# Design and Testing of Wearable and Long-Term Subdermal Implantable Electroencephalograms

by

Jason Yang

S.B. (EECS), Massachusetts Institute of Technology (2015)

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

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017



Author .....
Department of Electrical Engineering and Computer Science
May 26, 2017

Certified by .....
Charles G. Sodini
LeBel Professor of Electrical Engineering
Department of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Christopher J. Terman
Chairman, Department Committee on Graduate Theses

# Design and Testing of Wearable and Long-Term Subdermal Implantable Electroencephalograms

by

Jason Yang

Submitted to the Department of Electrical Engineering and Computer Science
on May 26, 2017, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

## Abstract

Electroencephalography (EEG) has long been used by neurologists to aid in diagnosing and treating neurological disorders ranging from sleep apnea to epilepsy. However, inherent difficulties still exist in capturing EEG data for extended periods of time on the order of days to weeks in humans. Such difficulties brought on by implementation challenges and usability ultimately lead to patient non-compliance. These challenges also curtail EEG use in sleep studies due to complexity of setup.

This work aims to address these issues by extending the functionality and performance of a previously designed EEG ASIC. A design for a subdermal, implantable EEG recording system for long term EEG monitoring as well as a simplified wearable and wireless EEG sensor is realized.

The implantable design is an 8-channel, 250Hz bandwidth EEG system in a 14.0mm x 15.5mm package that is wirelessly powered by an external device through inductively coupled coils with backscattering. The device is placed subdermally above the skull for continuous patient EEG monitoring for up to a month to aid neurologists with epilepsy diagnosis. This device is especially important given the severity of antiepileptic drugs' side-effects. The performance of this device is verified through animal studies in pigs.

This work is also extended to the design of a wearable, wireless EEG patch for sleep studies. The device is a 20mm x 24mm, 4-channel, 250Hz bandwidth, Bluetooth Low-Energy (BTLE) electronics package with adhesive electrodes that can be quickly applied. This device is aimed to assist neurologists and clinicians perform sleep studies. Results are verified using sleep data collected on the author.

Thesis Supervisor: Charles G. Sodini
Title: LeBel Professor of Electrical Engineering
Department of Electrical Engineering and Computer Science

# Acknowledgments

*"If I have seen further, it is by standing on the shoulders of giants"*

*- Issac Newton*

I cannot say that I have seen further than my peers, but I can say I have stood on the shoulders of many giants during my time at MIT to which I have too many thanks to give. First and foremost, I would like to thank my advisor, Charles Sodini. Although a seemingly simple concept, I now have a better idea what it means to make opportunity rather than chase opportunity. I would not be the person I am today if not for his guidance in the research world and advice in life. Thank you for all the advice and opportunities that you have given me for my time here, not only as a Master's student but also through my undergraduate career.

This work could not have been possible if not for the work and guidance of Bruno DoValle. This thesis builds upon his work while at MIT and could not have happened without his continued support during and after his Ph.D. Furthermore Bruno was a fantastic mentor during my undergraduate years.

Dr. Sydney Cash and his team at MGH including, but not limited to Lynde Folsom and Michael Duggan, were instrumental in completing this thesis. They provided hours of support from on the neural and physiological standpoints and helped carry out the animal experiments.

Some further giants need to be acknowledged - Nevan Hanumara and Daniel Teo spent many late-night hours assisting in the design of the mechanics of the device enclosures. They were both great references in coming up with a design that was easy to deploy and 'pig-proof'. Furthermore, Nevan provided great contacts and support in acquiring proper materials to complete this work.

I would also like to acknowledge VSIParylene and Analog Devices for providing their expertise and resources in manufacturing these devices.

Finally I need to acknowledge all the friends I have made through my tenure at MIT. Thank you for all the memories and understanding that sometimes life and the people around you is the most important thing a someone needs. I would like

to thank my friends who are always there, through the upsets and triumphs, always cheering me on. To those giants, thank you for letting me stand on your shoulders - thank you for making my family a bit bigger.

*This work is dedicated to you.*

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Electroencephalograms (EEGs) are devices that detect and record the bioelectric signals in the brain [1]. EEGs are used to diagnose and treat a wide range of neurological conditions by providing insight into a patient's brain activity. Their applications range from assisting neurologists in sleep studies to diagnosing epilepsy and other brain disorders [2].

Achieving long-term continuous EEG data in a minimally invasive and usable form-factor has been a long-standing problem [3]. In conventional EEG studies, the patient must routinely go to the hospital and be connected to bulky equipment. This process is prohibitively expensive in the long run and prevents the patient from going about their daily lives thereby reducing patient compliance in some situations.

Here, improvements to previous designs for minimally invasive and wearable EEGs are proposed to allow for more reliable operation over a longer duration of time.

## 1.1 Description of EEG

### 1.1.1 Physiology of EEG

EEG records the aggregate electrical activity of the brain produced by neurons firing. In a typical neuron, there is a 65mV potential difference between the inside of the neuron and surrounding fluid due to the distribution of ions in the neuron. When

a neuron becomes active, the membrane depolarizes and the potential difference between the inside and outside of the cell reverses [2]. This depolarization causes an impulse to travel down the neuron and trigger subsequent neurons to fire. However, the signal from a single neuron firing is too small to be detected by EEG [4]. Thus, EEG records the aggregate activity of large groups of neurons near the EEG electrode site.

### EEG Waveforms

EEG signals at the scalp typically are on the order of of $10\mu$V to $100\mu$V in amplitude with a frequency range of about 0.1Hz to 70Hz. However, during a seizure event, amplitudes can reach about 1mV with oscillations at about 250Hz. Historically, neurologists have divided typical EEG signals into 4 bands [5]:

- **Delta** (0.5Hz - 4Hz) - Typically the largest in amplitude and occur most prominently frontally in adults.
- **Theta** (4Hz - 8Hz) - These waves are seen in drowsiness.
- **Alpha** (8Hz - 14Hz) - Found in the posterior regions of the brain and occur most prominently during a relaxed state.
- **Beta** (14Hz - 30Hz)- Dominant pattern in alert subjects and are located in the frontal regions of the brain.

Overall the frequency of the EEG waveform increases with age and decreases during sleep.

