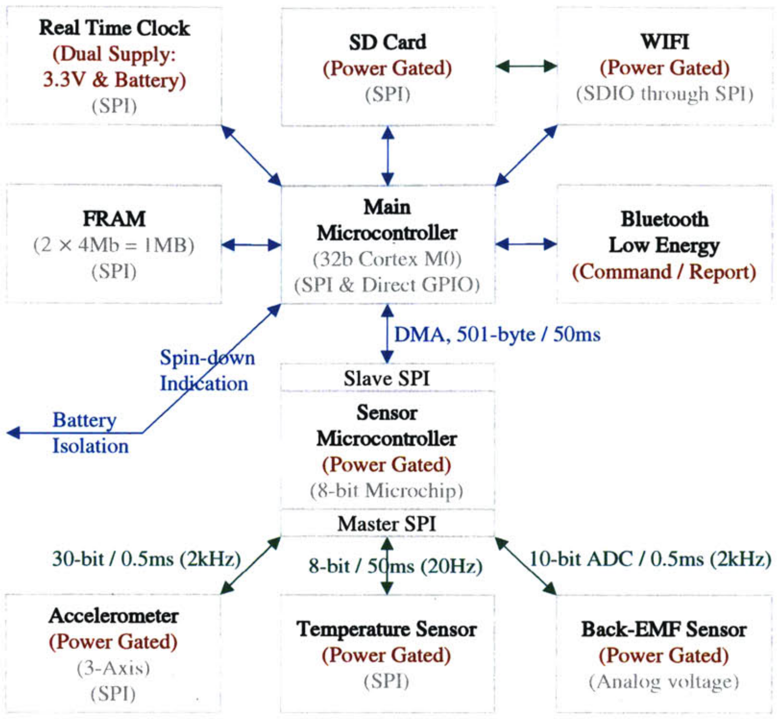


Figure 4-5: Sensor Package Block Diagram

Autonomous Data Flow

The main microcontroller immediately writes the received data into the FRAM block that consists of two 4Mb FRAM modules [46] that are all connected in SPI as slaves to the main microcontroller. It can store approximately 100 seconds worth of data samples. The FRAM block is written in a ring fashion, overwriting the oldest data. The FRAM block also keeps track of the most recently written address. If a spin-down event is detected, the address for the FRAM block to end its ring writing is set. The end address is calculated based on the preset spin-down window length. Once the end address is reached, the entire 100 s of recording is transferred to the more spacious storage, the SD card. The SD card is also connected in SPI as a slave to the main microcontroller, and it is power gated as it consumes a relatively high amount of power compared to other sensors and microcontrollers. The file system for the SD card is FAT32, and currently 150 data files are supported in the embedded system. To provide file descriptions as accurate as possible, the time stamp for each file is

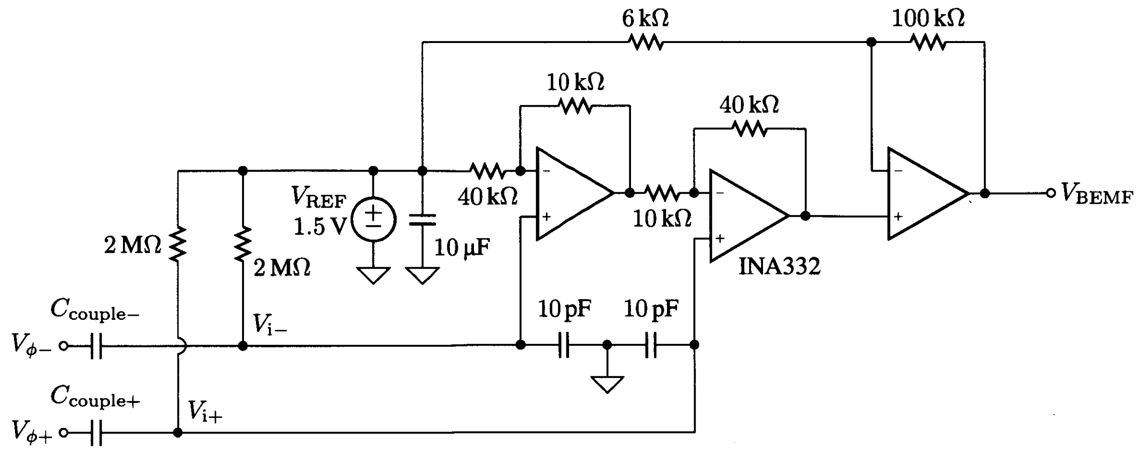


Figure 4-6: Back-EMF Voltage Sensor Circuit

also stored. The time stamp is acquired by querying the real time clock, which is also connected in SPI as a slave to the main microcontroller. Currently, at the full 2 kHz, each spin-down event is contained in a data file of exactly 1MB in the SD card. The SD card is required in the design because an FRAM module is pricey and small in data storage compared to its physical size. FRAM modules are also essential in the design because the SD card consumes much higher power than FRAM modules. Furthermore using the SD card as the primary data storage is inappropriate because it is not adequate for low latency writings, such as 20 ms in between data streams. It necessitates an additional data buffer layer, like FRAM or SRAM.

Back-EMF Voltage Sensor Design

The schematics for the back-EMF voltage sensor are presented in Fig. 4-6. The copper deposits on the PCB, as shown in Fig. 4-7, serve as bottom plates of the capacitors, and the phase wires, $V_{\phi+}$ and $V_{\phi-}$, above them act as top plates of the capacitors. Two capacitors designated as $C_{\mathrm{couple}+}$ and $C_{\mathrm{couple}-}$ in Fig. 4-6 represent these voltage sensing capacitors. The capacitively coupled phase voltages then go through differential amplifiers. Due to the capacitive input to the amplifiers' positive terminals, a differentiator is inherently implied. The physical circuit does not have the integrator, and the integration is done in signal processing to recover the real phase voltages. The gain of the amplifier is set to sufficiently high that the differential

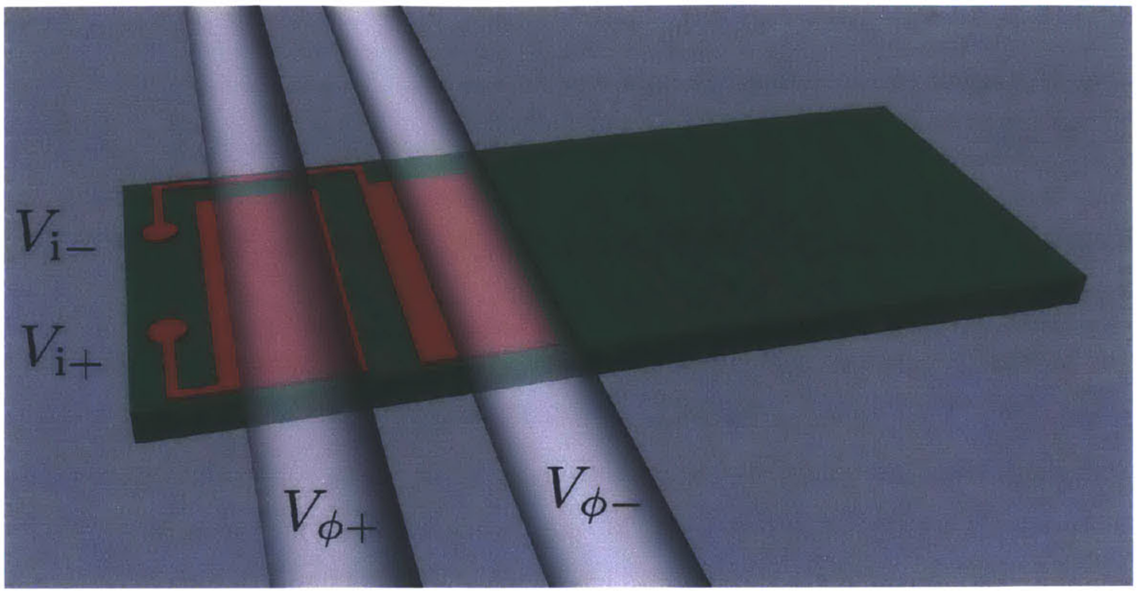


Figure 4-7: Back-EMF Voltage Sensing Structure

inputs are clipped to the rail voltages most of the time. The reason the clipped sinusoidal waveforms do not matter is that the speed of the rotor can still be estimated by tracking the zero crossings in the back-EMF voltage, instead of performing a Hilbert transform on the non-clipped decay of the sinusoidal voltage waveform [1]. This implementation is consisted of low power amplifiers and a low power voltage reference to bias the inputs and outputs of the amplifiers. Unlike the back-EMF design introduced in [47], which requires a high amount of current to operate (around 10 mA), this voltage sensor can be operated with less than 1 mA. The supply voltage that is omitted from the figure is 3.3 V, and the reference voltage, V_{REF}, is 1.5 V.

Spin-down Event Auto Trigger

If a motor loses its electrical power, the current is suddenly stopped and the magnetic fields collapse. The magnetic core of the harvester can no longer extract power from its electromagnetic coupling. Also, high power consumption is expected for the SD card writing. Therefore, as soon as the harvester microcontroller asserts the spin-down state, the energy buffer must be connected to prevent the supercapacitor block from over-discharged. The harvester controller will automatically provide a current path between the battery and the supercapacitor block by closing the switch M_{E1} and M_{E2} upon the detection of a spin-down event.

Even after the spin-down event is detected, the ongoing writing of the sensor data into the FRAM block is further continued for a pre-specified duration as spin-downs take a few to tens of seconds. When the end of the window is reached, the entire FRAM content is automatically transferred to the SD card. During this data move, the power consumption is significantly increased as the SD card requires high current during a write. Since the vibration and the back-EMF data after the completion of the spin-down does not hold a significant meaning, the incoming sensor data packets are not received nor stored into the FRAM block when the main microcontroller is fully commited to finishing the data transfer between the FRAM and the SD card. After the recent spin-down event is recorded as a file in the SD card, the real-time clock is queried, and the time-stamp is also stored. Then, the harvester microcontroller goes into the state where it looks for a rising edge in either of the core terminals to detect the primary power-on again, and the main microcontroller idles.

4.3.2 Interface between the Power Package and the Sensor Package

The current prototype exchanges two signals between the harvester and the digital sensor package. One signal is transmitted from the harvester microcontroller to pass the information that the primary side is turned off, indicating the initiation of a spin-down event. The other signal is transmitted from the main microcontroller to request a disconnection of the energy buffer from the supercapacitor, indicating that VAMPIRE now needs to completely shut off for power saving. For example, the disconnection request will be made by a technician with a tablet after he retrieves all the required data for analysis. As discussed, any signal sent to the digital sensor package from the harvester requires a voltage translation through a resistive divider. However, the signal from the sensor package can be directly read by the harvester microcontroller, since they share the same ground, and the supply voltage of the sensor package is lower than the voltage of the supercapacitor but higher than the threshold voltage of a digital high for the harvester microcontroller.

4.3.3 Wireless Communication

The Bluetooth Low Energy (BLE) module is the normal way of communication for VAMPIRE with the external environments. BLE is a standard and prevailing technology at the time of writing. VAMPIRE can be easily accessed by mobile phones, tablets, and laptops. BLE is extremely power efficient. It is not difficult to design a BLE device that consumes a few mW with infrequent message delivery. However, its data bandwidth is not suitable for a big data file transmission. In our experiments, the practical maximum data transmission rate was approximately 2kB/s with BL600SA [48]. A 20-byte message can be issued approximately every 10 ms, but reliable delivery is not guaranteed. Even if the perfect reliability is assumed for delivery, each data file needs more than 500 seconds to complete the transfer. Such a long time for a single file download quickly becomes undesirable for a practical application. Furthermore, depending upon the operating system of a user device, the minimum connection interval varies significantly, directly lowering an already low data transmission rate, for example, iOS forces 30 ms or longer, and Android forces 11.25 ms or longer, whereas the minimum connection interval specification of BLE is 7.5 ms. For this reason, the BLE connection for VAMPIRE is primarily used for issuing commands, such as checking and changing the configuration tables and stored file tables. If a user wants to download the full resolution raw data file, the data transmission is handed off to the WIFI module instead. A user issues a command to VAMPIRE over the BLE connection to turn on the WIFI module, and the WIFI connection can be made to VAMPIRE by a user device, e.g., a Windows 10 tablet or an Android phone. Data files can be downloaded at a far greater rate, and the WIFI connection is terminated at user's request. Note that maintaining the WIFI channel is very costly in terms of power. The detailed power discussion is explored in the next subsection.

The BLE connection can be made to VAMPIRE any time except for the SD card writing period, which takes about 10 seconds after the completion of a spin-down

Table 4.1: BLE Characteristics

BLE Characteristics	Address	Remark
Command Input from User	h'9F3F	Refer to BLE Command*
File Table	h'9F40	MSB: Most Recent Slot Number Otherwise: Each Bit = Slot Written
Power Table	h'9F41	Refer to BLE Command*
Current Sensor Value Report	h'9F42	AccMSB X/Y/Z, BEMF, AccLSB, T
Status Report	h'9F43	Energy Buffer Voltage (h'03FF) Spindown Indication (h'1) Record (Future) Indication (h'1) Powersave Mode Indication (h'1) Primary On Indication (h'1)
BLE Connection Parameters Report	h'9F44	Connection Interval (h'FFFF) Connection Timeout (h'FFFF) Connection Latency (h'FFFF)
RTC Report	h'9F45	Y/M/D/H/M/S

Table 4.2: BLE Commands

BLE Command*	Cmd Hex	Remark
Power Gating Config (Power Table*)	h'01xx	Sensor(5), SD(3), WIFI(1)
		Persist(7), Energy Buffer(0)
File Slot Deletion	h'02xxyy	Delete One Slot xx = 01 – 96
		Reset Table: xx = 00, yy = FF
Current Sensor Value Checkup	h'030x	On/Off: x = 1/0
Record 100s from now	h'040x	On/Off: x = 1/0
Record the most recent 100s til now	h'050x	On/Off: x = 1/0
BLE GATT Force Disconnect	h'06	
WIFI Module SSID/Password	h'07xxY[8]	SSID = xx, Y[8] = 8 Byte PW
Retrieve RTC	h'08	
Set RTC	h'09Y[6]	Y[6] = Y/M/D/H/M/S

event. This assumes that the power is supplied to VAMPIRE, either by the harvester or by the energy buffer, LiPo battery. The BLE module maintains multiple characteristics as described in Table. 4.1. It uses one characteristic to take in the command from a user. The BLE command structure is described in Table. 4.2. The remaining characteristics are for reporting back the status of VAMPIRE, e.g., the current file table with the most recent file order, the current power gating configuration, the voltage of the energy buffer, the BLE connection parameters, the real-time clock information, etc. A power gating command can be issued to individually turn on and off the sensor package, the SD card, or the WIFI module. It can also request the disconnection of the energy buffer from the supercapacitor block, indicating a complete shut off of VAMPIRE. Once VAMPIRE completely shuts off, it can be only revived by turning on the motor to induce the rising edge in one of the core terminals. A file management command can be issued to erase one or the entire file slots. A real-time data monitor command can be issued, however, the data received through the BLE connection is down-sampled at 80 Hz due to the low BLE bandwidth. VAMPIRE can also record the incoming data to the SD card regardless of a spin-down event by a direct user command: it can select the immediate 100 second starting from the command issuance, or the most recent 100 seconds ending at the command issuance. They go through the same process as the spin-down file generation, and get stored in the SD card with a proper time stamping as well. In case of multiple VAMPIREs in a close proximity, the WIFI SSIDs must be uniquely assigned from each other for proper identification. For this purpose, users can issue a command to configure WIFI settings, such as the name of the SSID and the password. Also, there is a command to update the real-time clock. The other characteristics for reporting the status of VAMPIRE, e.g., the battery level, the indication for an ongoing spin-down, the indication for an ongoing user-requested file storing, the power status of the primary side, etc., are updated every second.

Once a spin-down event is completed, the most likely scenario for a staff is to make a BLE connection to VAMPIRE, check the recently stored file table by checking the 2nd BLE characteristic, turn on the WIFI module by changing the power gating configuration in the BLE power table, and download the file through the WIFI connection. At the time when the WIFI module is turned on by a BLE command, the WIFI module initializes an http server, which takes approximately 20 seconds to be ready for data transfer. Files can be directly downloaded by providing a proper http file address to their browser or the VAMPIRE App, a Windows 10 application developed in-house. With the WIFI connection, each file transfer takes less than a second, as the WIFI module supports IEEE802.11b/g/n.

4.3.4 Remarks on Digital Modules

The main microcontroller is in fact an embedded microcontroller inside the BLE module. Within the module, a relatively powerful processor (32-bit ARM Cortex M0 CPU) is packed for the BLE stack, and the BLE module is always powered on for user communication. So, using it as the main microcontroller for VAMPIRE can be beneficial in that it does not need to add one more microcontroller, saving power and space. The WIFI module is packaged together with the SD card [49]. The WIFI module can be accessed by software through SDIO standards, which is an extension to the existing SD card command structures. This package also has a powerful microcontroller inside, however, their quiescent current is unmanageably high, around 100 mW, even if it is idling. Furthermore, VAMPIRE does not always need the SD card or the WIFI connection. Therefore, the power hungry package, the SD card and the WIFI module, is actively power gated.

4.4 Power Consumption of the Sensor Package

Except for the SD card writing and WIFI data transfers, both of which happen after spin-down events, the nominal operating power for VAMPIRE is 38.2 mW. The average current consumption is approximately 9.55 mA at the supercapacitor voltage of 4.0 V. The breakdown of the power consumption within VAMPIRE is described in Table. 4.3. The nominal operation includes harvester switchings for rectification, power maximization, and regulation, sampling all the sensors at the full 2 kHz, storing the sampled data into the FRAM block, and maintaining the BLE connection continuously. The SD card needs at least 92.4 mW to operate (even idling), which is why it is power gated. Though it is not listed in the table, the SD card writing after the completion of a spin-down requires additional power dissipation of 70 mW. If a WIFI connection is established by user's request, it requires additional 627 mW,

Table 4.3: Power Consumption

Block	Current	Voltage	Power Consumption
Harvester Microcontroller	1.3mA	4.0V	5.20mW
Sensor Microcontroller	3.9mA	3.3V	12.9mW
Accelerometer + Temperature	0.15mA	3.3V	0.495mW
Back-EMF Sensor	0.85mA	3.3V	2.81mW
Main Microcontroller + BLE	3.6mA	3.3V	11.9mW
FRAM	1.5mA	3.3V	4.95mW
Real Time Clock	0.005mA	3.3V	0.0165mW
Sum: Nominal Operation	-	-	38.2mW
SD Card	28mA	3.3V	92.4mW
WIFI	190mA	3.3V	627mW
Sum: Maximum Peak	-	-	757.6mW

making the entire power consumption of VAMPIRE reaching up to 757.6 mW (or 827.6 mW if the SD card is being written with the WIFI on, which is not a good use case).

Since the WIFI connection requires up to 20 times more power (approximately 800 mW) than the quiescent operation of VAMPIRE (approximately 40 mW) and the minimum time overhead for the initialization of the WIFI module is 20 seconds, the energy buffer can be significantly drained. The frequency of accessing VAMPIRE through WIFI must be tightly controlled in order to guarantee nonnegative power flow into the energy buffer. For example, if the motor runs with 10 A_{RMS} with the minimum run-time of 10 minutes, VAMPIRE can harvest approximately 120 mW for 10 minutes. Out of the total power harvest, 40 mW will be dissipated for nominal operations, such as control, sensing, and data storage operations. The remaining 80 mW can be stored in the energy buffer. With this net positive energy stored, the maximum power dissipation of VAMPIRE with the WIFI connection (800 mW) can be sustained for approximately 60 seconds.

A user can decide to drain the energy buffer extensively, beyond the nonnegative

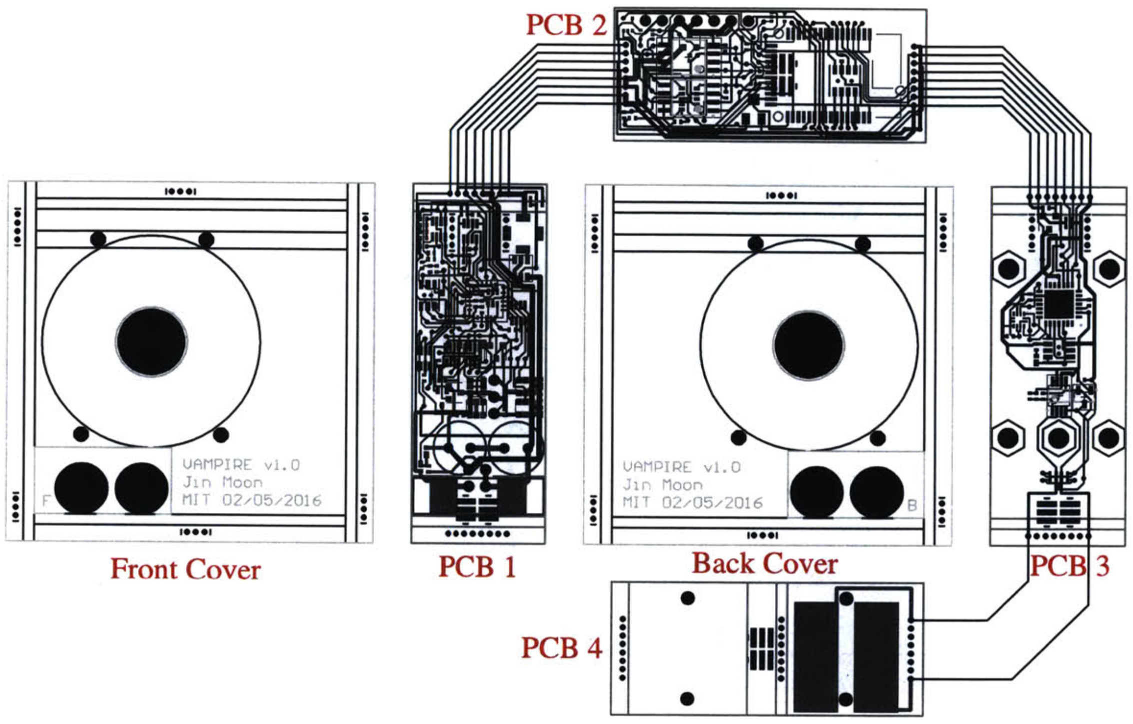


Figure 4-8: VAMPIRE PCBs Layout

energy flow constraint. However, the user will require more time to charge it up in a later access, if the user wants VAMPIRE to be an independent self-powered embedded system again. In any case, the LiPo battery is still protected by the circuit such that it cannot be drained lower than 3.85 V as a precaution to maintain a chemical stability and avoid a permanent damage to it.

4.5 Physical Construction

The physical structure of VAMPIRE is based on the printed circuit boards (PCBs). Six PCBs are used as six faces to create a rectangular box, and joints between neighboring PCBs are formed by soldering or socket-plug pairs. The layout of the entire PCBs is presented in Fig. 4-8. 'PCB 1' is the power package including the harvester front end, the power electronics circuits, and controls. 'PCB 2' is the part of the digital sensor package, consisted of the BLE module with the main microcontroller, the FRAMs, the real-time clock, and a mount for the SD card and the WIFI module.

'PCB 3' is the remaining sensor package, including the accelerometer, the temperature sensor, and the all the circuitries for the back-EMF sensor except for the front end that has capacitive couplings to motor's phase wires. The front end for the back-EMF sensor is placed on the PCB on the bottom, 'PCB 4'. The illustration of the capacitive coupling with this PCB is drawn in Fig. 4-7. Four PCBs from 'PCB 1' to 'PCB 4' join together, forming a rectangular structure. Two neighboring PCBs in this rectangular structure are connected by soldering a row of 90-degree, 0.5 mm pitch male pins. These soldering connections provide both physical structure integrity and electrical signal connection between two boards. In order to provide a firm and stiff environment for the accelerometer, the mounts which VAMPIRE uses to attach itself to the motor cover are placed on the sensor board, 'PCB 3'. The five mounting holes are emphasized with hexagonal shapes. The front and the back cover provide physical protection for the four boards on side faces, and do not have any electrical connection. The structure is designed such that all the outside faces of the PCBs are protected by the raised walls from neighboring PCBs. For example, 'PCB 1', 'PCB 3', 'Front Cover', and 'Back Cover' form raised walls to protect electrical components on the outside face of 'PCB 2'. The height of the raised wall is calculated based on the tallest component in the entire design.

The magnetic core is placed in the middle of this rectangular structure. Big circles in the front and the back covers designate the location. One of the two phase wires used for the back-EMF voltage sensor can go through the center of the core for energy harvesting as well. Two supercapacitors for the supercapacitor block and the 220 mAh LiPo battery are placed inside of this structure. They are placed in the lower part of 'PCB 1': two circles being the supercapacitors in series and the LiPo battery right below. The 3-D model of the final assembly with the internal components is illustrated in Fig. 4-9, and the actual VAMPIRE prototype is presented in Fig. 4-10 and Fig. 4-11.

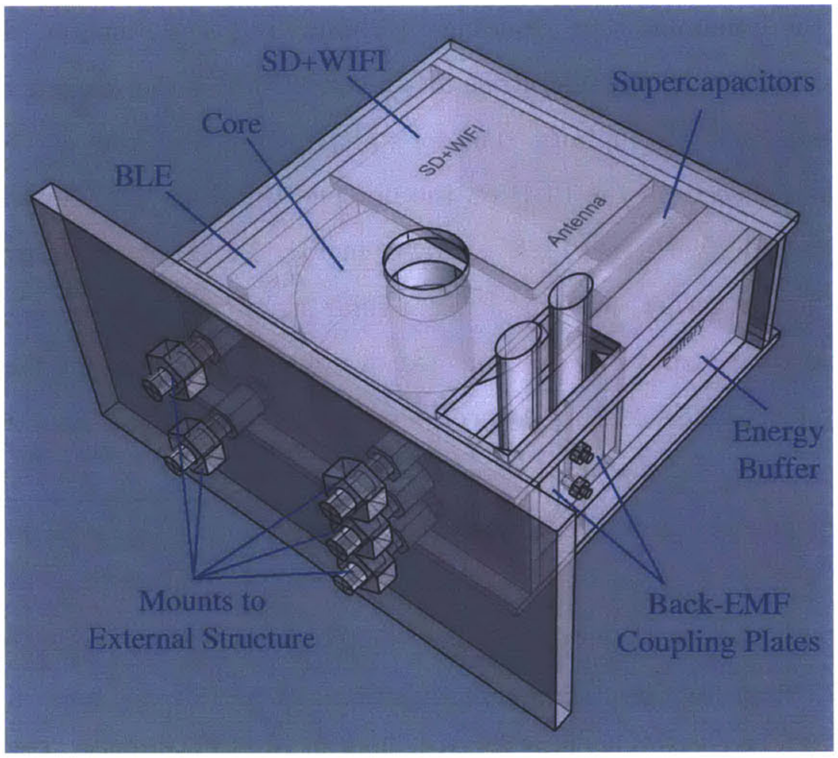


Figure 4-9: VAMPIRE 3-D Model with Internal Components

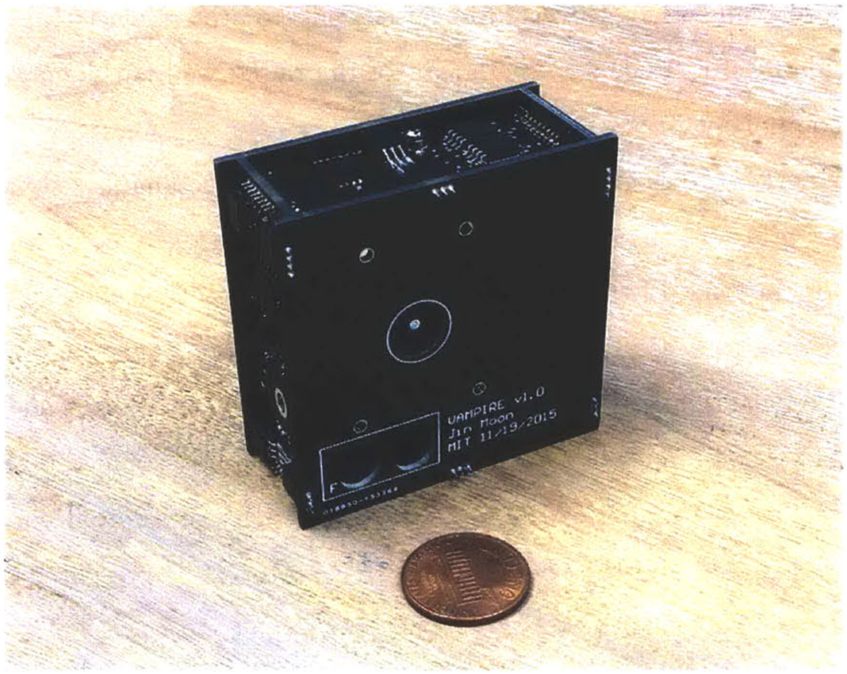


Figure 4-10: VAMPIRE Final Prototype - Assembled

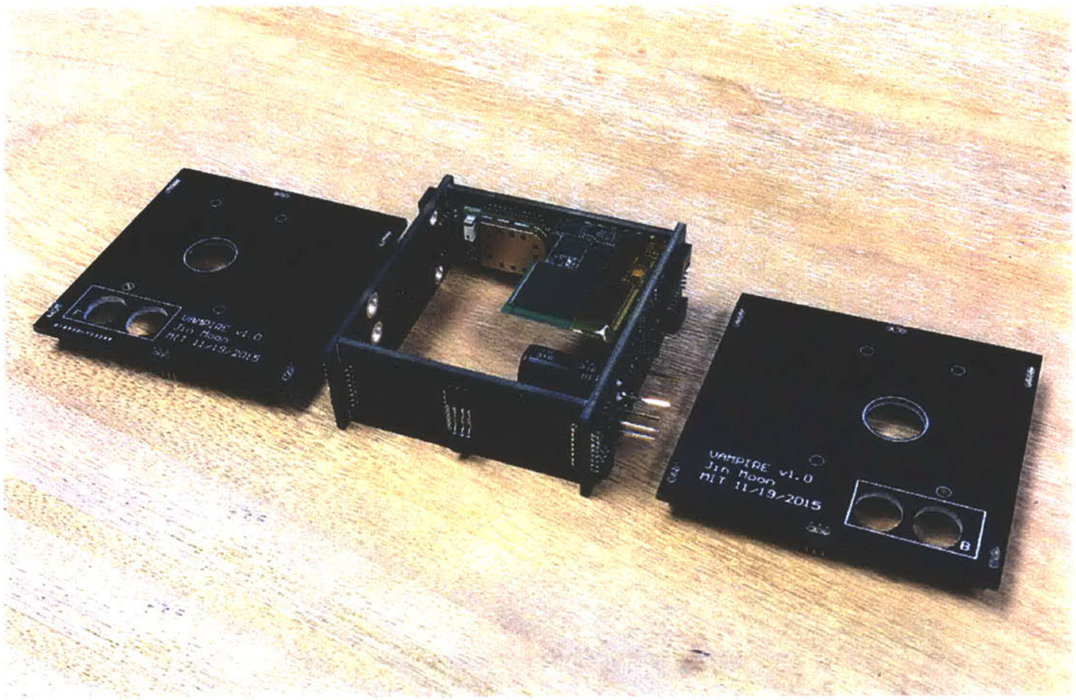
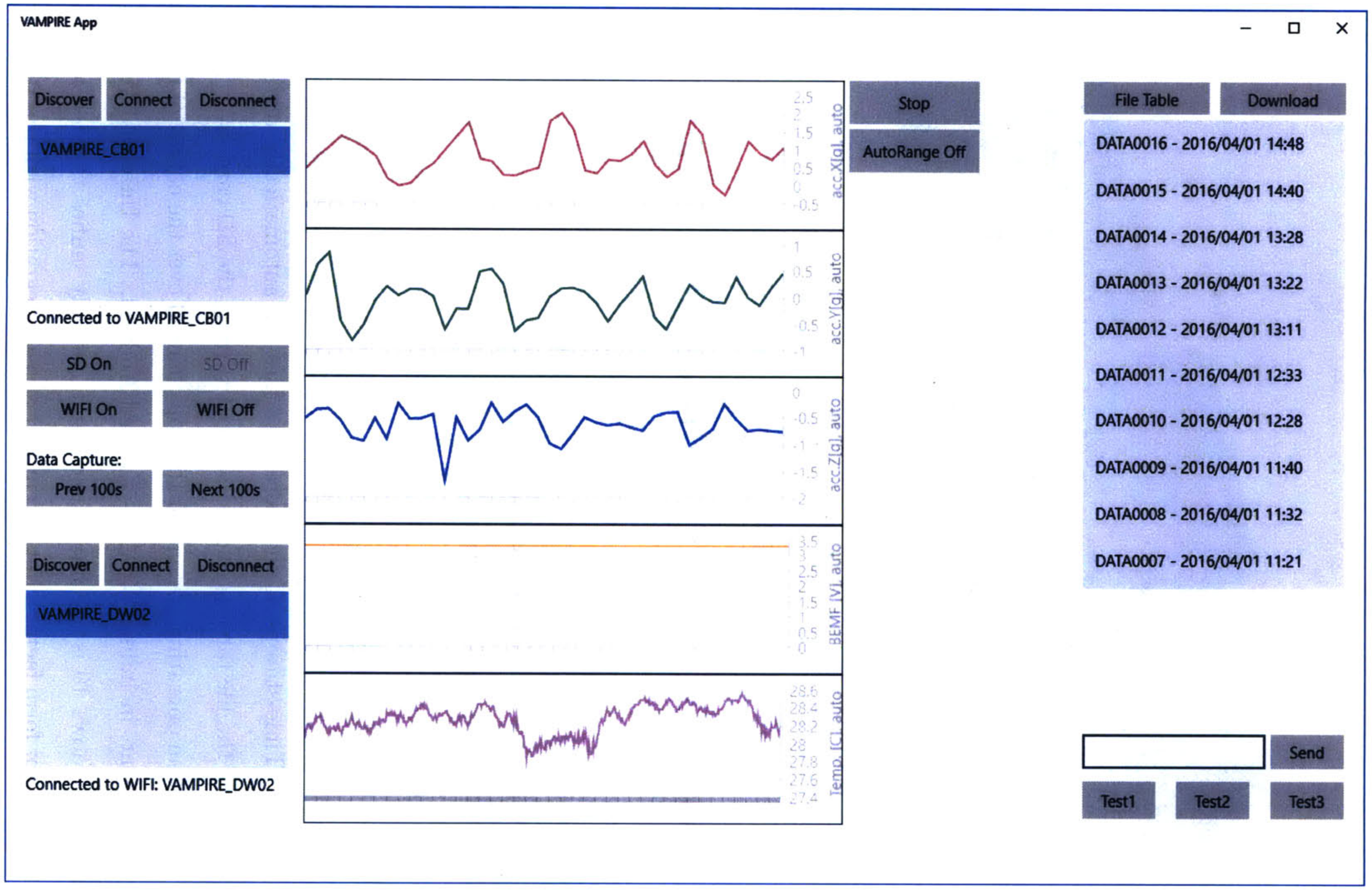


Figure 4-11: VAMPIRE Bare Structure - Expanded

4.6 User Application for Wireless Communication

The transparent BLE command structure of VAMPIRE allows any device that has a BLE capability to access and modify its configurations. As a demonstration, an app for the Windows 10 operating system is developed with C#. A user device is suggested to be equipped with both BLE and WIFI capabilities. The screenshot of the Windows 10 app is illustrated in Fig. 4-12. The 'VAMPIRE App' contains hardware knowledge for user conveniences, for example, time to boot and initialize the WIFI module is coded to prevent the users from applying ineffective commands during the boot-up period. Time-stamps of the store data files are automatically retrieved in background when the file table is queried without turning the SD card on.

Our choice of an operating system for the app changed over the course of the prototype development, including Windows XP and Android. The Universal Windows Platform for Windows 10 is the final choice as it not only unifies the development process for different form factors, but also provides easy integration with the Windows PC environment for intensive signal processing, like MATLAB or Octave.



Connected to VAMPIRE_CB01

Connected to WIFI: VAMPIRE_DW02

DATA0016 - 2016/04/01 14:48
DATA0015 - 2016/04/01 14:40
DATA0014 - 2016/04/01 13:28
DATA0013 - 2016/04/01 13:22
DATA0012 - 2016/04/01 13:11
DATA0011 - 2016/04/01 12:33
DATA0010 - 2016/04/01 12:28
DATA0009 - 2016/04/01 11:40
DATA0008 - 2016/04/01 11:32
DATA0007 - 2016/04/01 11:21

Figure 4-12: VAMPIRE Windows 10 App

4.7 Signal Processing

For condition-based maintenance, actionable information on the health of a motor needs to be extracted from the raw sensor data. The extraction of actionable information from the raw data — signal processing — can occur in a deployed embedded system or off-site. If signal processing is to be taken onboard the embedded system, only a signal-processed indication, which is usually extremely small in data size, is transmitted from the embedded system. It obviously saves communication power, potentially eliminating the necessity of a high-power, high-bandwidth communication protocol, such as WIFI. However, the computation power is drastically increased, even to the level comparable to or higher than WIFI power dissipation. On the other hand, if signal processing is to be performed off-site, there is no need for a powerful digital signal processor capable of fast floating point calculations in the embedded system. It also provides flexibility in choosing digital signal processing (DSP) tools and gives an option to easily update DSP algorithms later on. However, the entire raw data must be offloaded within a relatively short time, using a power intensive communication protocol like WIFI. Between these two approaches, VAMPIRE chooses the latter, offloading the stored raw data through WIFI.

Taking any approach, the implementation of the signal processing in any environment should be mathematically identical. Here, MATLAB is used, assuming Windows environment, on the extent that it can be applied to any system with sufficient computation power. In fact, the VAMPIRE App is a demonstration of how the collected sensor data can be accessed, assuming a technician with a tablet. A more realistic approach to the continuous condition-based maintenance is to have a central network that all the smaller sensor nodes, like VAMPIRE, report to. One of the options is discussed in [50]. Instead of a sporadic presence of a technician, a nearby NilmDB server can receive the data after every spin-down event, seamlessly process the data, and continuously relay the information to users through a web interface.

In the next subsection, mechanical impedance spectroscopy — signal processing to extract actionable information from the raw sensor data that VAMPIRE generates —

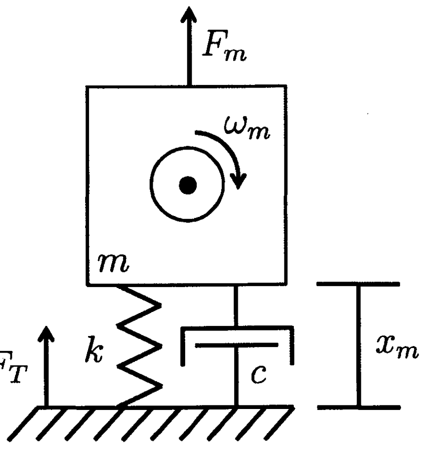


Figure 4-13: Spring-Mass-Damper Electromechanical System (Image courtesy of [1])

is discussed. Subsequently, the experiment is set up and carried out to illustrate the effectiveness and validity of the self-powered embedded system and signal processing for condition-based maintenance.

4.7.1 Mechanical Impedance Spectroscopy

The mathematical derivation in this subsection up to (4.9) is introduced in C. J. Schantz's dissertation [52] and R. Zachar's paper [1] in more detail. For the scope of this thesis, the epitome of the analyses is excerpted and explained here. Figure. 4-13 illustrates a second order spring-mass-damper mechanical system that can be used to model an electric motor or generator on resilient mounts [51]. Its governing equation is

$$m\,\ddot{x}_m(t) + c\,\dot{x}_m(t) + k\,x_m(t) = F_m(t), \tag{4.4}$$

where m is mass, x_m is the position of the system, k is the spring constant, c is the damping ratio associated with the mount, and $F_m(t)$ is a forcing function of time t. Rewriting (4.4) in terms of acceleration, $a_m(t)$, gives

$$m\,a_m(t) + c\int_{-\infty}^{t} a_m(\tau)\;\mathrm{d}\tau + k\int\int_{-infty}^{t} a_m(\tau)\;\mathrm{d}\tau = F_m(t). \tag{4.5}$$

After the Laplace transform, the transfer function from $F_m(s)$ to $A_m(s)$ is given as:

$$\frac{A_m(s)}{F_m(s)} = \frac{\frac{s^2}{m}}{s^2 + \frac{c}{m}s + \frac{k}{m}}, \tag{4.6}$$

where $A_m(s)$ and $F_m(s)$ are the Laplace transform of $a_m(t)$ and $F_m(t)$, respectively. At the steady-state, $F_m(t)$ for a rotating machine with an eccentric mass, m, can be generally written as

$$F_m(t) = C\,\omega_m^2 \cos(\omega_m\, t), \tag{4.7}$$

where C and ω_m denote a constant related to load mass and imbalance and the speed of the rotating shaft, respectively. With an omission of C in (4.7), a new function, $\Phi_m(t)$, is defined:

$$\Phi_m(t) = \omega_m^2 \cos(\omega_m\, t). \tag{4.8}$$

Note that (4.8) has only one unknown parameter in it: the rotor speed, ω_m. The corresponding transfer function from Φ_m to A_m is then:

$$\frac{A_m(s)}{\Phi_m(s)} = \frac{C\,\frac{s^2}{m}}{s^2 + \frac{c}{m}s + \frac{k}{m}}. \tag{4.9}$$

Except for the constant C, the transfer function (4.9) is identical to the transfer function (4.6). This implies that (4.9) can provide the same frequency dynamics, e.g., structural resonance frequency, as (4.6), except for the absolute magnitude. Moreover, this transfer function now scales as C changes, e.g., due to a change in load imbalance, unlike (4.6). Therefore, by obtaining (4.9), multiple phenomena can be identified in theory.

To obtain the transfer function (4.9) experimentally, $a_m(t)$ and $\Phi_m(t)$ are required. The acceleration can be obtained by querying an accelerometer at a sufficiently high sampling frequency. Experimentally obtaining $\Phi_m(t)$, which is called a virtual input, is also relatively easy as it only requires the speed of the rotor, ω_m. By measuring the acceleration and the speed of the rotor during a motor spin-down, (4.9) — an empirical vibration transfer function (eVTF) — can be generated. By analyzing the eVTF, an indication to motor's mechanical problems can be obtained.

Each file stored in the SD card of VAMPIRE contains 100 seconds worth of acceleration and back-EMF voltage data, sampled at 2 kHz. First, the zero crossing locations are extracted from the back-EMF voltage data. The number of pole-pairs in a rotational machine affects the number of zero crossings in a full revolution. The motor used in the experiment for VAMPIRE generates two zero crossings when the rotor makes a full revolution. Therefore, the extracted zero crossings can be directly converted to the speed of the rotor. The conversion of the zero crossing locations to the rotor speed uses the following equation:

$$\tilde{\omega}_m(t_{\mathrm{ZC}}[n]) = 2 \cdot \frac{2\pi}{t_{\mathrm{ZC}}[n+1] - t_{\mathrm{ZC}}[n]}, \tag{4.10}$$

where $t_{\mathrm{ZC}}[n]$ denotes the n^{th} zero crossing location in time and $\tilde{\omega}_m(t_{\mathrm{ZC}}[n])$ denotes the estimated rotor speed at $t_{\mathrm{ZC}}[n]$.

One thing to note is that due to the differentiator formed by the capacitive input at the front stage amplifiers of the back-EMF voltage sensor, as shown in Fig. 4-6 and Fig. 4-7, the direct output from the back-EMF sensor is the time derivative of the back-EMF voltage between the motor phase wires. Prior to the extraction of the zero crossing locations, software integration is first required to fully reconstruct the back-EMF voltage. In order to eliminate any DC drift during the software integration, a high pass filter with the cut-off frequency set at slightly higher than the DC is applied after the integration. For the results provided in this thesis, the 4th order high-pass Butterworth filter with the cut-off frequency at 6 Hz is applied. Figure 4-14 illustrates the raw back-EMF sensor output, the software integrated result, the high-pass filtered result, and the rotor speed estimation.

Now that $\Phi_m(t)$ and $a_m(t)$ during the spin-down event are obtained, the eVTF can be constructed by performing Fast Fourier transforms (FFT) on each signal. According to [52], calculating an eVTF with spin-down data can be susceptible to noise from other nearby machinery. According to [1], the change in excitation frequency during the spin-down (as the rotor slows down) further complicates the accuracy and calculation of an eVTF. It suggests to use the Short-time Fourier Transform (STFT) to reduce the uncorrelated noise and minimize frequency of excitation spreading to a wide frequency range. The spin-down period is split into multiple Hanning windows. The virtual input and the acceleration data are multiplied in the time-domain with a series of these time-binned Hanning windows, forming the 'masked' input (Hanning windowed virtual input) and the 'masked' output (Hanning windowed acceleration). Figure 4-15 illustrates the Hanning windows and masked input and output.

FFTs are performed on these masked inputs and outputs. The maximum envelopes are extracted from the overlaid FFTs of the masked inputs and outputs. Finally, the eVTF can be constructed by dividing the maximum envelope of the FFT of the masked outputs by the maximum envelope of the FFT of the masked inputs. In order to decouple environmental vibration projected onto the motor structure, e.g., other nearby electromechanical systems, the FFTs of the masked outputs are filtered with 6 Hz passband centered at the excitation frequency, which decays as the motor spins down. The extraction of the maximum envelopes and the resulting eVTF are illustrated in Fig. 4-16. The filtered FFTs can be verified in the second plot of the figure as well.

As discussed, an eVTF is rich in condition monitoring information. It can be used to observe the structural resonance properties and to check the change in load imbalance, as stated in (4.9). In the next section, experiments to link these properties to practical structural problems of an electromechanical system are presented.

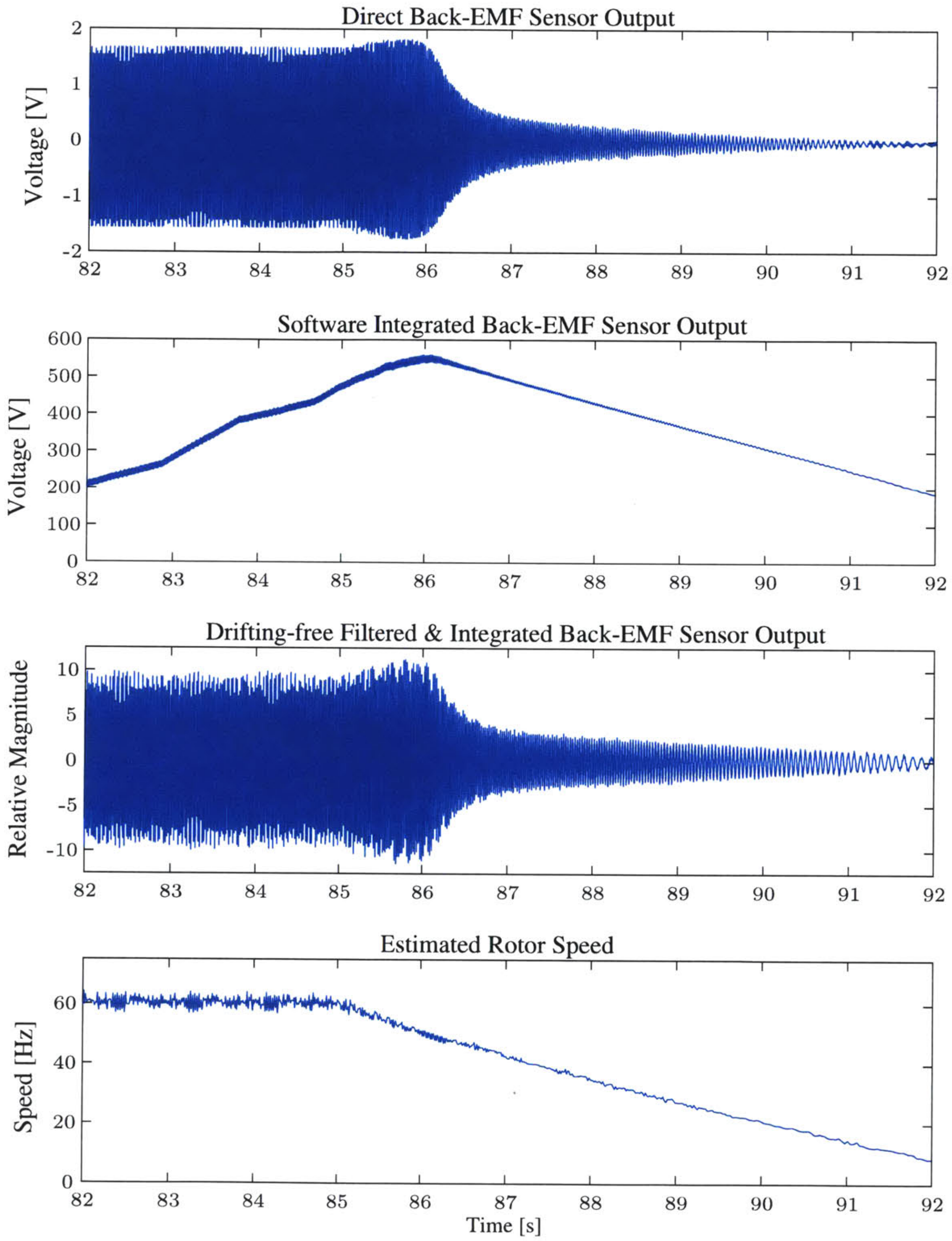


Figure 4-14: Back-EMF Sensor Output Processing

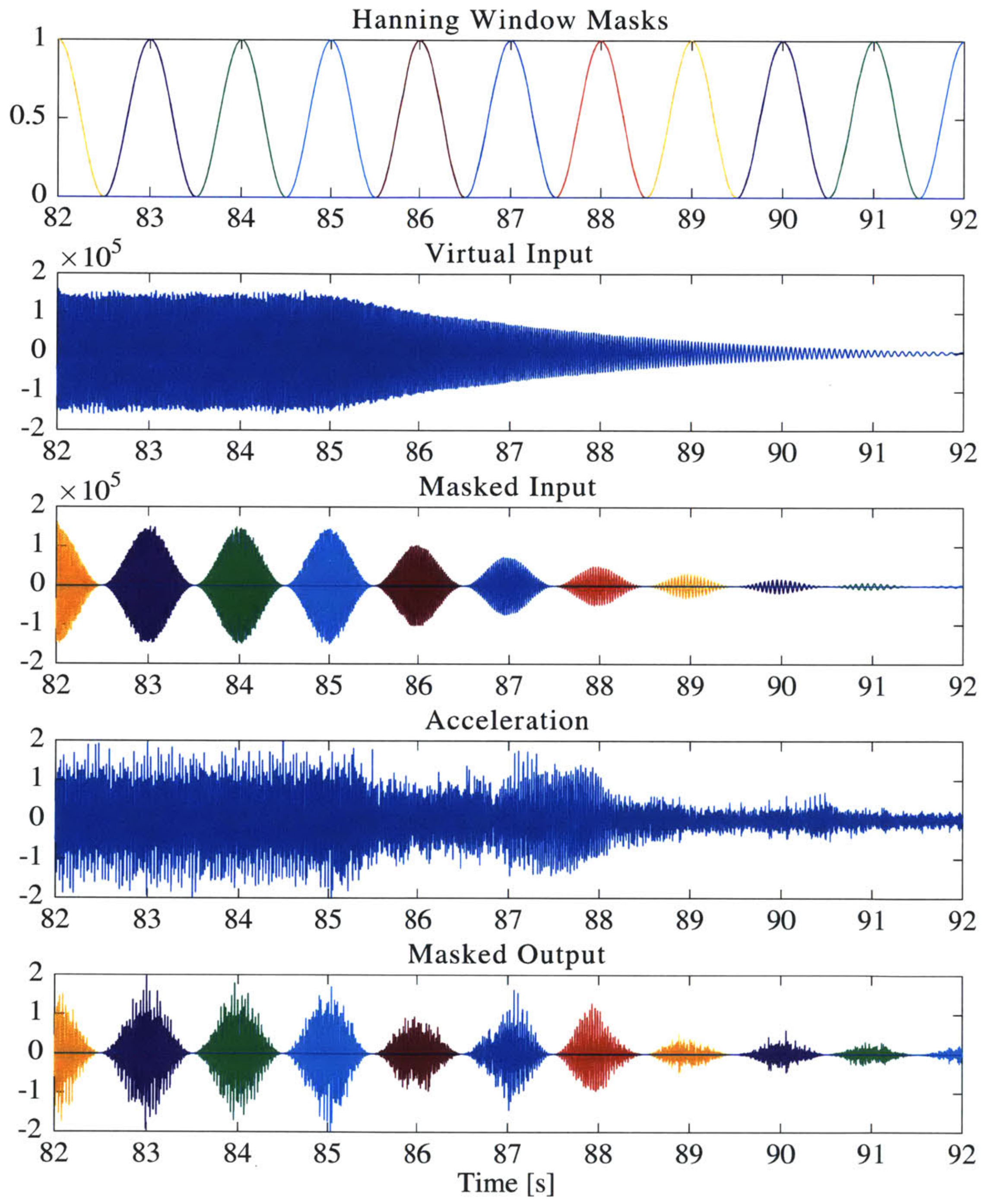


Figure 4-15: Hanning Window Mask and Masked Input/Output

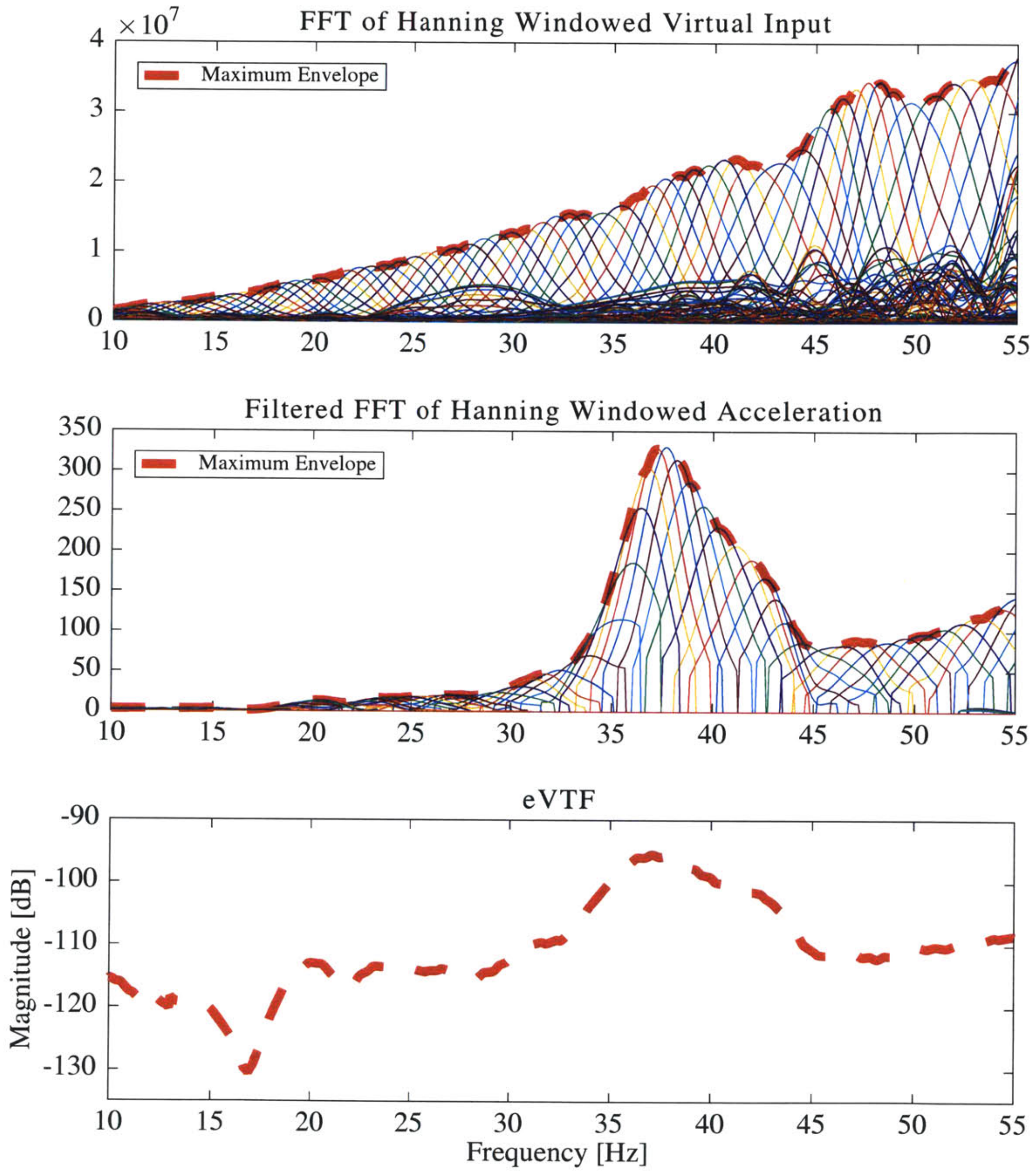


Figure 4-16: eVTF Construction

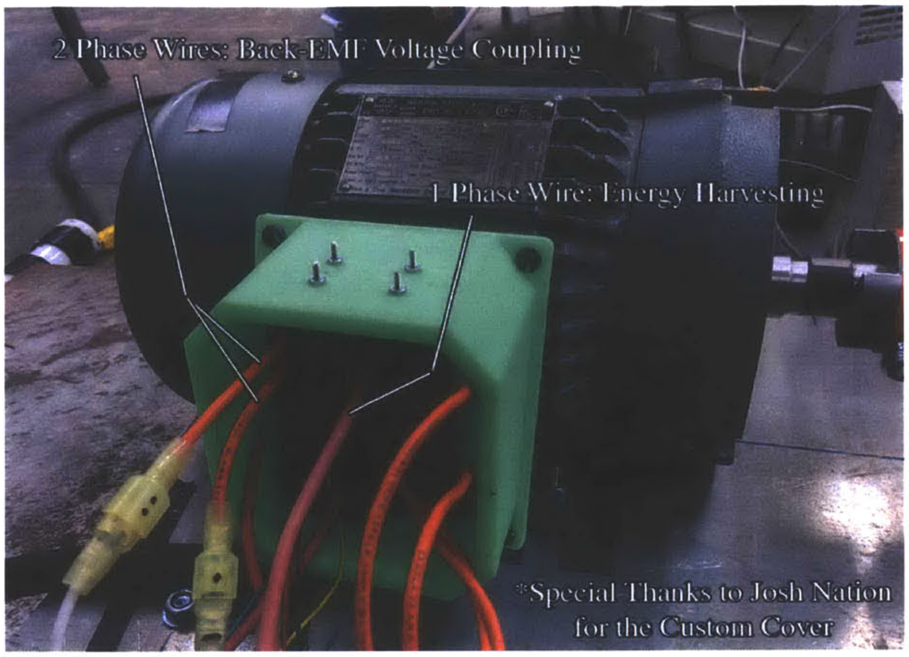

2 Phase Wires: Back-EMF Voltage Coupling
1 Phase Wire: Energy Harvesting
*Special Thanks to Josh Nation for the Custom Cover

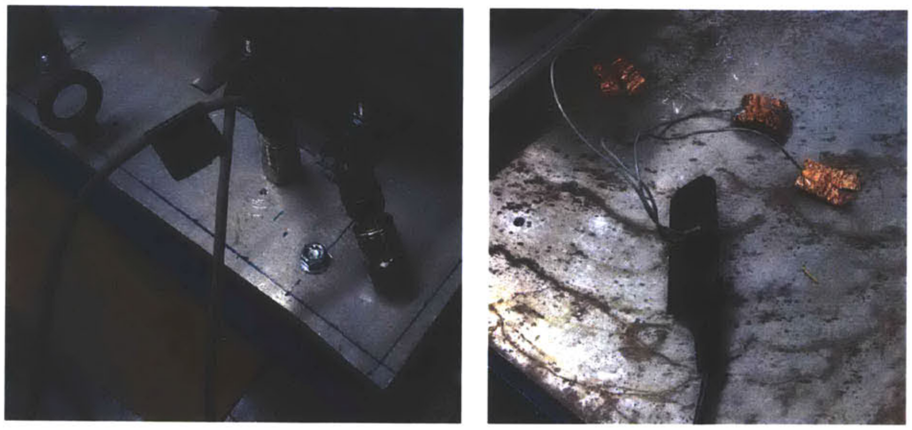
Figure 4-17: VAMPIRE in the Experiment Setup

4.8 Experiment

4.8.1 Experiment Setup

VAMPIRE is installed in a real vibration testing setup as shown in Fig. 4-17. The bright green cover for motor's terminal box is 3-D printed to provide a secure mount for VAMPIRE, replacing the existing external cover. For the same task of generating an eVTF, this retrofit installation of VAMPIRE replaces all the equipment shown in Fig. 4-18, which were required for data collection and analysis for [1]. The replaced components are: commercial off-the-shelf accelerometers; standalone back-EMF sensors; data acquisition blocks; data storage; wireless network capability; and power supply and wiring.

In this experiment, VAMPIRE harvests energy from 3-phase AC motor's wiring and provides health information of the electromechanical system that this motor is connected to. One of the three phase wires for the AC motor, the center wire shown in the figure, feeds magnetic fields to the magnetic core inside the embedded system. This wire and one additional phase wire are capacitively coupled to provide a differ-


(a) Commercial Accelerometers (Left) and Standalone Back-EMF Sensors (Right)

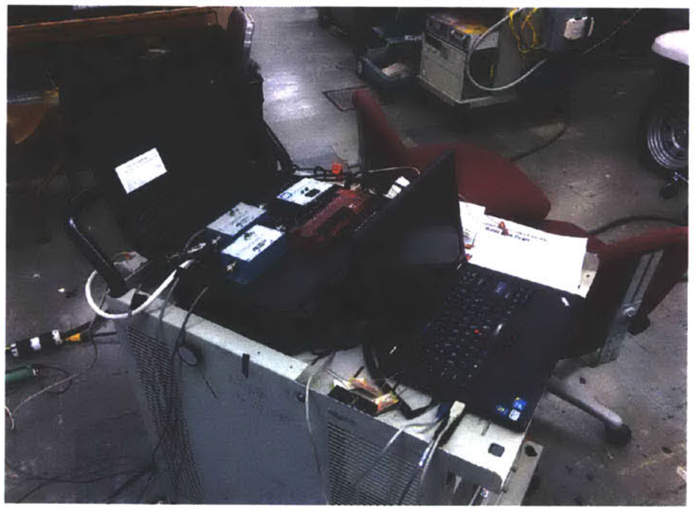
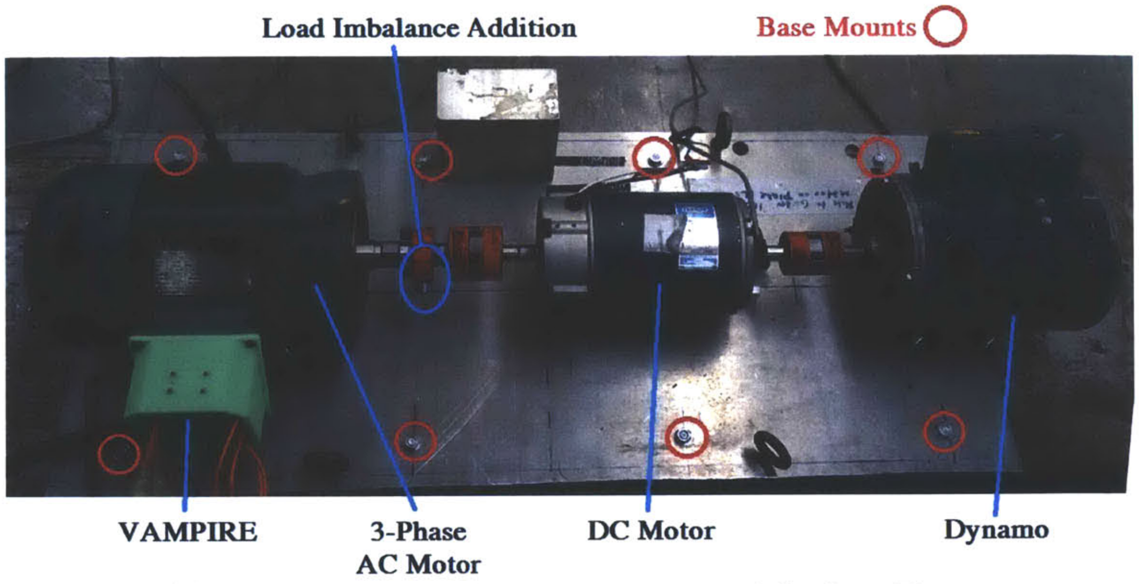

(b) Data Acquisition Blocks, Data Storage, and Power Supply and Wiring

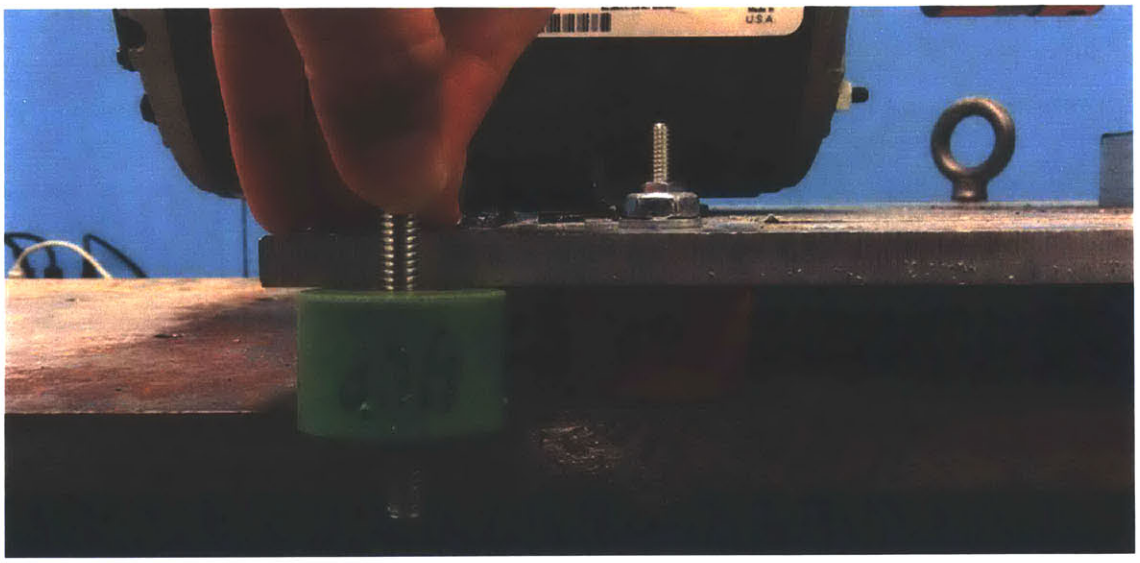
Figure 4-18: Equipment of [1] that is Replaced by VAMPIRE

ential voltage input to the back-EMF sensor. Though Fig. 4-17 shows test wirings outside the cover for debug and measurement purposes, wirings can be contained inside the box and the front face of the cover can be sealed in real applications.

The overall electromechanical test structure is depicted in Fig. 4-19 (a). The AC motor takes in 3-phase power input, and spins the load connected to the shaft: the DC motor and the dummy dynamo. The AC motor is rated up to 1.5 hp (1.1 kW), with the steady state revolutions per minute (RPM) of approximately 3,600 at the line frequency of 60 Hz. The electrical connections of the DC motor are left disconnected when the AC motor is powered. This electromechanical system can be also excited



(a) Electromechanical Connections on top of the Base Plate



(b) The Base Plate Mounted on a Box Beam with Changeable Stiffness

Figure 4-19: Test Setup - Electromechanical System under Diagnosis

by the DC motor by directly applying a DC voltage to it. In this case, the electrical connections to the grid are disconnected on the AC motor side. When the DC motor is operated for a spin-down analysis, the DC voltage supplied to the DC motor is controlled such that the same RPM of 3,600 is achieved at the beginning of the spin-down. Our intention is to provide the same mechanical spin-down response regardless of the excitation method. This electromechanical test structure is mounted on a thick metal plate that acts as the base, and the base plate sits on a metal box beam with mounts in between, as shown in Fig. 4-19 (b). Eight mounts between the base plate and the box beam can be seen in Fig. 4-19 (a). Note that when the AC motor is powered, VAMPIRE can operate on its own by harvesting energy from AC magnetic fields from the AC motor. On the other hand, when the DC motor is powered, the harvester inside VAMPIRE cannot extract any energy. In this case, the energy buffer — LiPo battery — is temporarily connected to the digital sensor package as a power supply. The battery voltage is maintained at approximately 4.1 V. The DC motor excitation is considered for cross-validation of VAMPIRE against the existing analysis environment and equipment of [1].

In the next subsection, the cross-validation of VAMPIRE results is presented first. The actionable information that can be extracted from the signal processing is described subsequently, identifying two important issues in the electromechanical structure: changes in the stiffness of mounts and changes in the load imbalance.

4.8.2 Results

Figure 4-20 presents eVTFs generated by various methods. The first plot is generated by exciting the DC motor at multiple frequencies and collecting the steady-state vibration data. The second plot is generated from a spin-down event by VAMPIRE powered by the energy buffer with the electromechanical structure excited by the DC motor. The third is generated from a spin-down event by the existing system, the motor mount test platform (MMTP), that was used in [1]. The DC motor is excited in this case. The fourth plot is generated from a spin-down event by VAMPIRE with the AC motor excitation. VAMPIRE is powered on by the internal energy harvester. The stiffness of the mounts used in these four cases is the same: durometer of 60A. Also added is the load imbalance of 17 g to the shaft for these cases.

With any excitation method, either the DC excitation or the AC excitation, VAMPIRE provides an almost identical primary resonance location, compared to the steady-state and the existing spin-down analysis method [1]. The overall shapes of the eVTFs that VAMPIRE generate also agree well with the existing methods. The

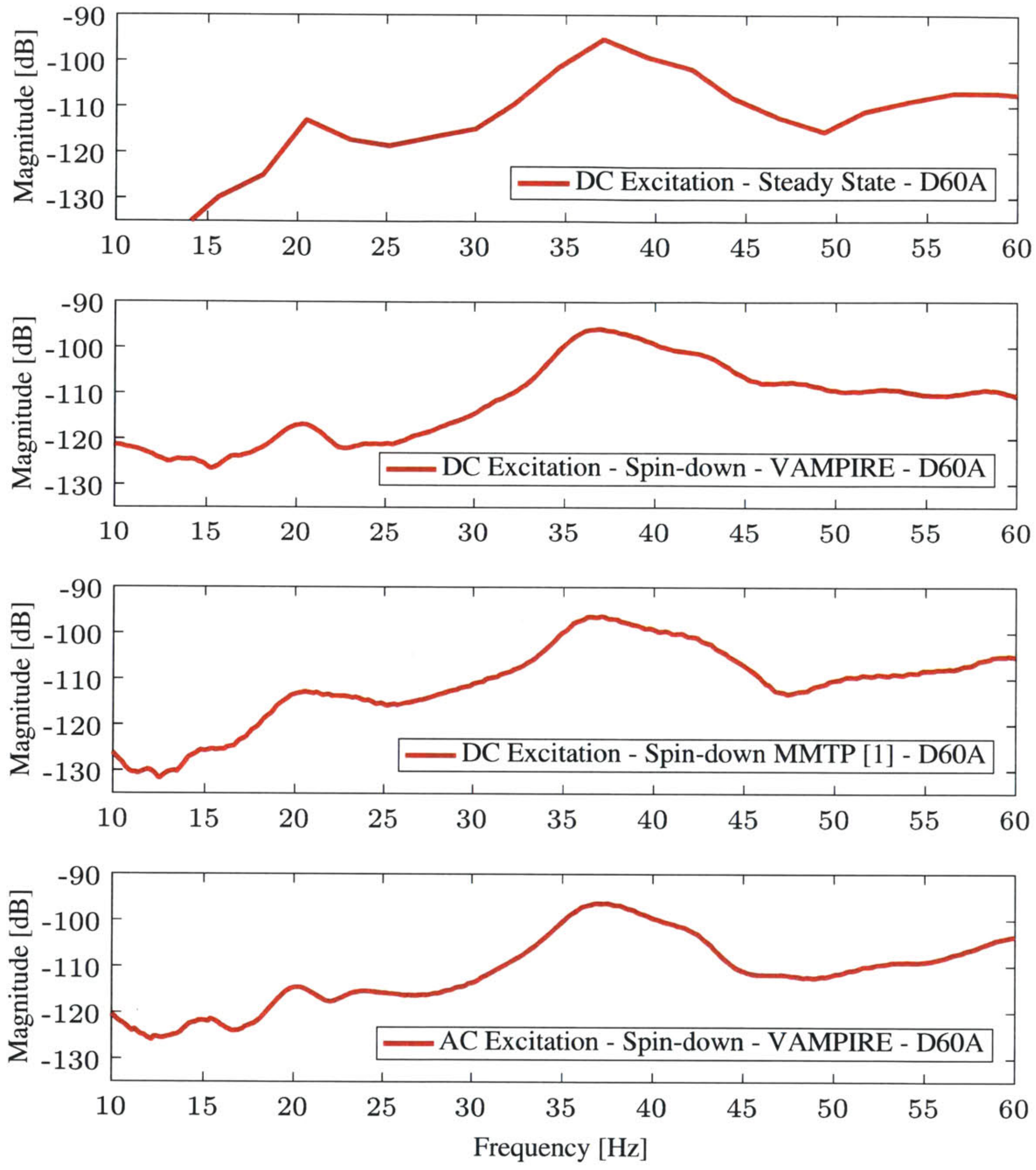


Figure 4-20: Crossvalidation of VAMPIRE Results

different shapes of the spin-down results at the low frequency region compared to the steady-state result are due to the fact that the rotor speed is low and can be easily affected by ambient noise. The different shapes at the higher frequency tail primarily derive from the transient effects at the initiation of a spin-down event.

The primary peak area is zoomed up for a closer verification in Fig. 4-21. As seen

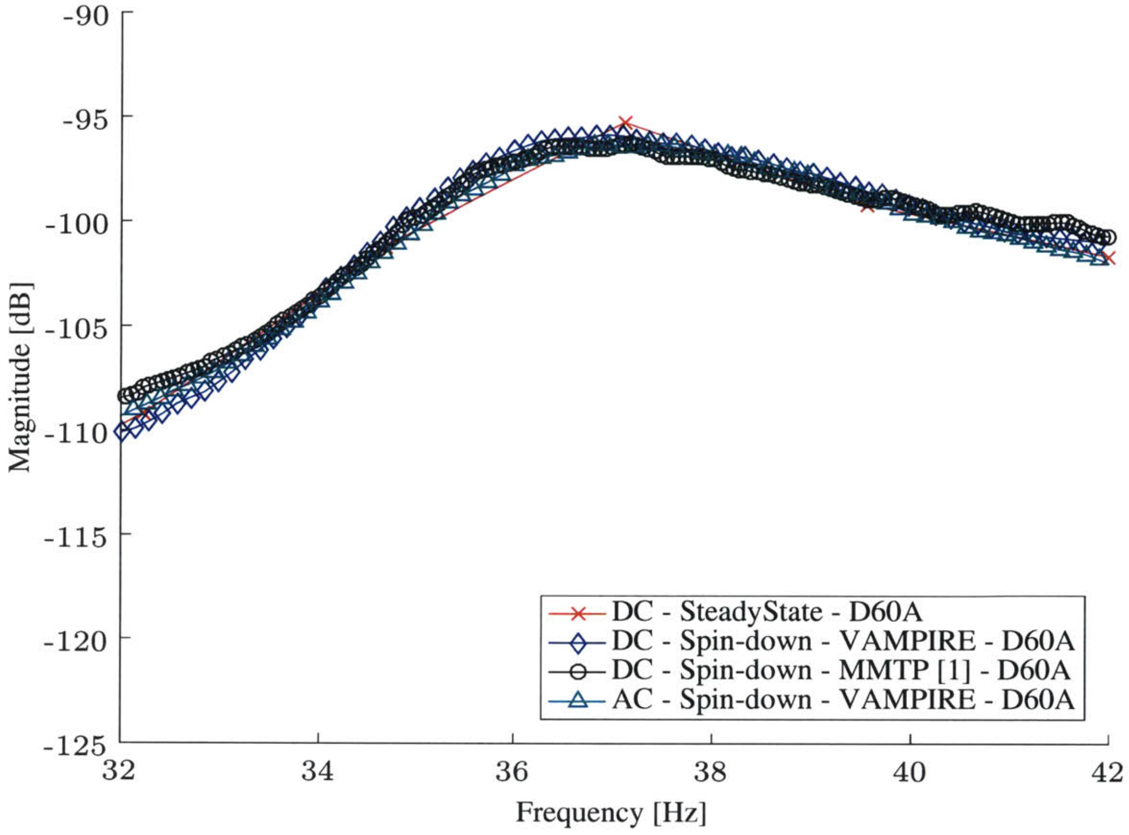


Figure 4-21: Closer Look at the Primary Peaks of the eVTFs

from the figure, the location of the peak, which is the one of the most important characteristics in an eVTF, for every case matches accurately. This implies that an eVTF generated by VAMPIRE can identify the mechanical self-resonant frequency as accurate as other methods.

Now, a series of eVTFs are generated with VAMPIRE data under different conditions. Two different mounts with durometer of 40A and 60A are assumed. At the same time, a load imbalance of 17 g is considered for each mount as well, giving a total of 4 cases. The result is illustrated in Fig. 4-22. Consider the red curve, 'D40A', as the base case. When the imbalance is added to the shaft, C in (4.9) is increased. Since the y-axis is in dB, the increased coefficient results in a vertical shift, as seen in the blue curve, 'D40A Imbalance.' When the D40A mounts are replaced with the D60A mounts (but still with the load imbalance), the change in the stiffness of the mounts results in different vibration dampening and brings the change in the me-

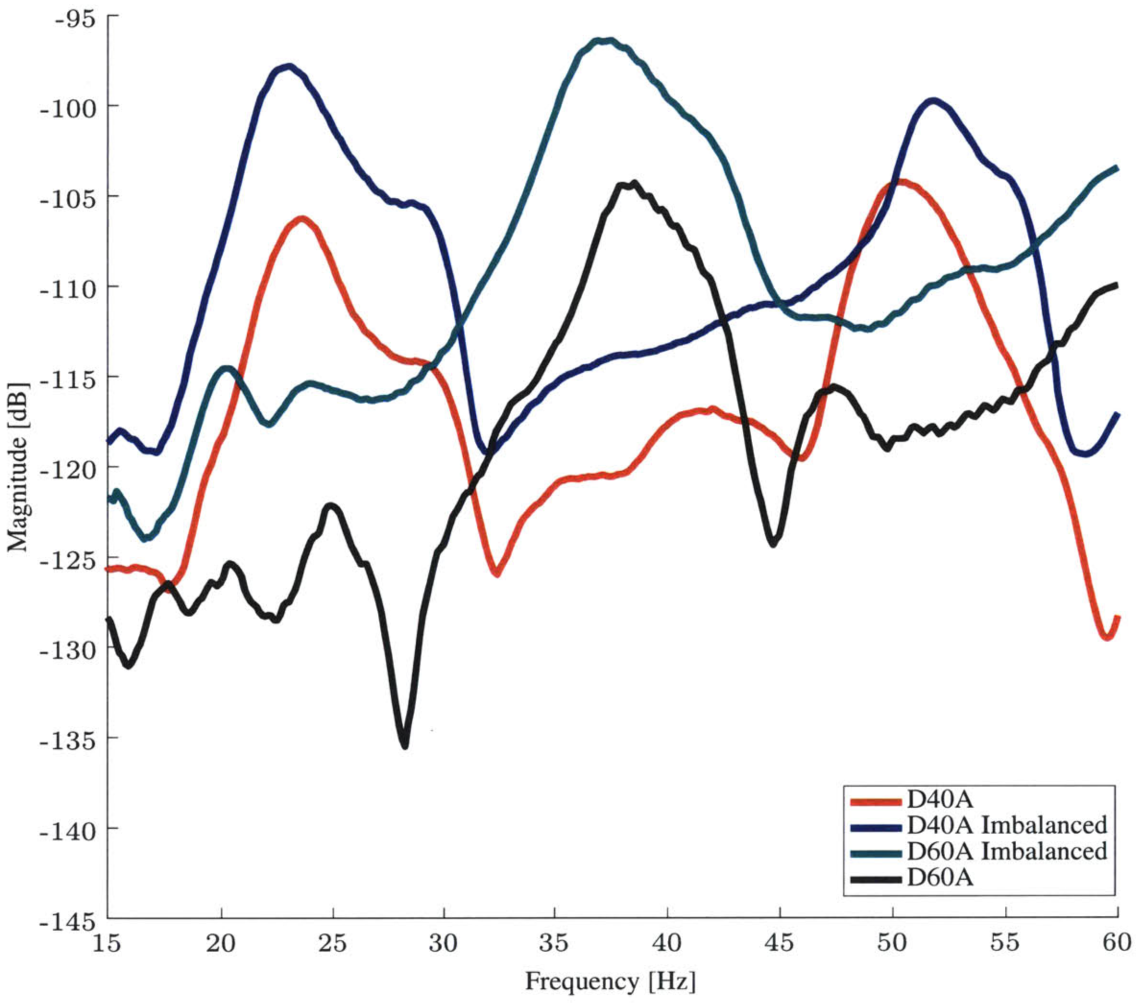


Figure 4-22: Extracting Actionable Information from VAMPIRE eVTFs

chanical self-resonant frequency. This clearly manifests as a horizontal shift, as seen in the green curve, 'D60A Imbalance.' When the load imbalance is removed with the D60A mounts, the vertical shift in the reverse direction happens as C is decreased, presenting the black curve, 'D60A.'