### EEG and Sleep

During sleep, EEG waveforms change significantly depending on the stage of sleep. In a normal sleep pattern, brain activity typically cycles between rapid-eye movement (REM) and non-REM sleep. A typical sleep cycle starts with Beta waves from waking consciousness and drops in frequency to Delta as sleep progresses. Over the course of sleep, EEG frequency oscillates between Delta and Beta waves of REM sleep,

Table 1.1: Sleep Stages and Associated Waveforms [6].

| Sleep Stage | EEG Waveform | Description |
|---|---|---|
| Awake (Eyes Open) | Beta | Active mental concentration |
| Awake (Eyes Closed) | Alpha | Relaxation |
| Non-REM Stage-1 | Theta | Light sleep |
| Non-REM Stage-2 | Sleep Spindles / K-Complexes | Deeper sleep |
| Non-REM Stage-3 | Delta | Slow wave deep sleep |
| REM | Beta | Dreaming |

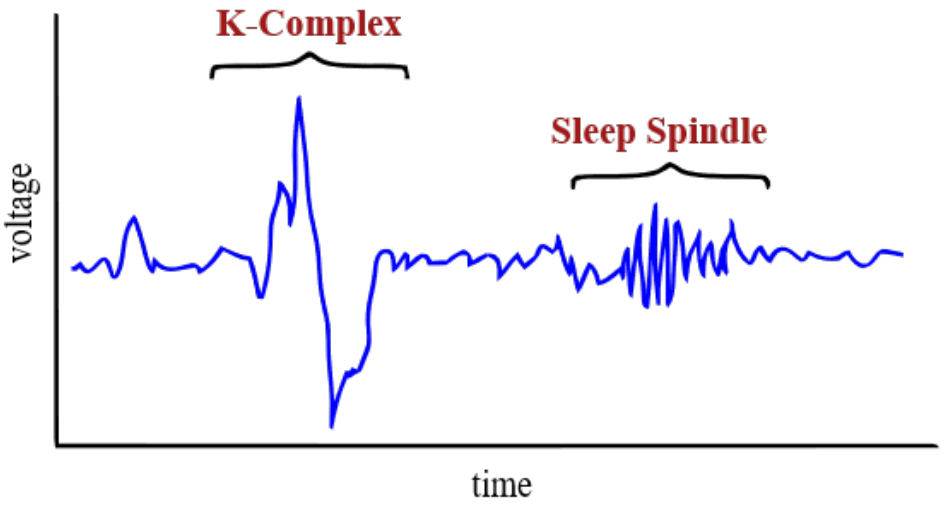


Figure 1-1: Sleep Spindle and K-Complex Waveforms.

transitioning between all EEG frequency bands. This cycle typically occurs over a 90 minute period [6]. Table 1.1 describes the stages of sleep and the associated EEG waveforms.

During non-REM stage-2 sleep, artifacts such as sleep spindles and K-complexes appear on EEG. A sleep spindle is a short 12Hz - 14Hz burst waveform that occurs for at least 0.5 seconds [7]. K-complexes are large amplitude events that occur due to external stimuli. These complexes are typically have a peak greater than $100\mu V$ in amplitude when measured at the scalp [8]. Figure 1-1 shows a sleep spindle and a K-complex.

Table 1.2: American Clinical Neurophysiology Society Guidelines for Long-Term Monitoring

| Parameter | Recommended Value |
|---|---|
| Low-Frequency Cutoff | $< 0.5Hz$ |
| High-Frequency Cutoff | $> 70Hz$ |
| Noise Level | $< 1\mu V_{RMS}$ |
| Input Impedance | $> 1M\Omega$ |
| Common Mode Rejection | $> 60dB$ |
| Dynamic Range | $> 40dB$ |

### 1.1.2 EEG Methods

#### Amplifier and Electrodes

For external bioelectric sensors, such as those used in conventional EEG studies, amplifier design is of paramount importance. For an arbitrary amplifier with a finite gain $K$, finite input impedance $Z_{in}$, and finite common-mode rejection ratio (CMRR), but otherwise ideal in every other parameter, voltage at the output is given by:

$$V_{out} = KV_{biol} + KV_{cm}\left(\frac{1}{CMRR} + \left(1 - \frac{Z_{in}}{Z_{in} + Z_{eletrode} - Z_{ref}}\right)\right) \tag{1.1}$$

where $V_{out}$ is the output voltage, $V_{biol}$ is the bioelectric signal of interest, and $V_{cm}$ is the common-mode interference signal [9][10]. This equation suggests that two factors are of great importance to ensure a maximal signal to noise ratio. Foremost, the input to the amplifier in the analog front end (AFE) must be of sufficiently high input impedance. State of the art amplifiers have an impedance in the hundreds of MΩ to few GΩ range. High impedance is required in order to not draw current from the bioelectric source and load it, as well as ensure sufficient common-mode noise and interference rejection between measurement and reference electrodes. Second, a low *and* balanced contact impedance between the bioelectric source and AFE is required. This is required, again, for proper common-mode noise rejection [1]. Table 1.2 shows the recommended values for long-term EEG recording [11].

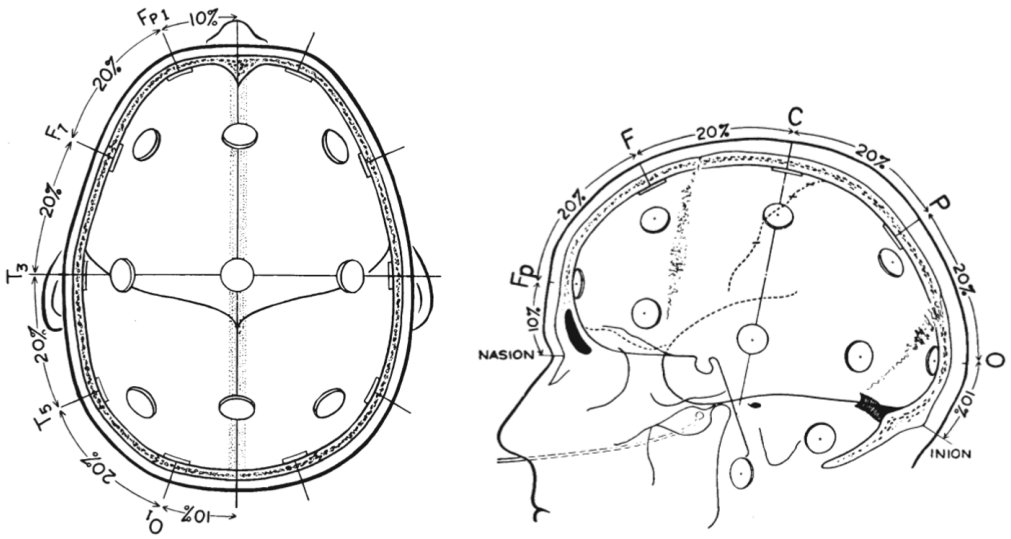


Figure 1-2: 10-20 Electrode System Placement [12]. Electrodes are placed in increments of 10% and 20% of the front-back or left-right distance of the skull.