By monitoring changes in eVTFs over time, two separate electromechanical issues can be diagnosed: a change in mount stiffness and a change in load imbalance. A change in stiffness can imply a damaged or degraded mounts. A change in load imbalance can imply a damaged load, a changed load profile, or a damaged shaft axle. Heuristics to quantify the shape of an eVTF, for example, locations of the primary peaks and corresponding magnitudes, slopes at certain regions, etc., can be implemented on a monitoring system with a networking capability, like NilmDB, and automatically generate an indication for users for condition-based maintenance, enabling failure prevention and low-cost maintenance for the electromechanical systems.

Chapter 5

Conclusion

This thesis has presented a complete analysis and design of building a self-powered embedded system based on electromagnetic energy harvesting. The research has been motivated by a need for condition-based maintenance for electromechanical systems to reduce time and cost involved in a potential repair. A self-powering mechanism is extremely valuable for sensor systems as it widens the area where the sensor systems can be deployed, eventually providing a more distributed and deeper sensing at finer granularity. A new way of extracting energy from an electromagnetic coupling that is introduced in this thesis targets the environment where a significant amount of ambient energy is not available by nature, e.g., dark enclosed space inside a motor, and especially where the installation of such a mechanism is constrained in physical size. The design example in this thesis is engineered to fit specifically for electromechanical systems, and a vibration sensor node for rotational machines, VAMPIRE, is developed to demonstrate the capability and usefulness of electromagnetic energy harvesting.

This thesis covers the entire spectrum of the design of a practical self-powered sensor node application: physics modeling, simulator design, power electronics circuit design, digital system design, software design, and physical construction. Since the new energy harvesting method using an electromagnetic coupling is still relatively unknown, this thesis starts with developing new analytical models for circuit designs. It also explores power electronic circuits designs to optimize and maximize power harvest and provide power regulation and path controls to be used as a practical power supply to the digital sensor package. Detailed design method and analysis on the internals of the embedded system for electromechanical diagnosis are discussed. A user software to wirelessly communicate with VAMPIRE is also developed, and a signal processing method to extract actionable information from a raw data collection is explained.

Electromagnetic energy harvesting can provide a reliable power flow and an adequate power level to sustain a sensor node with relatively small dimensions compared to other commonly used energy harvesting methods. Using electromagnetic energy harvesting, a magnetic core as small as $2.9\,\mathrm{cm}^3$ can provide approximately $12\,\mathrm{mW}$ per $1\,\mathrm{A_{RMS}}$ on the primary side. Electromagnetic coupling does not require a physical, ohmic contact to the existing electrical system, extracting energy nonintrusively from magnetic fields emanating from a power line of equipment under monitoring. Against the intuitive and conventional usage of inductors and transformers, a magnetic core used in electromagnetic energy harvesting is periodically forced into saturation. It is revealed in this thesis that an unsaturated core always results in suboptimal power harvest, and saturating the core at proper times maximizes power harvest. A simple explanation for the need of magnetic saturation is that the core is essentially a non-ideal current transformer where the linearity of the core is not relevant. Magnetic saturation allows either higher average current into a fixed load voltage or higher load voltage with a fixed average current. Chapter 2 presents a detailed discussion on creating a core mode — the Maxwell model – in the presence of magnetic saturation and building a numerical simulator based on the Maxwell model. The numerical simulator is built in MATLAB, using Newton and the GCR. The core model and the numerical simulator are verified with experiments and proven to be extremely accurate.

Chapter 3 starts with developing a SPICE-friendly model. It derives from the Maxwell model with a dimensional approximation. It is validated to be accurate and easy to use in commercially available circuit simulation tools. Two circuit design techniques to bring major enhancement in power harvest are introduced: the flux-shaping capacitor (FSC) method and the transfer window alignment (TWA) method. The FSC method shapes the flux accumulated in the core such that the transfer window is significantly lengthened, directly increasing the amount of harvested power. The power boost can be achieved without adding any active component or control effort. The downside of this method is that it is only appropriate for the cases where the primary side current does not change much across operations. The TWA method aligns the transfer window such that the maximum average current is obtained from the electromagnetic coupling. It is more resilient to the environmental changes at the expense of more control efforts. Since control efforts are relatively cheap with sensor nodes equipped with microcontrollers already, the TWA method is chosen to be the default method in the prototype to enhance power extraction. A detailed discussion on control implementation of the TWA method is also presented. Finally, the deviation from the expected responses with the TWA method in the presence of other flux dissipating components are covered. The example is explored with the assumption of very high-power primary side, which illuminated the importance of the DC wire resistance of the core windings in the flux equality.

In Chapter 4, the design of the self-powered embedded system is presented. The power flow and regulation stages in the power package is explored and the necessity of the energy buffer for VAMPIRE is explained. Since there are multiple power sources that require charging, discharging, and regulation, power paths are accordingly controlled. The spin-down event, which is crucial for electromechanical diagnosis, is detected using the inherent nature of a nonideal current transformer, detecting a sudden change in magnetic fields. The sensors and microcontrollers to acquire data are also discussed in detail. Two wireless communication protocols are employed in the design: BLE and WIFI. The BLE connection is used for issuing commands, checking the status of the embedded system, and configuring options. The WIFI connection is used for raw data transfers. VAMPIRE is operated with 40 mW nominal power consumption, with the option of performing power-intensive operations, such as SD card writing and WIFI data transmission after spin-down events, provided that the energy buffer supplies the net-positive energy that it accumulated during the normal motor run-time. A physical frame of VAMPIRE is formed by PCBs and soldered pin-socket pairs only. The neighboring PCBs provide a raised wall to protect exposed electrical components on the outside of the PCBs. It also includes the back-EMF coupling plates in its physical structure. A user software is developed using C# and the Universal Windows Platform, targeting Windows 10 environment. The signal processing to extract actionable information from the raw data collected from VAMPIRE is also covered. It is based on the second order mechanical (spring-mass-damper) system, and the resulting empirical vibration transfer function (eVTF) is rich in condition monitoring information. A change in mount stiffness, which implies damaged or degraded mounts, and a change in load imbalance, which implies a damaged load, a changed load profile, or a damaged shaft axle, can be identified, enabling this process to be used for condition-based maintenance for electromechanical diagnosis.

Appendix A

Numerical Solver — Circuit Simulator

This is the code for the numerical solver written in MATLAB for circuit simulation based on the Maxwell method described in Chapter 2.6.4.

A.1 Top Level Runcode

The code below is the top level MATLAB script.

```
clear all;
clc;

tspan = [0 1/60];
M = 100 * (tspan(2)-tspan(1))/(1/60);
dt = (tspan(2)-tspan(1))/M;

t = tspan(1):dt:tspan(2);
i = zeros(1, length(t));

IP = 0.87*sqrt(2);
N = 500;
freq = 60;
w = 2*pi*freq;
Rwire = 0.014*N;
Rload = 6.46e3;
h = 9e-3;
OD = 24.5e-3;
```

```
ID = 16.5e-3;
BSAT = 1.19;
alpha = 2.2;    % beta = 2.2 * 6.44e-2 = 0.14168
Pcoremax = 0.125e-3;

%% Steady-State Solver including Rcore
fprintf('Solving for the steady-state.\n');

state_handle = @(init_condition) expand_state_implicit(t, i, init_condition, IP, N,
    freq, Rwire, Rload, Pcoremax, BSAT, alpha, h, OD, ID);
zero_init = [0;1000e6];
steady_state_condition = engine_newtonNdGCR(state_handle, zero_init, 1e-4, 1e-4);

fprintf('Steady-state found.\n');

%% Drawing Part
fprintf('Evaluating the differential equation.\n');
tspan = [0 3/60];
M = 5000 * (tspan(2)-tspan(1))/(1/60);  % M segments & (M+1) points
dt = (tspan(2)-tspan(1))/M;

t = tspan(1):dt:tspan(2);
i = zeros(1, length(t));

dt = t(2) - t(1);
i(1) = steady_state_condition(1);
Rcore = steady_state_condition(2);

i = expand_time_fullrange(t, i, dt, IP, N, freq, Rwire, Rcore, Rload, Pcoremax, BSAT,
     alpha, h, OD, ID);
v = i*(Rwire+Rload);
iRMS = sqrt(1/(length(t)-1) * sum(i(1:(end-1)).^2));

fprintf('Calculation done.\n');
fprintf('Now plotting.\n');

plot(t,i);
avg_power_mW = iRMS^2*Rload/1e-3
```

A.2 engine_newtonNdGCR()

This is the Newton solver core, which calls the GCR routine for every iteration.

```
function [x0, nf] = engine_newtonNdGCR(fhand, x0, psi_tol, f_tol)

if nargin<4
    error('Must provide four input arguments.  Type ''help newtonNd'' for details');
end

maxIters=500;       % max # of iterations

epsilon_GCR = 0.1;
maxiter_GCR = maxIters;
alpha_GCR = 1e-3;

nf = zeros(maxIters,1);
ndx = zeros(maxIters,1);
x = zeros(max(size(x0)),maxIters);

% Newton loop
for iter=1:maxIters
    F = fhand(x0); % evaluate function
    dx = engine_tgcr(x0, fhand, alpha_GCR, -F, epsilon_GCR, maxiter_GCR);

    nf(iter)=norm(F,2); % norm of f at step k+1
    ndx(iter)=norm(dx,2); % norm of dx at step k+1

        x(:,iter)=x0+dx; % solution x at step k+1
    x0=x(:,iter); % set value for next guess

    if nf(iter) < f_tol && ndx(iter) < psi_tol && (ndx(iter)/norm(x0,2)*0 < psi_tol
        || isnan(ndx(iter)/norm(x0,2)) == 1)
        break; % converged
    end
end

nf = nf(1:iter);

if iter==maxIters,
    fprintf('Non-Convergence after %d iterations!!!\n',iter);
end
```

A.3 engine_tgcr()

This is the GCR routine for constructing the next search direction.

```
function [x, r_norms] = engine_tgcr(x0, fhand, alpha_GCR, b, tol, maxiters)
x = zeros(size(b));

r = b;
r_norms = zeros(maxiters+1,1);
r_norms(1) = norm(r,2);

p = zeros(max(size(r)), maxiters);
Mp = zeros(max(size(r)), maxiters);

for iter = 1:maxiters
  p(:,iter) = r;
  Mp(:, iter) = 1/alpha_GCR*(fhand(x0+alpha_GCR*r) - fhand(x0));

  for j=1:iter-1,
    beta = Mp(:,iter)' * Mp(:,j);
    p(:,iter) = p(:,iter) - beta * p(:,j);
    Mp(:,iter) = Mp(:,iter) - beta * Mp(:,j);
  end;

  norm_Mp = norm(Mp(:,iter),2);
  Mp(:,iter) = Mp(:,iter)/norm_Mp;
  p(:,iter) = p(:,iter)/norm_Mp;

  alpha = r' * Mp(:,iter);
  x = x + alpha * p(:,iter);
  r = r - alpha * Mp(:,iter);

  r_norms(iter+1) = norm(r,2);

  if r_norms(iter+1) < (tol * r_norms(1))
    break; % converged
  end;
end;

if r_norms(iter+1) > (tol * r_norms(1)) % Nonconvergence
  x = [];
end;

r_norms = r_norms / r_norms(1);
r_norms = r_norms(1:iter+1);
```

A.4 expand_state_implicit()

This is the convergence solver for the steady-state and the core loss resistance.

```
function F = expand_state_implicit(t, i, init_condition, IP, N, freq, Rwire, Rload,
    Pcoremax, BSAT, alpha, h, OD, ID)

    dt = t(2) - t(1);
    i(1) = init_condition(1);
    Rcore = init_condition(2);

    i = expand_time_fullrange(t, i, dt, IP, N, freq, Rwire, Rcore, Rload, Pcoremax,
        BSAT, alpha, h, OD, ID);

    iRMS = sqrt(1/(length(t)-1) * sum(i(1:(end-1)).^2));
    vRMS = iRMS*(Rwire+Rload);
    iSAT = 2*pi*(OD/2+ID/2)/2 * alpha * pi/(2*N);
    iPEAK = abs(max(i) * (Rcore+Rwire+Rload)/(Rcore));
    scalingH = min(1, iPEAK/iSAT);

    v = i*(Rwire+Rload);
    Integration_cycle = (t(end)-t(1))/(1/freq);
    BPEAK = 1/(2*2*(OD/2-ID/2)*h*N*Integration_cycle) * sum(abs(v)) * dt;    %
        important to exclude 'dt' outside of sum() for computation speed
    scalingB = min(1, BPEAK/BSAT);

    Pcore = Pcoremax * scalingH * scalingB;

    if Pcore == 0
        Rcore_new = 1000e6;          % Assumes Rcore_init = 1000e6
    else
        Rcore_new = vRMS^2/Pcore;
    end

    F = zeros(2,1);
    F(1) = i(end)-i(1);
    F(2) = Rcore_new - Rcore;
```

A.5 expand_time_fullrange()

This is the time range solver, calling the Newton solver core for every time point.

```
function F = expand_time_fullrange(t, i, dt, IP, N, freq, Rwire, Rcore, Rload,
    Pcoremax, BSAT, alpha, h, OD, ID)

    for n=1:length(t)-1
        i_n = i(n);
        t_n = t(n);

        function_handle = @(i_next) expand_time_implicit(t_n, i_n, dt, i_next, IP, N,
             freq, Rwire, Rcore, Rload, Pcoremax, BSAT, alpha, h, OD, ID);
        i(n+1) = engine_newtonNdGCR(function_handle, i_n, 1e-4, 1e-4);
    end

    F = i;
```

A.6 expand_time_implicit() - Resistive Load

This function describes the load characteristics. Described here is the resistive load case.

```
function F = expand_time_implicit(t_n, i_n, dt, i_next, IP, N, freq, Rwire, Rcore,
    Rload, Pcoremax, BSAT, alpha, h, OD, ID)

    w = 2*pi*freq;
    if Pcoremax == 0
        gamma = 1;
    else
        gamma = (Rcore+Rwire+Rload)/Rcore;
    end

    fhand = @(tau, x) (IP/N*w/gamma*cos(w*tau) - (Rwire+Rload)*x/(gamma*N^2*h*BSAT/2/
        pi^2/alpha * log((OD^2+(IP*sin(w*tau)-gamma*N*x)^2/pi^2/alpha^2)/(ID^2+(IP*
        sin(w*tau)-gamma*N*x)^2/pi^2/alpha^2))/log(exp(1))));
    F = i_next - i_n - 1/2*dt*(fhand(t_n, i_n) + fhand(t_n + dt, i_next));
```

A.7 expand_time_implicit() - Voltage Load

This function describes the load characteristics. Described here is the constant voltage load case.

```
function F = expand_time_implicit(t_n, i_n, dt, i_next, IP, N, freq, Rwire, Rcore,
    Resr, Vload, Pcoremax, BSAT, alpha, h, OD, ID)

    w = 2*pi*freq;
    if Pcoremax == 0
        gamma = 1;
    else
        gamma = (Rcore+Rwire+Resr)/Rcore;
    end

    fhand = @(tau, x) (IP/N*w/gamma*cos(w*tau) - ((Rwire+Resr)*x + Vload*sign(sign(x)
        +0.5))/(gamma*N^2*h*BSAT/2/pi^2/alpha * log((OD^2+(IP*sin(w*tau)-gamma*N*x-N/
        Rcore*Vload*sign(sign(x)+0.5))^2/pi^2/alpha^2)/(ID^2+(IP*sin(w*tau)-gamma*N*x
        -N/Rcore*Vload*sign(sign(x)+0.5))^2/pi^2/alpha^2))/log(exp(1))));

    F = i_next - i_n - 1/2*dt*(fhand(t_n, i_n) + fhand(t_n + dt, i_next));
```

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B

Electromagnetic Energy Harvester and Relevant Circuits — Power Package Hardware

This part describes the schematic drawings and the PCB layout for the power package described in Chapter 4.2. The PCB layout presented here physically makes up a side wall of VAMPIRE. It is shown as 'PCB 1' in Fig. 4-8. This appendix presents:

- Schematic Drawings
- PCB Layout
- Bill of Materials
- Software

B.1 Schematic Drawings

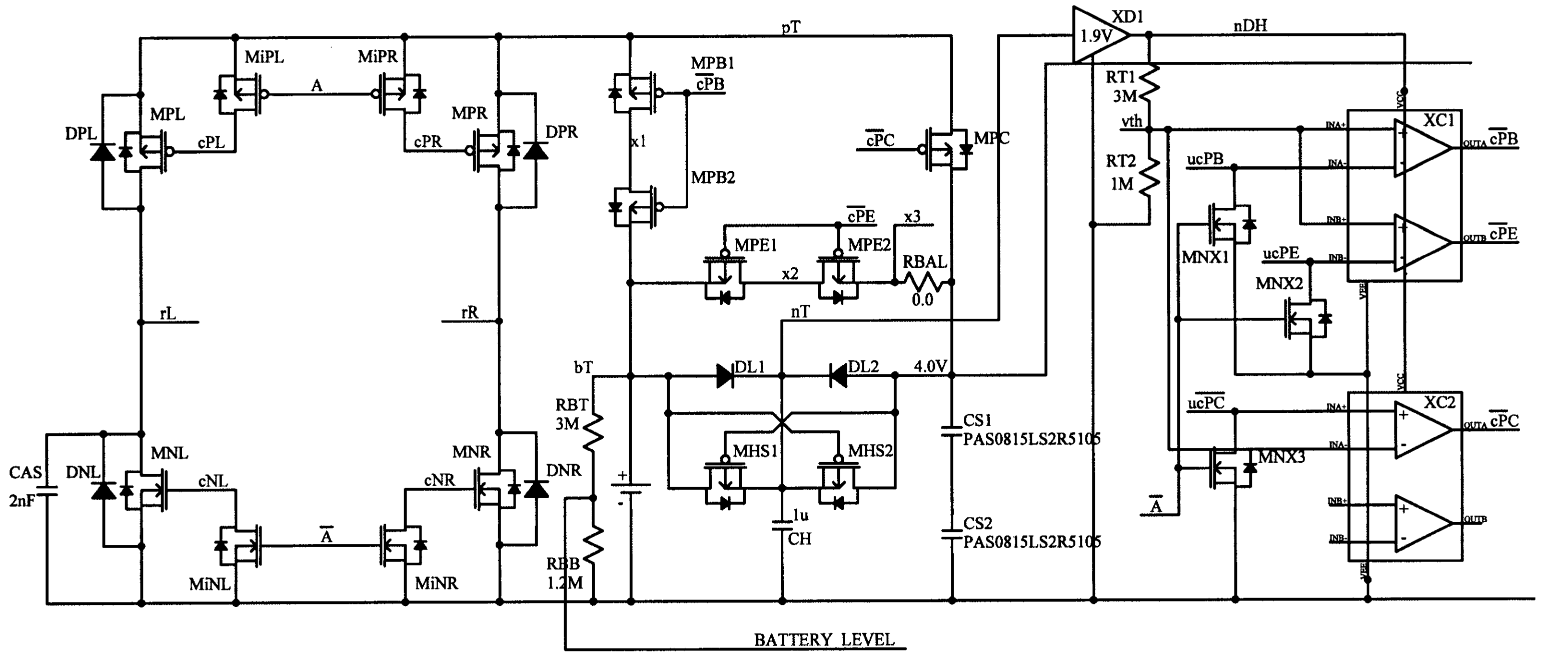


Figure B-1: Schematics of the Power Package - Part 1

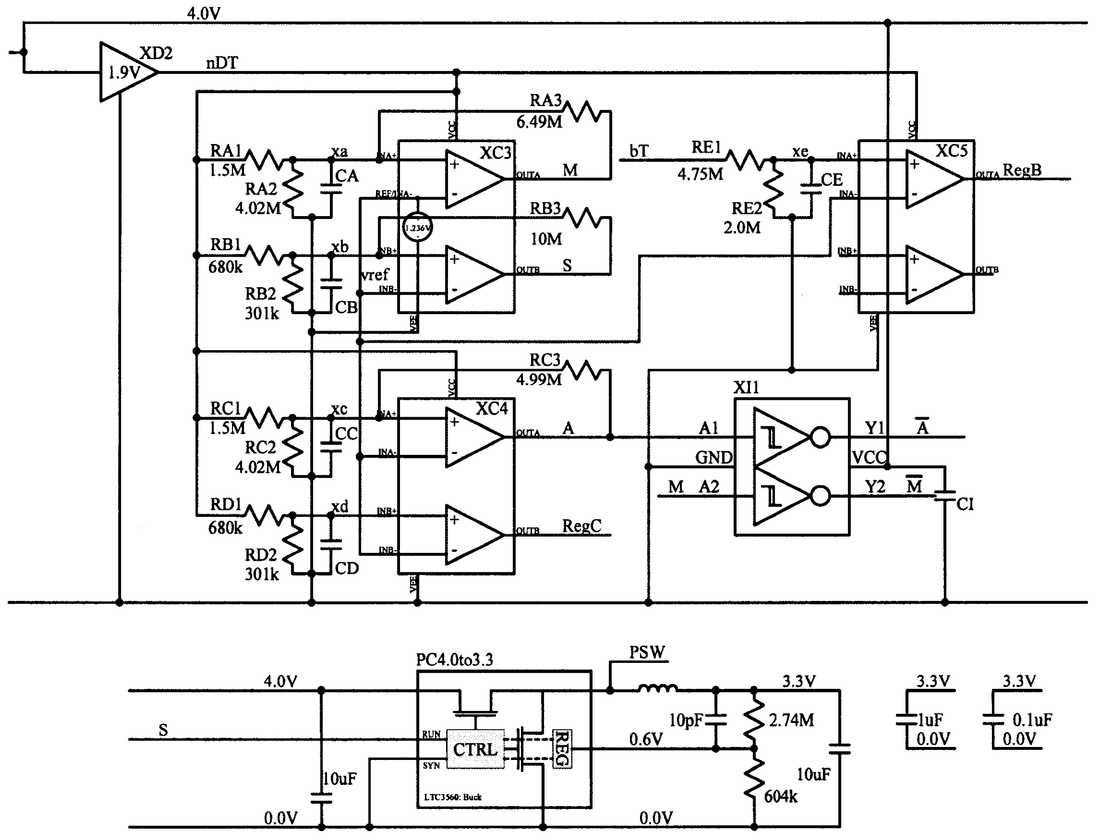


Figure B-2: Schematics of the Power Package - Part 2

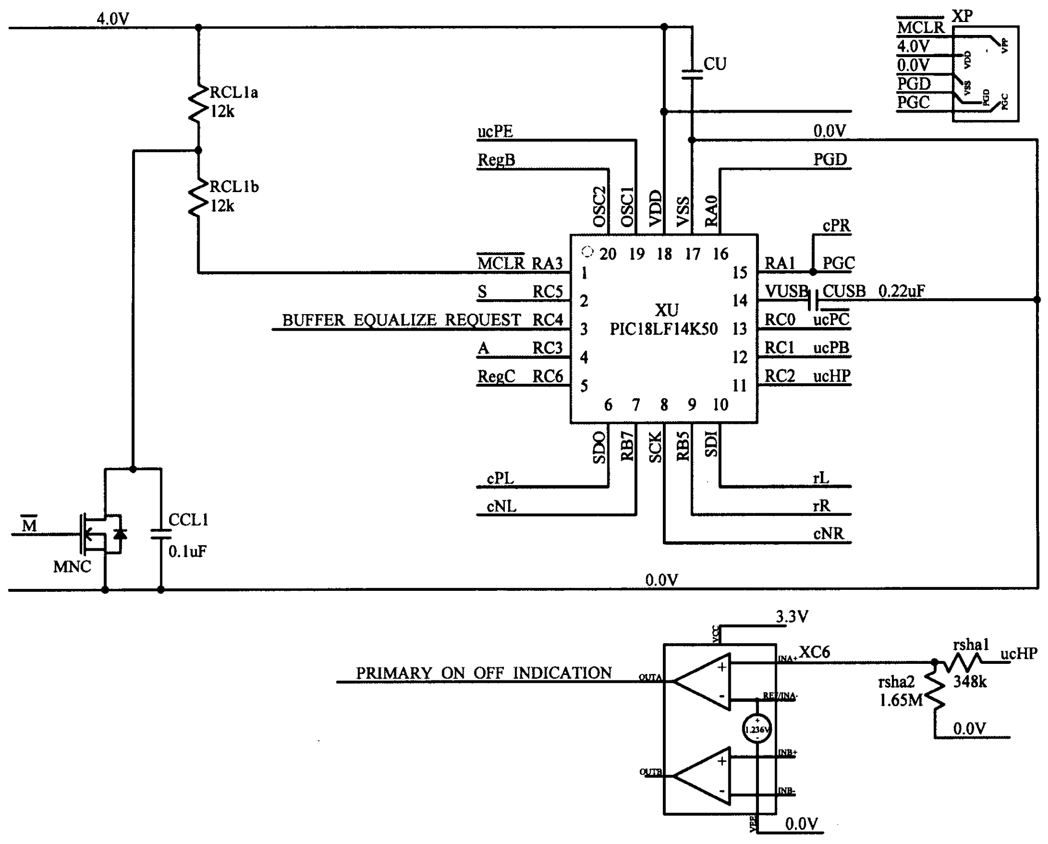


Figure B-3: Schematics of the Power Package - Part 3

B.2 PCB Layout

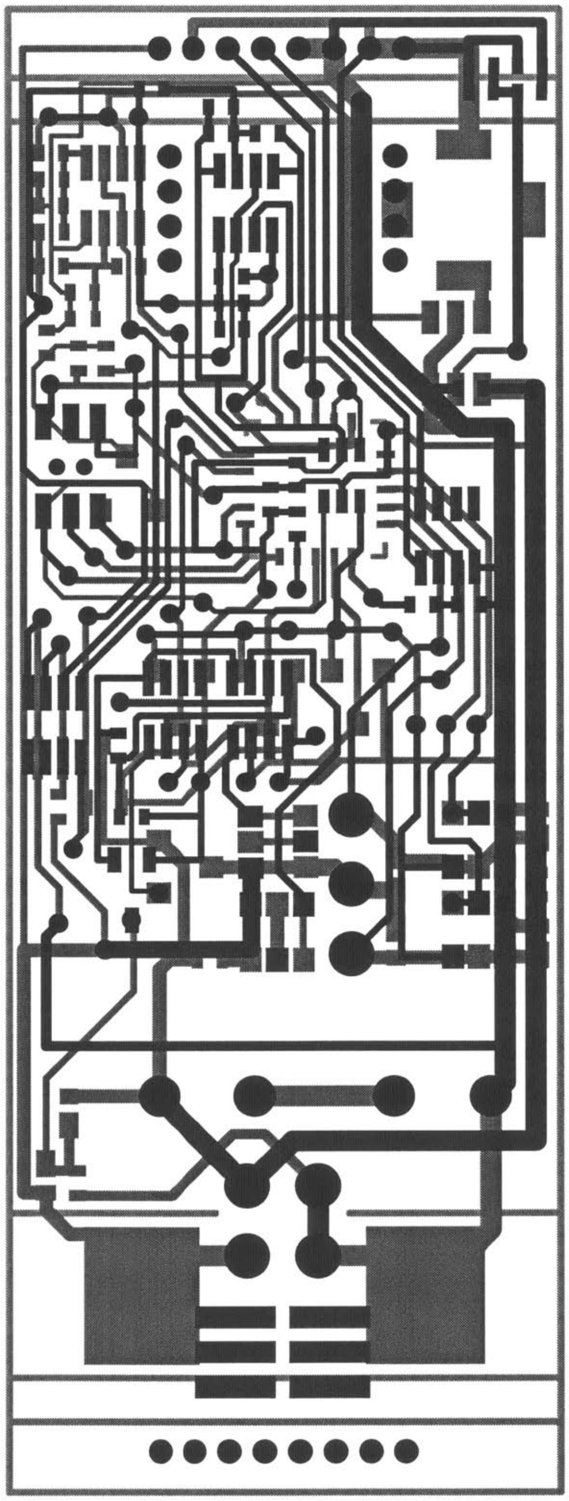


Figure B-4: PCB Layout of the Power Package - Full Stack

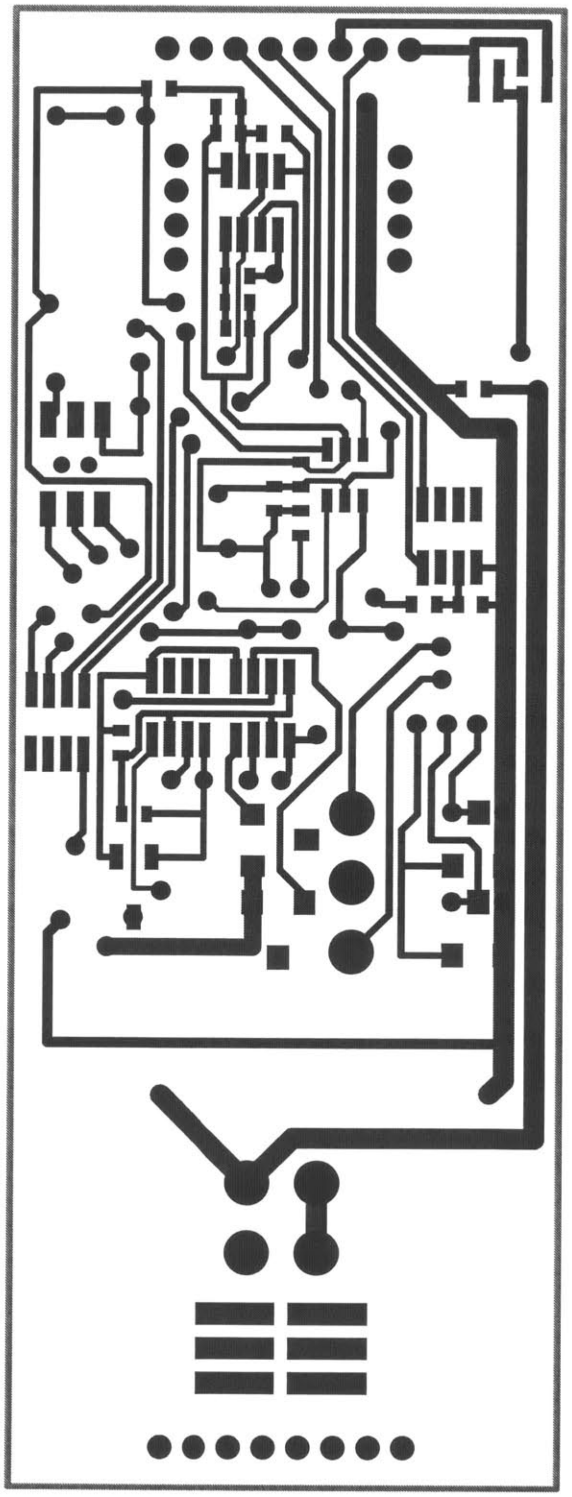


Figure B-5: PCB Layout of the Power Package - Top Copper

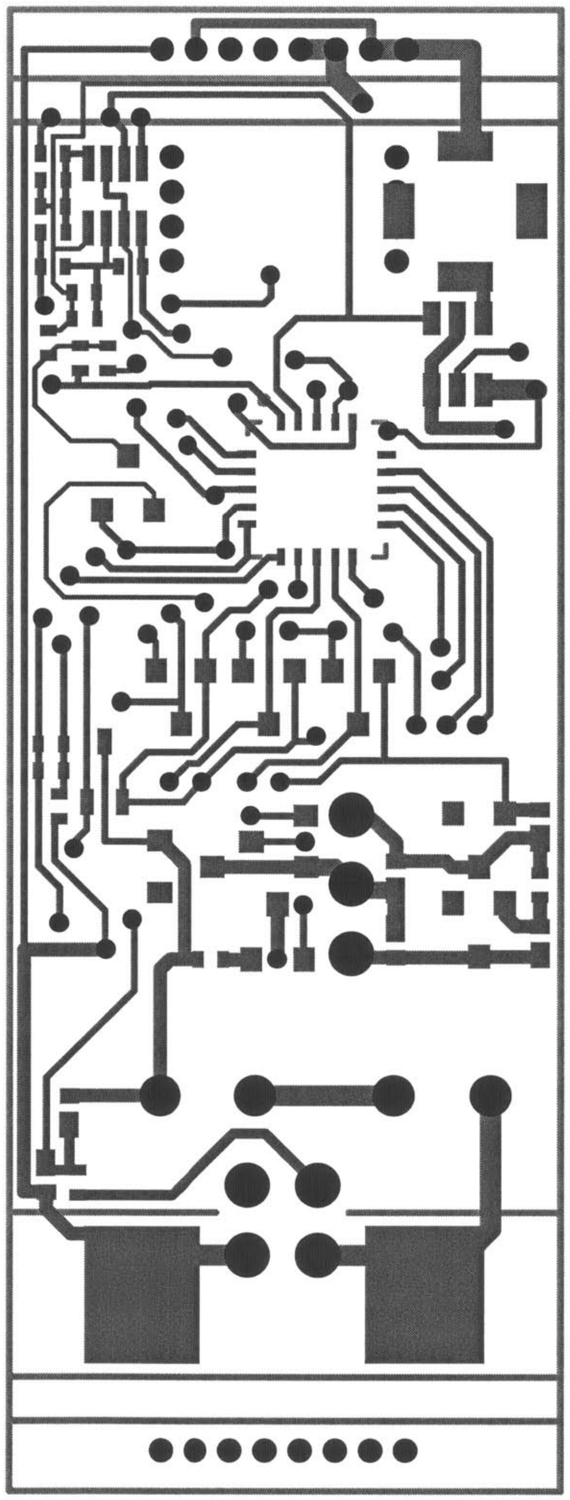


Figure B-6: PCB Layout of the Power Package - Bottom Copper

B.3 Bill of Materials

Table B.1: Bill of Materials for the Power Package

Device	Value	Quantity	Designator
LiPo Battery	220mAh	1	BAT
SMD Capacitor 0402	10pF	1	C1
SMD Capacitor 0402	2nF	1	CAS
SMD Capacitor 0402	0.1uF	8	CA, CB, CC, CD, CE, CCL1, cd2, CI
SMD Capacitor 0402	0.22uF	1	CUSB
SMD Capacitor 0402	1uF	3	cd1, CU, CH
SMD Capacitor 0402	10uF	2	Ci1, C1o
PAS0815LS2R5105	1F	2	CS1, CS2
PMEG2010EPK	PMEG2010EPK	6	DL1, DL2, DNL, DNR, DPL, DPR
SRU5018-100Y	10uH	1	L1
DMG3415U-7	DMG3415U-7	11	MHS1, MHS2, MiPL, MiPR, MPB1, MPB2, MPC, MPE1, MPE2, MPL, MPR
DMG6968U-7	DMG6968U-7	8	MiNL, MiNR, MNC, MNL, MNR, MNX1, MNX2, MNX3
LTC3560	LTC3560	1	PC4.0to3.3
SMD Resistor 0402	0 Ohm	1	RBAL
SMD Resistor 0402	12k Ohm	2	RCL1a, RCL1b
SMD Resistor 0402	301k Ohm	2	RB2, RD2
SMD Resistor 0402	348k Ohm	1	rsha1
SMD Resistor 0402	604k Ohm	1	R2
SMD Resistor 0402	680k Ohm	2	RB1, RD1
SMD Resistor 0402	1.2M Ohm	1	RBB
SMD Resistor 0402	1.5M Ohm	2	RA1, RC1
SMD Resistor 0402	1.65M Ohm	1	rsha2
SMD Resistor 0402	10M Ohm	1	RB3
SMD Resistor 0402	1M Ohm	1	RT2
SMD Resistor 0402	2.74M Ohm	1	R1
SMD Resistor 0402	2M Ohm	1	RE2
SMD Resistor 0402	3M Ohm	2	RBT, RT1
SMD Resistor 0402	4.02M Ohm	2	RA2, RC2
SMD Resistor 0402	4.75M Ohm	1	RE1
SMD Resistor 0402	4.99M Ohm	1	RC3
SMD Resistor 0402	6.49M Ohm	1	RA3
MAX9019EKA+T	MAX9019EKA+T	4	XC1, XC2, XC4, XC5
MAX9017AEKA+T	MAX9017AEKA+T	2	XC3, XC6
MCP103T-195I/LB	MCP103T-195I/LB	2	XD1, XD2
NC7WZ14EP6X	NC7WZ14EP6X	1	XI1
M40-3200345R	M40-3200345R	1	XP
PIC18LF14K50-MCU	PIC18LF14K50-MCU	1	XU
LPPB041NGCN-RC	LPPB041NGCN-RC	2	Upper Structural Mounts 4-pin
851-87-003-40-252101	851-87-003-40-252101	2	Lower Structural Mounts 3-pin