### Electrode Placement

Because EEG records brain activity over a small region of the brain, electrode placement is of paramount importance in order to ensure the right EEG signals are detected. Conventional scalp based EEG uses a standardized International 10-20 placement system in order to describe and apply electrodes for an EEG experiment. The convention dictates that electrodes are placed in increments of 10% or 20% of the total front-back or left-right arc distance of the skull. [1][12]. Figure 1-2 shows the top and side views of the conventional 10-20 electrode placement. Figure 1-3 shows the naming convention for the 10-20 system.

### Montages

Because EEGs and other bioelectric sensors measure a potential difference between two points, display of EEG can be set up in several ways depending on what two points the EEG measurement is taken across. Such different measurement systems are referred to as EEG montages. Several standard montages exist [13]:

- **Bipolar Montage** - EEG measurements are taken between two arbitrary electrodes. No common reference for all the electrodes exists.

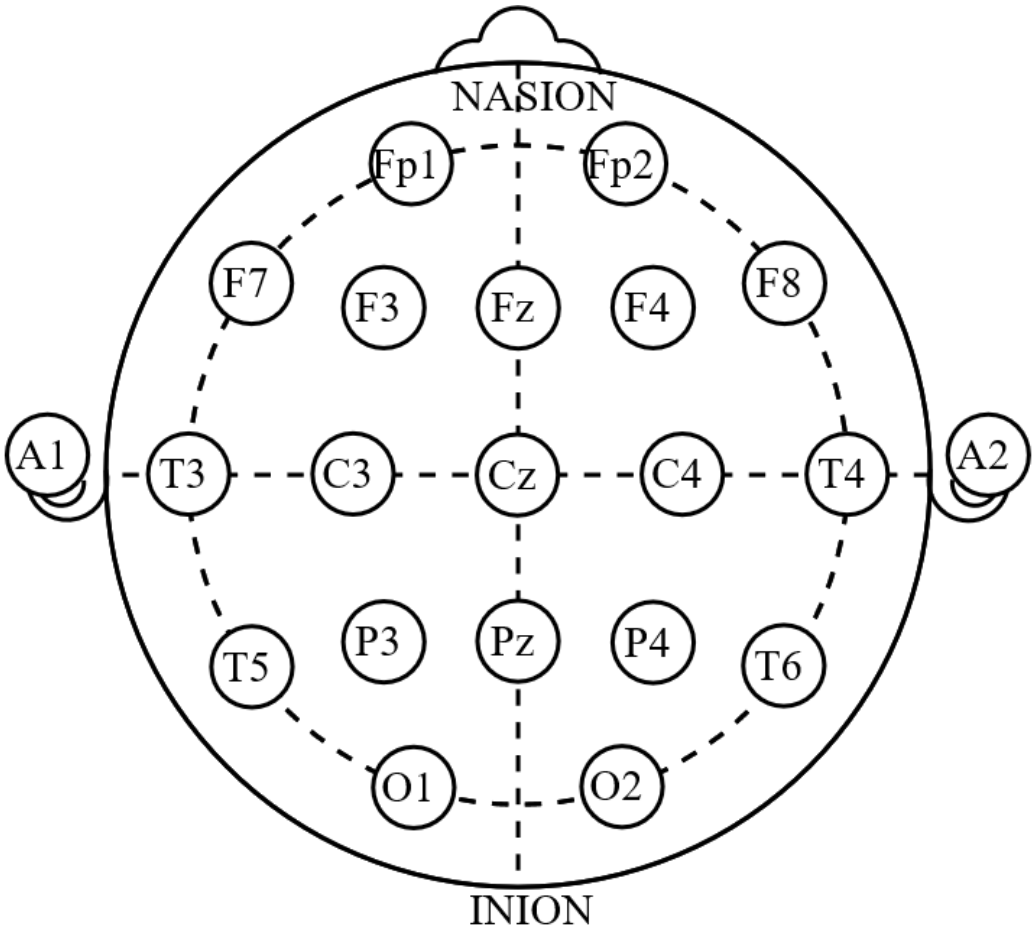


Figure 1-3: 10-20 Electrode System Naming.

- **Referential Montage** - All EEG electrodes are taken in reference to a common reference electrode. Several such standard references exist. One is placing electrodes on both earlobes as the ears present little electrical activity. Another common reference is along the mid-line of the head as they do not amplify the signal in one hemisphere over the other.

- **Average Montage** - All the electrodes are equally wighted and averaged together. This average serves as the virtual reference for each electrode.

- **Laplacian Montage** - A slight variation from the above average montage where each electrode is referenced from a weighted average of the surrounding electrodes.

Generally each montage can be mathematically derived from every other montage.

## 1.2 Motivations for Long-Term Subdermal Implantable EEG

Being able to accurately monitor neural signals continuously for long periods of time not only gives researchers information on how the brain changes through different natural factors such as seasons or sleep cycles, but also enables doctors to accurately diagnose and treat neurological disorders. In the case of epilepsy, a patient suffers from recurring seizures that are typically the result of excessive abnormal neural activity in the brain [14]. In 70% of cases, epilepsy is controllable with medication; however, in order to determine the type of medication to treat an epileptic patient, neurologists must determine the location of the seizure through recordings using an EEG signal [15]. Unfortunately epileptic events can be quite varied in occurrence and the patient may require long hospital stays or be forced to wear bulky equipment for extended durations in order to successfully capture a seizure recording on EEG.

One of the largest hurdles in achieving long-term EEG recordings is overcoming patient non-compliance caused by disruption of everyday tasks. Current conventional

EEG systems require 21 electrodes to be placed in specific locations on the scalp in locations dictated by the international 10-20 system [13][16]. In this setup, patients may be asked to wear these precisely placed electrodes for up to 72 hours at a time. Even with ambulatory EEG systems, the user is tethered to a portable device and must make special and disruptive accommodations such as not showering since water interferes with the electrodes [17]. Furthermore, typical wet gel electrodes used for EEG recordings have a tendency to dry and be absorbed into the skin thereby increasing the skin impedance and consequently increasing the noise in the recording. Due to these disruptions in daily life, and problems in long-term signal integrity, extended duration EEG recordings on timescales of about a month remain a significant challenge.

To address inherent issues with long-term collection of EEG data, a design for a system that is minimally invasive not only in terms of surgery but also in terms of impact in daily life is required. One possibility to solve this is to have the electrodes and electronics placed underneath the scalp, above the skull, and powered by an external device. Such a scheme would not require constant maintenance of the electrodes and would not require having to be tethered to bulky electronics for extended periods of time. This approach is differs from electrocorticography (ECOG) where electrodes are placed underneath the skull and directly on the patient's brain.

## 1.3 Motivations for Wearable EEG

In addition to treating epilepsy, EEGs are also commonly used for sleep studies and sleep disorder diagnosis. However, much like epilepsy diagnosis, accurate sleep EEG data for sleep studies is difficult to acquire. During a polysomnogram, a type of sleep study, a user is attached to a conventional EEG system and asked to sleep. Such experiments are difficult to carry out and have a high degree of impact on the user's daily life, as the user must sleep in a foreign location under non-ideal conditions all while interrupted by numerous wired electrodes. Several ideas have been presented to circumvent these issues ranging from motion trackers to breathing monitors, but

these devices provide little insight into the brain activity during sleep. By designing a easy-to-use wireless EEG patch that can be deployed anywhere, even outside of a hospital, many issues caused by the disruptive nature of conventional EEG systems can be avoided.

## 1.4 Previous Work

### 1.4.1 ASIC

This work revolves around a previously deigned ASIC targeted towards EEG acquisition with seizure detection and monitoring. The ASIC includes eight EEG channels designed and fabricated in a TSMC 0.18 $\mu$m CMOS process and packaged in a 20mm x 20mm 144-pin LQFP package. The ASIC has a miniaturized, power-efficient analog front end that consumes only 2.75$\mu$W per channel and has an input-reffered noise of 1.1$\mu V_{RMS}$. The EEG front end of this ASIC has high and low cutoff frequencies of 500Hz and 0.07Hz respectively when driven by a 491kHz clock and has an input impedance of 1.6$G\Omega$ [18].

### 1.4.2 Implantable System

The implantable EEG system presented in this work leverages existing work done both in [18] and in unpublished work. The implantable implementation is described in [18] and will be referred to as 'the previous design' throughout the rest of this work.

## 1.5 Aims of Thesis Work

The research presented in this document aims to complete the following:

- Optimize an implantable EEG design such that it is electrically more robust.
- Revise the external device used with the implant device such that it is more animal resilient for use in swine model testing.

- Perform animal testing using a swine model in a hospital.
- Design an easy to use, wearable and wireless EEG patch for sleep studies.
- Perform testing with the aforementioned EEG patch.

## 1.6 Thesis Organization

This thesis is organized as follows:

- Chapter 2 describes an optimized subdermal implantable EEG system for use in swine testing. This chapter details designs for both the implant and external devices, presents fabrication results, animal testing, and concludes analysis of the experimental data.
- Chapter 3 presents a design for a wearable EEG patch for use in clinical settings. Fabrication, experimental results, and analysis are also presented.
- Chapter 4 gives a design of a wireless receiver and server for remote communication of both the subdermal and wearable EEG designs. This chapter covers both the hardware and software implementations of the design.
- Chapter 5 draws the thesis conclusions and presents possible future research directions.

# Chapter 2

# Subdermal Implantable EEG System

This chapter presents a design for an optimized implantable EEG recorder for use in in-vivo testing with pigs. The design has two main parts: the implant design and external device design, covering both the electrical and mechanical design of each component. Design success is gauged on two metrics:

- Viability for long-term data recording on timescales of approximately 1 month.
- Quality of EEG data through analysis procedures detailed in Section 2.6.

Fabrication data and experimental results are also presented.

## 2.1 System Overview

Figure 2-1 shows the high-level system diagram of the subdermal system for testing in a swine model, which consists of two parts - an external device that is placed on top of the skin, and an implanted device that will be surgically placed subcutaneously. Here the external device provides configuration data and power to the implant while also receiving EEG data to be relayed to a nearby device using BTLE. This design is described in more detail in 2.4. The implanted device design revolves around the ASIC described in 1.4.1 which provides the EEG analog front-end and digitization. Support electronics such as impedance modulator and power management (PM) allow for power and data to be delivered to the ASIC and is detailed in 2.3.

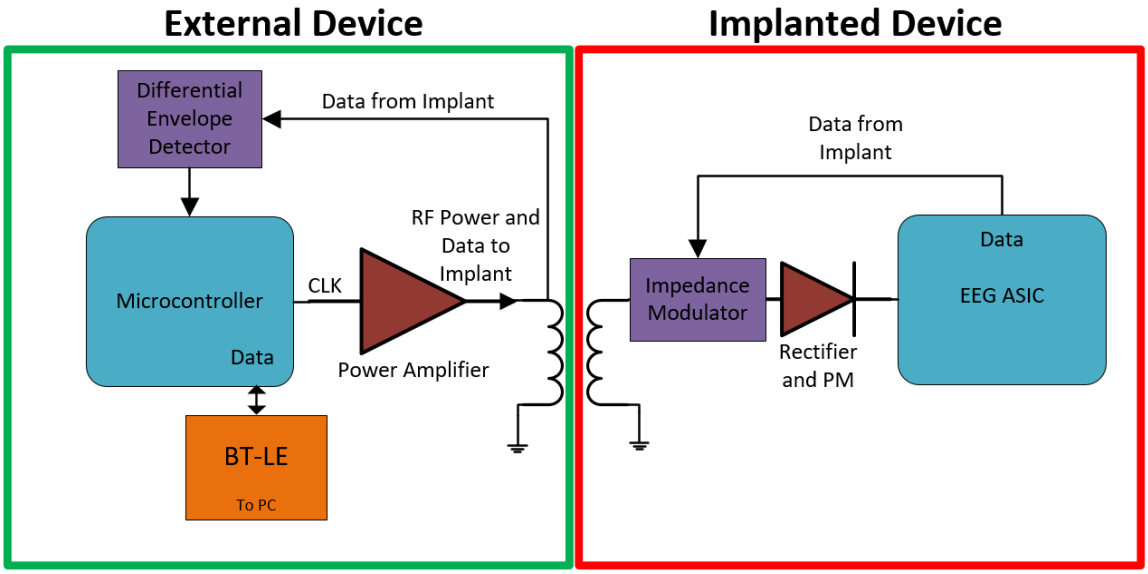


Figure 2-1: Subdermal System Diagram. External system block diagram is shown on the left. Implant System block diagram is shown on the right.