B.4 Software

The software is written in C with MPLAB as the microcontroller in the power package is Microchip's PIC processor. The compiler used in MPLAB is CCS.

```
#include "18F14K50.h"
#device PIC18F14K50 ADC=8

#fuses USBDIV1, CPUDIV1
#fuses NOIESO, NOFCMEN, NOPCLKEN, NOPLLEN, INTRC_IO
#fuses BORV19, NOBROWNOUT, NOPUT
#fuses WDT256, NOWDT
#fuses MCLR, HFOFST
#fuses NODEBUG, NOXINST, NOLVP, NOSTVREN
#fuses NOPROTECT
#fuses NOCPD, NOCPB
#fuses NOWRT
#fuses NOWRTD, NOWRTB, NOWRTC
#fuses NOEBTR
#fuses NOEBTRB

#use delay(clock=8000000)

#define PIN_READ_L PIN_B4
#define PIN_READ_R PIN_B5
#define PIN_CTRL_NL PIN_B7
#define PIN_CTRL_NR PIN_B6

#define PIN_CTRL_uB PIN_C1
#define PIN_CTRL_uC PIN_C0
#define PIN_CTRL_uE PIN_A5

#define PIN_OUTPUT_P PIN_C2

#define PIN_INPUT_M PIN_A3
#define PIN_INPUT_A PIN_C3
#define PIN_INPUT_S PIN_C5

#define PIN_INPUT_REGB PIN_A4
#define PIN_INPUT_REGC PIN_C6

#define PIN_INPUT_REQ_BATT_EQ PIN_C4

#define TDISCONNECTED 25000 // (1/120Hz) / (1/(8MHz/4)) * (1 + 50%) = 25000 cycles
#define TDISCONNECTED_LONG 41667 // (1/120Hz) / (1/(8MHz/4)) * (2 + 50%) = 41667
```

```
    cycles
#define TSHUTOFF_UNIT 25000 // 4us * 25000 = 0.1s
#define TSHUTOFF_COUNT 50 // deciseconds
#define TCLEAREQ_UNIT 25000
#define TCLEAREQ_COUNT 50

#byte WDTCON = 0xFD1
#byte RCON = 0xFD0
#bit SWDTEN = WDTCON.0
#bit TOb = RCON.3


void init_MCU() {
 setup_oscillator(OSC_PLL_OFF | OSC_8MHZ | OSC_INTRC);
 setup_spi(SPI_SS_DISABLED);
 setup_timer_1(T1_INTERNAL | T1_DIV_BY_1);
 setup_timer_3(T3_DISABLED);
 delay_ms(80);
}

void init_ports() {
 output_bit(PIN_CTRL_PL, 1); // PMOS (default:off by H) - RECTIFIER UPPER LEFT
 output_bit(PIN_CTRL_PR, 1); // PMOS (default:off by H) - RECTIFIER UPPER RIGHT
 output_bit(PIN_CTRL_NL, 0); // NMOS (default:off by L) - RECTIFIER LOWER LEFT
 output_bit(PIN_CTRL_NR, 0); // NMOS (default:off by L) - RECTIFIER LOWER RIGHT

 output_bit(PIN_CTRL_uB, 0); // Externally Initialized to 0 // Disconnects the
     battery from the rectifier
 output_bit(PIN_CTRL_uC, 0); // Externally Initialized to 0 // Connects the supercap
     to the rectifier
 output_bit(PIN_CTRL_uE, 0); // Externally Initialized to 0 // Disconnects the
     equalizing path

 input(PIN_INPUT_M);
 input(PIN_INPUT_A);
 input(PIN_INPUT_S);
 input(PIN_INPUT_REGB);
 input(PIN_INPUT_REGC);

 output_bit(PIN_OUTPUT_P, 0);
}

void main(void) {
 unsigned int8 dataADC_L = 0;
 unsigned int8 dataADC_R = 0;
unsigned int1 currL = 0;
unsigned int1 currR = 0;
unsigned int1 prevL = 0;
unsigned int1 prevR = 0;
unsigned int1 RedgeL = 0;
unsigned int1 RedgeR = 0;
unsigned int1 FedgeL = 0;
unsigned int1 FedgeR = 0;

unsigned int8 state = 0;
unsigned int16 tSAT_counter = 0;
unsigned int16 tHALF_counter = 0;
unsigned int16 OTF_counter = 0;
unsigned int1 regulate_cap = 0;
unsigned int1 regulate_battery = 0;
unsigned int1 sensor_on = 0;
unsigned int1 spin_down = 0;
unsigned int1 retain_battery = 0;
unsigned int1 charging_mode = 0; // 0 : supercap 1: battery
unsigned int16 disconnect_counter = 0;
unsigned int1 first_cycle = 1;

unsigned int16 shut_off_counter = 0;
unsigned int1 complete_shutoff = 0;
unsigned int1 request_equalize = 0;
unsigned int16 take_eq_counter = 0;

init_MCU(); delay_ms(80);
init_ports(); delay_ms(10);

for(;;) {
 if(state == 0) {
  init_ports();
  state = input(PIN_INPUT_A);
  first_cycle = 1;
 }
 else if(state == 1) {
  setup_adc_ports(sAN10 | sAN11);
  setup_adc(ADC_CLOCK_DIV_2);
  state = 2;
  set_timer1(0);
  first_cycle = 1;
 }
 else {
  set_adc_channel(10); delay_us(10); dataADC_L = read_adc();
set_adc_channel(11); delay_us(10); dataADC_R = read_adc();

prevL = currL;
prevR = currR;

if( (dataADC_L > 223) && (dataADC_R < 32) ) { currL = 1; currR = 0; }
else if( (dataADC_L < 32) && (dataADC_R > 223)) { currL = 0; currR = 1; }
else { currL = 0; currR = 0; }

RedgeL = currL > prevL;
RedgeR = currR > prevR;
FedgeL = currL < prevL;
FedgeR = currR < prevR;

if(state == 2) {
 if(RedgeL) {
  set_timer1(0);
  if(spin_down == 1) {
   charging_mode = 0;
   output_bit(PIN_CTRL_uE, request_equalize);
   output_bit(PIN_CTRL_uB, 0);
   output_bit(PIN_CTRL_uC, 0);
  }
  spin_down = 0;
  setup_timer_3(T3_DISABLED);
  take_eq_counter = 0;
  output_bit(PIN_OUTPUT_P, 0);
  state = 3;
 }

 if(spin_down == 1) {
  retain_battery = input(PIN_INPUT_S);
  if(retain_battery == 0) { // Deep shutdown by critical VBATTERY
   charging_mode = 0;
   output_bit(PIN_CTRL_uE, 0);
   output_bit(PIN_CTRL_uB, 0);
   output_bit(PIN_CTRL_uC, 0);
   setup_timer_3(T3_DISABLED);
  }
  else {
   if(get_timer3() > TCLEAREQ_UNIT) {
    take_eq_counter = take_eq_counter + 1;
    set_timer3(0);
   }
   if(take_eq_counter > TCLEAREQ_COUNT) {
        request_equalize = input(PIN_INPUT_REQ_BATT_EQ);
        if(request_equalize == 0) {
         output_bit(PIN_CTRL_uE, 0);
         complete_shutoff = 1;
        }
        else {
         output_bit(PIN_CTRL_uE, 1);
        }
       }
      }
     }
    }
    else if(state == 3) {
     first_cycle = 0;
     complete_shutoff = 0;
     if(FedgeL) {
      tSAT_counter = get_timer1();
      state = 4;
     }
    }
    else if(state == 4) {
     if(FedgeL) { // it will never occur from state 3, only from 9
      tSAT_counter = get_timer1() - OTF_counter;
      tSAT_counter = tSAT_counter - (tSAT_counter >> 5);
     }
     if(RedgeR) {
      tHALF_counter = get_timer1();
      set_timer1(0);
      output_bit(PIN_CTRL_PL, 1);
      output_bit(PIN_CTRL_PR, 1);
      output_bit(PIN_CTRL_NL, 1);
      output_bit(PIN_CTRL_NR, 1);
      state = 5;
     }
    }
    else if(state == 5) {
     if(get_timer1() > (tHALF_counter >> 1) - (tSAT_counter >> 1) ) {
      OTF_counter = get_timer1();
      output_bit(PIN_CTRL_NR, 0);
      output_bit(PIN_CTRL_PR, 0);
      state = 6;
     }
    }
    else if(state == 6) {
     if(FedgeR) { // if saturation happens before timer
```

```
   output_bit(PIN_CTRL_PR, 1);
   tSAT_counter = get_timer1() - OTF_counter; // update tSAT
   tSAT_counter = tSAT_counter - (tSAT_counter >> 5);
   output_bit(PIN_CTRL_NL, 0); // Now diodes only for rectifier
   state = 7;
  }
  if(get_timer1() > (tHALF_counter >> 1) + (tSAT_counter >> 1) - (tSAT_counter >>
      5)) {
   output_bit(PIN_CTRL_PR, 1);
   output_bit(PIN_CTRL_NL, 0);
   state = 7;
  }
 }
 else if(state == 7) {
  if(FedgeR) { // in case of earlier time-up than saturation
   tSAT_counter = get_timer1() - OTF_counter; // update tSAT
   tSAT_counter = tSAT_counter - (tSAT_counter >> 5);
  }
  if(RedgeL) {
   tHALF_counter = get_timer1();
   set_timer1(0);
   output_bit(PIN_CTRL_PL, 1);
   output_bit(PIN_CTRL_PR, 1);
   output_bit(PIN_CTRL_NL, 1);
   output_bit(PIN_CTRL_NR, 1);
   state = 8;
  }
 }
 else if(state == 8) {
  if(get_timer1() > (tHALF_counter >> 1) - (tSAT_counter >> 1) ) {
   OTF_counter = get_timer1();
   output_bit(PIN_CTRL_NL, 0);
   output_bit(PIN_CTRL_PL, 0);
   state = 9;
  }
 }
 else if(state == 9) {
  if(FedgeL) {
   output_bit(PIN_CTRL_PL, 1); // Float PL to avoid leak
   tSAT_counter = get_timer1() - OTF_counter;
   tSAT_counter = tSAT_counter - (tSAT_counter >> 5);
   output_bit(PIN_CTRL_NR, 0); // Cut off active paths
   state = 4;
  }
  if(get_timer1() > (tHALF_counter >> 1) + (tSAT_counter >> 1) - (tSAT_counter >>
      5)) {
    output_bit(PIN_CTRL_PL, 1);
    output_bit(PIN_CTRL_NR, 0);
    state = 4;
   }
   if(state == 4) {
    request_equalize = input(PIN_INPUT_REQ_BATT_EQ);
    output_bit(PIN_CTRL_uE, request_equalize);

    sensor_on = input(PIN_INPUT_S);
    if(sensor_on == 0) {
     regulate_cap = 0;
    }
    else {
     regulate_cap = input(PIN_INPUT_REGC);
    }

    if(regulate_cap == 1) {
    state = 10;
    }
    else {
     if(charging_mode == 1) { // No regulation for the supercap but
      state = 2; // previously in the battery charging mode:
      charging_mode = 0; // supercap charging mode
      output_bit(PIN_CTRL_uE, request_equalize);
      output_bit(PIN_CTRL_uB, 0);
      output_bit(PIN_CTRL_uC, 0);
     }
    }
   }
  }
  else if(state == 10) { // Regulation Control Stage #2
   regulate_battery = input(PIN_INPUT_REGB);

   if(regulate_battery == 1) {
    output_bit(PIN_CTRL_uC, 1);
    output_bit(PIN_CTRL_uB, 0);
    output_bit(PIN_CTRL_uE, request_equalize);

    output_bit(PIN_CTRL_PL, 1);
    output_bit(PIN_CTRL_PR, 1);
    output_bit(PIN_CTRL_NL, 1);
    output_bit(PIN_CTRL_NR, 1);

    regulate_cap = input(PIN_INPUT_REGC); // Check again
    if(regulate_cap == 0) {
     state = 2;
     output_bit(PIN_CTRL_NL, 0);
     output_bit(PIN_CTRL_NR, 0); // diodes-only stage

     charging_mode = 0; // supercap charging mode
     output_bit(PIN_CTRL_uE, request_equalize);
     output_bit(PIN_CTRL_uB, 0);
     output_bit(PIN_CTRL_uC, 0); // Connect the supercap
    }
   }
   else {
    if(charging_mode == 1) {
     state = 4;
    }
    else {
     state = 2;
     charging_mode = 1; // battery charging mode
     output_bit(PIN_CTRL_NL, 0);
     output_bit(PIN_CTRL_NR, 0);

     output_bit(PIN_CTRL_uE, request_equalize);
     output_bit(PIN_CTRL_uC, 1);
     output_bit(PIN_CTRL_uB, 1);
    }
   }
  }

  disconnect_counter = get_timer1();
  if( ((state == 2) && (first_cycle == 0) && (complete_shutoff == 0) && (
      disconnect_counter > TDISCONNECTED_LONG) && (take_eq_counter == 0)) || ((state
       > 2) && (state < 11) && (disconnect_counter > TDISCONNECTED)) ) {
   if(input(PIN_INPUT_S)) {
    output_bit(PIN_OUTPUT_P, 1);
    spin_down = 1;
    state = 2;
    output_bit(PIN_CTRL_PL, 1);
    output_bit(PIN_CTRL_PR, 1);
    output_bit(PIN_CTRL_NL, 0);
    output_bit(PIN_CTRL_NR, 0);

    output_bit(PIN_CTRL_uC, 1);
    output_bit(PIN_CTRL_uB, 0);
    output_bit(PIN_CTRL_uE, 1);
    setup_timer_3(T3_INTERNAL | T3_DIV_BY_8);
```

```
      set_timer3(0);
      shut_off_counter = 0;
      take_eq_counter = 1;
     }
    }
    else if ((state == 2) && (first_cycle == 1) && (complete_shutoff == 0) && (
        disconnect_counter > TDISCONNECTED_LONG)) {
     request_equalize = input(PIN_INPUT_REQ_BATT_EQ);
     output_bit(PIN_CTRL_uE, request_equalize);
    }
   }
  }
 }
```

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix C

Data Sampling Circuits — Sensors Hardware and Sensor Microcontroller

This part describes the schematic drawings and the PCB layouts for the data sampling circuits described in Chapter 4.3.1. The PCB layouts presented here physically make up a side wall and a bottom plate of VAMPIRE. They are shown as 'PCB 3' and 'PCB 4' in Fig. 4-8. The bottom plate 'PCB 4', shown in Fig. C-5 in detail, only contains the capacitive coupling plates without any physical component, and the rest of the data sampling circuits is placed on 'PCB 3', shown in Fig. C-2 in detail. This appendix presents:

- Schematic Drawings
- PCB Layout
- Bill of Materials
- Software

C.1 Schematic Drawings

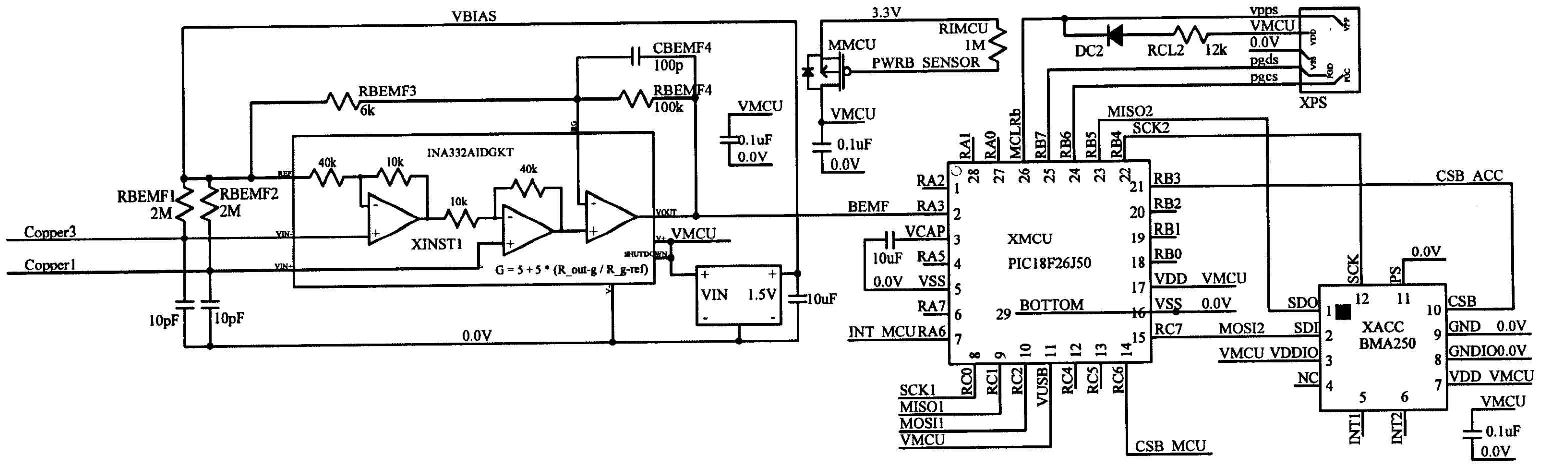


Figure C-1: Schematics of the Data Sampler

C.2 PCB Layout

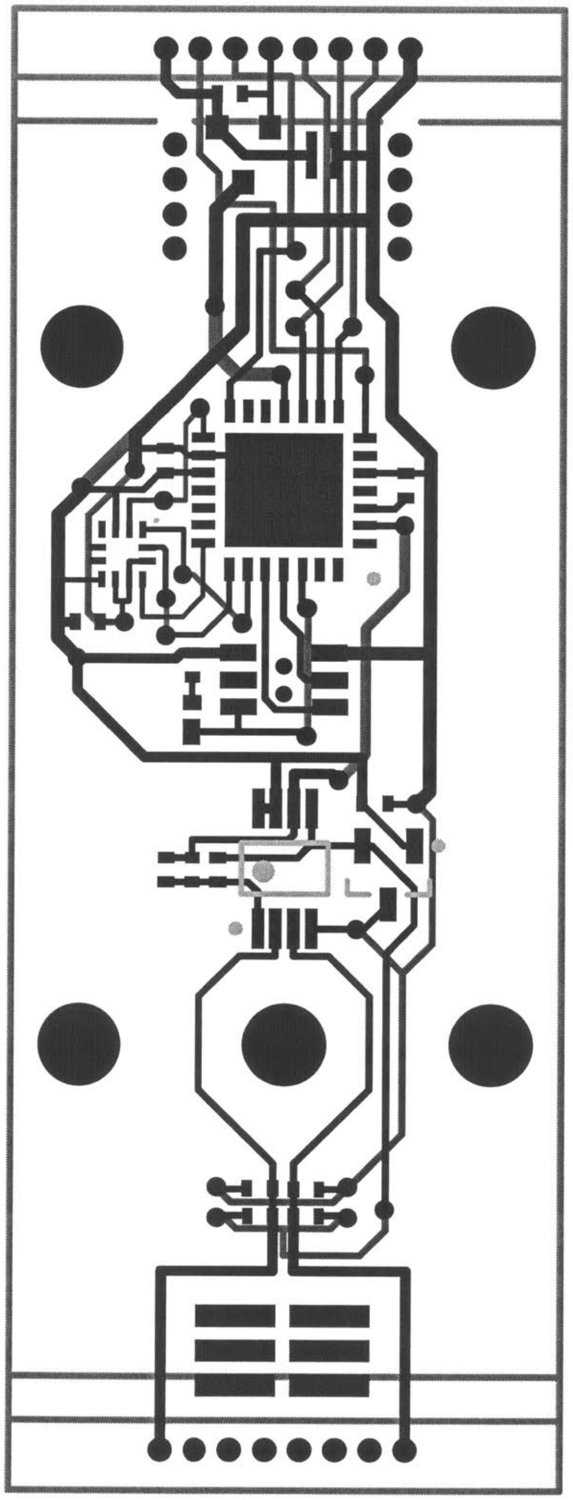


Figure C-2: PCB Layout of the Data Sampler - Full Stack

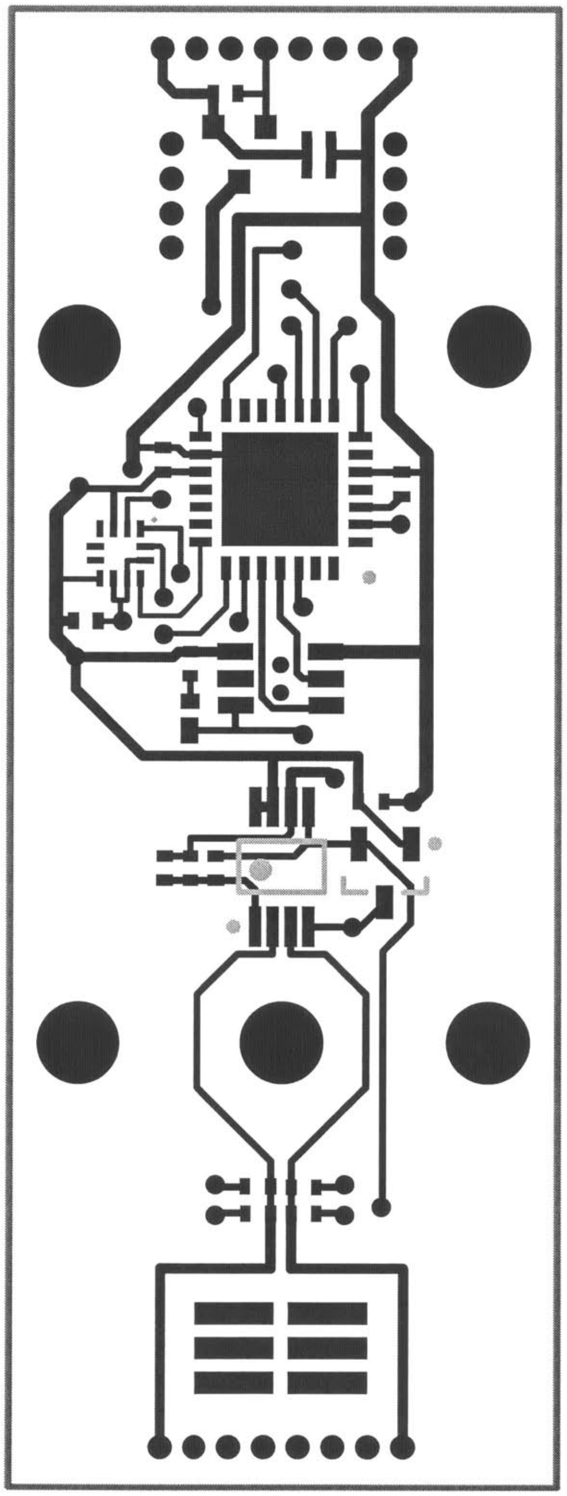


Figure C-3: PCB Layout of the Data Sampler - Top Copper

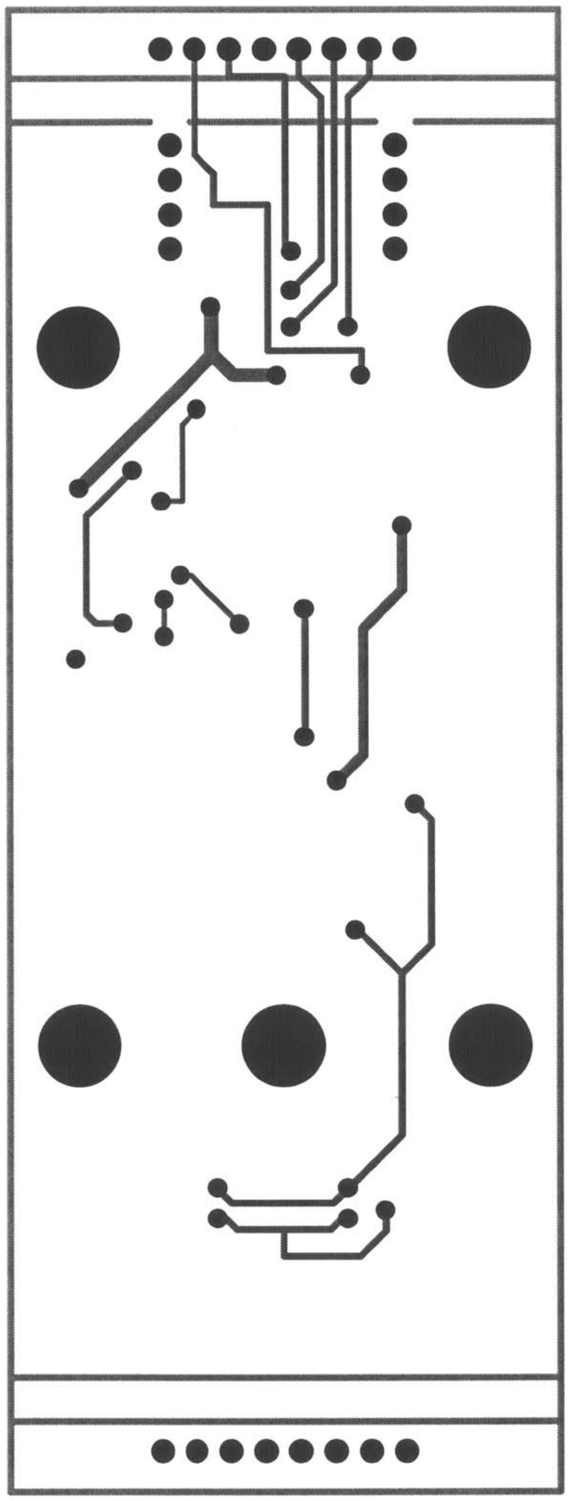


Figure C-4: PCB Layout of the Data Sampler - Bottom Copper

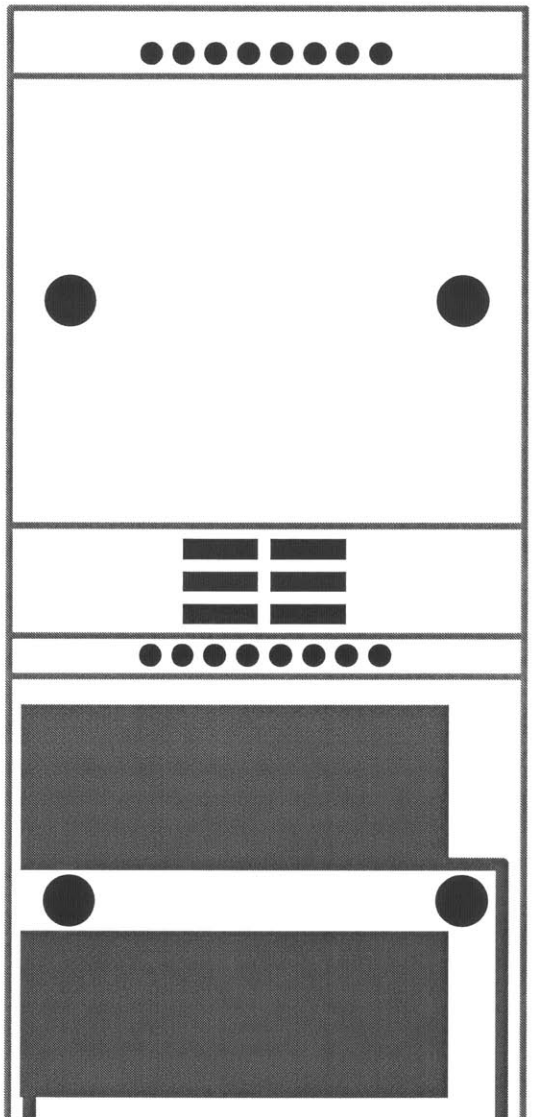


Figure C-5: PCB Layout of the Back-EMF Plate Board - Full Stack

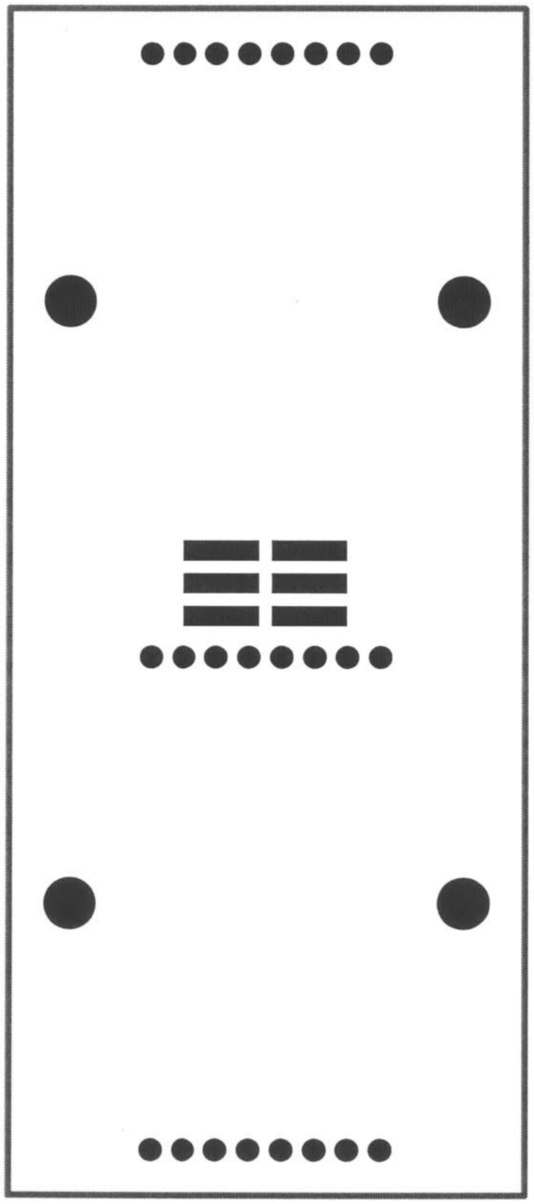


Figure C-6: PCB Layout of the Back-EMF Plate Board - Top Copper

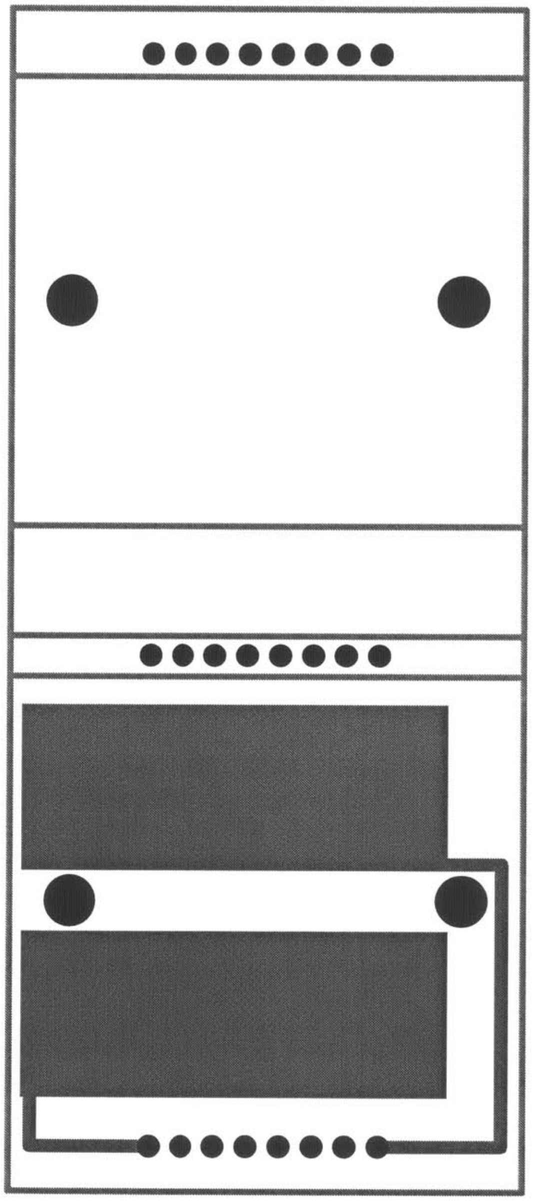


Figure C-7: PCB Layout of the Back-EMF Plate Board - Bottom Copper

C.3 Bill of Materials

Table C.1: Bill of Materials for the Data Sampler

Device	Value	Quantity	Designator
SMD Capacitor 0402	10pF	2	CBEMF1, CBEMF2
SMD Capacitor 0402	100pF	1	CBEMF4
SMD Capacitor 0402	10uF	2	cbias, ccapm
SMD Capacitor 0402	0.1uF	3	cdacc, cdbemf, cdmcu
PMEG2010EPK	PMEG2010EPK	1	DC2
DMG3415U-7	DMG3415U-7	1	MMCU
SMD Resistor 0402	2M Ohm	2	RBEMF1, RBEMF2
SMD Resistor 0402	6.04k Ohm	1	RBEMF3
SMD Resistor 0402	12k Ohm	1	RCL2
SMD Resistor 0402	100k Ohm	1	RBEMF4
SMD Resistor 0402	1M Ohm	1	RIMCU
BMA250	BMA250	1	XACC
INA332AIDGKT	INA332AIDGKT	1	XINST1
PIC18F26J50	PIC18F26J50	1	XMCU
M40-3200345R	M40-3200345R	1	XPS
ISL21080CIH315Z-TK	ISL21080CIH315Z-TK	1	XR
LPPB041NGCN-RC	LPPB041NGCN-RC	2	Upper Structural Mounts 4-pin
851-87-003-40-252101	851-87-003-40-252101	4	Lower Structural Mounts 3-pin
GRPB081VWCN-RC	GRPB081VWCN-RC	2	Interboard Mounts 8-pin