## 2.2 Changes From Previous Design

This design revision addresses several issues from the original design detailed in DoValle, where previous in-vivo pig tests failed after a few hours due to mechanical instability of the system [18]. This is largely attributed to the bulkiness of the devices used in the test setup. The revisions on the original design detailed in this document are aimed to reduce the physical size and increase the electrical efficiency of the entire system in hopes to streamline the device for longer term in vivo tests with pigs. The primary changes detailed here include a more efficient power amplifier design, power management, optimization of impedance detection, use of Bluetooth Low Energy (BTLE), and an optimized external mechanical housing for animal survival and operator serviceability.

## 2.3 Implant Design

The main goal of the implant device design revision is further miniaturization and electronic robustness over the previous device in [18]. By reevaluating use cases of this device, many redundant systems from the previous design could be removed,

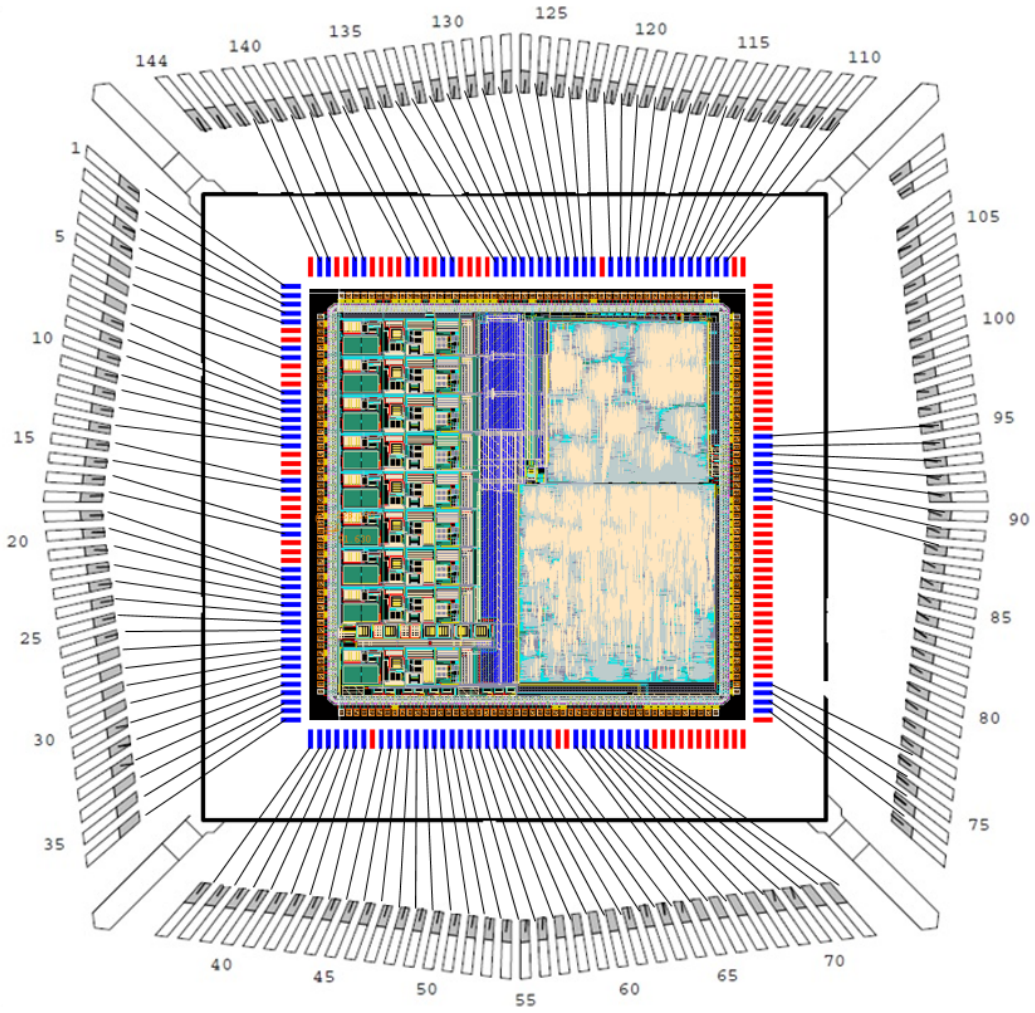


Figure 2-2: Previous Bonding Diagram for 144 pin LFCSP Package.

resulting in a smaller device. Furthermore, testing of the previous device revealed further optimizations for long-term electrical viability during in-vivo testing using pig models. The design revision has the following goals:

- Reduction of device size for ease of implantation.

- Electrical optimizations to mitigate fault conditions.