C.4 Software

The software is written in C. The compiler used is CCS.

```
#include "18F26J50.h"
#device  ADC=10

#fuses  NOWDT, PLL1, NOSTVREN, NOXINST, NODEBUG
#fuses  NOCPUDIV, NOPROTECT
#fuses  INTRC_IO, NOT1DIG, NOLPT1OSC, NOFCMEN, NOIESO
#fuses  WDT32768
#fuses  DSWDTOSC_INT, RTCOSC_INT, NODSBOR, NODSWDT, DSWDT2
#fuses  NOIOL1WAY, MSSPMSK5
#fuses  NOWPFP, WPEND, NOWPCFG
#fuses  WPDIS
#use  delay(clock=8000000)

#pin_select SCK2OUT = PIN_C0
#pin_select SCK2IN = PIN_C0
#pin_select SDI2 = PIN_C2
#pin_select SDO2 = PIN_C1
#pin_select SS2IN = PIN_C6

#byte  DMACON1 = 0xF88
#byte  DMACON2 = 0xF86
#byte  DMABCH = 0xF66
#byte  DMABCL = 0xF67
#byte  TXADDRH = 0xF6A
#byte  TXADDRL = 0xF6B
#bit  DMAEN = DMACON1.0

#define  SCK1      PIN_B4
#define  SDO1      PIN_C7
#define  SDI1      PIN_B5

#define  SCK2      PIN_C0
#define  SDI2      PIN_C2
#define  SDO2      PIN_C1

#define  CSB1_ACC     PIN_B3
#define  ACC_ON()     output_bit(CSB1_ACC, 0)
#define  ACC_OFF()     output_bit(CSB1_ACC, 1)

#define  BEMF      PIN_A3
#define  INT_SAMPLE_READY  PIN_A6
```

```
#define  T2_DIVISION     T2_DIV_BY_1  // 1/(8MHz/4/T2_DIVISION) * T2_OVERFLOW *
    T2_REPETITION
#define  T2_OVERFLOW     200     // PIC interrupt happens 0.5us*DIV*OVER*REP
#define  T2_REPETITION    5     // Actual data sampling freq = 1/(0.5us*DIV*OVER*REP)

#reserve 0x500:0xCFF  // 1st set: 0x500 to 0x8FF, 2nd set: 0x900 to 0xCFF
#byte  dma_set1  =  0x500
#byte  dma_set2  =  0x900

#define  LEDU      PIN_B1
#define  LEDL      PIN_B0
#define  DEC        10
#define  HEX        16

int8 ACC_X_MSB=0;
int8 ACC_X_LSB=0;
int8 ACC_Y_MSB=0;
int8 ACC_Y_LSB=0;
int8 ACC_Z_MSB=0;
int8 ACC_Z_LSB=0;
int8 ACC_T=0;
int16 rADC_BEMF=0;
int8* dma_ptr;

void MCU_init() {
 setup_oscillator(OSC_PLL_OFF | OSC_8MHZ | OSC_NORMAL);
 delay_ms(100);
 setup_adc_ports(sAN3);
 setup_adc(ADC_CLOCK_DIV_2);
 set_adc_channel(3);
 delay_ms(10);
 enable_interrupts(GLOBAL);
 dma_ptr = &dma_set1;
}

void MSSP_init() {
 output_bit(SCK1, 0); output_bit(SDO1, 0);    input(SDI1);
 output_bit(CSB1_ACC, 1);
 output_bit(INT_SAMPLE_READY, 0);
 setup_spi(SPI_MASTER | SPI_CLK_DIV_4 | SPI_SCK_IDLE_LOW | SPI_SAMPLE_AT_MIDDLE |
     SPI_XMIT_L_TO_H);
 input(SCK2);    output_bit(SDO2, 0);    input(SDI2);
 setup_spi2(SPI_SLAVE | SPI_SCK_IDLE_LOW | SPI_SAMPLE_AT_MIDDLE | SPI_XMIT_L_TO_H);
 DMACON1 = 0b00110100;
 DMACON2 = 0b00000000;
 delay_ms(10);
}

void ACC_init() {
 ACC_ON();
  spi_write(0x0F);  // Range selection
  spi_write(0x05);  // 05: +-4g, 03: +-2g // 08: +-8g, 0C: +-16g
 ACC_OFF();

 ACC_ON();
  spi_write(0x10);  // Bandwidth Configuration
  spi_write(0x0F);
 ACC_OFF();
}

void Timer_init() {
 setup_timer_2(T2_DIVISION, T2_OVERFLOW - 1, T2_REPETITION);
 delay_ms(10);
 enable_interrupts(INT_TIMER2);
 set_timer2(0);
}

void Data_sample() {
 ACC_ON();
  spi_write(0x82);
  ACC_X_LSB = spi_read(0x00);  // XLSB
  ACC_X_MSB = spi_read(0x00);  // XMSB
  ACC_Y_LSB = spi_read(0x00);  // YLSB
  ACC_Y_MSB = spi_read(0x00);  // YMSB
  ACC_Z_LSB = spi_read(0x00);  // ZLSB
  ACC_Z_MSB = spi_read(0x00);  // ZMSB
  ACC_T  = spi_read(0x00);  // NEED TO: (VAL+48) >> 1 => Celsius
 ACC_OFF();
 rADC_BEMF = read_adc();
}

#int_timer2
void timer2_interrupt_routine() {
 static int8 i=0;
 static int8* temp_ptr;

 output_bit(PIN_A0, 1);
 delay_us(10);
 output_bit(PIN_A0, 0);
```

```
Data_sample();
*(dma_ptr+0) = ACC_X_MSB;
*(dma_ptr+1) = ACC_Y_MSB;
*(dma_ptr+2) = ACC_Z_MSB;
*(dma_ptr+3) = ((rADC_BEMF) >> 2) & 0xFF;
*(dma_ptr+4) = (ACC_X_LSB & 0xC0) + ((ACC_Y_LSB & 0xC0) >> 2) + ((ACC_Z_LSB & 0xC0)
    >> 4) + (rADC_BEMF & 0x03);
dma_ptr = dma_ptr + 5;

if(dma_ptr == &dma_set1 + 500) {
 *(dma_ptr) = ACC_T;

 temp_ptr = &dma_set1;
 *(dma_ptr+1) = *(temp_ptr + 0);
 *(dma_ptr+2) = *(temp_ptr + 1);
 *(dma_ptr+3) = *(temp_ptr + 2);
 *(dma_ptr+4) = *(temp_ptr + 3);
 *(dma_ptr+5) = ((*(temp_ptr + 4)) & 0xFC) + ((ACC_T & 0xC0) >> 6);
 *(dma_ptr+6) = *(temp_ptr + 125);
 *(dma_ptr+7) = *(temp_ptr + 126);
 *(dma_ptr+8) = *(temp_ptr + 127);
 *(dma_ptr+9) = *(temp_ptr + 128);
 *(dma_ptr+10) = ((*(temp_ptr + 129)) & 0xFC) + ((ACC_T & 0x30) >> 4);
 *(dma_ptr+11) = *(temp_ptr + 250);
 *(dma_ptr+12) = *(temp_ptr + 251);
 *(dma_ptr+13) = *(temp_ptr + 252);
 *(dma_ptr+14) = *(temp_ptr + 253);
 *(dma_ptr+15) = ((*(temp_ptr + 254)) & 0xFC) + ((ACC_T & 0x0C) >> 2);
 *(dma_ptr+16) = *(temp_ptr + 375);
 *(dma_ptr+17) = *(temp_ptr + 376);
 *(dma_ptr+18) = *(temp_ptr + 377);
 *(dma_ptr+19) = *(temp_ptr + 378);
 *(dma_ptr+20) = ((*(temp_ptr + 379)) & 0xFC) + ((ACC_T) & 0x03);

 dma_ptr = &dma_set2;
 TXADDRH = 0x05;
 TXADDRL = 0x00;
 DMABCH = 0x02;  // DMABC = samples - 1 = 500+1+20-1 = 520 = 0x208
 DMABCL = 0x08;  // 500 samples + 1 temperature + 20 subsamples
 DMAEN = 1;
 delay_us(50);
 output_bit(INT_SAMPLE_READY, 1);
 delay_us(50);
 output_bit(INT_SAMPLE_READY, 0);
}
```

```
 else if(dma_ptr == &dma_set2 + 500) {
  *(dma_ptr) = ACC_T;

  temp_ptr = &dma_set2;
  *(dma_ptr+1) = *(temp_ptr + 0);
  *(dma_ptr+2) = *(temp_ptr + 1);
  *(dma_ptr+3) = *(temp_ptr + 2);
  *(dma_ptr+4) = *(temp_ptr + 3);
  *(dma_ptr+5) = ((*(temp_ptr + 4)) & 0xFC) + ((ACC_T & 0xC0) >> 6);
  *(dma_ptr+6) = *(temp_ptr + 125);
  *(dma_ptr+7) = *(temp_ptr + 126);
  *(dma_ptr+8) = *(temp_ptr + 127);
  *(dma_ptr+9) = *(temp_ptr + 128);
  *(dma_ptr+10) = ((*(temp_ptr + 129)) & 0xFC) + ((ACC_T & 0x30) >> 4);
  *(dma_ptr+11) = *(temp_ptr + 250);
  *(dma_ptr+12) = *(temp_ptr + 251);
  *(dma_ptr+13) = *(temp_ptr + 252);
  *(dma_ptr+14) = *(temp_ptr + 253);
  *(dma_ptr+15) = ((*(temp_ptr + 254)) & 0xFC) + ((ACC_T & 0x0C) >> 2);
  *(dma_ptr+16) = *(temp_ptr + 375);
  *(dma_ptr+17) = *(temp_ptr + 376);
  *(dma_ptr+18) = *(temp_ptr + 377);
  *(dma_ptr+19) = *(temp_ptr + 378);
  *(dma_ptr+20) = ((*(temp_ptr + 379)) & 0xFC) + ((ACC_T) & 0x03);

  dma_ptr = &dma_set1;
  TXADDRH = 0x09;
  TXADDRL = 0x00;
  DMABCH = 0x02;
  DMABCL = 0x08;
  DMAEN = 1;
  delay_us(50);
  output_bit(INT_SAMPLE_READY, 1);
  delay_us(50);
  output_bit(INT_SAMPLE_READY, 0);
 }
}

void main(void) {
 MCU_init();
 MSSP_init();
 ACC_init();
 Timer_Init();

 for(;;) {
  }
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix D

Main Microcontroller and Wireless Communication Circuits — Sensor Package Hardware

This part describes the schematic drawings and the PCB layouts for the main microcontroller and wireless communication circuits described in Chapter 4.3, excluding the data sampling circuits. The PCB layout presented here physically makes up a top plate for VAMPIRE. It is shown as 'PCB 2' in Fig. 4-8. The detailed PCB layout is shown in Fig. C-2. This appendix presents:

- Schematic Drawings
- PCB Layout
- Bill of Materials
- Software

D.1 Schematic Drawings

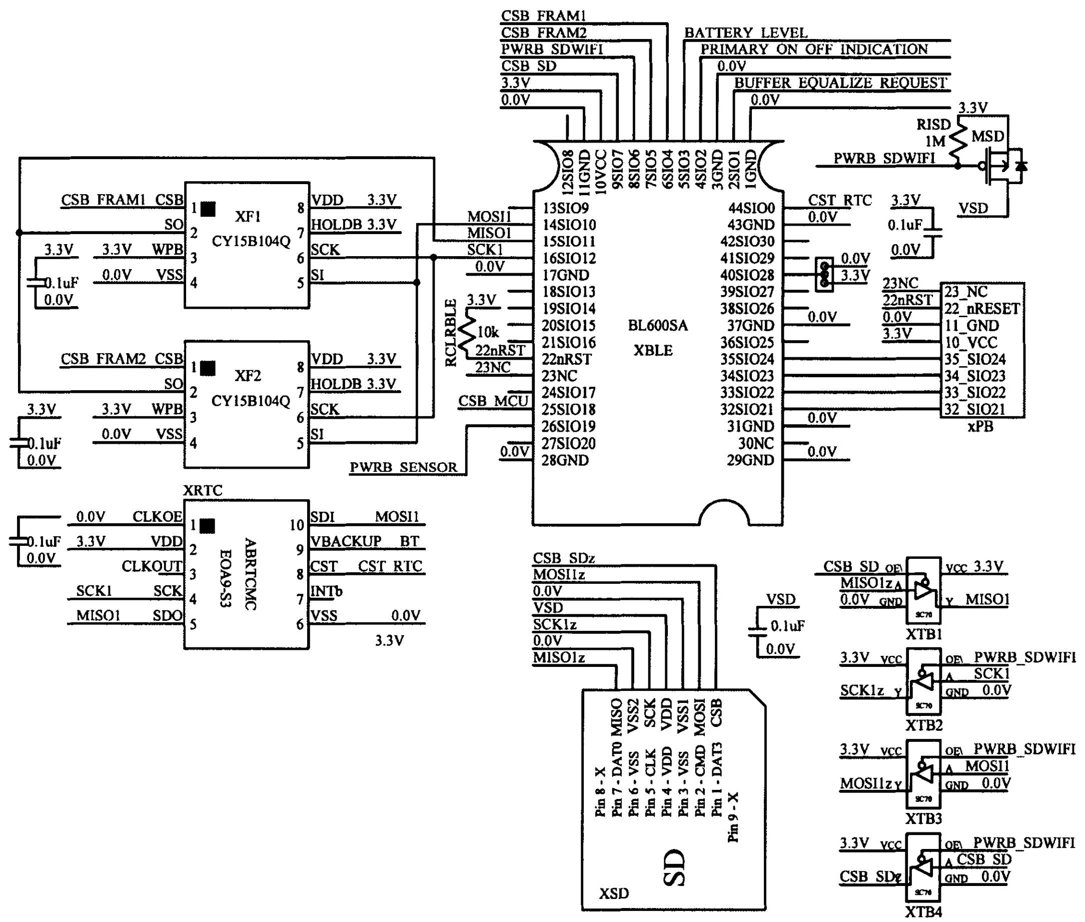


Figure D-1: Schematics of the Sensor Package (without sensors)

D.2 PCB Layout

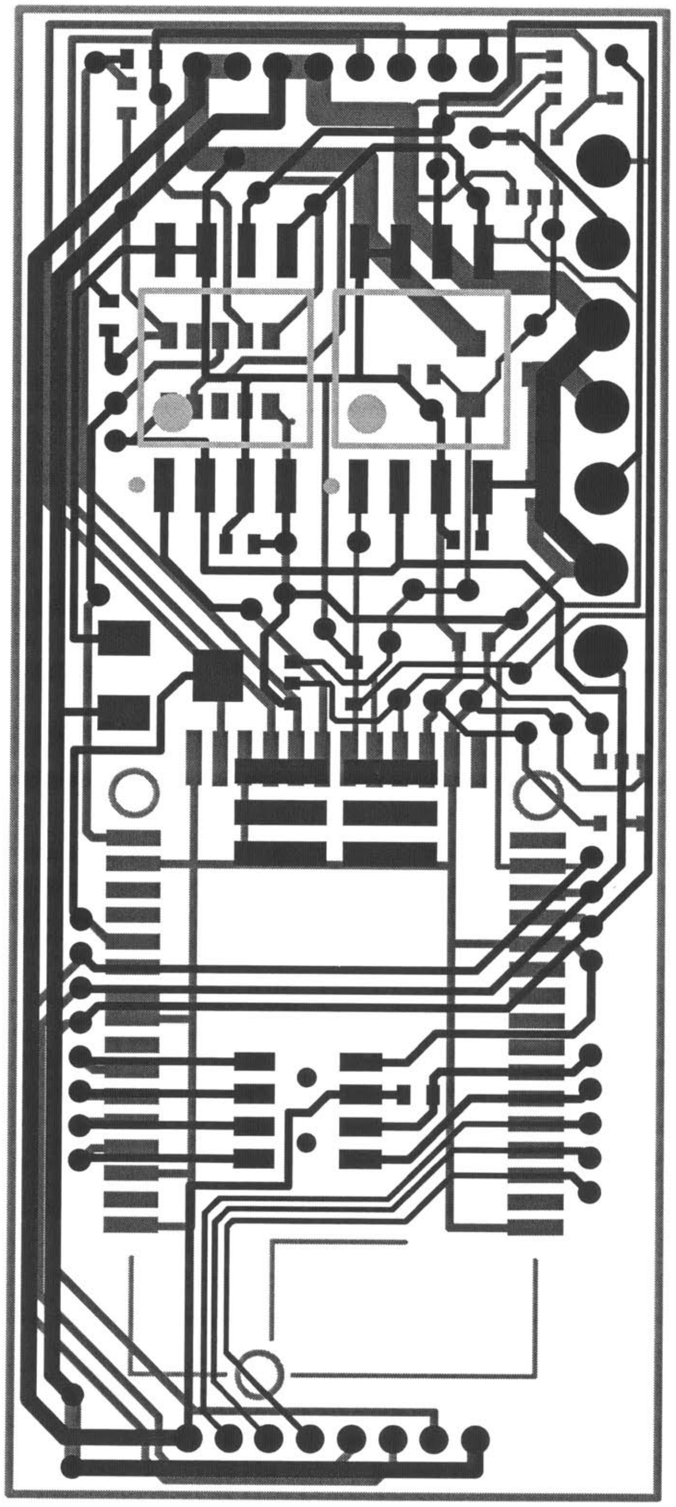


Figure D-2: PCB Layout of the Sensor Package (without sensors) - Full Stack

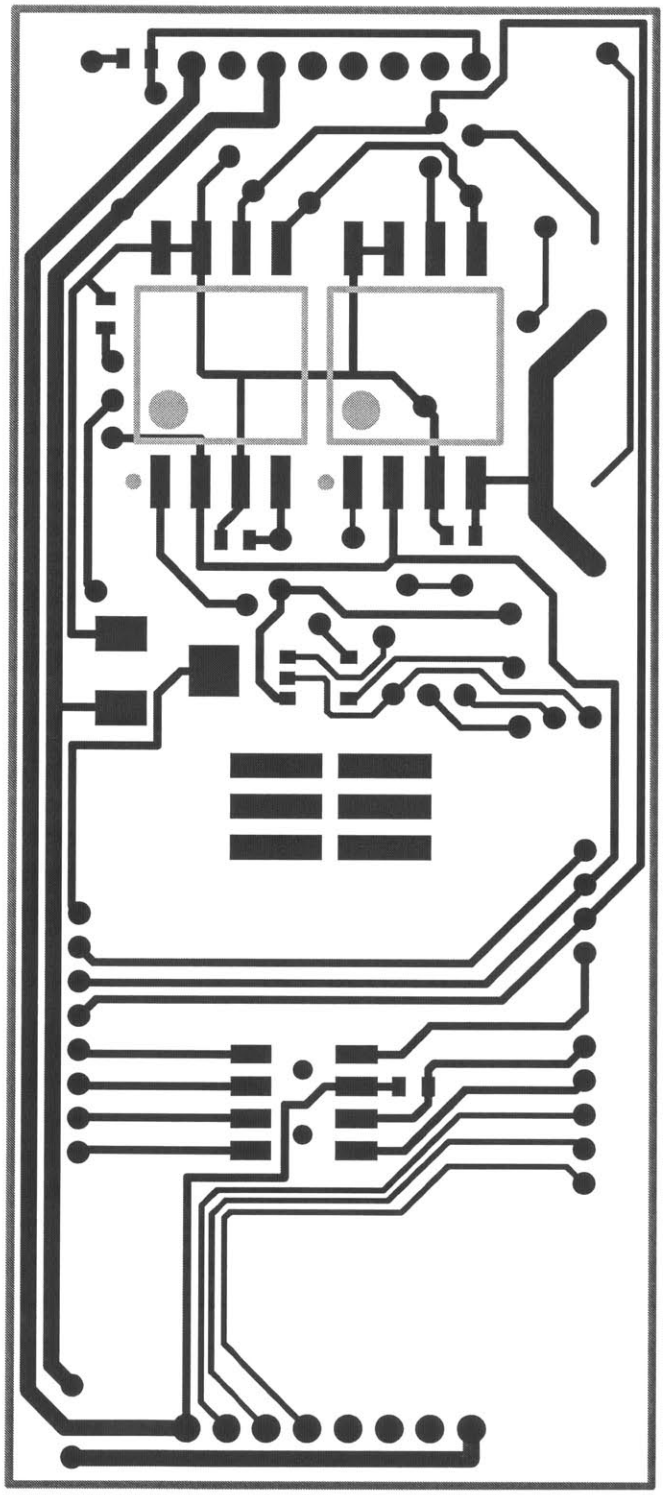


Figure D-3: PCB Layout of the Sensor Package (without sensors) - Top Copper

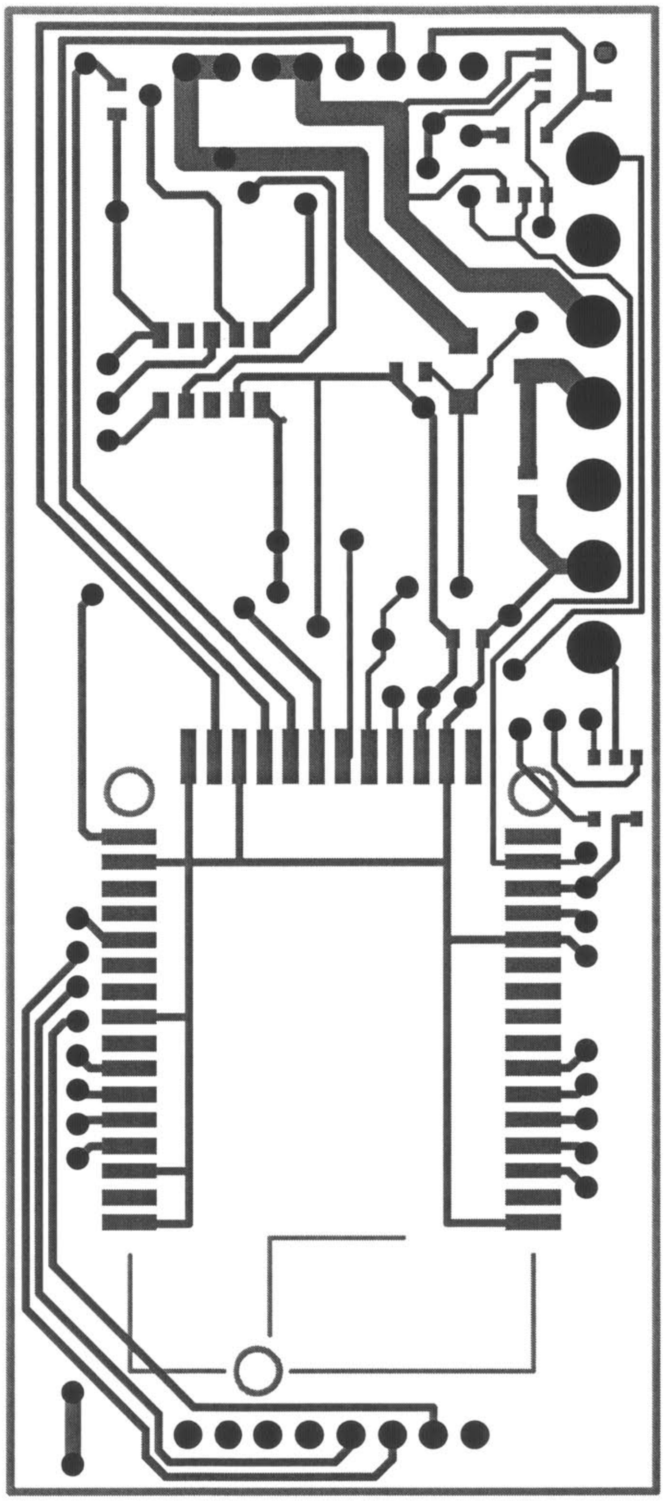


Figure D-4: PCB Layout of the Sensor Package (without sensors) - Bottom Copper

D.3 Bill of Materials

Table D.1: Bill of Materials for the Sensor Package

Device	Value	Quantity	Designator
SMD Capacitor 0402	0.1uF	5	cdble, cdf1, cdf2, cdrtc, cdsd
DMG3415U-7	DMG3415U-7	1	MSD
SMD Resistor 0402	10k Ohm	1	RCLRBLE
SMD Resistor 0402	1M Ohm	1	RISD
SW-3WAY-7813J-1-023E	SW-3WAY-7813J-1-023E	1	swble
BL600SA	BL600SA	1	XBLE
CY15B104Q	CY15B104Q	2	XF1, XF2
M40-3200445R	M40-3200445R	1	xPB
AB-RTCMC-32.768KHZ	AB-RTCMC-32.768KHZ	1	XRTC
FlashAir W-03	FlashAir W-03	1	XSD
NC7SZ125P5X	NC7SZ125P5X	4	XTB1, XTB2, XTB3, XTB4
851-87-003-40-252101	851-87-003-40-252101	4	Structural Mounts 3-pin
GRPB081VWCN-RC	GRPB081VWCN-RC	2	Interboard Mounts 8-pin

D.4 Software

The software language used for programming the main microcontroller is SmartBASIC from Laird Technology. As it is the embedded microcontroller inside the BLE module, all the BLE activities are directly controlled in this code.

```
#define MAX_DEVNAME_CHRS 20

#define ADV_IND 0
#define ADV_FILTERPOLICY_ANY 0
#define BLE_DISCOVERABILITY_GENERAL 2
#define BLE_APPEARANCE_GENERIC_BLOOD_PRESSURE 896
#define BLE_APPEARNNCE_UNKNOWN 0

#define BLE_EVBLEMSGID_CONNECT_0 0
#define BLE_EVBLEMSGID_DISCONNECT_1 1
#define BLE_EVBLEMSGID_AUTH_KEY_REQUEST_11 11 // msgCtx = 1 for passkey, 2 for 16
    byte OOB data
#define BLE_EVBLEMSGID_CONN_PARMS_UPDATE_14 14
#define BLE_EVBLEMSGID_CONN_PARMS_UPDATE_FAIL_15 15

#define DIN 1
#define DOUT 2

#define AIN 3

#define PIN_SDO 10
#define PIN_SDI 11
#define PIN_SCK 12

#define PIN_CSB_MCU 18
#define PIN_CSB_FRAM1 4
#define PIN_CSB_FRAM2 5
#define PIN_CSB_SD 7

#define PIN_CST_RTC 0

#define PIN_INT_MCU 17

#define PIN_PWRB_SENSOR 19
#define PIN_PWRB_SDWIFI 6

#define PIN_PRIMARY_OFF 2
#define PIN_BATTERY_LEVEL 3
#define PIN_BUFFER_EQ_REQ 1
```

```
#define WIFI_ON 1
#define WIFI_OFF 0

dim device_name$ : device_name$ = "VAMPIRE_CB03"
dim vampire_no : vampire_no = 3

// BLE Connectivity Variables
dim advertise_report$ : advertise_report$=""
dim scan_report$ : scan_report$=""
dim advFlags : advFlags = BLE_DISCOVERABILITY_GENERAL
dim peer_addr$ : peer_addr$ = ""
dim system_id$ : system_id$="\01\02\03\04\05\06\07\08"
// BLE Essential Variables
dim handle_service_01
dim handle_char_01   // command
dim handle_char_02   // file table
dim handle_char_03   // power table
dim handle_char_04   // realtime monitor
dim handle_char_05   // status report
dim handle_char_06   // test
dim handle_char_07   // realtime clock
dim state_connected
dim ble_handle
// General Essential Variables
dim rc                // result code for functions
dim handle_spi
dim data_upto_512byte$
dim file_table$             // 20 bytes (150bit + 2bit + 8bit)
dim power_table$ : power_table$   = "\20"   // 1 bytes
dim realtime_data$ : rc = strfill(realtime_data$, 0xFF, 20)
dim status_report$ : status_report$ = "\00\00\00\00\00"
dim rtc_timedata$
// Application Essential Variables
dim FRAM_logical_addr : FRAM_logical_addr  = 0
dim FRAM_physical_addr
dim spindown_addr : spindown_addr   = 1002000  // init value = unachievable
dim force_record_addr : force_record_addr  = 1002000
// Constants for Faster Computation
dim fram_cmd_wr$ : fram_cmd_wr$   = "\02\00\00\00"
dim fram_cmd_rd$ : fram_cmd_rd$   = "\03\00\00\00"
dim table_wr$ : table_wr$    = "\02\07\A5\08"
dim table_rd$ : table_rd$    = "\03\07\A5\08"
dim fram_wren$ : fram_wren$   = "\06"
dim write_01byte_FF$ : write_01byte_FF$  = "\FF"
dim write_02byte_01AA$ : write_02byte_01AA$ = "\01\AA"
dim write_02byte_FFFC$ : write_02byte_FFFC$ = "\FF\FC"
dim write_02byte_FFFE$ : write_02byte_FFFE$ = "\FF\FE"
dim write_02byte_FFs$ : write_02byte_FFs$  = "\FF\FF"
dim write_03byte_FFFDFF$ : write_03byte_FFFDFF$ = "\FF\FD\FF"
dim write_04byte_FFs$ : write_04byte_FFs$  = "\FF\FF\FF\FF"
dim write_13byte_FFs$ : rc = strfill(write_13byte_FFs$, 0xFF, 13)
dim command_06byte$ : command_06byte$  = "\FF\FF\FF\FF\FF\FF"
dim token_ssid$ : token_ssid$   = "APPSSID="
dim token_passwd$ : token_passwd$   = "APPNETWORKKEY="
dim token_remainder$ : token_remainder$  = "\0D\0A\4C\4F\43\4B\3D\31\0D\0A\44\4E\53\4
    D\4F\44\45\3D\31\0D\0A"
// -----
dim option_datamonitor : option_datamonitor = 0
dim option_force_record : option_force_record = 0

sub DbgMsg(BYVAL text$ as string)
 print text$; "\n"
endsub

sub DbgMsgVal(BYVAL text$ as string, BYVAL number as integer)
 print text$; number; "\n"
endsub

sub ShowConnParms(nCtx as integer)
 dim rc
 dim intrvl,sprvto,slat
 rc = BleGetCurConnParms(nCtx,intrvl,sprvto,slat)
 if rc==0 then
  DbgMsgVal("      --- Conn Interval = ",intrvl)
  DbgMsgVal("      --- Conn Supervision Timeout = ",sprvto)
  DbgMsgVal("      --- Conn Slave Latency = ",slat)
 endif
endsub

function HandlerBleMsg(BYVAL nMsgId AS INTEGER, BYVAL nCtx AS INTEGER) as integer
 dim rc
 dim conn_interval, conn_timeout, conn_latency
 dim conn_string$

 TimerCancel(0)
 select nMsgId

  case BLE_EVBLEMSGID_CONNECT_0
   state_connected = 1
   DbgMsg(" ooo Connected")
   rc = BleSetCurconnParms(nCtx, 7500, 12500, 4000000, 0)
   ShowConnParms(nCtx)
   ble_handle = nCtx

  case BLE_EVBLEMSGID_DISCONNECT_1
   state_connected = 0
   DbgMsg(" xxx Disconnected")
   rc = BleAdvertStart(ADV_IND, peer_addr$, 100, 300000, ADV_FILTERPOLICY_ANY)
   exitfunc 0 // one of the two ways to get out to the main loop's WAITEVENT

  case BLE_EVBLEMSGID_AUTH_KEY_REQUEST_11
   rc = BleSecMngrPassKey(nCtx, 123456) // key value = 123456

  case BLE_EVBLEMSGID_CONN_PARMS_UPDATE_14
   DbgMsg(" +++ Connection Parameter Update")
   rc = BleGetCurConnParms(nCtx, conn_interval, conn_timeout, conn_latency)
   conn_interval = conn_interval/1250
   conn_timeout = conn_timeout/10000
   conn_string$ = ""
   rc = StrSetChr(conn_string$, (conn_interval >> 8) & 0xFF, 0)
   rc = StrSetChr(conn_string$, (conn_interval) & 0xFF, 1)
   rc = StrSetChr(conn_string$, (conn_timeout >> 8) & 0xFF, 2)
   rc = StrSetChr(conn_string$, (conn_timeout) & 0xFF, 3)
   rc = StrSetChr(conn_string$, (conn_latency >> 8) & 0xFF, 4)
   rc = StrSetChr(conn_string$, (conn_latency) & 0xFF, 5)

   rc = BleCharValueWrite(handle_char_06, conn_string$)
   ShowConnParms(nCtx)
   exitfunc 0 // one of the two ways to get out to the main loop's WAITEVENT

  case BLE_EVBLEMSGID_CONN_PARMS_UPDATE_FAIL_15
   DbgMsg(" ??? Connection Parameter Negotiation Fail")
   ShowConnParms(nCtx)
   rc = BleSetCurconnParms(nCtx, 7500, 12500, 4000000, 0)

  case 18
   DbgMsg(" *** Connection Encrypted")
   ble_handle = nCtx
   exitfunc 0

  case else
   DbgMsgVal(" xxx Ble Msg: ", nMsgId)
 endselect
endfunc 1
```

```
function HandlerBLEADVTIMEOUT() as integer
 if !state_connected then
  rc = BleAdvertStart(ADV_IND, peer_addr$, 100, 300000, ADV_FILTERPOLICY_ANY)
 endif
endfunc 0

sub sample_data_from_MCU()
 gpiowrite(PIN_CSB_MCU, 0)
  rc = strfill(data_upto_512byte$, 0xFF, 501)       // strfill truncates too!
  rc = spireadwrite(data_upto_512byte$, data_upto_512byte$)   // will read 501 bytes
  rc = strfill(realtime_data$, 0xFF, 20)
  rc = spireadwrite(realtime_data$, realtime_data$)
 gpiowrite(PIN_CSB_MCU, 1)

 if state_connected & option_datamonitor then
  rc = BleCharValueNotify(handle_char_04, realtime_data$)
 endif
endsub

function store_data_to_FRAM()
 TimerCancel(0)
 sample_data_from_MCU()

 if FRAM_logical_addr < 501000 then
  FRAM_physical_addr = FRAM_logical_addr
  rc = StrSetChr(fram_cmd_wr$, (FRAM_physical_addr >> 16) & 0xFF, 1)
  rc = StrSetChr(fram_cmd_wr$, (FRAM_physical_addr >> 8) & 0xFF, 2)
  rc = StrSetChr(fram_cmd_wr$, (FRAM_physical_addr) & 0xFF, 3)

  gpiowrite(PIN_CSB_FRAM1, 0)
   rc = spiwrite(fram_wren$)
  gpiowrite(PIN_CSB_FRAM1, 1)

  gpiowrite(PIN_CSB_FRAM1, 0)
   rc = spiwrite(fram_cmd_wr$)
   rc = spiwrite(data_upto_512byte$)
  gpiowrite(PIN_CSB_FRAM1, 1)
 else
  FRAM_physical_addr = FRAM_logical_addr - 501000
  rc = StrSetChr(fram_cmd_wr$, (FRAM_physical_addr >> 16) & 0xFF, 1)
  rc = StrSetChr(fram_cmd_wr$, (FRAM_physical_addr >> 8) & 0xFF, 2)
  rc = StrSetChr(fram_cmd_wr$, (FRAM_physical_addr) & 0xFF, 3)

  gpiowrite(PIN_CSB_FRAM2, 0)
```

```
   rc = spiwrite(fram_wren$)
  gpiowrite(PIN_CSB_FRAM2, 1)

  gpiowrite(PIN_CSB_FRAM2, 0)
   rc = spiwrite(fram_cmd_wr$)
   rc = spiwrite(data_upto_512byte$)
  gpiowrite(PIN_CSB_FRAM2, 1)
 endif

 FRAM_logical_addr = FRAM_logical_addr + 501
 if FRAM_logical_addr == 1002000 then
  FRAM_logical_addr = 0
 endif
endfunc 0


sub read_data_from_FRAM(byval address as integer)
 if address < 501000 then
  FRAM_physical_addr = address
  rc = StrSetChr(fram_cmd_rd$, (FRAM_physical_addr >> 16) & 0xFF, 1)
  rc = StrSetChr(fram_cmd_rd$, (FRAM_physical_addr >> 8) & 0xFF, 2)
  rc = StrSetChr(fram_cmd_rd$, (FRAM_physical_addr) & 0xFF, 3)

  gpiowrite(PIN_CSB_FRAM1, 0)
   rc = strfill(data_upto_512byte$, 0xFF, 501)
   rc = spiwrite(fram_cmd_rd$)
   rc = spireadwrite(data_upto_512byte$, data_upto_512byte$)
  gpiowrite(PIN_CSB_FRAM1, 1)
 else
  FRAM_physical_addr = address - 501000
  rc = StrSetChr(fram_cmd_rd$, (FRAM_physical_addr >> 16) & 0xFF, 1)
  rc = StrSetChr(fram_cmd_rd$, (FRAM_physical_addr >> 8) & 0xFF, 2)
  rc = StrSetChr(fram_cmd_rd$, (FRAM_physical_addr) & 0xFF, 3)

  gpiowrite(PIN_CSB_FRAM2, 0)
   rc = strfill(data_upto_512byte$, 0xFF, 501)
   rc = spiwrite(fram_cmd_rd$)
   rc = spireadwrite(data_upto_512byte$, data_upto_512byte$)
  gpiowrite(PIN_CSB_FRAM2, 1)
 endif
endsub

sub load_file_table()
 gpiowrite(PIN_CSB_FRAM1, 0)
  rc = strfill(file_table$, 0xFF, 20)
  rc = spiwrite(table_rd$)
  rc = spireadwrite(file_table$, file_table$)
 gpiowrite(PIN_CSB_FRAM1, 1)
endsub

sub load_power_table()
 gpiowrite(PIN_CSB_FRAM2, 0)
  rc = spiwrite(table_rd$)
  rc = spireadwrite(write_01byte_FF$, power_table$)
 gpiowrite(PIN_CSB_FRAM2, 1)
endsub

sub store_file_table()
 gpiowrite(PIN_CSB_FRAM1, 0)
  rc = spiwrite(fram_wren$)
 gpiowrite(PIN_CSB_FRAM1, 1)

 gpiowrite(PIN_CSB_FRAM1, 0)
  rc = spiwrite(table_wr$)
  rc = spiwrite(file_table$)
 gpiowrite(PIN_CSB_FRAM1, 1)
endsub

sub store_power_table()
 gpiowrite(PIN_CSB_FRAM2, 0)
  rc = spiwrite(fram_wren$)
 gpiowrite(PIN_CSB_FRAM2, 1)

 gpiowrite(PIN_CSB_FRAM2, 0)
  rc = spiwrite(table_wr$)
  rc = spiwrite(power_table$)
 gpiowrite(PIN_CSB_FRAM2, 1)
endsub

sub store_timedata(byval slot as integer, byval timedata$ as string)
 dim local_addr
 dim local_write$
 local_addr = 501020 + 7*(slot - 1)
 local_write$ = "\02\00\00\00"

 rc = StrSetChr(local_write$, (local_addr >> 16) & 0xFF, 1)
 rc = StrSetChr(local_write$, (local_addr >> 8) & 0xFF, 2)
 rc = StrSetChr(local_write$, (local_addr) & 0xFF, 3)

 gpiowrite(PIN_CSB_FRAM1, 0)
```

```
  rc = spiwrite(fram_wren$)
 gpiowrite(PIN_CSB_FRAM1, 1)

 gpiowrite(PIN_CSB_FRAM1, 0)
  rc = spiwrite(local_write$)
  rc = spiwrite(timedata$)
 gpiowrite(PIN_CSB_FRAM1, 1)
endsub

sub load_timedata(byval slot as integer)
 dim local_addr
 dim local_write$
 dim local_07byte_FFs$

 local_addr = 501020 + 7*(slot - 1)
 local_write$ = "\03\00\00\00"

 rc = StrSetChr(local_write$, (local_addr >> 16) & 0xFF, 1)
 rc = StrSetChr(local_write$, (local_addr >> 8) & 0xFF, 2)
 rc = StrSetChr(local_write$, (local_addr) & 0xFF, 3)

 local_07byte_FFs$ = ""
 rc = strfill(local_07byte_FFs$, 0xFF, 7)

 gpiowrite(PIN_CSB_FRAM1, 0)
  rc = spiwrite(local_write$)
  rc = spireadwrite(local_07byte_FFs$, rtc_timedata$)
 gpiowrite(PIN_CSB_FRAM1, 1)
endsub

sub get_timedata()
 dim local_write$
 dim local_read$

 gpiowrite(PIN_CST_RTC, 1)
 local_write$ = "\88\FF\FF\FF\FF\FF\FF\FF"
 rc = spireadwrite(local_write$, local_read$)
 gpiowrite(PIN_CST_RTC, 0)

 rtc_timedata$ = right$(local_read$,7)
endsub

sub set_timedata(byval local_timedata$ as string)
 dim local_write$
 dim local_read$
 gpiowrite(PIN_CST_RTC, 1)
 local_write$ = "\08\00\00\00\00\01\00\00"      // ignore weekday (set to \01 =
     always sunday!)
 rc = StrSetChr(local_write$, StrGetChr(local_timedata$, 0), 1) // second
 rc = StrSetChr(local_write$, StrGetChr(local_timedata$, 1), 2) // minute
 rc = StrSetChr(local_write$, StrGetChr(local_timedata$, 2), 3) // hour
 rc = StrSetChr(local_write$, StrGetChr(local_timedata$, 3), 4) // day
 rc = StrSetChr(local_write$, StrGetChr(local_timedata$, 5), 6) // month
 rc = StrSetChr(local_write$, StrGetChr(local_timedata$, 6), 7) // year

 rc = spiwrite(local_write$)
 gpiowrite(PIN_CST_RTC, 0)
endsub


function command_to_SD(byval CMD as integer, byval SDaddress as integer, byval CRC as
     integer)
 dim local_read$

 gpiowrite(PIN_CSB_SD, 1)
 rc = spiwrite(write_01byte_FF$)
 gpiowrite(PIN_CSB_SD, 0)
 rc = spiwrite(write_01byte_FF$)

 rc = StrSetChr(command_06byte$, CMD, 0)
 rc = StrSetChr(command_06byte$, (SDaddress >> 24) & 0xFF, 1)
 rc = StrSetChr(command_06byte$, (SDaddress >> 16) & 0xFF, 2)
 rc = StrSetChr(command_06byte$, (SDaddress >> 8) & 0xFF, 3)
 rc = StrSetChr(command_06byte$, (SDaddress) & 0xFF, 4)
 rc = StrSetChr(command_06byte$, CRC, 5)
 rc = spiwrite(command_06byte$)

 DO
  rc = spireadwrite(write_01byte_FF$, local_read$)
  rc = StrGetChr(local_read$, 0)
 DOWHILE rc == 0xFF
endfunc rc

sub initialize_SD()
 dim local_read$

 SPIclose(handle_spi)
 rc = SPIopen(0, 250000, 0, handle_spi)       // Slower Clock During Initialization
 rc = spiwrite(write_13byte_FFs$)        // SDI high at least 10T before init
```

```
DO
 rc = command_to_SD(0x40, 0x00000000, 0x95)      // CMD0, and 0x95 for CRC of CMD0
 if rc == 0x01 then
  rc = command_to_SD(0x48, 0x000001AA, 0x87)     // CMD8, and 0x87 for CRC of CMD8
  if rc == 0x01 then
   rc = spiwrite(write_02byte_FFs$)
   local_read$ = ""
   rc = spireadwrite(write_02byte_FFs$, local_read$)
   if strcmp(local_read$, write_02byte_01AA$) == 0 then
    DO
     rc = command_to_SD(0x77, 0x00000000, 0x00)  // CMD55
     if rc == 0x01 then
      rc = command_to_SD(0x69,0x40000000,0x00) // ACMD41
     endif
    DOWHILE rc != 0
    rc = command_to_SD(0x7A, 0x00000000, 0x00)   // CMD58
    if rc == 0x00 then // Okay to CMD58, init okay
     rc = spireadwrite(write_04byte_FFs$, local_read$)
     rc = (StrGetChr(local_read$, 0) >> 6) & 0x01 // block_access
     if rc == 0x00 then // 0 = byte address, 1 = block
      DO
       rc = command_to_SD(0x50, 512, 0x00) // CMD16 (byte size set to 512)
      DOWHILE rc != 0
     endif
     gpiowrite(PIN_CSB_SD, 1)
     rc = spiwrite(write_01byte_FF$)
     break
    else // Error to CMD58, init failed
     gpiowrite(PIN_CSB_SD, 1)
     rc = spiwrite(write_01byte_FF$)
     print "init error\n"
    endif
   endif
  endif
 endif
DOWHILE 1
SPIclose(handle_spi)
rc = SPIopen(0, 2000000, 0, handle_spi) // Recover the nominal 2MHz speed
endsub

sub tx_from_FRAM_to_SD(byval file_number as integer, byval starting_address as
    integer)
 dim local_read$
 dim fram_address : fram_address = starting_address
dim sd_address : sd_address = 42976 + (file_number-1)*2048 - 8192
dim new_data_ready : new_data_ready = 0

SPIclose(handle_spi)
rc = SPIopen(0, 8000000, 0, handle_spi)

read_data_from_FRAM(fram_address)
rc = command_to_SD(0x59, sd_address, 0x00) // CMD25 with 1 nWR
if rc == 0x00 then
 rc = spiwrite(write_02byte_FFFC$) // 1 nWR + Data Token = \FC for CMD25
 rc = spiwrite(data_upto_512byte$)
 rc = spiwrite(write_13byte_FFs$) // now 501 + 11 (padding) + 2 (crc) bytes
 rc = spireadwrite(write_02byte_FFs$, local_read$)
 fram_address = fram_address + 501
 if fram_address == 1002000 then
  fram_address = 0
 endif
endif
gpiowrite(PIN_CSB_SD, 1)
DO
 if new_data_ready == 0 then
  read_data_from_FRAM(fram_address)
  new_data_ready = 1
 endif

 gpiowrite(PIN_CSB_SD, 0)
 rc = spireadwrite(write_01byte_FF$, local_read$)
 rc = StrGetChr(local_read$, 0)
 if rc == 0xFF then
  rc = spiwrite(write_02byte_FFFC$)
  rc = spiwrite(data_upto_512byte$)
  rc = spiwrite(write_13byte_FFs$)
  rc = spireadwrite(write_02byte_FFs$, local_read$)
  new_data_ready = 0
  fram_address = fram_address + 501
  if fram_address == 1002000 then
   fram_address = 0
  endif

  if fram_address == starting_address then
   rc = spiwrite(write_03byte_FFFDFF$)    // Data Token = \FD for CMD25 stop
   DO
    rc = spireadwrite(write_01byte_FF$, local_read$)
    rc = StrGetChr(local_read$, 0)
   DOWHILE rc != 0xFF
```

```
    gpiowrite(PIN_CSB_SD, 1)
    rc = spiwrite(write_01byte_FF$)
    break
   endif
  endif
  gpiowrite(PIN_CSB_SD, 1)
 DOWHILE 1
 SPIclose(handle_spi)
 rc = SPIopen(0, 2000000, 0, handle_spi)
endsub

sub power_WIFI(byval onoff as integer)
 dim local_read$

 rc = command_to_SD(0x51, 41376 - 8192, 0x00) // CMD17: single read, \A1\A0 = 41376
 if rc == 0x00 then
  DO
   rc = spireadwrite(write_01byte_FF$, local_read$)
   rc = StrGetChr(local_read$, 0) // StartToken for CMD17 = \FE
  DOWHILE rc != 0xFE
  rc = strfill(data_upto_512byte$, 0xFF, 512)     // Important! SDI held at high
     level
  rc = spireadwrite(data_upto_512byte$, data_upto_512byte$)
  rc = spireadwrite(write_02byte_FFs$, local_read$)
  rc = spiwrite(write_01byte_FF$)
 endif
 gpiowrite(PIN_CSB_SD, 1)

 rc = StrSetChr(data_upto_512byte$, 0x23 - onoff ,75)   // 0x23 = Read-only, 0x22 =
    unprotected

 rc = command_to_SD(0x58, 41376 - 8192, 0x00)     // CMD24: single write, \A1\A0 =
    41376
 if rc == 0x00 then
  rc = spiwrite(write_02byte_FFFE$)
  rc = spiwrite(data_upto_512byte$)
  rc = spiwrite(write_02byte_FFs$)
  rc = spireadwrite(write_01byte_FF$, local_read$)
  DO
   rc = spireadwrite(write_01byte_FF$, local_read$)
   rc = StrGetChr(local_read$, 0)
  DOWHILE rc != 0xFF
  rc = spiwrite(write_01byte_FF$)
 endif
 gpiowrite(PIN_CSB_SD, 1)
```

```
endsub

sub config_WIFI(byval ssid$ as string, byval passwd$ as string)
 dim i
 dim local_read$
 dim location_ssid
 dim location_passwd
 dim location_remainder

 rc = command_to_SD(0x51, 41392, 0x00)       // CMD17: single read, \A1\A0 = 41376
 if rc == 0x00 then
  DO
   rc = spireadwrite(write_01byte_FF$, local_read$)
   rc = StrGetChr(local_read$, 0)
  DOWHILE rc != 0xFE
  rc = strfill(data_upto_512byte$, 0xFF, 512)
  rc = spireadwrite(data_upto_512byte$, data_upto_512byte$)
  rc = spireadwrite(write_02byte_FFs$, local_read$)
  rc = spiwrite(write_01byte_FF$)
 endif
 gpiowrite(PIN_CSB_SD, 1)

 location_passwd = StrPos(data_upto_512byte$, token_passwd$, 0) + 14
 for i = 0 to StrLen(passwd$)-1
  rc = StrSetChr(data_upto_512byte$, StrGetChr(passwd$, i), location_passwd + i)
 next

 location_ssid = StrPos(data_upto_512byte$, token_ssid$, 0) + 8
 for i = 0 to StrLen(ssid$)-1
  rc = StrSetChr(data_upto_512byte$, StrGetChr(ssid$, i), location_ssid + i)
 next

 location_remainder = location_ssid + StrLen(ssid$)
 for i = 0 to StrLen(token_remainder$) - 1
  rc = StrSetChr(data_upto_512byte$, StrGetChr(token_remainder$,i),
      location_remainder+i)
 next

 rc = command_to_SD(0x58, 41392, 0x00) // CMD24: single write, \A1\A0 = 41376
 if rc == 0x00 then
  rc = spiwrite(write_02byte_FFFE$)
  rc = spiwrite(data_upto_512byte$)
  rc = spiwrite(write_02byte_FFs$)
  rc = spireadwrite(write_01byte_FF$, local_read$)
  DO
```

```
   rc = spireadwrite(write_01byte_FF$, local_read$)
   rc = StrGetChr(local_read$, 0)
  DOWHILE rc != 0xFF
  rc = spiwrite(write_01byte_FF$)
 endif
 gpiowrite(PIN_CSB_SD, 1)
endsub

sub delay_ms(byval duration as integer)
 dim start_tick

 start_tick = GetTickCount()
 DO
  if GetTickSince(start_tick) > duration then
   break
  endif
 DOWHILE 1
endsub

function force_event_expiry()
endfunc 0

OnEvent EVBLEMSG call HandlerBLEMSG
OnEvent EVBLE_ADV_TIMEOUT call HandlerBLEADVTIMEOUT
OnEvent EVDETECTCHAN0 call store_data_to_FRAM
OnEvent EVTMR0 call force_event_expiry

dim received_BLE_CMD$ : received_BLE_CMD$ = "\00\00\00\00"
dim previous_BLE_CMD$

dim record_tick
dim record_ms

dim daddr : daddr = 0
dim dms

dim ble_cmd
dim ble_arg1
dim ble_arg2
dim ble_arg3
dim ble_arg4

dim received_primary_off
dim previous_primary_off : previous_primary_off = 1
dim new_force_record
dim old_force_record : old_force_record = 0

dim state_spindown_on : state_spindown_on = 0
dim state_force_record : state_force_record = 0
dim file_number

dim battery_level : battery_level = 0

dim option_power
dim option_power_persist
dim option_power_mcu
dim option_power_sd
dim option_power_wifi

dim current_power
dim power_control

dim request_battery_eq

dim state_powersave : state_powersave = 0
dim state_sdinitialized : state_sdinitialized = 0

dim wloc
dim wbit

dim new_wifi_ssid$
dim new_wifi_passwd$

dim rtc_command$
dim rtc_response$
dim rtc_report$

dim single_read$
dim single_address
dim single_index

rc = gpiosetfunc(PIN_SDO,  DOUT, 1)
rc = gpiosetfunc(PIN_SDI,  DIN, 0)
rc = gpiosetfunc(PIN_SCK,  DOUT, 0)
rc = gpiosetfunc(PIN_CSB_MCU, DOUT, 1)
rc = gpiosetfunc(PIN_CSB_FRAM1, DOUT, 1)
rc = gpiosetfunc(PIN_CSB_FRAM2, DOUT, 1)
rc = gpiosetfunc(PIN_CSB_SD, DOUT, 1)
rc = gpiosetfunc(PIN_CST_RTC, DOUT, 0)
rc = gpiosetfunc(PIN_INT_MCU, DIN, 0)


rc = gpiosetfunc(PIN_PWRB_SENSOR, DOUT, 1)
rc = gpiosetfunc(PIN_PWRB_SDWIFI, DOUT, 1)

rc = gpiosetfunc(PIN_PRIMARY_OFF, DIN, 0)
rc = gpiosetfunc(PIN_BATTERY_LEVEL, AIN, 0x13) // 0.0v (0x000) to 3.6v (0x400)

rc = gpiosetfunc(PIN_BUFFER_EQ_REQ, DOUT, 0)

rc = BleSecmngrIOCap(0)
rc = BleGapSvcInit(device_name$, 1, BLE_APPEARNNCE_UNKNOWN, 7500, 11250, 4000000, 0)
rc = BleSvcRegDevInfo("MFG NAME", "MODEL NAME", "SERIAL NO", "HwRev1.0", "SwRev1.0",
    system_id$, "", "")

rc = BleSvcCommit(1, BleHandleUUID16(0x9F3E), handle_service_01) // 1 = PRIMARY
rc = BleCharNew(0x0E, BleHandleUUID16(0x9F3F), 1289, 0, 0)    // Attr(R/W/20)
rc = BleCharCommit(handle_service_01, received_BLE_CMD$, handle_char_01) // BLE CMD

rc = BleCharNew(0x02, BleHandleUUID16(0x9F40), 1281, 0, 0)    // Attr(R/20)
rc = BleCharCommit(handle_service_01, received_BLE_CMD$, handle_char_02) // FILE
    TABLE

rc = BleCharNew(0x02, BleHandleUUID16(0x9F41), 1281, 0, 0)    // Attr(R/20)
rc = BleCharCommit(handle_service_01, received_BLE_CMD$, handle_char_03) // POWER
    TABLE

rc = BleCharNew(0x12, BleHandleUUID16(0x9F42), 1281, 137, 0)   // Attr(R/20), Cccd(R/
    W/2)
rc = BleCharCommit(handle_service_01, realtime_data$, handle_char_04)  // MONITOR

rc = BleCharNew(0x02, BleHandleUUID16(0x9F43), 1281, 0, 0)    // Attr(R/20)
rc = BleCharCommit(handle_service_01, realtime_data$, handle_char_05)  // STATUS
    REPORT

rc = BleCharNew(0x02, BleHandleUUID16(0x9F44), 1281, 0, 0)    // Attr(R/20)
rc = BleCharCommit(handle_service_01, realtime_data$, handle_char_06)  // CONNECTION
    PARAMETERS

rc = BleCharNew(0x12, BleHandleUUID16(0x9F45), 1281, 137, 0)   // Attr(R/20), Cccd(R/
    W/2)
rc = BleCharCommit(handle_service_01, realtime_data$, handle_char_07)  // RTC REPORT
rc = BleAdvRptInit(advertise_report$, advFlags, 1, MAX_DEVNAME_CHRS)
rc = BleAdvRptAddUuid16(advertise_report$,-1,-1,-1,-1,-1,-1)
rc = BleScanRptInit(scan_report$)
rc = BleAdvRptsCommit(advertise_report$, scan_report$)
rc = BleAdvertStart(ADV_IND, peer_addr$, 100, 300000, ADV_FILTERPOLICY_ANY)

rc = gpioassignevent(0, PIN_INT_MCU, 0)
rc = SPIopen(0, 2000000, 0, handle_spi)

load_file_table()
file_number = StrGetChr(file_table$, 0)
rc = BleCharValueWrite(handle_char_02, file_table$)

load_power_table()
option_power = StrGetChr(power_table$, 0)
option_power_persist = (option_power >> 7) & 0x01   // 8th bit = persist option

if option_power_persist then
 option_power_mcu = (option_power >> 5) & 0x01   // 6th bit = mcu power
 option_power_sd = (option_power >> 3) & 0x01   // 4th bit = SD power
 option_power_wifi = (option_power >> 1) & 0x01   // 2nd bit = WIFI power
 request_battery_eq = 0         // this cannot be stored
else
 option_power_mcu = 1
 option_power_sd = 0
 option_power_wifi = 0
 request_battery_eq = 0
 power_table$ = "\20"
 store_power_table()
endif

rc = BleCharValueWrite(handle_char_03, power_table$)

gpiowrite(PIN_PWRB_SENSOR, 1 - option_power_mcu)
gpiowrite(PIN_PWRB_SDWIFI, 1 - option_power_sd)

if option_power_sd then
 delay_ms(20)
 if !state_sdinitialized then
  initialize_SD()
  state_sdinitialized = 1
 endif
 power_WIFI(WIFI_OFF)
 if option_power_wifi then
  power_WIFI(WIFI_ON)
 endif
endif

gpiowrite(PIN_BUFFER_EQ_REQ, 0)

record_tick = GetTickCount()
DO
 record_ms = GetTickSince(record_tick)
 if record_ms >= 995 then
  record_tick = GetTickCount()
  if state_connected then
   rc = BleCharValueRead(handle_char_01, received_BLE_CMD$)
   rc = BleCharValueRead(handle_char_02, file_table$)
   rc = BleCharValueRead(handle_char_03, power_table$)

   if strcmp(received_BLE_CMD$, previous_BLE_CMD$) then
    ble_cmd = StrGetChr(received_BLE_CMD$, 0)
    ble_arg1 = StrGetChr(received_BLE_CMD$, 1)
    ble_arg2 = StrGetChr(received_BLE_CMD$, 2)

    if ble_cmd == 1 then
     current_power = StrGetChr(power_table$, 0)
     power_control = current_power ^ ble_arg1

     if (power_control >> 5) & 0x01 then
      option_power_mcu  = (ble_arg1 >> 5) & 0x01 // 6th bit = mcu power
      gpiowrite(PIN_PWRB_SENSOR, 1 - option_power_mcu) // remember PMOS control
      if !option_power_mcu then
       rc = gpiounassignevent(0)
      else
       rc = gpioassignevent(0, PIN_INT_MCU, 0)
      endif
     endif
     if (power_control >> 3) & 0x01 then
      option_power_sd   = (ble_arg1 >> 3) & 0x01 // 4th bit = SD power
      if option_power_sd then
       gpiowrite(PIN_PWRB_SDWIFI, 0)
       if !state_sdinitialized then
        initialize_SD()
        state_sdinitialized = 1
       endif
      endif
      if !option_power_sd then
       if option_power_wifi then
        power_WIFI(option_power_wifi)
    option_power_wifi = 0
   endif
   gpiowrite(PIN_PWRB_SDWIFI, 1)
   state_sdinitialized = 0
   ble_arg1 = ble_arg1 & 0xFD
  endif
 endif
 if (power_control >> 1) & 0x01 then     // 2nd bit = WIFI power
  if option_power_sd then
   option_power_wifi  = (ble_arg1 >> 1) & 0x01 // 2nd bit = WIFI power
   if !state_sdinitialized then
    initialize_SD()
    state_sdinitialized = 1
   endif
   power_WIFI(WIFI_OFF)
   rc = StrSetChr(status_report$, 0x00, 4)
   if option_power_wifi then
    power_WIFI(WIFI_ON)
    rc = StrSetChr(status_report$, vampire_no, 4)
   endif
  else
   option_power_wifi = 0
   ble_arg1 = ble_arg1 & 0xFD
   rc = StrSetChr(status_report$, 0x00, 4)
  endif
 endif
 if power_control & 0x01 then
  request_battery_eq = (ble_arg1 & 0x01)
  gpiowrite(PIN_BUFFER_EQ_REQ, request_battery_eq)
 endif

 rc = StrSetChr(power_table$, ble_arg1, 0)
 rc = BleCharValueWrite(handle_char_03, power_table$)

 if (power_control >> 7) & 0x01 then
  option_power_persist = (ble_arg1 >> 7) & 0x01
  store_power_table()
 endif

 if option_power_persist then
  store_power_table()
 endif
elseif ble_cmd == 2 then
 if ble_arg1 > 0 then  // nonzero filenumber (1 - 150)
  wloc = (ble_arg1  + 7)/8
```

```
   wbit = 0xFF - (1 << (7 - ((ble_arg1 - 1) % 8)))
   rc = StrSetChr(file_table$, StrGetChr(file_table$, wloc) & wbit , wloc)

   if ble_arg1 == file_number then
    file_number = file_number - 1
    if file_number == 0 then
     file_number = 150
    endif
    rc = StrSetChr(file_table$, file_number, 0)
   endif

   store_file_table()
   rc = BleCharValueWrite(handle_char_02, file_table$)
  elseif ble_arg2 == 0xFF then
   rc = strfill(file_table$, 0x00, 20)
   file_number = 150
   rc = StrSetChr(file_table$, file_number, 0)
   store_file_table()
   rc = BleCharValueWrite(handle_char_02, file_table$)
  endif
 elseif ble_cmd == 3 then
  if option_power_mcu then
   option_datamonitor = ble_arg1
  endif
 elseif ble_cmd == 4 then // record the future 100seconds starting now
  if option_power_mcu then
   option_force_record = ble_arg1
  endif
 elseif ble_cmd == 5 then // screenshot of the most recent 100seconds
  if ble_arg1 then
   if !option_power_sd then
    gpiowrite(PIN_PWRB_SDWIFI, 0)
    initialize_SD()
    state_sdinitialized = 1
   endif
   if !state_sdinitialized then
    initialize_SD()
    state_sdinitialized = 1
   endif
   file_number = file_number + 1
   if file_number == 151 then
    file_number = 1
   endif
   get_timedata()
   store_timedata(file_number, rtc_timedata$)
```

```
    tx_from_FRAM_to_SD(file_number, FRAM_logical_addr)
    state_spindown_on = 0
    state_force_record = 0
    spindown_addr = 1002000
    force_record_addr = 1002000

    rc = StrSetChr(file_table$, file_number, 0)

    wloc = (file_number  + 7)/8
    wbit = 1 << (7 - ((file_number - 1) % 8))
    rc = StrSetChr(file_table$, StrGetChr(file_table$, wloc) | wbit , wloc)
    store_file_table()
    rc = BleCharValueWrite(handle_char_02, file_table$)

    if !option_power_sd then
     gpiowrite(PIN_PWRB_SDWIFI, 1)
     state_sdinitialized = 0
    endif
   endif
  elseif ble_cmd == 6 then
   rc = BleDisconnect(ble_handle)
  elseif ble_cmd == 7 then
   if !option_power_sd then
    gpiowrite(PIN_PWRB_SDWIFI, 0)
    initialize_SD()
    state_sdinitialized = 1
   endif
   if !state_sdinitialized then
    initialize_SD()
    state_sdinitialized = 1
   endif
   new_wifi_ssid$ = "VAMPIRE_DW" + mid$(received_BLE_CMD$, 1, 2)
   new_wifi_passwd$ = mid$(received_BLE_CMD$, 3, 8)

   if strlen(new_wifi_passwd$) < 8 then
    new_wifi_passwd$ = "01234567"
   endif

   config_WIFI(new_wifi_ssid$, new_wifi_passwd$)

   if !option_power_sd then
    gpiowrite(PIN_PWRB_SDWIFI, 1)
    state_sdinitialized = 0
   endif
```

```
elseif ble_cmd == 8 then // report RTC data to BLE char
 get_timedata()

 rtc_report$ = "\00\00\00\00\00\00"
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 6), 0)  // year
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 5), 1)  // month
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 3), 2)  // day
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 2), 3)  // hour
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 1), 4)  // minute
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 0), 5)  // second

 rc = BleCharValueNotify(handle_char_07, rtc_report$)

elseif ble_cmd == 9 then  // set realtime clock to SPI RTC now
 rtc_response$ = right$(received_BLE_CMD$, 6)
 rtc_command$ = "\00\00\00\00\01\00\00"       // ignore weekday (set to \01 =
     always sunday!)
 rc = StrSetChr(rtc_command$, StrGetChr(rtc_response$, 5), 0) // second
 rc = StrSetChr(rtc_command$, StrGetChr(rtc_response$, 4), 1) // minute
 rc = StrSetChr(rtc_command$, StrGetChr(rtc_response$, 3), 2) // hour
 rc = StrSetChr(rtc_command$, StrGetChr(rtc_response$, 2), 3) // day
 rc = StrSetChr(rtc_command$, StrGetChr(rtc_response$, 1), 5) // month
 rc = StrSetChr(rtc_command$, StrGetChr(rtc_response$, 0), 6) // year

 set_timedata(rtc_command$)

elseif ble_cmd == 0x10 then  // FRAM timestamp loading
 load_timedata(ble_arg1)
 rtc_report$ = "\00\00\00\00\00\00"
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 6), 0)  // year
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 5), 1)  // month
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 3), 2)  // day
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 2), 3)  // hour
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 1), 4)  // minute
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 0), 5)  // second

elseif ble_cmd == 0x11 then  // FRAM timestamp saving

 if ble_arg1 == 0xFF then // direct input or from SPI
  get_timedata()
  rtc_command$ = rtc_timedata$

  rtc_report$ = "\00\00\00\00\00\00"
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_command$, 6), 0)  // year
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_command$, 5), 1)  // month
```

```
   rc = StrSetChr(rtc_report$, StrGetChr(rtc_command$, 3), 2)  // day
   rc = StrSetChr(rtc_report$, StrGetChr(rtc_command$, 2), 3)  // hour
   rc = StrSetChr(rtc_report$, StrGetChr(rtc_command$, 1), 4)  // minute
   rc = StrSetChr(rtc_report$, StrGetChr(rtc_command$, 0), 5)  // second
  else
   rtc_response$ = right$(received_BLE_CMD$, 6)
   rtc_command$ = "\00\00\00\00\01\00\00"      // ignore weekday (set to \01 =
       always sunday!)
   rc = StrSetChr(rtc_command$, StrGetChr(rtc_response$, 5), 0) // second
   rc = StrSetChr(rtc_command$, StrGetChr(rtc_response$, 4), 1) // minute
   rc = StrSetChr(rtc_command$, StrGetChr(rtc_response$, 3), 2) // hour
   rc = StrSetChr(rtc_command$, StrGetChr(rtc_response$, 2), 3) // day
   rc = StrSetChr(rtc_command$, StrGetChr(rtc_response$, 1), 5) // month
   rc = StrSetChr(rtc_command$, StrGetChr(rtc_response$, 0), 6) // year
  endif
  store_timedata(ble_arg2, rtc_command$)

 elseif ble_cmd == 0x12 then // requesting 4 different time data, corresponding to
     4 slots specified, stored in FRAM
  ble_arg3 = StrGetChr(received_BLE_CMD$, 3)
  ble_arg4 = StrGetChr(received_BLE_CMD$, 4)

  rc = strfill(rtc_report$, 0xFF, 20)

  load_timedata(ble_arg1)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 6), 0)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 5), 1)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 3), 2)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 2), 3)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 1), 4)

  load_timedata(ble_arg2)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 6), 5)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 5), 6)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 3), 7)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 2), 8)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 1), 9)

  load_timedata(ble_arg3)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 6), 10)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 5), 11)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 3), 12)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 2), 13)
  rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 1), 14)
 load_timedata(ble_arg4)
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 6), 15)
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 5), 16)
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 3), 17)
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 2), 18)
 rc = StrSetChr(rtc_report$, StrGetChr(rtc_timedata$, 1), 19)

 rc = BleCharValueNotify(handle_char_07, rtc_report$)

elseif ble_cmd == 0x61 then  // check a block in the SD card at the address,
    return the block data through the notification. // Chr 'a' in ascii code

 single_read$ = right$(received_BLE_CMD$, strlen(received_BLE_CMD$) - 2)

 if ble_arg1 == 0x68 then  // Hex Byte Input Chr 'h'
  single_address = 0
  for single_index = 1 to strlen(single_read$)
   single_address = single_address * 256 + StrGetChr(single_read$, single_index -
        1)
  next
 elseif ble_arg1 == 0x64 then // Decimal Integer String Input Chr 'd'
  single_address = StrValDec(single_read$)
 endif

 single_read$ = ""
 rc = command_to_SD(0x51, single_address, 0x00) // CMD17: single read, \A1\A0 =
     41376
 if rc == 0x00 then
  DO
   rc = spireadwrite(write_01byte_FF$, single_read$) // nWR
   rc = StrGetChr(single_read$, 0) // StartToken for CMD17 = \FE
  DOWHILE rc != 0xFE
  rc = strfill(data_upto_512byte$, 0xFF, 512)
  rc = spireadwrite(data_upto_512byte$, data_upto_512byte$)
  rc = spireadwrite(write_02byte_FFs$, single_read$)
  rc = spiwrite(write_01byte_FF$)
 endif
 gpiowrite(PIN_CSB_SD, 1)

 print "\nRead Data:\n"
 for single_index = 1 to 32
  single_read$ = mid$(data_upto_512byte$, 16*(single_index-1), 16)
  print strhexize$(single_read$);"\n"
 next
```

```
    print "Transferring through BLE\n"
    for single_index = 1 to 32
     single_read$ = mid$(data_upto_512byte$, 16*(single_index-1), 16)
     rc = BleCharValueNotify(handle_char_07, single_read$)
     print "Hex #"; single_index; "\n"
     delay_ms(100)
    next
    print "\n"
   endif
   previous_BLE_CMD$ = "\00\00\00\00"
   rc = BleCharValueWrite(handle_char_01,previous_BLE_CMD$)
  endif
 endif

 battery_level = gpioread(PIN_BATTERY_LEVEL)
 rc = StrSetChr(status_report$, (battery_level >> 8) & 0x03, 0)
 rc = StrSetChr(status_report$, (battery_level & 0xFF), 1)
 rc = StrSetChr(status_report$, (state_spindown_on << 4) + state_force_record, 2)
 rc = StrSetChr(status_report$, (state_powersave << 4) + !received_primary_off, 3)
 rc = BleCharValueWrite(handle_char_05, status_report$)

endif

if option_power_mcu then
 received_primary_off = gpioread(PIN_PRIMARY_OFF)
 if received_primary_off & !previous_primary_off & !state_force_record then
  if FRAM_logical_addr < 501*20*85 then
   spindown_addr = FRAM_logical_addr + 501*20*15  // forward +15s
  else
   spindown_addr = FRAM_logical_addr - 501*20*85  // backward -85s
  endif
  state_spindown_on = 1
  // auto EQ update in power table start
  ble_arg1 = StrGetChr(power_table$, 0)
  rc = StrSetChr(power_table$, ble_arg1 | 0x01, 0)
  gpiowrite(PIN_BUFFER_EQ_REQ, 1)
  rc = BleCharValueWrite(handle_char_03, power_table$)
  if option_power_persist then
   store_power_table()
  endif
  // auto EQ update in power table end
  state_force_record = 0
  force_record_addr = 1002000
  option_force_record = 0
 elseif state_spindown_on & !received_primary_off then
 spindown_addr = 1002000
 state_spindown_on = 0
endif
previous_primary_off = received_primary_off

new_force_record = option_force_record
if new_force_record & !old_force_record then
 if FRAM_logical_addr < 501 then
  force_record_addr = FRAM_logical_addr + 1001499
 else
  force_record_addr = FRAM_logical_addr - 501
 endif
 state_force_record = 1
 state_spindown_on = 0
 spindown_addr = 1002000
elseif state_force_record & !new_force_record then
 force_record_addr = 1002000
 state_force_record = 0
endif
old_force_record = new_force_record

if state_spindown_on & (FRAM_logical_addr == spindown_addr) then
 if !option_power_sd then
  gpiowrite(PIN_PWRB_SDWIFI, 0)
  initialize_SD()
  state_sdinitialized = 1
 endif
 if !state_sdinitialized then
  initialize_SD()
  state_sdinitialized = 1
 endif
 file_number = file_number + 1
 if file_number == 151 then
  file_number = 1
 endif
 get_timedata()
 store_timedata(file_number, rtc_timedata$)
 tx_from_FRAM_to_SD(file_number, spindown_addr)
 state_spindown_on = 0
 spindown_addr = 1002000

 rc = StrSetChr(file_table$, file_number, 0)

 wloc = (file_number  + 7)/8
 wbit = 1 << (7 - ((file_number - 1) % 8))
```

```
  rc = StrSetChr(file_table$, StrGetChr(file_table$, wloc) | wbit , wloc)
  store_file_table()
  rc = BleCharValueWrite(handle_char_02, file_table$)
  if !option_power_sd then
   gpiowrite(PIN_PWRB_SDWIFI, 1)
   state_sdinitialized = 0
  endif
 endif

 if state_force_record & (FRAM_logical_addr == force_record_addr) then
  if !option_power_sd then
   gpiowrite(PIN_PWRB_SDWIFI, 0)
   initialize_SD()
   state_sdinitialized = 1
  endif
  if !state_sdinitialized then
   initialize_SD()
   state_sdinitialized = 1
  endif
  file_number = file_number + 1
  if file_number == 151 then
   file_number = 1
  endif
  get_timedata()
  store_timedata(file_number, rtc_timedata$)
  tx_from_FRAM_to_SD(file_number, force_record_addr)
  state_force_record = 0
  force_record_addr = 1002000

  rc = StrSetChr(file_table$, file_number, 0)

  wloc = (file_number  + 7)/8
  wbit = 1 << (7 - ((file_number - 1) % 8))
  rc = StrSetChr(file_table$, StrGetChr(file_table$, wloc) | wbit , wloc)
  store_file_table()
  rc = BleCharValueWrite(handle_char_02, file_table$)
  if !option_power_sd then
   gpiowrite(PIN_PWRB_SDWIFI, 1)
   state_sdinitialized = 0
  endif
 endif
endif

TimerStart(0,2,0)  // in case there is no mcu activity
WAITEVENT
```

```
DOWHILE 1
WAITEVENT
```

Appendix E

Miscellaneous — Front and Back Covers

This appendix illustrates the PCB layouts for the front and back covers for VAMPIRE. Both covers act as protective layers, and do not have any electrical connection to the actual circuits on other PCBs. The biggest hole is for the power wire to pass through for energy harvesting purposes, and the next two big holes are for back-EMF coupling. This appendix presents:

- PCB Layout
- Bill of Materials

E.1 PCB Layout

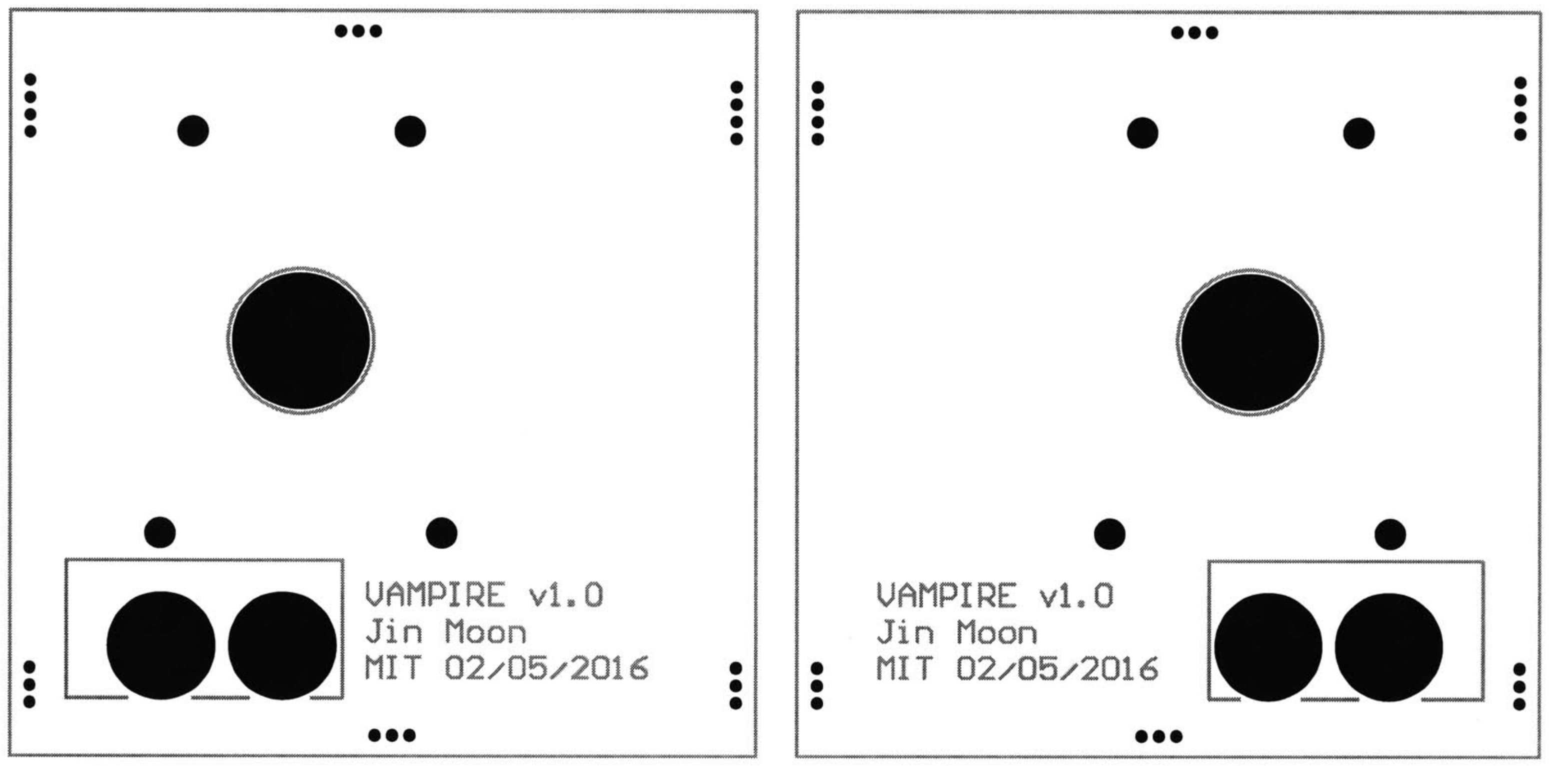


Figure E-1: PCB Layout of the Front and Back Covers

E.2 Bill of Materials

Table E.1: Bill of Materials for the Front and Cover

Device	Value	Quantity	Designator
GRPB041VWVN-RC	GRPB041VWVN-RC	4	Structural Plug 4-pin
850-80-003-10-001101	850-80-003-10-001101	8	Structural Plug 3-pin

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix F

User Software — VAMPIRE App

This part describes the user software that communicates with VAMPIRE. The application is written in C# using Visual Studio 2015 Community. It is developed for Universal Windows Platform, targeting Windows 10 environment.

F.1 User Interface Design - XAML Code

```
<Page
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:VAMPIRE_App"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:telerikChart="using:Telerik.UI.Xaml.Controls.Chart"
    x:Class="VAMPIRE_App.MainPage"
    mc:Ignorable="d">
    <Grid x:Name="Form" Background="{ThemeResource
        ApplicationPageBackgroundThemeBrush}" MinWidth="1280" MinHeight="720"
        MaxWidth="1280" MaxHeight="720" ScrollViewer.VerticalScrollBarVisibility="
        Disabled" Width="1280" Height="720" UseLayoutRounding="False" Margin
        ="0,-10,0,10">
        <Button x:Name="buttonDiscoverBLE" Content="Discover" HorizontalAlignment="
            Left" Height="40" Margin="10,10,0,0" VerticalAlignment="Top" Width="70"
            Padding="0" Click="buttonDiscoverBLE_Click"/>
```

```
<ListBox x:Name="listBoxBLE" HorizontalAlignment="Left" Margin="10,55,0,500"
    Width="250                  " DoubleTapped="listBoxBLE_DoubleTapped"/>
<Button x:Name="buttonConnectBLE" Content="Connect" HorizontalAlignment="Left
    " Height="40" Margin="85,10,0,0" VerticalAlignment="Top" Width="70"
    Padding="0" Click="buttonConnectBLE_Click"/>
<Button x:Name="buttonDisconnectBLE" Content="Disconnect" HorizontalAlignment
    ="Left" Height="40" Margin="160,10,0,0" VerticalAlignment="Top" Width
    ="100" Padding="0" RenderTransformOrigin="0.5,0.5" Click="
    buttonDisconnectBLE_Click">
    <Button.RenderTransform>
        <CompositeTransform SkewY="0.398" TranslateY="0.243"/>
    </Button.RenderTransform>
</Button>
<TextBlock x:Name="statusConnectionBLE" HorizontalAlignment="Left" Height
    ="25" Margin="10,225,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="
    Top" Width="250"/>
<TextBox x:Name="textBoxDirectCommand" HorizontalAlignment="Left" Height="20"
     Margin="1020,730,0,-42" Text="" VerticalAlignment="Top" Width="175"
    Padding="7.5,3,18,3" UseLayoutRounding="False" MaxLength="100"
    ScrollViewer.HorizontalScrollBarVisibility="Disabled" ScrollViewer.
    VerticalScrollBarVisibility="Disabled" KeyUp="textBoxDirectCommand_KeyUp
    "/>
<Button x:Name="buttonDirectCommand" Content="Send" HorizontalAlignment="Left
    " Height="32" Margin="1200,730,0,-42" VerticalAlignment="Top" Width="70"
    Padding="0" Click="buttonDirectCommand_Click"/>
<Button x:Name="buttonMonitor" Content="Monitor" HorizontalAlignment="Left"
    Height="40" Margin="795,10,0,0" VerticalAlignment="Top" Width="125"
    Padding="0" Click="buttonMonitor_Click"/>
<Button x:Name="buttonTest" Content="Test1" HorizontalAlignment="Left" Height
    ="35" Margin="805,584,0,0" VerticalAlignment="Top" Width="70" Click="
    buttonTest_Click"/>

<telerikChart:RadCartesianChart x:Name="chartX" Margin="275,10,490,570"
    BorderThickness="1" Background="White" BorderBrush="Black" Opacity="0.5"
    Tapped="chartX_Tapped">
    <telerikChart:RadCartesianChart.HorizontalAxis>
        <telerikChart:CategoricalAxis ShowLabels="False" Title="">
        </telerikChart:CategoricalAxis>
    </telerikChart:RadCartesianChart.HorizontalAxis>
    <telerikChart:RadCartesianChart.VerticalAxis>
        <telerikChart:LinearAxis Title="acc.X [g], auto" HorizontalLocation="
            Right"/>
    </telerikChart:RadCartesianChart.VerticalAxis>
    <telerikChart:LineSeries ItemsSource="{Binding}" Stroke="Crimson">
    </telerikChart:LineSeries>
```

```
</telerikChart:RadCartesianChart>

<telerikChart:RadCartesianChart x:Name="chartY" Margin="275,150,490,430"
    BorderThickness="1" Background="White" BorderBrush="Black" Opacity="0.5"
    Tapped="chartY_Tapped">
    <telerikChart:RadCartesianChart.HorizontalAxis>
        <telerikChart:CategoricalAxis ShowLabels="False" Title="">
        </telerikChart:CategoricalAxis>
    </telerikChart:RadCartesianChart.HorizontalAxis>
    <telerikChart:RadCartesianChart.VerticalAxis>
        <telerikChart:LinearAxis Title="acc.Y [g], auto" HorizontalLocation="
            Right"/>
    </telerikChart:RadCartesianChart.VerticalAxis>
    <telerikChart:LineSeries ItemsSource="{Binding}" Stroke="DarkGreen">
    </telerikChart:LineSeries>
</telerikChart:RadCartesianChart>

<telerikChart:RadCartesianChart x:Name="chartZ" Margin="275,290,490,290"
    BorderThickness="1" Background="White" BorderBrush="Black" Opacity="0.5"
    Tapped="chartZ_Tapped">
    <telerikChart:RadCartesianChart.HorizontalAxis>
        <telerikChart:CategoricalAxis ShowLabels="False" Title="">
        </telerikChart:CategoricalAxis>
    </telerikChart:RadCartesianChart.HorizontalAxis>
    <telerikChart:RadCartesianChart.VerticalAxis>
        <telerikChart:LinearAxis Title="acc.Z [g], auto" HorizontalLocation="
            Right"/>
    </telerikChart:RadCartesianChart.VerticalAxis>
    <telerikChart:LineSeries ItemsSource="{Binding}" Stroke="Blue">
    </telerikChart:LineSeries>
</telerikChart:RadCartesianChart>

<telerikChart:RadCartesianChart x:Name="chartB" Margin="275,430,490,150"
    BorderThickness="1" Background="White" BorderBrush="Black" Opacity="0.5"
    Tapped="chartB_Tapped">
    <telerikChart:RadCartesianChart.HorizontalAxis>
        <telerikChart:CategoricalAxis ShowLabels="False" Title="">
        </telerikChart:CategoricalAxis>
    </telerikChart:RadCartesianChart.HorizontalAxis>
    <telerikChart:RadCartesianChart.VerticalAxis>
        <telerikChart:LinearAxis Title="BEMF [V], auto" LastLabelVisibility="
            Visible" HorizontalLocation="Right"/>
        <!--Minimum="0.0" Maximum="3.3"-->
    </telerikChart:RadCartesianChart.VerticalAxis>
    <telerikChart:LineSeries ItemsSource="{Binding}" Stroke="DarkOrange">
    </telerikChart:LineSeries>
</telerikChart:RadCartesianChart>

<telerikChart:RadCartesianChart x:Name="chartT" Margin="275,570,490,10"
    BorderThickness="1" Background="White" BorderBrush="Black" Opacity="0.5"
    Tapped="chartT_Tapped">
    <telerikChart:RadCartesianChart.HorizontalAxis>
        <telerikChart:CategoricalAxis ShowLabels="False" Title="">
        </telerikChart:CategoricalAxis>
    </telerikChart:RadCartesianChart.HorizontalAxis>
    <telerikChart:RadCartesianChart.VerticalAxis>
        <telerikChart:LinearAxis Title="Temp. [C], auto" LastLabelVisibility
            ="Visible" HorizontalLocation="Right"/>
        <!--Minimum="-40" Maximum="90"-->
    </telerikChart:RadCartesianChart.VerticalAxis>
    <telerikChart:LineSeries ItemsSource="{Binding}" Stroke="Violet">
    </telerikChart:LineSeries>
</telerikChart:RadCartesianChart>

<Button x:Name="buttonTest2" Content="Test2" HorizontalAlignment="Left"
    Margin="805,624,0,0" VerticalAlignment="Top" Click="buttonTest2_Click"
    Width="70" Height="35"/>
<Button x:Name="buttonTest3" Content="Test3" HorizontalAlignment="Left"
    Height="35" Margin="805,664,0,0" VerticalAlignment="Top" Width="70" Click
    ="buttonTest3_Click"/>
<Button x:Name="buttonAutoRange" Content="AutoRange Off" HorizontalAlignment
    ="Left" Height="40" Margin="795,55,0,0" VerticalAlignment="Top" Width
    ="125" Click="buttonAutoRange_Click"/>
<ListBox x:Name="listBoxWIFI" HorizontalAlignment="Left" Margin="10,494,0,61"
     Width="250                  " DoubleTapped="listBoxWIFI_DoubleTapped"/>
<Button x:Name="buttonDiscoverWIFI" Content="Discover" HorizontalAlignment="
    Left" Height="40" Margin="10,449,0,0" VerticalAlignment="Top" Width="70"
    Padding="0" Click="buttonDiscoverWIFI_Click"/>
<Button x:Name="buttonConnectWIFI" Content="Connect" HorizontalAlignment="
    Left" Height="40" Margin="85,449,0,0" VerticalAlignment="Top" Width="70"
    Padding="0" Click="buttonConnectWIFI_Click"/>
<Button x:Name="buttonDisconnectWIFI" Content="Disconnect"
    HorizontalAlignment="Left" Height="40" Margin="160,449,0,0"
    VerticalAlignment="Top" Width="100" Padding="0" RenderTransformOrigin
    ="0.5,0.5" Click="buttonDisconnectWIFI_Click">
    <Button.RenderTransform>
        <CompositeTransform SkewY="0.398" TranslateY="0.243"/>
    </Button.RenderTransform>
</Button>
```