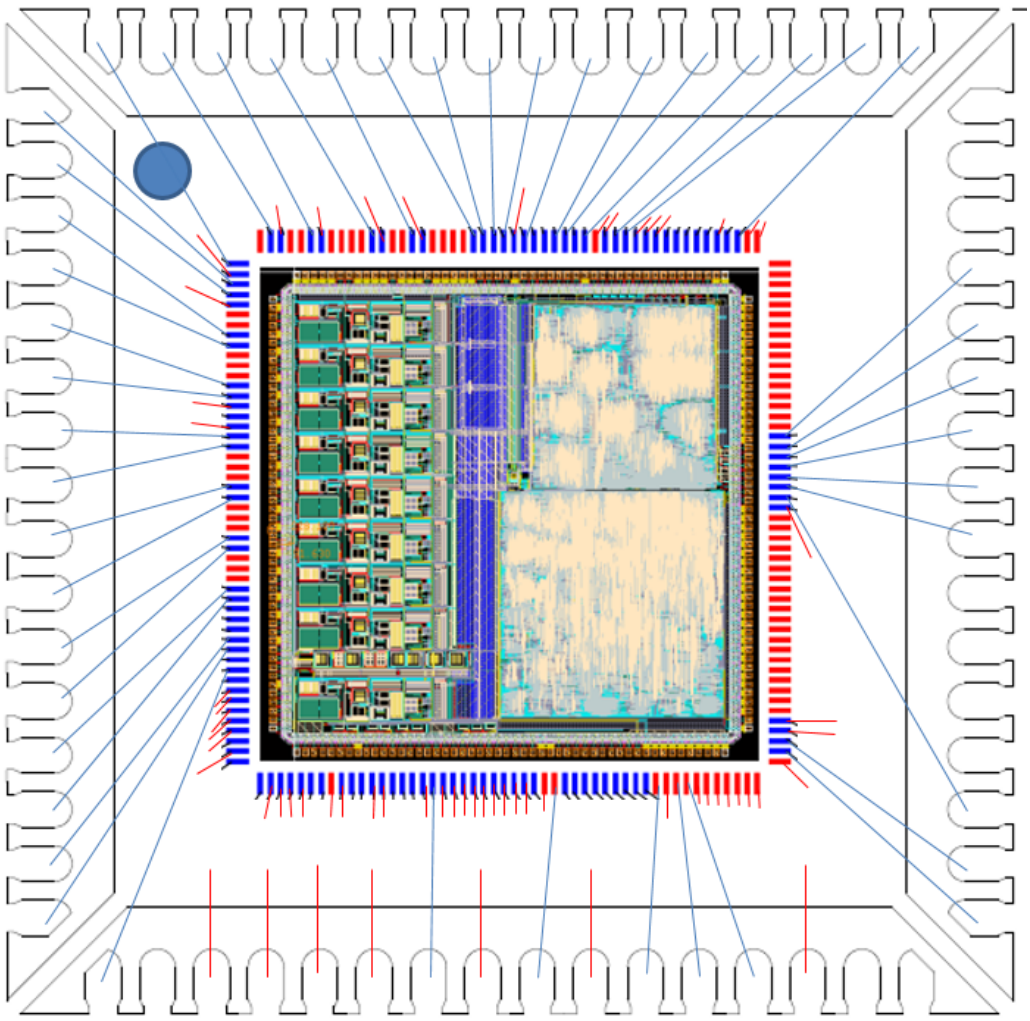

Figure 2-3: Modified Bonding Diagram for reduced 64 pin QFN Package.

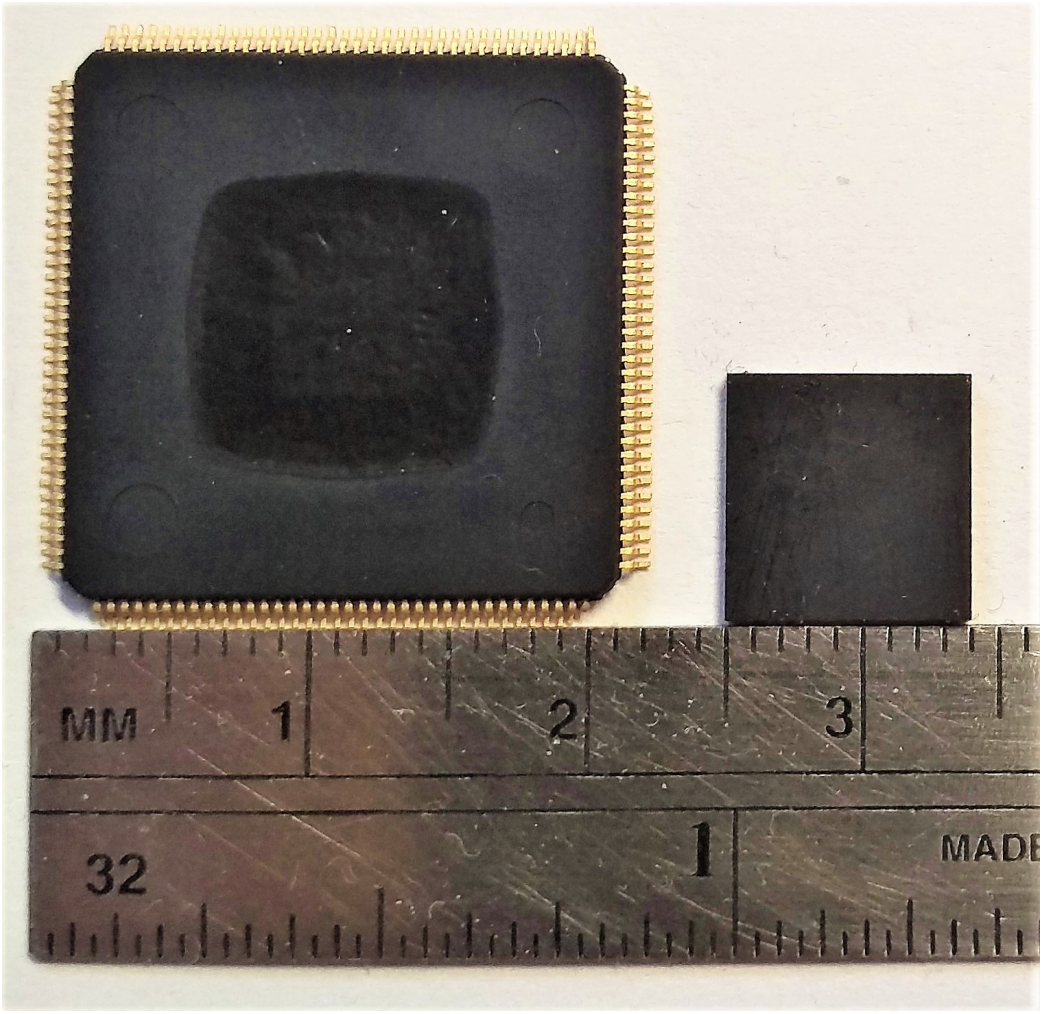


Figure 2-4: Packaged ASICs. 144 pin LFCSP (left), 64 pin QFN (right).

### 2.3.1 Electrical

#### ASIC

The original ASIC detailed in [18] is a 5mm x 5mm die fabricated using the TSMC 0.18 $\mu$m process and packaged in a 20mm x 20mm 144 pin package. This package size was selected due to an uneven pad distribution on the die, resulting in one or more sides requiring more bondwires and an overall larger size as shown in 2-2. The ASIC contains two modes of operation. In Mode 1, EEG is constantly being sampled at the maximum clock rate and transmitted continuously to a receiving device. In Mode 2, sampling is reduced by a factor of 5 and data is only recorded to an off-chip SPI memory for later transmission when a seizure is detected.

For our application, seizure detection was not needed; thus, Mode 2 could be eliminated completely. By removing this mode of operation, many associated pads on the ASIC can be left unconnected and bond diagram can be balanced as shown in 2-3. With the reduced pad requirement, the ASIC can be fitted into a 9mm x 9mm QFN, minimizing overall device footprint. Figure 2-4 shows the result of the revised packaging.

#### Wireless Power Link

The device is powered by magnetically coupled coils at 2.5MHz with resonance in the secondary coil. A pre-fabricated 12.6$\mu$H external receiver coil is placed in parallel with a 270pF capacitor as shown in Figure 2-5 as $L_1$ and $C_1$ respectively. This value for the coil is selected in order to achieve the largest peak-peak voltage gain from external to implant and maximal load gain when reflecting an impedance from the implant to external device for communication. For a coupled coil system with an arbitrary coupling coefficient $k$, voltage gain is given by:

$$\frac{V_o}{V_i} = \frac{k\sqrt{\frac{L_s}{L_p}}}{1 + (1 - k^2)\frac{j\omega L_s}{Z_s}} \tag{2.1}$$

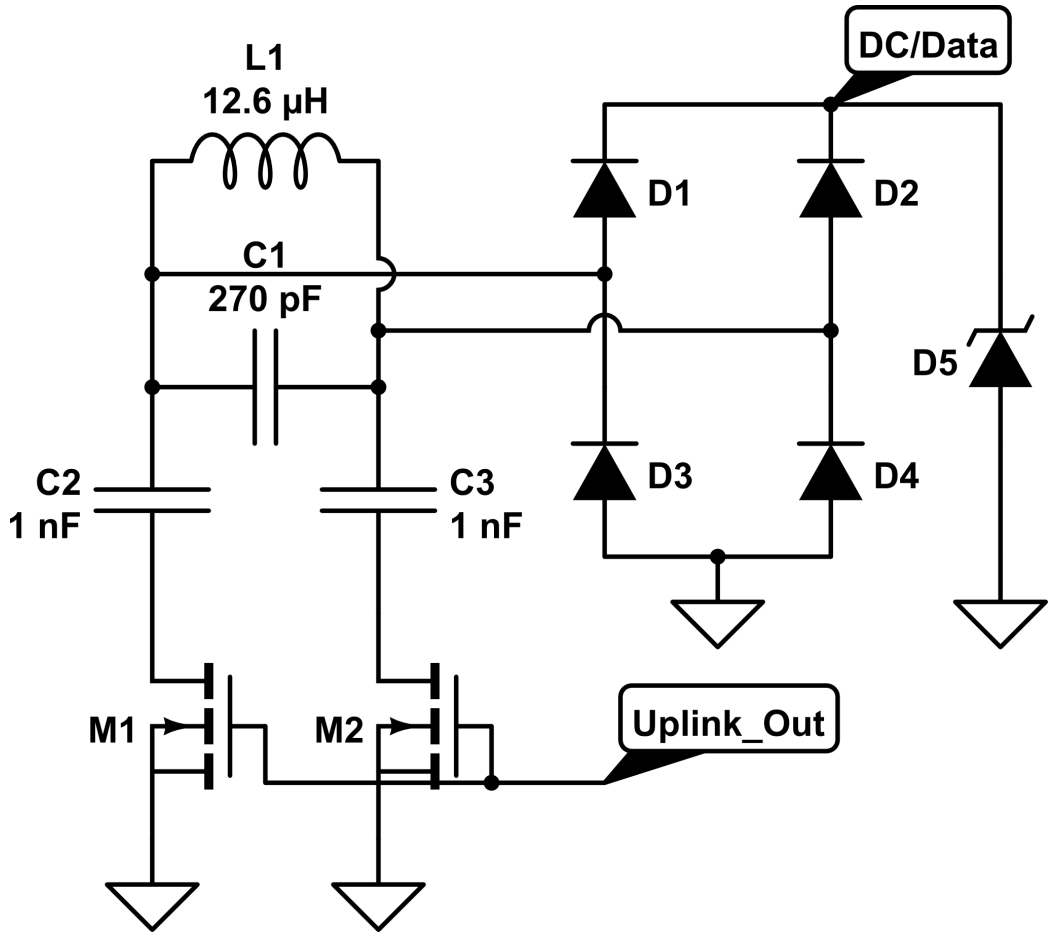


Figure 2-5: Schematic for Implant Resonant Coil. $L_1, C_1$ forms a resonant tank for power delivery. $C_2, C_3, M_1$, and $M_2$ modulate coil impedance for communication to the external device. $D_1, D_2, D_3$, and $D_4$ form a bridge rectifier. $D_5$ is a protective Zener diode clamp.

where $L_p$ denotes the primary side inductance and $L_s$ denotes the secondary inductance ($L_1$ in Figure 2-5) [19]. $Z_s$ is the impedance across $L_s$. Assuming a capacitive and resistive load across $L_s$, as is the case with a parallel resonant secondary coil driving a load, voltage gain becomes:

$$\frac{V_2}{V_1} = \frac{k\sqrt{\frac{L_s}{L_p}}}{1 + (1-k^2)(j\omega\frac{L_s}{R_l} - \omega^2 L_s C_s)} \tag{2.2}$$

where $R_l$ is a resistive load and $C_s$ is the resonant capacitor across $L_s$.

Resonance is given by:

$$\omega_0 = \frac{1}{\sqrt{L_{eff}C_{eff}}} \tag{2.3}$$

where $L_{eff}, C_{eff}$ are the effective inductance and capacitance respectively, which is the rated inductance and capacitance added with any non-ideal or parasitic effects. At resonance, equation 2.2 combines with equation 2.3 to yield:

$$\frac{V_2}{V_1} = \frac{k\sqrt{\frac{L_s}{L_p}}}{1 + (1-k^2)(j\omega\frac{L_s}{R_l} - 1)} \tag{2.4}$$

which for very light loads (large $R_l$), becomes:

$$\frac{V_2}{V_1} \approx \frac{1}{k}\sqrt{\frac{L_s}{L_p}} \tag{2.5}$$

Reflected impedance from the $R_l$ across the primary coil $L_p$ from the coupled resonant secondary with an arbitrary coupling constant at the resonant frequency is given by:

$$Z_r = k^2(\frac{L_p R_l}{L_s}) + (1-k^2)(j\omega L_p) \tag{2.6}$$

where $Z_r$ is the reflected impedance [20].