```
<Button x:Name="buttonTurnOnWIFI" Content="WIFI On" HorizontalAlignment="Left
    " Height="35" Margin="10,305,0,0" VerticalAlignment="Top" Width="120"
    Click="buttonTurnOnWIFI_Click"/>
<Button x:Name="buttonTurnOffWIFI" Content="WIFI Off" HorizontalAlignment="
    Left" Height="35" Margin="140,305,0,0" VerticalAlignment="Top" Width
    ="120" Click="buttonTurnOffWIFI_Click"/>
<Button x:Name="buttonShowFileTable" Content="File Table" HorizontalAlignment
    ="Left" Height="30" Margin="1020,10,0,0" VerticalAlignment="Top" Width
    ="120" Click="buttonShowFileTable_Click"/>
<ListBox x:Name="listBoxFileTable" HorizontalAlignment="Left" Margin
    ="1020,45,0,231" Width="250                  " SelectionChanged="
    listBoxFileTable_SelectionChanged" RightTapped="
    listBoxFileTable_RightTapped"/>
<Button x:Name="buttonDownload" Content="Download" HorizontalAlignment="Left"
     Height="30" Margin="1150,0,0,680" VerticalAlignment="Bottom" Width="120"
     Click="buttonDownload_Click" DoubleTapped="buttonDownload_DoubleTapped
    "/>
<TextBlock x:Name="statusConnectionWIFI" HorizontalAlignment="Left" Height
    ="25" Margin="10,664,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="
    Top" Width="250"/>
<TextBlock x:Name="statusFileDownload" HorizontalAlignment="Left" Height
    ="131" Margin="1020,494,0,0" TextWrapping="Wrap" Text=""
    VerticalAlignment="Top" Width="250" IsTextSelectionEnabled="True"
    RightTapped="statusFileDownload_RightTapped"/>
<Button x:Name="buttonTurnOnSD" Content="SD On" HorizontalAlignment="Left"
    Height="35" Margin="10,261,0,0" VerticalAlignment="Top" Width="120" Click
    ="buttonTurnOnSD_Click"/>
<Button x:Name="buttonTurnOffSD" Content="SD Off" HorizontalAlignment="Left"
    Height="35" Margin="140,261,0,0" VerticalAlignment="Top" Width="120"
    Click="buttonTurnOffSD_Click"/>
<Button x:Name="buttonCapturePrev100s" Content="Prev 100s"
    HorizontalAlignment="Left" Height="35" Margin="10,380,0,0"
    VerticalAlignment="Top" Width="120" Click="buttonCapturePrev100s_Click"/>
<Button x:Name="buttonCaptureNext100s" Content="Next 100s"
    HorizontalAlignment="Left" Height="35" Margin="140,380,0,0"
    VerticalAlignment="Top" Width="120" Click="buttonCaptureNext100s_Click"/>
<TextBlock x:Name="statusCapture" HorizontalAlignment="Left" Height="20"
    Margin="10,360,0,0" TextWrapping="Wrap" Text="Data Capture:"
    VerticalAlignment="Top" Width="250"/>
<Button x:Name="buttonRefreshTime" Content="RTC:" HorizontalAlignment="Left"
    Height="30" Margin="1020,630,0,0" VerticalAlignment="Top" Width="50"
    Click="buttonRefreshTime_Click"/>
<TextBlock x:Name="statusTime" HorizontalAlignment="Left" VerticalAlignment="
    Center" Height="22" Margin="1075,635,0,63" TextWrapping="Wrap" Text="RTC
    unreachable" Width="150" IsTextSelectionEnabled="True" MaxLines="1"/>
```

```
        <TextBox x:Name="textBoxChangeTime" HorizontalAlignment="Left" Height="20"
            Margin="1020,667,0,0" Text="" VerticalAlignment="Top" Width="205" Padding
            ="7.5,3,18,3" UseLayoutRounding="False" MaxLength="100" ScrollViewer.
            HorizontalScrollBarVisibility="Disabled" ScrollViewer.
            VerticalScrollBarVisibility="Disabled" PlaceholderText="0000/00/00
            00:00:00"/>
        <Button x:Name="buttonChangeTime" Content="Send" HorizontalAlignment="Left"
            Height="32" Margin="1230,667,0,0" VerticalAlignment="Top" Width="40"
            Padding="0" Click="buttonChangeTime_Click"/>
        <DatePicker x:Name="rtcDatePicker" HorizontalAlignment="Left" Margin
            ="974,552,0,0" VerticalAlignment="Top" DateChanged="
            rtcDatePicker_DateChanged" Visibility="Collapsed"/>
        <TimePicker x:Name="rtcTimePicker" HorizontalAlignment="Left" Margin
            ="1028,590,0,0" VerticalAlignment="Top" TimeChanged="
            rtcTimePicker_TimeChanged" Visibility="Collapsed"/>
        <Button x:Name="buttonRTCPicker" Content="Pick" HorizontalAlignment="Left"
            Height="32" Margin="1230,630,0,0" VerticalAlignment="Top" Width="40"
            Padding="0" Click="buttonRTCPicker_Click"/>
    </Grid>
</Page>
```

F.2 Function Design - C# Code

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Text.RegularExpressions;
using System.Threading;
using Telerik.Charting;
using Telerik.UI.Xaml.Controls.Chart;
using Windows.Devices.Bluetooth;
using Windows.Devices.Bluetooth.GenericAttributeProfile;
using Windows.Devices.Enumeration;
using Windows.Devices.WiFi;
using Windows.Foundation;
using Windows.Security.Credentials;
using Windows.Storage;
using Windows.Storage.AccessCache;
using Windows.Storage.Pickers;
using Windows.Storage.Streams;
using Windows.UI;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Media;

namespace VAMPIRE_App
{
    public sealed partial class MainPage : Page
    {
        public GattDeviceService vampireService;
        public IReadOnlyList<GattCharacteristic> vampireCharCommand;
        public IReadOnlyList<GattCharacteristic> vampireCharFileTable;
        public IReadOnlyList<GattCharacteristic> vampireCharPowerTable;
        public IReadOnlyList<GattCharacteristic> vampireCharMonitor;
        public IReadOnlyList<GattCharacteristic> vampireCharStatusReport;
        public IReadOnlyList<GattCharacteristic> vampireCharConnection;
        public IReadOnlyList<GattCharacteristic> vampireCharRealtimeClock;

        public bool vampireConnected;
        public bool sdOn;
        public bool wifiOn;

        public object Utilities { get; private set; }

        public MainPage()
        {
            this.InitializeComponent();
            statusConnectionBLE.Text = vampireConnected ? "Connected to " +
                vampireService.Device.Name : "Disconnected";
            statusConnectionBLE.InvalidateMeasure();

            buttonMonitor.IsEnabled = false;
            buttonTurnOnWIFI.IsEnabled = false;
            buttonTurnOffWIFI.IsEnabled = false;
        }

        private async void buttonDiscoverBLE_Click(object sender, RoutedEventArgs e)
        {
            statusConnectionBLE.Text = "Searching for BLE";
            statusConnectionBLE.InvalidateMeasure();
            Guid vampireGuid = new Guid("00009f3e-0000-1000-8000-00805f9b34fb".
                ToUpper());
            var vampireDevices = await DeviceInformation.FindAllAsync(
                GattDeviceService.GetDeviceSelectorFromUuid(vampireGuid));
    listBoxBLE.Items.Clear();
    foreach (var device in vampireDevices)
    {
        Debug.WriteLine("Service Enabled for " + device.Name + " ? " + device
            .IsEnabled);
        if (device.IsEnabled)
        {
            myDeviceInformation currentDevice = new myDeviceInformation();
            currentDevice.deviceInformation = device;
            currentDevice.displayName = device.Name;
            listBoxBLE.Items.Add(currentDevice);
        }
    }
    statusConnectionBLE.Text = vampireConnected ? "Connected to " +
        vampireService.Device.Name : "Disconnected";
    statusConnectionBLE.InvalidateMeasure();
}

private DispatcherTimer connectionTimerBLE = new DispatcherTimer();

public void connectionHandlerBLE()
{
    connectionTimerBLE.Tick += connectionHandlerBLE_Tick;
    connectionTimerBLE.Interval = new TimeSpan(0, 0, 0, 1);
    connectionTimerBLE.Start();
}

private async void connectionHandlerBLE_Tick(object sender, object e)
{
    GattReadResult x = await vampireCharConnection[0].ReadValueAsync(
        BluetoothCacheMode.Uncached);
    byte[] byteArray = new byte[x.Value.Length];
    DataReader.FromBuffer(x.Value).ReadBytes(byteArray);

    int connectionInterval = byteArray[0] * 256 + byteArray[1];
    int supervisionTimeOut = byteArray[2] * 256 + byteArray[3];
    int slaveLatency = byteArray[4] * 256 + byteArray[5];

    if (connectionInterval <= 9)
    {
        Debug.WriteLine("Connection Interval = " + connectionInterval * 1.25
            + "ms");
        Debug.WriteLine("Supervision Timeout = " + supervisionTimeOut / 100 +
            "s");
        Debug.WriteLine("      Slave Latency = " + slaveLatency);

        Debug.WriteLine(vampireService.Device.ConnectionStatus);
        connectionTimerBLE.Stop();
        connectionTimerBLE.Tick -= connectionHandlerBLE_Tick;
        buttonTest3.IsEnabled = true;
        statusConnectionBLE.Text = vampireConnected ? "Connected to " +
            vampireService.Device.Name : "Disconnected";
        statusConnectionBLE.InvalidateMeasure();
        buttonConnectBLE.IsEnabled = true;
        listBoxBLE.IsEnabled = true;
        buttonMonitor.IsEnabled = true;

        buttonRefreshTime_Click(null, null);
    }
}

private async void buttonConnectBLE_Click(object sender, RoutedEventArgs e)
{
    if (listBoxBLE.SelectedItem != null)
    {
        if (vampireConnected == false)
        {
            buttonConnectBLE.IsEnabled = false;
            listBoxBLE.IsEnabled = false;
            buttonMonitor.IsEnabled = false;

            statusConnectionBLE.Text = "Connecting to " + listBoxBLE.
                SelectedItem.ToString();
            statusConnectionBLE.InvalidateMeasure();

            vampireService = await GattDeviceService.FromIdAsync(((
                myDeviceInformation)listBoxBLE.SelectedItem).
                deviceInformation.Id);
            vampireConnected = true;
            while (vampireService.Device.ConnectionStatus ==
                BluetoothConnectionStatus.Disconnected) { }
            vampireCharCommand = vampireService.GetCharacteristics(new Guid
                ("00009F3F-0000-1000-8000-00805F9B34FB"));
            var writer = new DataWriter();
            writer.WriteByte(0);
            GattCommunicationStatus writeStatus = await vampireCharCommand
                [0].WriteValueAsync(writer.DetachBuffer());
                vampireCharFileTable = vampireService.GetCharacteristics(new Guid
                    ("00009F40-0000-1000-8000-00805F9B34FB"));
                vampireCharPowerTable = vampireService.GetCharacteristics(new
                    Guid("00009F41-0000-1000-8000-00805F9B34FB"));
                vampireCharMonitor = vampireService.GetCharacteristics(new Guid
                    ("00009F42-0000-1000-8000-00805F9B34FB"));
                vampireCharStatusReport = vampireService.GetCharacteristics(new
                    Guid("00009F43-0000-1000-8000-00805F9B34FB"));
                vampireCharConnection = vampireService.GetCharacteristics(new
                    Guid("00009F44-0000-1000-8000-00805F9B34FB"));
                vampireCharRealtimeClock = vampireService.GetCharacteristics(new
                    Guid("00009F45-0000-1000-8000-00805F9B34FB"));

                connectionTimerBLE.Tick -= connectionHandlerBLE_Tick;
                connectionHandlerBLE();
            }
        }
    }

    private void buttonDisconnectBLE_Click(object sender, RoutedEventArgs e)
    {
        if (vampireConnected == true && vampireService != null)
        {
            statusConnectionBLE.Text = "Disconnecting from " + vampireService.
                Device.Name;
            statusConnectionBLE.InvalidateMeasure();
            vampireService.Device.Dispose();
            vampireService.Dispose();
            vampireConnected = false;
            while (vampireService.Device.ConnectionStatus ==
                BluetoothConnectionStatus.Connected) { }
            statusConnectionBLE.Text = vampireConnected ? "Connected to " +
                vampireService.Device.Name : "Disconnected";
            statusConnectionBLE.InvalidateMeasure();

            statusTime.Text = "RTC Unreachable";
        }
    }

    private void listBoxBLE_DoubleTapped(object sender, Windows.UI.Xaml.Input.
        DoubleTappedRoutedEventArgs e)
    {
        buttonConnectBLE_Click(sender, e);
    }
```

```
private async void buttonDirectCommand_Click(object sender, RoutedEventArgs e
    )
{
    if (vampireConnected)
    {
        string hexString = Regex.Replace(textBoxDirectCommand.Text, "\\s+",
            "").ToUpper();
        if (hexString.Length % 2 == 1)
        {
            hexString += "0";
            textBoxDirectCommand.Text = hexString;
        }
        byte[] commandBytes = StringToByteArray(hexString);

        var writer = new DataWriter();
        writer.WriteBytes(commandBytes);
        GattCommunicationStatus writeStatus = await vampireCharCommand[0].
            WriteValueAsync(writer.DetachBuffer());
        textBoxDirectCommand.Text = "";
        Debug.WriteLine(statusConnectionBLE.ToString());
    }
}

public static byte[] StringToByteArray(string hex)
{
    byte[] byteArray = new byte[hex.Length >> 1];
    for (int i = 0; i < hex.Length >> 1; ++i)
    {
        byteArray[i] = (byte)((GetHexVal(hex[i << 1]) << 4) + (GetHexVal(hex
            [(i << 1) + 1])));
    }
    return byteArray;
}

public static int GetHexVal(char hex)
{
    int val = (int)hex;
    return val - (val < 58 ? 48 : 55);
}

private void textBoxDirectCommand_KeyUp(object sender, Windows.UI.Xaml.Input.
    KeyRoutedEventArgs e)
{
    if (e.Key == Windows.System.VirtualKey.Enter)
    {
            buttonDirectCommand_Click(sender, e);
        }
    }

    private async void buttonMonitor_Click(object sender, RoutedEventArgs e)
    {
        if (!vampireConnected)
        {
            if (vampireCharMonitor != null)
            {
                vampireCharMonitor[0].ValueChanged -= dataChanged;
            }
            buttonMonitor.Content = "Monitor";
            realTimeX.RemoveRange(0, realTimeX.Count);
            realTimeY.RemoveRange(0, realTimeY.Count);
            realTimeZ.RemoveRange(0, realTimeZ.Count);
            realTimeB.RemoveRange(0, realTimeB.Count);
            realTimeT.RemoveRange(0, realTimeT.Count);
            movingAvT.RemoveRange(0, movingAvT.Count);
        }
        else
        {
            if (buttonMonitor.Content.ToString() == "Monitor")
            {
                byte[] command = { 3, 1 };
                var writer = new DataWriter();
                writer.WriteBytes(command);
                GattCommunicationStatus writeStatus = await vampireCharCommand
                    [0].WriteValueAsync(writer.DetachBuffer());
                vampireCharMonitor[0].ValueChanged += dataChanged;
                await vampireCharMonitor[0].
                    WriteClientCharacteristicConfigurationDescriptorAsync(
                    GattClientCharacteristicConfigurationDescriptorValue.Notify);
                buttonMonitor.Content = "Stop";
            }
            else
            {
                byte[] command = { 3, 0 };
                var writer = new DataWriter();
                writer.WriteBytes(command);
                GattCommunicationStatus writeStatus = await vampireCharCommand
                    [0].WriteValueAsync(writer.DetachBuffer());
                vampireCharMonitor[0].ValueChanged -= dataChanged;
                await vampireCharMonitor[0].
                    WriteClientCharacteristicConfigurationDescriptorAsync(
                GattClientCharacteristicConfigurationDescriptorValue.None);
            buttonMonitor.Content = "Monitor";
            realTimeX.RemoveRange(0, realTimeX.Count);
            realTimeY.RemoveRange(0, realTimeY.Count);
            realTimeZ.RemoveRange(0, realTimeZ.Count);
            realTimeB.RemoveRange(0, realTimeB.Count);
            realTimeT.RemoveRange(0, realTimeT.Count);
            movingAvT.RemoveRange(0, movingAvT.Count);
        }
    }
}

public List<Double> realTimeX = new List<Double>();
public List<Double> realTimeY = new List<Double>();
public List<Double> realTimeZ = new List<Double>();
public List<Double> realTimeB = new List<Double>();
public List<Double> realTimeT = new List<Double>();
public List<Double> movingAvT = new List<Double>();
public int movingAvTDepth = 20;


async void dataChanged(GattCharacteristic sender, GattValueChangedEventArgs
    eventArgs)
{
    byte[] byteArray = new byte[eventArgs.CharacteristicValue.Length];
    double[] doubleArrayX = new double[4];
    double[] doubleArrayY = new double[4];
    double[] doubleArrayZ = new double[4];
    double[] doubleArrayB = new double[4];
    double[] doubleArrayT = new double[1];

    int samplesPerPacket = 4;
    int packetsPerSecond = 20;
    int secondsToDisplay = 1;
    int numberOfPoints = samplesPerPacket * packetsPerSecond *
        secondsToDisplay / 2;

    DataReader.FromBuffer(eventArgs.CharacteristicValue).ReadBytes(byteArray)
        ;

    if (realTimeX.Count >= numberOfPoints)
    {
        realTimeX.RemoveRange(0, 4);
        realTimeY.RemoveRange(0, 4);
        realTimeZ.RemoveRange(0, 4);
        realTimeB.RemoveRange(0, 4);
        realTimeT.RemoveRange(0, 1);
        movingAvT.RemoveRange(0, 1);
    }

    for (int i = 0; i < 4; i++)
    {
        doubleArrayX[i] = (double)((sbyte)byteArray[0 + 5 * i] * 4 + ((
            byteArray[4 + 5 * i] & 0xC0) >> 6)) / 256;
        doubleArrayY[i] = (double)((sbyte)byteArray[1 + 5 * i] * 4 + ((
            byteArray[4 + 5 * i] & 0x30) >> 4)) / 256;
        doubleArrayZ[i] = (double)((sbyte)byteArray[2 + 5 * i] * 4 + ((
            byteArray[4 + 5 * i] & 0x0C) >> 2)) / 256;
        doubleArrayB[i] = (double)(byteArray[3 + 5 * i]) / 256 * 3.3;
    }
    doubleArrayT[0] = (double)(sbyte)(((byteArray[4 + 5 * 0] & 0x03) << 6) +
        ((byteArray[4 + 5 * 1] & 0x03) << 4) + ((byteArray[4 + 5 * 2] & 0x03)
         << 2) + ((byteArray[4 + 5 * 3] & 0x03))) / 2 + 24.0;

    for (int i = 0; i < 3; i++)
    {
        realTimeX.Add(doubleArrayX[i]);
        realTimeY.Add(doubleArrayY[i]);
        realTimeZ.Add(doubleArrayZ[i]);
        realTimeB.Add(doubleArrayB[i]);
    }
    realTimeT.Add(doubleArrayT[0]);

    if (realTimeT.Count <= movingAvTDepth)
    {
        movingAvT.Add(realTimeT.Sum() / realTimeT.Count);
    }
    else
    {
        double midSum = 0;
        for (int i = realTimeT.Count - 1; i >= realTimeT.Count -
            movingAvTDepth; i--)
        {
            midSum += realTimeT[i];
        }
        movingAvT.Add(midSum / movingAvTDepth);
    }


    var dispatcher = Dispatcher;
    await Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.Normal,
        () =>
    {
        chartX.DataContext = null;
        chartX.DataContext = realTimeX;
        chartY.DataContext = null;
        chartY.DataContext = realTimeY;
        chartZ.DataContext = null;
        chartZ.DataContext = realTimeZ;
        chartB.DataContext = null;
        chartB.DataContext = realTimeB;
        chartT.DataContext = null;
        chartT.DataContext = movingAvT;
    });
}

private void chartX_Tapped(object sender, Windows.UI.Xaml.Input.
    TappedRoutedEventArgs e)
{
    if ((chartX.VerticalAxis as NumericalAxis).Maximum == Double.
        PositiveInfinity)
    {
        (chartX.VerticalAxis as NumericalAxis).Maximum = 2.0;
        (chartX.VerticalAxis as NumericalAxis).Minimum = -2.0;
        chartX.VerticalAxis.Title = "acc.X[g], fixed";
    }
    else
    {
        (chartX.VerticalAxis as NumericalAxis).Maximum = Double.
            PositiveInfinity;
        (chartX.VerticalAxis as NumericalAxis).Minimum = Double.
            NegativeInfinity;
        chartX.VerticalAxis.Title = "acc.X[g], auto";
    }
}

private void chartY_Tapped(object sender, Windows.UI.Xaml.Input.
    TappedRoutedEventArgs e)
{
    if ((chartY.VerticalAxis as NumericalAxis).Maximum == Double.
        PositiveInfinity)
    {
        (chartY.VerticalAxis as NumericalAxis).Maximum = 2.0;
        (chartY.VerticalAxis as NumericalAxis).Minimum = -2.0;
        chartY.VerticalAxis.Title = "acc.Y[g], fixed";
    }
    else
    {
        (chartY.VerticalAxis as NumericalAxis).Maximum = Double.
            PositiveInfinity;
        (chartY.VerticalAxis as NumericalAxis).Minimum = Double.
            NegativeInfinity;
        chartY.VerticalAxis.Title = "acc.Y[g], auto";
    }
}

private void chartZ_Tapped(object sender, Windows.UI.Xaml.Input.
    TappedRoutedEventArgs e)
{
    if ((chartZ.VerticalAxis as NumericalAxis).Maximum == Double.
        PositiveInfinity)
    {
        (chartZ.VerticalAxis as NumericalAxis).Maximum = 2.0;
        (chartZ.VerticalAxis as NumericalAxis).Minimum = -2.0;
        chartZ.VerticalAxis.Title = "acc.Z[g], fixed";
    }
    else
    {
        (chartZ.VerticalAxis as NumericalAxis).Maximum = Double.
            PositiveInfinity;
        (chartZ.VerticalAxis as NumericalAxis).Minimum = Double.
            NegativeInfinity;
        chartZ.VerticalAxis.Title = "acc.Z[g], auto";
    }
}

private void chartB_Tapped(object sender, Windows.UI.Xaml.Input.
    TappedRoutedEventArgs e)
{
    if ((chartB.VerticalAxis as NumericalAxis).Maximum == Double.
        PositiveInfinity)
    {
        (chartB.VerticalAxis as NumericalAxis).Maximum = 3.5;
        (chartB.VerticalAxis as NumericalAxis).Minimum = -0.5;
        chartB.VerticalAxis.Title = "BEMF [V], fixed";
    }
    else
    {
        (chartB.VerticalAxis as NumericalAxis).Maximum = Double.
            PositiveInfinity;
            (chartB.VerticalAxis as NumericalAxis).Minimum = Double.
                NegativeInfinity;
            chartB.VerticalAxis.Title = "BEMF [V], auto";
        }
    }

    private void chartT_Tapped(object sender, Windows.UI.Xaml.Input.
        TappedRoutedEventArgs e)
    {
        if ((chartT.VerticalAxis as NumericalAxis).Maximum == Double.
            PositiveInfinity)
        {
            (chartT.VerticalAxis as NumericalAxis).Maximum = 90;
            (chartT.VerticalAxis as NumericalAxis).Minimum = -40;
            chartT.VerticalAxis.Title = "Temp. [C], fixed";
        }
        else
        {
            (chartT.VerticalAxis as NumericalAxis).Maximum = Double.
                PositiveInfinity;
            (chartT.VerticalAxis as NumericalAxis).Minimum = Double.
                NegativeInfinity;
            chartT.VerticalAxis.Title = "Temp. [C], auto";
        }
    }

    private void buttonAutoRange_Click(object sender, RoutedEventArgs e)
    {
        if ((string)(buttonAutoRange.Content) == "AutoRange On")
        {
            (chartX.VerticalAxis as NumericalAxis).Maximum = Double.
                PositiveInfinity;
            (chartX.VerticalAxis as NumericalAxis).Minimum = Double.
                NegativeInfinity;
            chartX.VerticalAxis.Title = "acc.X[g], auto";
            (chartY.VerticalAxis as NumericalAxis).Maximum = Double.
                PositiveInfinity;
            (chartY.VerticalAxis as NumericalAxis).Minimum = Double.
                NegativeInfinity;
            chartY.VerticalAxis.Title = "acc.Y[g], auto";
            (chartZ.VerticalAxis as NumericalAxis).Maximum = Double.
                PositiveInfinity;
            (chartZ.VerticalAxis as NumericalAxis).Minimum = Double.
                NegativeInfinity;
            chartZ.VerticalAxis.Title = "acc.Z[g], auto";
        (chartB.VerticalAxis as NumericalAxis).Maximum = Double.
            PositiveInfinity;
        (chartB.VerticalAxis as NumericalAxis).Minimum = Double.
            NegativeInfinity;
        chartB.VerticalAxis.Title = "BEMF [V], auto";
        (chartT.VerticalAxis as NumericalAxis).Maximum = Double.
            PositiveInfinity;
        (chartT.VerticalAxis as NumericalAxis).Minimum = Double.
            NegativeInfinity;
        chartT.VerticalAxis.Title = "Temp. [C], auto";
        buttonAutoRange.Content = "AutoRange Off";
    }
    else
    {
        (chartX.VerticalAxis as NumericalAxis).Maximum = 2.0;
        (chartX.VerticalAxis as NumericalAxis).Minimum = -2.0;
        chartX.VerticalAxis.Title = "acc.X[g], fixed";
        (chartY.VerticalAxis as NumericalAxis).Maximum = 2.0;
        (chartY.VerticalAxis as NumericalAxis).Minimum = -2.0;
        chartY.VerticalAxis.Title = "acc.Y[g], fixed";
        (chartZ.VerticalAxis as NumericalAxis).Maximum = 2.0;
        (chartZ.VerticalAxis as NumericalAxis).Minimum = -2.0;
        chartZ.VerticalAxis.Title = "acc.Z[g], fixed";
        (chartB.VerticalAxis as NumericalAxis).Maximum = 3.5;
        (chartB.VerticalAxis as NumericalAxis).Minimum = -0.5;
        chartB.VerticalAxis.Title = "BEMF [V], fixed";
        (chartT.VerticalAxis as NumericalAxis).Maximum = 90;
        (chartT.VerticalAxis as NumericalAxis).Minimum = -40;
        chartT.VerticalAxis.Title = "Temp. [C], fixed";
        buttonAutoRange.Content = "AutoRange On";
    }
}

private IReadOnlyList<WiFiAdapter> localWifiAdapters;
private WiFiAdapter localWifiAdapter;

private async void buttonDiscoverWIFI_Click(object sender, RoutedEventArgs e)
{
    buttonDiscoverWIFI.IsEnabled = false;
    if (localWifiAdapters == null)
    {
        localWifiAdapters = await WiFiAdapter.FindAllAdaptersAsync();
    }

    listBoxWIFI.Items.Clear();
```

```
        localWifiAdapter = localWifiAdapters[0];

        await localWifiAdapter.ScanAsync();
        var localWifiNetworks = localWifiAdapter.NetworkReport.AvailableNetworks;
        foreach (var network in localWifiNetworks)
        {
            if (network.Ssid.Contains("VAMPIRE"))
            {
                myLocalNetwork localNetwork = new myLocalNetwork();
                localNetwork.displayName = network.Ssid;
                localNetwork.wiFiAvailableNetwork = network;
                listBoxWIFI.Items.Add(localNetwork);
            }
        }
        buttonDiscoverWIFI.IsEnabled = true;
    }

    private async void buttonConnectWIFI_Click(object sender, RoutedEventArgs e)
    {
        if (listBoxWIFI.Items.Count > 0 && listBoxWIFI.SelectedItem != null)
        {
            PasswordCredential networkPassword = new PasswordCredential();
            networkPassword.Password = "01234567";
            statusConnectionWIFI.Text = "Connecting to WIFI: " + (listBoxWIFI.
                SelectedItem.ToString());
            WiFiConnectionResult x = await localWifiAdapter.ConnectAsync((
                listBoxWIFI.SelectedItem as myLocalNetwork).wiFiAvailableNetwork,
                 WiFiReconnectionKind.Automatic, networkPassword);
            if (x.ConnectionStatus == WiFiConnectionStatus.Success)
            {
                statusConnectionWIFI.Text = "Connected to WIFI: " + (listBoxWIFI.
                    SelectedItem.ToString());
            }
            else
            {
                statusConnectionWIFI.Text = "WIFI Connection Failure";
            }

        }
    }

    private void buttonDisconnectWIFI_Click(object sender, RoutedEventArgs e)
    {
        if (localWifiAdapter != null)
        {
```

```
        localWifiAdapter.Disconnect();
        statusConnectionWIFI.Text = "WIFI Disconnected";
    }
}

private void listBoxWIFI_DoubleTapped(object sender, Windows.UI.Xaml.Input.
    DoubleTappedRoutedEventArgs e)
{
    buttonConnectWIFI_Click(sender, e);
}

private async void buttonTurnOnWIFI_Click(object sender, RoutedEventArgs e)
{
    if (vampireConnected & sdOn)
    {
        GattReadResult x = await vampireCharPowerTable[0].ReadValueAsync(
            BluetoothCacheMode.Uncached);
        byte[] byteArray = new byte[x.Value.Length];
        DataReader.FromBuffer(x.Value).ReadBytes(byteArray);

        if (byteArray.Length > 0)
        {
            Debug.WriteLine("read value: " + byteArray[0].ToString("X2"));
        }

        byteArray[0] = (byte)((byte)byteArray[0] | (byte)0x0A);
        byte[] command = { 1, byteArray[0] };

        Debug.WriteLine("changed value: " + byteArray[0].ToString("X2"));

        var writer = new DataWriter();
        writer.WriteBytes(command);
        GattCommunicationStatus writeStatus = await vampireCharCommand[0].
            WriteValueAsync(writer.DetachBuffer());
        wifiOn = true;

        buttonTurnOffSD.IsEnabled = false;
    }
}

private async void buttonTurnOffWIFI_Click(object sender, RoutedEventArgs e)
{
    if (vampireConnected & sdOn)
    {
            GattReadResult x = await vampireCharPowerTable[0].ReadValueAsync(
                BluetoothCacheMode.Uncached);
            byte[] byteArray = new byte[x.Value.Length];
            DataReader.FromBuffer(x.Value).ReadBytes(byteArray);

            if (byteArray.Length > 0)
            {
                Debug.WriteLine("read value: " + byteArray[0].ToString("X2"));
            }

            byteArray[0] = (byte)((byte)byteArray[0] & (byte)0xFD);
            byte[] command = { 1, byteArray[0] };

            Debug.WriteLine("changed value: " + byteArray[0].ToString("X2"));

            var writer = new DataWriter();
            writer.WriteBytes(command);
            GattCommunicationStatus writeStatus = await vampireCharCommand[0].
                WriteValueAsync(writer.DetachBuffer());
            wifiOn = false;

            buttonTurnOffSD.IsEnabled = true;
        }
    }

    private DispatcherTimer refreshTimerRealtimeClock = new DispatcherTimer();
    private int currentFileTableIndex = 0;
    private byte[] tableRTCData;
    private int[] listboxRTCIndex;
    private int selectedDataSlot = 0;

    private void backgroundRTC()
    {
        refreshTimerRealtimeClock.Tick += RefreshTimerRealtimeClock_Tick;
        refreshTimerRealtimeClock.Interval = new TimeSpan(0, 0, 0, 1);
        refreshTimerRealtimeClock.Start();
        currentFileTableIndex = 0;
    }

    private async void RefreshTimerRealtimeClock_Tick(object sender, object e)
    {
        for (int j = 0; j < listBoxFileTable.Items.Count; j++)
        {
            if ((listBoxFileTable.Items[j] as myFileName).fileNumber ==
                selectedDataSlot)
    {
        listBoxFileTable.SelectedIndex = j;
    }
}
if (listBoxFileTable.Items.Count - currentFileTableIndex < 4)
{
    listboxRTCIndex = new int[listBoxFileTable.Items.Count -
        currentFileTableIndex];
    tableRTCData = new byte[listBoxFileTable.Items.Count -
        currentFileTableIndex + 1];
    tableRTCData[0] = 0x12;
    for (int i = 1; i < tableRTCData.Length; i++)
    {
        listboxRTCIndex[i - 1] = currentFileTableIndex + i - 1;
        tableRTCData[i] = (byte)(listBoxFileTable.Items[
            currentFileTableIndex + i - 1] as myFileName).fileNumber;
        Debug.WriteLine("slot: " + tableRTCData[i]);
    }
    currentFileTableIndex += tableRTCData.Length - 1;
}
else
{
    listboxRTCIndex = new int[listBoxFileTable.Items.Count -
        currentFileTableIndex];
    tableRTCData = new byte[4 + 1];
    tableRTCData[0] = 0x12;
    for (int i = 1; i < tableRTCData.Length; i++)
    {
        listboxRTCIndex[i - 1] = currentFileTableIndex + i - 1;
        tableRTCData[i] = (byte)(listBoxFileTable.Items[
            currentFileTableIndex + i - 1] as myFileName).fileNumber;
        Debug.WriteLine("slot: " + tableRTCData[i]);
    }
    currentFileTableIndex += 4;
}

vampireCharRealtimeClock[0].ValueChanged += tableClockReceived;
await vampireCharRealtimeClock[0].
    WriteClientCharacteristicConfigurationDescriptorAsync(
    GattClientCharacteristicConfigurationDescriptorValue.Notify);
var writer = new DataWriter();
writer.WriteBytes(tableRTCData);
GattCommunicationStatus writeStatus = await vampireCharCommand[0].
    WriteValueAsync(writer.DetachBuffer());
    if (currentFileTableIndex >= listBoxFileTable.Items.Count)
    {
        refreshTimerRealtimeClock.Stop();
        currentFileTableIndex = 0;
        refreshTimerRealtimeClock.Tick -= RefreshTimerRealtimeClock_Tick;
        Debug.WriteLine("refresher ended!");
    }

    for (int j = 0; j < listBoxFileTable.Items.Count; j++)
    {
        if ((listBoxFileTable.Items[j] as myFileName).fileNumber ==
            selectedDataSlot)
        {
            listBoxFileTable.SelectedIndex = j;
        }
    }
}

private async void tableClockReceived(GattCharacteristic sender,
    GattValueChangedEventArgs args)
{
    GattReadResult x = await vampireCharRealtimeClock[0].ReadValueAsync(
        BluetoothCacheMode.Uncached);
    byte[] byteArray = new byte[x.Value.Length];
    DataReader.FromBuffer(x.Value).ReadBytes(byteArray);

    for (int i = 0; i < tableRTCData.Length - 1; i++) // number of slots,
        each slot of 5 bytes (year/month/day/hour/minute)
    {
        string timestamp = "20" + byteArray[5 * i + 0].ToString("X2") + "/" +
             byteArray[5 * i + 1].ToString("X2") + "/" + byteArray[5 * i +
            2].ToString("X2") + " " + byteArray[5 * i + 3].ToString("X2") +
            ":" + byteArray[5 * i + 4].ToString("X2");
        var dispatcher = Dispatcher;
        await Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.
            Normal, () =>
        {
            (listBoxFileTable.Items[listboxRTCIndex[i]] as myFileName).
                rtcData = timestamp;
            for (int j = 0; j < listBoxFileTable.Items.Count; j++)
            {
                if ((listBoxFileTable.Items[j] as myFileName).fileNumber ==
                    selectedDataSlot)
                {
                    listBoxFileTable.SelectedIndex = j;
                }
            }
        });

        Debug.WriteLine(timestamp);
        Debug.WriteLine("listbox index = " + listboxRTCIndex[i] + " -
            dataslot index = " + tableRTCData[i + 1]);
        //
    }

    for (int i = 0; i < tableRTCData.Length - 1; i++)
    {
        await Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.
            Normal, () =>
        {
            var y = (listBoxFileTable.Items[listboxRTCIndex[i]] as myFileName
                );
            listBoxFileTable.Items[listboxRTCIndex[i]] = y;
            for (int j = 0; j < listBoxFileTable.Items.Count; j++)
            {
                if ((listBoxFileTable.Items[j] as myFileName).fileNumber ==
                    selectedDataSlot)
                {
                    listBoxFileTable.SelectedIndex = j;
                }
            }
        });
    }

    vampireCharRealtimeClock[0].ValueChanged -= tableClockReceived;
    await vampireCharRealtimeClock[0].
        WriteClientCharacteristicConfigurationDescriptorAsync(
        GattClientCharacteristicConfigurationDescriptorValue.None);
}

private async void buttonShowFileTable_Click(object sender, RoutedEventArgs e
    )
{
    if (vampireConnected)
    {
        GattReadResult x = await vampireCharFileTable[0].ReadValueAsync(
            BluetoothCacheMode.Uncached);
        byte[] byteArray = new byte[x.Value.Length];
        DataReader.FromBuffer(x.Value).ReadBytes(byteArray);
        int currentIndex = byteArray[0];
```

```
listBoxFileTable.Items.Clear();

if (currentIndex < 1)
{
    currentIndex = 1;
}
if (currentIndex > 150)
{
    currentIndex = 150;
}

listBoxFileTable.Items.Clear();

for (int i = currentIndex; ; i--)
{
    if (i == 0)
    {
        if (currentIndex == 150)
        {
            break;
        }
        i = 150;
    }

    int q = (i - 1) / 8 + 1;     // byte index (1-based)
    int r = (i - 1) % 8;

    if (((byteArray[q] & (1 << (7 - r))) != 0))
    {
        myFileName fileName = new myFileName();
        fileName.fileNumber = i;
        fileName.displayName = "DATA" + i.ToString("D4");
        fileName.url = "http://flashair/FILES/" + fileName.
            displayName + ".RAW";
        listBoxFileTable.Items.Add(fileName);
    }

    if (i == currentIndex + 1)
    {
        break;
    }
}

for (int j = 0; j < listBoxFileTable.Items.Count; j++)
{
```

```
                if ((listBoxFileTable.Items[j] as myFileName).fileNumber ==
                    selectedDataSlot)
                {
                    listBoxFileTable.SelectedIndex = j;
                }
            }
            refreshTimerRealtimeClock.Tick -= RefreshTimerRealtimeClock_Tick;
            backgroundRTC();
        }
        else
        {
            listBoxFileTable.Items.Clear();
        }
    }

    private StorageFolder rawDataFolder;

    private async void buttonDownload_Click(object sender, RoutedEventArgs e)
    {
        if (!vampireConnected)
        {
            listBoxFileTable.Items.Clear();
        }
        if ((listBoxFileTable.SelectedItem != null) && (vampireConnected) &&
            wifiOn)
        {
            object currentItem = listBoxFileTable.SelectedItem;

            statusFileDownload.Text = "Initiating a data link";
            statusFileDownload.InvalidateMeasure();

            HttpClientHandler aHandler = new HttpClientHandler();
            aHandler.ClientCertificateOptions = ClientCertificateOption.Automatic
                ;

            HttpClient aClient = new HttpClient(aHandler);
            aClient.DefaultRequestHeaders.ExpectContinue = false;

            string url = (currentItem as myFileName).url;

            HttpResponseMessage response = new HttpResponseMessage();
            bool noExceptionOccurred = true;
            try
            {
```

```
    response = await aClient.GetAsync(url, HttpCompletionOption.
        ResponseHeadersRead);
}
catch (Exception)
{
    noExceptionOccurred = false;
    Debug.WriteLine("WIFI Data Connection Failed");
}

if (response.IsSuccessStatusCode && noExceptionOccurred)
{
    string filename = (currentItem as myFileName).displayName + ".RAW
        ";

    if (rawDataFolder == null)
    {
        FolderPicker folderPicker = new FolderPicker();
        folderPicker.SuggestedStartLocation = PickerLocationId.
            Desktop;
        folderPicker.FileTypeFilter.Add(".raw");
        rawDataFolder = await folderPicker.PickSingleFolderAsync();
        if (rawDataFolder != null)
        {
            StorageApplicationPermissions.FutureAccessList.
                AddOrReplace("VAMPIRE RawData", rawDataFolder);
            Debug.WriteLine("Accessable Folder: " + rawDataFolder.
                Name);
            Debug.WriteLine("Accessable Folder: " + rawDataFolder.
                Path);
        }
    }

    if (rawDataFolder != null)
    {
        StorageFile dataFile = await rawDataFolder.CreateFileAsync(
            filename, CreationCollisionOption.ReplaceExisting);
        var fs = await dataFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream stream = await response.Content.ReadAsStreamAsync();
        IInputStream inputStream = stream.AsInputStream();

        ulong totalBytesRead = 0;
        while (true)
        {
            IBuffer buffer = new Windows.Storage.Streams.Buffer(1024)
                ;
```

```
                        buffer = await inputStream.ReadAsync(buffer, buffer.
                            Capacity, InputStreamOptions.None);
                        if (buffer.Length == 0)
                        {
                            break;
                        }
                        totalBytesRead += buffer.Length;
                        statusFileDownload.Text = ((double)totalBytesRead /
                            1048576 * 100).ToString("F3") + "%";
                        statusFileDownload.InvalidateMeasure();
                        await fs.WriteAsync(buffer);
                    }
                    statusFileDownload.Text = "Successfully Downloaded to:\n" +
                        dataFile.Path;
                    statusFileDownload.InvalidateMeasure();
                    inputStream.Dispose();
                    fs.Dispose();
                }
                else
                {
                    statusFileDownload.Text = "Folder selection canceled";
                }
            }
            else
            {
                statusFileDownload.Text = "Error opening a data link";
                statusFileDownload.InvalidateMeasure();
            }
        }
    }



    private DispatcherTimer wifiBootTimer = new DispatcherTimer();
    private int elapsedWifiBootTime = 0;
    private int maxWifiBootTime = 7;

    private void wifiBootWaiter()
    {
        buttonTurnOnWIFI.Content = "WIFI On (" + (maxWifiBootTime).ToString() +
            ")";
        buttonTurnOffWIFI.Content = "WIFI Off (" + (maxWifiBootTime).ToString() +
             ")";

        wifiBootTimer.Tick += WifiBootTimer_Tick; ;
    wifiBootTimer.Interval = new TimeSpan(0, 0, 0, 1);
    wifiBootTimer.Start();
    elapsedWifiBootTime = 0;
}

private void WifiBootTimer_Tick(object sender, object e)
{
    elapsedWifiBootTime += 1;
    buttonTurnOnWIFI.Content = "WIFI On (" + (maxWifiBootTime -
        elapsedWifiBootTime).ToString() + ")";
    buttonTurnOffWIFI.Content = "WIFI Off (" + (maxWifiBootTime -
        elapsedWifiBootTime).ToString() + ")";
    if (elapsedWifiBootTime == maxWifiBootTime)
    {
        buttonTurnOnWIFI.Content = "WIFI On";
        buttonTurnOffWIFI.Content = "WIFI Off";
        buttonTurnOnWIFI.IsEnabled = true;
        buttonTurnOffWIFI.IsEnabled = true;
        wifiBootTimer.Tick -= WifiBootTimer_Tick;
    }
}

private async void buttonTurnOnSD_Click(object sender, RoutedEventArgs e)
{
    if (vampireConnected)
    {
        GattReadResult x = await vampireCharPowerTable[0].ReadValueAsync(
            BluetoothCacheMode.Uncached);
        byte[] byteArray = new byte[x.Value.Length];
        DataReader.FromBuffer(x.Value).ReadBytes(byteArray);

        if (byteArray.Length > 0)
        {
            Debug.WriteLine("read value: " + byteArray[0].ToString("X2"));
        }

        byteArray[0] = (byte)((byte)byteArray[0] | (byte)0x08);
        byte[] command = { 1, byteArray[0] };

        Debug.WriteLine("changed value: " + byteArray[0].ToString("X2"));

        var writer = new DataWriter();
        writer.WriteBytes(command);
        GattCommunicationStatus writeStatus = await vampireCharCommand[0].
            WriteValueAsync(writer.DetachBuffer());
            sdOn = true;
            wifiBootWaiter();
        }
    }

    private async void buttonTurnOffSD_Click(object sender, RoutedEventArgs e)
    {
        if (vampireConnected)
        {
            GattReadResult x = await vampireCharPowerTable[0].ReadValueAsync(
                BluetoothCacheMode.Uncached);
            byte[] byteArray = new byte[x.Value.Length];
            DataReader.FromBuffer(x.Value).ReadBytes(byteArray);

            if (byteArray.Length > 0)
            {
                Debug.WriteLine("read value: " + byteArray[0].ToString("X2"));
            }

            byteArray[0] = (byte)((byte)byteArray[0] & (byte)0xF5);
            byte[] command = { 1, byteArray[0] };

            Debug.WriteLine("changed value: " + byteArray[0].ToString("X2"));

            var writer = new DataWriter();
            writer.WriteBytes(command);
            GattCommunicationStatus writeStatus = await vampireCharCommand[0].
                WriteValueAsync(writer.DetachBuffer());
            sdOn = false;
            buttonTurnOnWIFI.IsEnabled = false;
            buttonTurnOffWIFI.IsEnabled = false;
        }
    }

    private void listBoxFileTable_SelectionChanged(object sender,
        SelectionChangedEventArgs e)
    {
        if (listBoxFileTable.SelectedItem != null)
        {
            object currentItem = listBoxFileTable.SelectedItem;
            selectedDataSlot = (currentItem as myFileName).fileNumber;
            Debug.WriteLine("data slot = " + selectedDataSlot);
        }
    }
```

```
private void statusFileDownload_RightTapped(object sender, Windows.UI.Xaml.
    Input.RightTappedRoutedEventArgs e)
{
    statusFileDownload.Text = "";
}

private async void buttonDownload_DoubleTapped(object sender, Windows.UI.Xaml
    .Input.DoubleTappedRoutedEventArgs e)
{
    FolderPicker folderPicker = new FolderPicker();
    folderPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    folderPicker.FileTypeFilter.Add(".raw");
    rawDataFolder = await folderPicker.PickSingleFolderAsync();
    if (rawDataFolder != null)
    {
        StorageApplicationPermissions.FutureAccessList.AddOrReplace("VAMPIRE
            RawData", rawDataFolder);
        Debug.WriteLine("Accessable Folder: " + rawDataFolder.Name);
        Debug.WriteLine("Accessable Folder: " + rawDataFolder.Path);
    }
}

private async void buttonCapturePrev100s_Click(object sender, RoutedEventArgs
     e)
{
    if (vampireConnected)
    {
        byte[] commandBytes = { 5, 1 };
        var writer = new DataWriter();
        writer.WriteBytes(commandBytes);
        GattCommunicationStatus writeStatus = await vampireCharCommand[0].
            WriteValueAsync(writer.DetachBuffer());
    }
}

private async void buttonCaptureNext100s_Click(object sender, RoutedEventArgs
     e)
{
    if (vampireConnected)
    {
        byte[] commandBytes = { 4, 1 };
        var writer = new DataWriter();
        writer.WriteBytes(commandBytes);
        GattCommunicationStatus writeStatus = await vampireCharCommand[0].
            WriteValueAsync(writer.DetachBuffer());
    }
}

        async void rtcChanged(GattCharacteristic sender,
            GattValueChangedEventArgs eventArgs)
{
    GattReadResult x = await vampireCharRealtimeClock[0].ReadValueAsync(
        BluetoothCacheMode.Uncached);
    byte[] byteArray = new byte[x.Value.Length];
    DataReader.FromBuffer(x.Value).ReadBytes(byteArray);

    DateTime rtcClock = new DateTime(2000 + int.Parse(byteArray[0].ToString("
        X2")), int.Parse(byteArray[1].ToString("X2")), int.Parse(byteArray
        [2].ToString("X2")), int.Parse(byteArray[3].ToString("X2")), int.
        Parse(byteArray[4].ToString("X2")), int.Parse(byteArray[5].ToString("
        X2")));
    vampireCharRealtimeClock[0].ValueChanged -= rtcChanged;
    await Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.Normal,
        () =>
    {
        statusTime.Text = rtcClock.ToString("yyyy/MM/dd - HH:mm:ss");
    });
}

public async void buttonRefreshTime_Click(object sender, RoutedEventArgs e)
{
    if (vampireConnected)
    {
        vampireCharRealtimeClock[0].ValueChanged += rtcChanged;
        await vampireCharRealtimeClock[0].
            WriteClientCharacteristicConfigurationDescriptorAsync(
            GattClientCharacteristicConfigurationDescriptorValue.Notify);

        byte[] commandBytes = { 8 };
        var writer = new DataWriter();
        writer.WriteBytes(commandBytes);
        GattCommunicationStatus writeStatus = await vampireCharCommand[0].
            WriteValueAsync(writer.DetachBuffer());
    }
}

private async void buttonChangeTime_Click(object sender, RoutedEventArgs e)
{
    byte rtc_Y, rtc_M, rtc_D, rtc_H, rtc_m, rtc_s;
```

```
        if (textBoxChangeTime.Text.Length != 19)
        {
            textBoxChangeTime.Text = DateTime.Now.ToString("yyyy/MM/dd HH:mm:ss")
                ;
            return;
        }
        rtc_Y = (byte)Convert.ToInt32(textBoxChangeTime.Text.Substring(2, 2), 16)
            ;
        rtc_M = (byte)Convert.ToInt32(textBoxChangeTime.Text.Substring(5, 2), 16)
            ;
        rtc_D = (byte)Convert.ToInt32(textBoxChangeTime.Text.Substring(8, 2), 16)
            ;
        rtc_H = (byte)Convert.ToInt32(textBoxChangeTime.Text.Substring(11, 2),
            16);
        rtc_m = (byte)Convert.ToInt32(textBoxChangeTime.Text.Substring(14, 2),
            16);
        rtc_s = (byte)Convert.ToInt32(textBoxChangeTime.Text.Substring(17, 2),
            16);

        if (vampireConnected)
        {
            byte[] commandBytes = { 9, rtc_Y, rtc_M, rtc_D, rtc_H, rtc_m, rtc_s
                };
            var writer = new DataWriter();
            writer.WriteBytes(commandBytes);
            GattCommunicationStatus writeStatus = await vampireCharCommand[0].
                WriteValueAsync(writer.DetachBuffer());
            textBoxChangeTime.Text = "";
        }
    }

    private void buttonRTCPicker_Click(object sender, RoutedEventArgs e)
    {
        if (rtcDatePicker.Visibility == Visibility.Collapsed)
        {
            buttonRTCPicker.Content = "Hide";
            rtcDatePicker.Visibility = Visibility.Visible;
            rtcTimePicker.Visibility = Visibility.Visible;

            textBoxChangeTime.Text = rtcDatePicker.Date.DateTime.ToString("yyyy/
                MM/dd") + " " + rtcTimePicker.Time.ToString();
        }
        else
        {
            buttonRTCPicker.Content = "Pick";
            rtcDatePicker.Visibility = Visibility.Collapsed;
            rtcTimePicker.Visibility = Visibility.Collapsed;
        }
    }

    private void rtcDatePicker_DateChanged(object sender,
        DatePickerValueChangedEventArgs e)
    {
        if (textBoxChangeTime.Text.Length != 19)
        {
            textBoxChangeTime.Text = rtcDatePicker.Date.DateTime.ToString("yyyy/
                MM/dd");
        }
        else
        {
            textBoxChangeTime.Text = rtcDatePicker.Date.DateTime.ToString("yyyy/
                MM/dd") + textBoxChangeTime.Text.Substring(textBoxChangeTime.Text
                .Length - 9);
        }
    }

    private void rtcTimePicker_TimeChanged(object sender,
        TimePickerValueChangedEventArgs e)
    {
        Debug.WriteLine(rtcTimePicker.Time.ToString());
        if (textBoxChangeTime.Text.Length != 19)
        {
            textBoxChangeTime.Text = DateTime.Now.ToString("yyyy/MM/dd") + " " +
                rtcTimePicker.Time.ToString();
        }
        else
        {
            textBoxChangeTime.Text = textBoxChangeTime.Text.Substring(0,11) +
                rtcTimePicker.Time.ToString();
        }
    }

    private void listBoxFileTable_RightTapped(object sender, Windows.UI.Xaml.
        Input.RightTappedRoutedEventArgs e)
    {
        Debug.WriteLine( (listBoxFileTable.SelectedItem as myFileName).rtcData.
            ToString() );
    }
}
```

```
    public class myDeviceInformation
    {
        public DeviceInformation deviceInformation { get; set; }
        public string displayName { get; set; }
        public override string ToString()
        {
            return displayName;
        }
    }

    public class myLocalNetwork
    {
        public WiFiAvailableNetwork wiFiAvailableNetwork { get; set; }
        public string displayName { get; set; }
        public override string ToString()
        {
            return displayName;
        }
    }

    public class myFileName
    {
        public string url { get; set; }
        public string rtcData { get; set; }
        public string displayName { get; set; }
        public int fileNumber { get; set; }
        public override string ToString()
        {
            if (rtcData == "" || rtcData == null)
            {
                return displayName;
            }
            else
            {
                return displayName + " - " + rtcData;
            }
        }
    }
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix G

Signal Processing — Impedance Spectroscopy

This appendix presents the signal processing MATLAB code for generating an eVTF, described in Chapter 4.7. The raw data file retrieved from VAMPIRE is directly inserted in the script.

G.1 Signal Processing Code

```
close all;
clear all;

display_mode = 0;    %0 = (1920 x 1080 x 3), 1 = (2560 x 1440 x 1)
plot_on = 0;
ylim_override = 1;
ylimov_min = -135;
ylimov_max = -90;

%% Measurement File Reading
path_directory = 'DCData';
path_filename = 'DATA0026.RAW'; SPINDOWN_L = 80; SPINDOWN_R = 95;
path_file = sprintf('%s\\%s',path_directory, path_filename);

handle_file = fopen(path_file, 'r');
bytesPerSample = 5;
samplesPerPacket = 100;
```

```
bytesPerPacket = bytesPerSample * samplesPerPacket;       % excluding temperature
    data, which will give +1
numPackets = 2000;

data_binary = zeros(bytesPerSample, samplesPerPacket * numPackets);
temp_binary = zeros(numPackets, 1);

index_write = 1;

for i=1:numPackets % binary file reading
 data_binary(index_write:index_write+bytesPerPacket-1) = fread(handle_file,
     bytesPerPacket, 'uint8');
 temp_binary(i) = fread(handle_file, 1, 'int8');
 fseek(handle_file, 512 - bytesPerPacket - 1, 'cof');
 index_write = index_write + bytesPerPacket;
end
fclose(handle_file);

data_binary(1:3, :) = data_binary(1:3, :) + (data_binary(1:3, :) > 127) * (-256);   %
     convert x/y/z to signed int

data = zeros(bytesPerSample - 1, samplesPerPacket * numPackets);
for i=1:samplesPerPacket * numPackets % binary to formatted values
 data(1,i) = double(4 * data_binary(1, i) + bitshift(data_binary(5, i),-6)) /1024 *
     8;      % -4g to +4g with 1024 steps;
 data(2,i) = double(4 * data_binary(2, i) + bitand(bitshift(data_binary(5, i),-4), 3)
     ) /1024 * 8;
 data(3,i) = double(4 * data_binary(3, i) + bitand(bitshift(data_binary(5, i),-2), 3)
     ) /1024 * 8;
 data(4,i) = double(4 * data_binary(4, i) + bitand(data_binary(5, i), 3)) / 1024 *
     3.3;
end

freq_sample = 2000;
axis_time = (1:samplesPerPacket * numPackets) / freq_sample;

%% First Plot Pane: Raw Data
if plot_on == 1
 tmin = 75;
 tmax = 100;
 amin = -4;
 amax = 4;
 vmin = -0.5;
 vmax = 3.8;
 handle_figure1 = figure(1);
 if display_mode == 0
  set(handle_figure1, 'Position', [-951     9   944  1108]);
 elseif display_mode == 1
  set(handle_figure1, 'Position', [0    42   640  1314]);
 end
 subplot(3,1,1); plot(axis_time, data(1,:),'r'); axis([tmin tmax amin amax]); title('
     Raw Acc X'); xlabel('time [s]'); ylabel('acc [g]');
 subplot(3,1,2); plot(axis_time, data(2,:),'g'); axis([tmin tmax amin amax]); title('
     Raw Acc Y'); xlabel('time [s]'); ylabel('acc [g]');
 subplot(3,1,3); plot(axis_time, data(3,:),'b'); axis([tmin tmax amin amax]); title('
     Raw Acc Z'); xlabel('time [s]'); ylabel('acc [g]');
%  subplot(4,1,4); plot(axis_time, data(4,:),'k'); axis([tmin tmax vmin vmax]); title
    ('Raw BEMF');  xlabel('time [s]'); ylabel('d(BEMF)/dt [V]');
end

%% Second Plot Pane: Rotor Speed Estimation
tb = data(4,:);
mean_tb = mean(tb);
tb = tb - mean_tb;
tb = sgolayfilt(tb, 3, 33);
tt = axis_time;

[fbb, fba] = butter(4, 0.006, 'high');
data_bemf = filter(fbb, fba, cumtrapz(tb));
numZ = 4;
data_bemf = [zeros(1,numZ) data_bemf(1:end-numZ)];

tb = data_bemf;   %% It's now integrated & DC-removed!
tb = tb - mean(tb);
wavelen = length(tb);
zc = zeros(wavelen,1);
zc_val = zeros(wavelen,1);
zc_ind = 0;
for i = 2:wavelen
 if ( (tb(i-1) < 0 && tb(i) >= 0) || (tb(i-1) > 0 && tb(i) <= 0) )

  t1 = tt(i-1);
  t2 = tt(i);
  y1 = abs(tb(i-1));
  y2 = abs(tb(i));

  zc_ind = zc_ind + 1;
  zc_val(zc_ind) = (t2*y1 + t1*y2)/(y1+y2);
  zc(i) = 0.05;
 end
end
zc_val = zc_val(1:zc_ind);
rotor_time = zc_val;
rotor_speed = medfilt1([0;1./diff(zc_val)]/2,4);

if plot_on == 1
 tmin = 85;
 tmax = 95;

 handle_figure2 = figure(2);
 if display_mode == 0
  set(handle_figure2, 'Position', [1929     9   944  1108]);
 elseif display_mode == 1
  set(handle_figure2, 'Position', [640     42   640  1314]);
 end

 subplot(2,1,1); plot(tt, data_bemf); xlim([tmin, tmax]); title('Integrated Raw BEMF
     '); xlabel('time [s]'); ylabel('BEMF');
 subplot(2,1,2); plot(rotor_time, rotor_speed); xlim([tmin, tmax]); ylim([0 75]);
     title('Rotor Speed Estimation from Raw Data'); xlabel('time [s]'); ylabel('speed
      [Hz]');
end

%% Third Plot Pane: Evenly Spaced Sampling & Interpolation
tmin = SPINDOWN_L;   % Spindown Range tMIN!
tmax = SPINDOWN_R;   % Spindown Range tMAX!

target_rotor_time = rotor_time;
target_rotor_speed = rotor_speed; %medfilt1(rotor_speed,40);

index_rotor_tmin = find(target_rotor_time >= tmin, 1, 'first');
index_rotor_tmax = find(target_rotor_time <= tmax, 1, 'last');
target_rotor_time = target_rotor_time(index_rotor_tmin:index_rotor_tmax);
target_rotor_speed = target_rotor_speed(index_rotor_tmin:index_rotor_tmax);

t_even = tmin:(1/freq_sample):tmax;
speed_even = zeros(size(t_even));

for j=1:length(t_even)
 time_point = t_even(j);

 for i=2:length(target_rotor_time)
  if time_point == target_rotor_time(i-1)
   speed_even(j) = target_rotor_speed(i-1);
  elseif time_point == target_rotor_time(i)
   speed_even(j) = target_rotor_speed(i);
  elseif (target_rotor_time(i-1) < time_point && time_point < target_rotor_time(i))
   t1 = target_rotor_time(i-1);
   t2 = target_rotor_time(i);
   y1 = target_rotor_speed(i-1);
   y2 = target_rotor_speed(i);
   estimated_speed = (y1 + (time_point - t1)/(t2 - time_point) * y2) / (1 + (
       time_point - t1) / (t2 - time_point));
   speed_even(j) = estimated_speed;
   break;
  end
 end
end

valid_range = find(t_even >= min(target_rotor_time) & t_even <= max(target_rotor_time
    ));
t_even = t_even(valid_range);
speed_even = speed_even(valid_range);
virtual_amp = (2*pi*speed_even).^2;
virtual_input = virtual_amp.*sin(cumtrapz(t_even,2*pi*speed_even));

if plot_on == 1
 handle_figure3 = figure(3);
 if display_mode == 0
  set(handle_figure3, 'Position', [2889      9   944  1108]);
 elseif display_mode == 1
  set(handle_figure3, 'Position', [1280     42   640  1314]);
 end
 subplot(4,1,1); plot(target_rotor_time, target_rotor_speed); xlim([tmin, tmax]);
     ylim([0 75]); title('Uneven Raw: Speed vs Time'); xlabel('time [s]'); ylabel('
     Speed [Hz]');
 subplot(4,1,2); plot(t_even, speed_even); xlim([tmin, tmax]); ylim([0 75]); title('
     Evenly Interpolated: Speed vs Time'); xlabel('time [s]'); ylabel('Speed [Hz]');
 subplot(4,1,3); plot(t_even, virtual_amp); xlim([tmin, tmax]); title('Virtual
     Amplitude'); xlabel('time [s]'); ylabel('Relative Amplitude');
 subplot(4,1,4); plot(t_even, virtual_input); xlim([tmin, tmax]); title('Virtual
     Input'); xlabel('time [s]'); ylabel('Relative Amplitude');
end

 acc_tmin = tmin;
 acc_tmax = tmax;

 index_acc = find(axis_time <= acc_tmin, 1, 'last');
 acc_x = data(1,index_acc:(index_acc+length(virtual_input)-1)); acc_x = acc_x - mean(
     acc_x);
 acc_y = data(2,index_acc:(index_acc+length(virtual_input)-1)); acc_y = acc_y - mean(
     acc_y);
 acc_z = data(3,index_acc:(index_acc+length(virtual_input)-1)); acc_z = acc_z - mean(
     acc_z);

%% Main Signal Processing
virtual_input = virtual_input * 1.0;
cx = 0.0;
cy = 0.0;
cz = 1.0;
magxyz = sqrt(cz^2+cx^2+cy^2);
acceleration_input = (cx * acc_x + cy * acc_y + cz * acc_z)/magxyz;

window_length = 1.0; %second
window_overlap = 0.9; %percent
numSamples = round(freq_sample * window_length);
hanning_window = hanning(numSamples);
numWindows = floor(length(virtual_input)/round(numSamples*(1-window_overlap)));

mask_input = zeros(length(virtual_input),numWindows);
mask_output = zeros(length(virtual_input),numWindows);
MASK_input = zeros(length(virtual_input),numWindows);
MASK_output = zeros(length(virtual_input),numWindows);
mFRF = zeros(length(virtual_input),numWindows);

if plot_on == 1
 handle_figure5 = figure(5);
 if display_mode == 0
  set(handle_figure5, 'Position', [2889     9   944  1108]);
 elseif display_mode == 1
  set(handle_figure5, 'Position', [0     42   640  1314]);
 end
end

for i=0:numWindows-1
 center_point = round(numSamples*(1-window_overlap))*i;
 mask = zeros(size(virtual_input));
 index_mask_start = center_point - round(numSamples/2);
 mask_length = numSamples+index_mask_start-1;
 index_hanning_start = 0;
 index_hanning_end = numSamples;
 %for the beginning remove the left half of the hanning window
 if(index_mask_start<=0)
```

```
  index_hanning_start = -index_mask_start;
  mask_length = numSamples - index_hanning_start;
  index_mask_start = 1;
 end
 %for the end remove the right half of the hanning window
 if(mask_length>length(mask))
  index_hanning_end = numSamples - (mask_length-length(mask));
  mask_length = length(mask);
 end

 %.) Now add the hanning window to the mask (which is all 0's)
 mask(index_mask_start:mask_length) = hanning_window(index_hanning_start+1:
     index_hanning_end);
 mask_input(:,i+1) = mask.*virtual_input;
 mask_output(:,i+1) = mask.*acceleration_input;

 if plot_on == 1;
  if(mod(i,10)==0) % plot every 10th window
   x = (0:1:(size(mask_input,1)-1))./freq_sample;
   subplot(3,1,1); hold on
   title('Hanning Window Masks');
   plot(x,mask)
   subplot(3,1,2); hold on
   title('Masked Input');
   plot(x,mask_input(:,i+1));
   subplot(3,1,3); hold on
   title('Masked Output');
   plot(x,mask_output(:,i+1));
  end
 end
end

freq = linspace(0,freq_sample,length(mask_input(:,1)));
revPerSecond = 60;

for i=1:numWindows
 MASK_input(:,i) = fft(mask_input(:,i));
 max_freq = round(revPerSecond/freq_sample*length(mask_input(:,i)));
 [M,k] = max(MASK_input(1:max_freq,i));
 center_freq = freq(k);
 MASK_output(:,i) = fft(mask_output(:,i));
 MASK_output(freq<(center_freq-3),i)=0;
 MASK_output(freq>(center_freq+3),i)=0;
end
```

```
MASK_output_env = max(MASK_output,[],2);
MASK_input_env = max(MASK_input,[],2);

env_FRF=abs(MASK_output_env)./abs(MASK_input_env);

%% Final FRF
if plot_on == 1
 handle_figure6 = figure(6);
 if display_mode == 0
  set(handle_figure6, 'Position', [2889     9   944  1108]);
 elseif display_mode == 1
  set(handle_figure6, 'Position', [640     42   640  1314]);
 end

 fmin = 0;
 fmax = 60;

 subplot(3,1,1); plot(freq,mag2db(env_FRF));
 xlabel('Frequency (Hz)');
 ylabel('Vibration Magnitude (dB)');
 title('Vibration Transfer Function Estimate');
 xlim([fmin, fmax]);
 ylim([-150 -100]);

 RAW_input = abs(fft(virtual_input));
 RAW_output = abs(fft(acceleration_input))./2;
 subplot(3,1,2); plot(freq, mag2db(RAW_output./RAW_input));
 xlabel('Frequency (Hz)');
 ylabel('Vibration Magnitude (dB)');
 title('Direct Vibration Transfer Function');
 xlim([fmin, fmax]);
 ylim([-150 -100]);

 subplot(3,1,3); plot(freq, mag2db(env_FRF), 'r', freq, mag2db(RAW_output./RAW_input)
     , 'b');
 xlabel('Frequency (Hz)');
 ylabel('Vibration Magnitude (dB)');
 title('Overlay');
 legend('FRF', 'Direct');
 xlim([fmin fmax]);
 ylim([-150 -100]);
end

handle_figure7 = figure(7);
if display_mode == 0
```

```
 set(handle_figure7, 'Position', [-1911   513   942   258]);
elseif display_mode == 1
 set(handle_figure7, 'Position', [623   276   985   764]);
end

set(handle_figure7, 'Position', [ 9 49   666   636]);

fmin = 15;
fmax = 55;

dB_FRF = mag2db(env_FRF);
[maxdB, maxIndex] = max(dB_FRF);
plot(freq,dB_FRF, 'r', freq(maxIndex), maxdB, 'bo', [freq(maxIndex) freq(maxIndex)
    freq(maxIndex)], [-150 maxdB -100], 'k--');
xlabel('Frequency (Hz)');
ylabel('Vibration Magnitude (dB)');
title('Frequency Response Function');
xlim([fmin, fmax]);

ifmin = find(fmin <= freq, 1, 'first');
ifmax = find(fmax >= freq, 1, 'last');
max_ylim = max(dB_FRF(ifmin:ifmax));
min_ylim = min(dB_FRF(ifmin:ifmax));
margin_ylim = (max_ylim - min_ylim) * 0.1;
ylim([min_ylim - margin_ylim max_ylim + margin_ylim]);
if ylim_override == 1
 ylim([ylimov_min ylimov_max]);
end
text(freq(maxIndex) + (max(xlim) - min(xlim))*0.01, maxdB + (max(ylim) - min(ylim))
    *0.02, sprintf('%4.2f Hz', freq(maxIndex)));
legend('FRF', sprintf('Maximum of [%4.2f dB] at [%4.2f Hz]',maxdB, freq(maxIndex)), '
    Location', 'SouthEast');
```

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] R. Zachar, P. Lindahl, J. Donnal, W. Cotta, C. Schantz, and S. B. Leeb. Utilizing spin-down transients for vibration-based diagnostics of resiliently mounted machines. *IEEE Transactions on Instrumentation and Measurement*, PP(99):1–10, 2016.

[2] S. Meninger, T.O. Mur-Miranda, R. Amirtharajah, A. Chandrakasan, and J. Lang. Vibration-to-electric energy conversion. In *Low Power Electronics and Design, 1999. Proceedings. 1999 International Symposium on*, pages 48–53, Aug 1999.

[3] G.K. Ottman, H.F. Hofmann, A.C. Bhatt, and G.A. Lesieutre. Adaptive piezoelectric energy harvesting circuit for wireless remote power supply. *Power Electronics, IEEE Transactions on*, 17(5):669–676, Sep 2002.

[4] J.A. Paradiso and T. Starner. Energy scavenging for mobile and wireless electronics. *Pervasive Computing, IEEE*, 4(1):18–27, Jan 2005.

[5] I. Stark. Invited talk: Thermal energy harvesting with thermo life. In *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, pages 19–22, April 2006.

[6] Y.K. Tan and S.K. Panda. Energy harvesting from hybrid indoor ambient light and thermal energy sources for enhanced performance of wireless sensor nodes. *Industrial Electronics, IEEE Transactions on*, 58(9):4424–4435, Sept 2011.

[7] A.G. Fowler, S.O.R. Moheimani, and S. Behrens. A 3-dof mems ultrasonic energy harvester. In *Sensors, 2012 IEEE*, pages 1–4, Oct 2012.

[8] D. Brunelli, C. Moser, L. Thiele, and L. Benini. Design of a solar-harvesting circuit for batteryless embedded systems. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 56(11):2519–2528, Nov 2009.

[9] S. B. Leeb, S. R. Shaw, and J. L. Kirtley. Transient event detection in spectral envelope estimates for nonintrusive load monitoring. *IEEE Transactions on Power Delivery*, 10(3):1200–1210, Jul 1995.

[10] S. R. Shaw, S. B. Leeb, L. K. Norford, and R. W. Cox. Nonintrusive load monitoring and diagnostics in power systems. *IEEE Transactions on Instrumentation and Measurement*, 57(7):1445–1454, July 2008.

[11] S. Sudevalayam and P. Kulkarni. Energy harvesting sensor nodes: Survey and implications. *Communications Surveys Tutorials, IEEE*, 13(3):443–461, Third 2011.

[12] R. Dayal, S. Dwari, and L. Parsa. A new design for vibration-based electromagnetic energy harvesting systems using coil inductance of microgenerator. *Industry Applications, IEEE Transactions on*, 47(2):820–830, March 2011.

[13] X. Xing, J. Lou, G.M. Yang, O. Obi, C. Driscoll, and N.X. Sun. Wideband vibration energy harvester with high permeability magnetic material. *Applied Physics Letters*, 95(13):134103–134103–3, Sep 2009.

[14] X. Xing, G.M. Yang, M. Liu, J. Lou, O. Obi, and N.X. Sun. High power density vibration energy harvester with high permeability magnetic material. *Journal of Applied Physics*, 109(7):07E514, 2011.

[15] Qian Sun, S. Patil, S. Stoute, Nian-Xiang Sun, and B. Lehman. Optimum design of magnetic inductive energy harvester and its ac-dc converter. In *Energy Conversion Congress and Exposition (ECCE), 2012 IEEE*, pages 394–400, Sept 2012.

[16] Qian Sun, S. Patil, Nian-Xiang Sun, and B. Lehman. Modeling and optimization of an inductive magnetic harvester considering nonlinear effects. In *Control and Modeling for Power Electronics (COMPEL), 2013 IEEE 14th Workshop on*, pages 1–6, June 2013.

[17] Qian Sun, S. Patil, Nian-Xiang Sun, and B. Lehman. Inductive magnetic harvester with resonant capacitive rectifier based on synchronized switch harvesting technique. In *Energy Conversion Congress and Exposition (ECCE), 2013 IEEE*, pages 4940–4947, Sept 2013.

[18] Paul Mali. *Magnetic amplifiers - principles and applications.* John F. Rider Publisher, Inc., New York, 1960.

[19] Soo-Hong Kim, Jae-Bum Park, Seung-Deog Choi, Yoon-Ho Kim, and M. Ehsani. Optimal control method of magnetic switch used in high-voltage power supply. *Power Electronics, IEEE Transactions on*, 28(3):1065–1071, March 2013.

[20] Soo-Hong Kim and M. Ehsani. Control and analysis of magnetic switch reset current in pulsed power systems. *Power Electronics, IEEE Transactions on*, 29(2):529–533, Feb 2014.

[21] R. Watson and F.C. Lee. Analysis, design, and experimental results of a 1-kw zvs-fb-pwm converter employing magamp secondary-side control. *Industrial Electronics, IEEE Transactions on*, 45(5):806–814, Oct 1998.

[22] E.R.C. da Silva, S.G. Abeyratne, and Y. Murai. Pwm series resonant dc-link converter with current clamping by the use of saturable core. *Power Electronics, IEEE Transactions on*, 14(1):82–89, Jan 1999.

[23] H. Gruening, K. Koyanagi, and M. Mukunoki. Low reverse-recovery stress in high-power converters achieved by self-resetting saturable cores. *Industry Applications, IEEE Transactions on*, 45(1):232–238, Jan 2009.

[24] S. Aldhaher, P.C.-K. Luk, and J.F. Whidborne. Tuning class e inverters applied in inductive links using saturable reactors. *Power Electronics, IEEE Transactions on*, 29(6):2969–2978, June 2014.

[25] Giselher Herzer. Amorphous and nanocrystalline soft magnets. In *Magnetic Hysteresis in Novel Magnetic Materials*, pages 711–730. Springer, 1997.

[26] Vacuumschmelze. VITROPERM 500F W380, `http://www.vacuumschmelze.com/en/products/cores_inductive_components/applications/cores/vitroperm_standard_types_encapsulated.html`.

[27] Jinyeong Moon, John Donnal, Jim Paris, and Steven B. Leeb. VAMPIRE: A magnetically self-powered sensor node capable of wireless transmission. In *Applied Power Electronics Conference and Exposition (APEC), 2013 Twenty-Eighth Annual IEEE*, pages 3151–3159, March 2013.

[28] P.L. Dowell. Effects of eddy currents in transformer windings. *Electrical Engineers, Proceedings of the Institution of*, 113(8):1387–1394, August 1966.

[29] W.G. Hurley, E. Gath, and J.G. Breslin. Optimizing the ac resistance of multilayer transformer windings with arbitrary current waveforms. *Power Electronics, IEEE Transactions on*, 15(2):369–376, Mar 2000.

[30] Eric Charles Snelling. *Soft ferrites: properties and applications.* Butterworths, London, second edition, 1988.

[31] Jieli Li, T. Abdallah, and C.R. Sullivan. Improved calculation of core loss with nonsinusoidal waveforms. In *Industry Applications Conference, 2001. Thirty-Sixth IAS Annual Meeting. Conference Record of the 2001 IEEE*, volume 4, pages 2203–2210 vol.4, Sept 2001.

[32] K. Venkatachalam, C.R. Sullivan, T. Abdallah, and H. Tacca. Accurate prediction of ferrite core loss with nonsinusoidal waveforms using only steinmetz parameters. In *Computers in Power Electronics, 2002. Proceedings. 2002 IEEE Workshop on*, pages 36–41, June 2002.

[33] Yehui Han, G. Cheung, An Li, C.R. Sullivan, and D.J. Perreault. Evaluation of magnetic materials for very high frequency power applications. *Power Electronics, IEEE Transactions on*, 27(1):425–435, Jan 2012.

[34] J. Muhlethaler, J. Biela, J.W. Kolar, and A. Ecklebe. Core losses under the dc bias condition based on steinmetz parameters. *Power Electronics, IEEE Transactions on*, 27(2):953–963, Feb 2012.

[35] J. Muhlethaler, J. Biela, J.W. Kolar, and A. Ecklebe. Improved core-loss calculation for magnetic components employed in power electronic systems. *Power Electronics, IEEE Transactions on*, 27(2):964–973, Feb 2012.

[36] Stanley C Eisenstat, Howard C Elman, and Martin H Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM Journal on Numerical Analysis*, 20(2):345–357, 1983.

[37] Yousef Saad. *Iterative Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, second edition, 2003.

[38] NXP Semiconductors. PMEG2010ER 1A low V_F MEGA Schottky barrier rectifier.

[39] NXP Semiconductors. PMV30UN μTrenchMOS^{TM} ultra low level fet.

[40] Diodes Inc. DMG3415Y P-channel enhancement mode MOSFET.

[41] A. Facen and A. Boni. Power supply generation in cmos passive uhf rfid tags. In *Research in Microelectronics and Electronics 2006, Ph. D.*, pages 33–36, 2006.

[42] S. Mandal and R. Sarpeshkar. Low-power cmos rectifier design for rfid applications. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 54(6):1177–1188, June 2007.

[43] Microchip Technology Inc. PIC18F/LF1XK50 20-pin USB flash microcontrollers with nanoWatt XLP technology.

[44] Taiyo Yuden. Supercapacitor PAS0815LS2R5105 - 1 F 2.5 V 70 mΩ, `http://www.t-yuden.com/ut/product/category/energy_device/`.

[45] Bosch Sensortec. BMA250 Digital Triaxial Acceleration Sensor, `https://www.bosch-sensortec.com/bst/products/motion/accelerometers/overview_accelerometers`.

[46] Cypress Semiconductor. CY15B104Q: 4-Mbit (512 K ÃŮ 8) Serial (SPI) F-RAM, `http://www.cypress.com/products/4-mb-f-ram`.

[47] J. S. Donnal and S. B. Leeb. Noncontact power meter. *IEEE Sensors Journal*, 15(2):1161–1169, Feb 2015.

[48] Laird Tech. BL600SA, `http://www.lairdtech.com/products/bl600-series`.

[49] Toshiba Corporation. FlashAir W-03, `http://www.toshiba-memory.com/cms/en/products/wireless-sd-cards/flashair/`.

[50] J. Paris, J. S. Donnal, and S. B. Leeb. Nilmdb: The non-intrusive load monitor database. *IEEE Transactions on Smart Grid*, 5(5):2459–2467, Sept 2014.

[51] Zhangjun Tang, P. Pillay, and A. M. Omekanda. Vibration prediction in switched reluctance motors with transfer function identification from shaker and force hammer tests. *IEEE Transactions on Industry Applications*, 39(4):978–985, July 2003.

[52] Christopher J. Schantz. Methods for non-intrusive sensing and system monitoring. *Ph.D. dissertation, Massachusetts Institute of Technology*, 2014.