By selecting the largest value for $L_2$, maximal voltage gain and largest load to be reflected to the primary can be achieved; however, equation 2.3 suggests that larger inductor values would require smaller capacitor values for a particular resonant frequency. Having systems that depend on small value capacitors is a concern, as

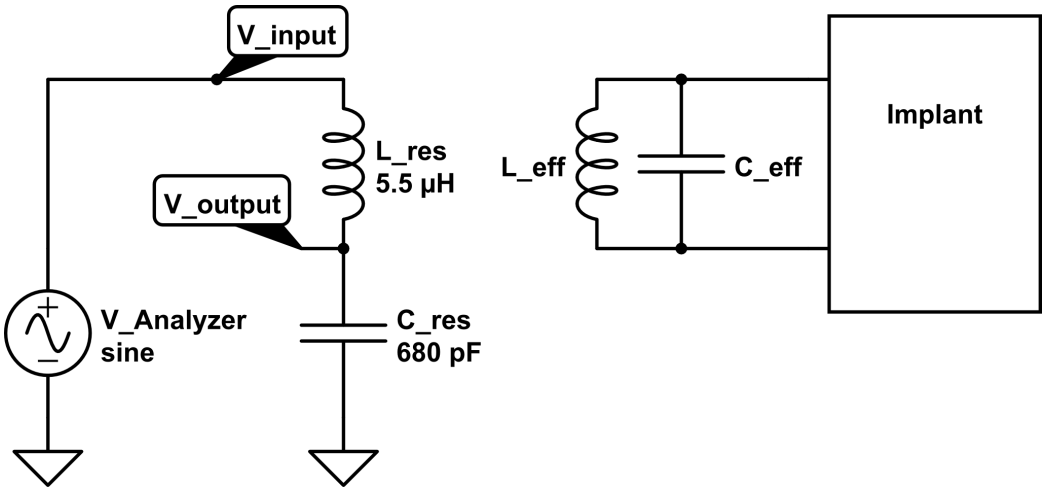


Figure 2-6: Schematic for Implant Resonance Test. $L_{eff} = L_1(= 12.6\mu H) + L_{parasitics}$, $C_{eff} = C_1(= 270pF) + C_{parasitics}$.

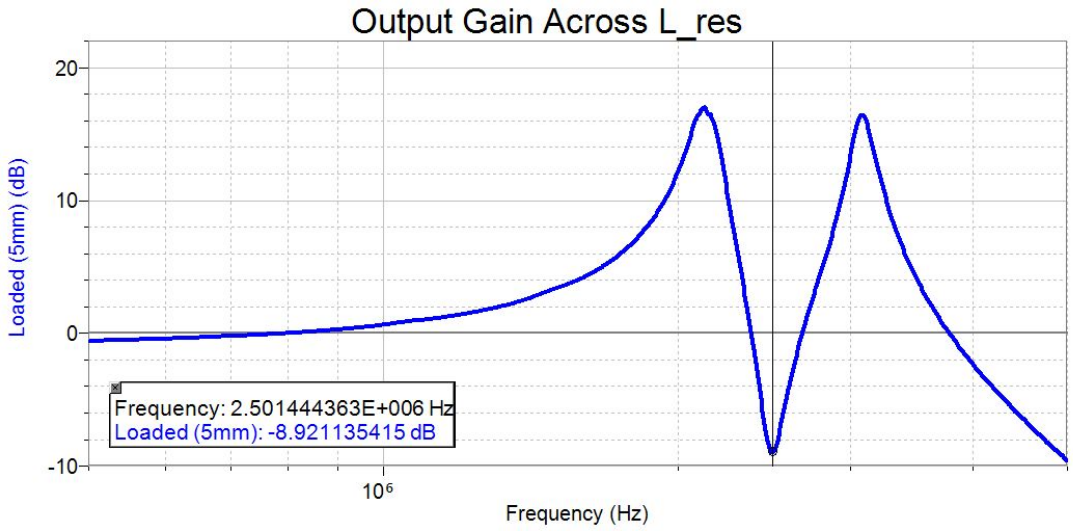


Figure 2-7: Implant Coil transfer Characteristic. Trough in curve represents the resonant frequency of the secondary coil.

parasitic capacitances, such as from inductor windings or diode reverse-bias junctions, start to cause a non-trivial and sometimes unpredictable variation in $C_{eff}$ from the desired value. It should be noted that the 270pF $C_1$ capacitor in Figure 2-5 is less than what is expected to resonate at 2.5MHz to account for such parasitics. Due to manufacturing variations in the coil, the resonance of $L_1$ with $C_1$ was hand-trimmed in the experimental setup by adding or removing small amounts of capacitance and measuring the frequency response with a network analyzer. The test setup schematic is shown in Figure 2-6. $L_{eff}$ and $C_{eff}$ is the resonant structure in the implant to be trimmed. It should be noted that $L_{eff}$ and $C_{eff}$ is the desired resonant inductance and capacitance respectively with non-idea parasitics added. $L_{res}$ and $C_{res}$ is an external resonant tank set to resonate at the 2.5MHz operating frequency. The output is measured at $V\_output$ relative to $V\_input$. A sample gain plot used for trimming is shown in Figure 2-7. Trimming is achieved when the trough of the voltage transfer curve is at 2.5MHz.

AC current is then rectified by a bridge rectifier shown in Figure 2-5 as $DC/Data$. Full-wave bridge rectification was chosen in order to keep the incoming AC signal symmetric to minimize distortion for the impedance-modulated link. Equation 2.4 suggests that the output voltage has a strong dependence on coupling coefficient $k$, which, is heavily dependent on relative coil placement and orientation. For our particular coupled coil system, $k$ ranges from approximately 0.08 to 0.35 over a range of 5mm to 15mm when coils are placed coaxially. This dependence causes large fluctuations in rectified voltage with positioning. In order to prevent large voltages from damaging downstream electronics, a 5.6V Zener $D_5$ is placed to clamp the output voltage.

### Uplink and Downlink Scheme

In order to both program the implanted ASIC and receive data from the implant to the external device, the data is encoded in an On-Off Keying (OOK) scheme. One bit of data is transmitted between rising edges of a signal where a duty cycle greater than 50% corresponds to a logic 1; likewise, a duty cycle less than 50% corresponds

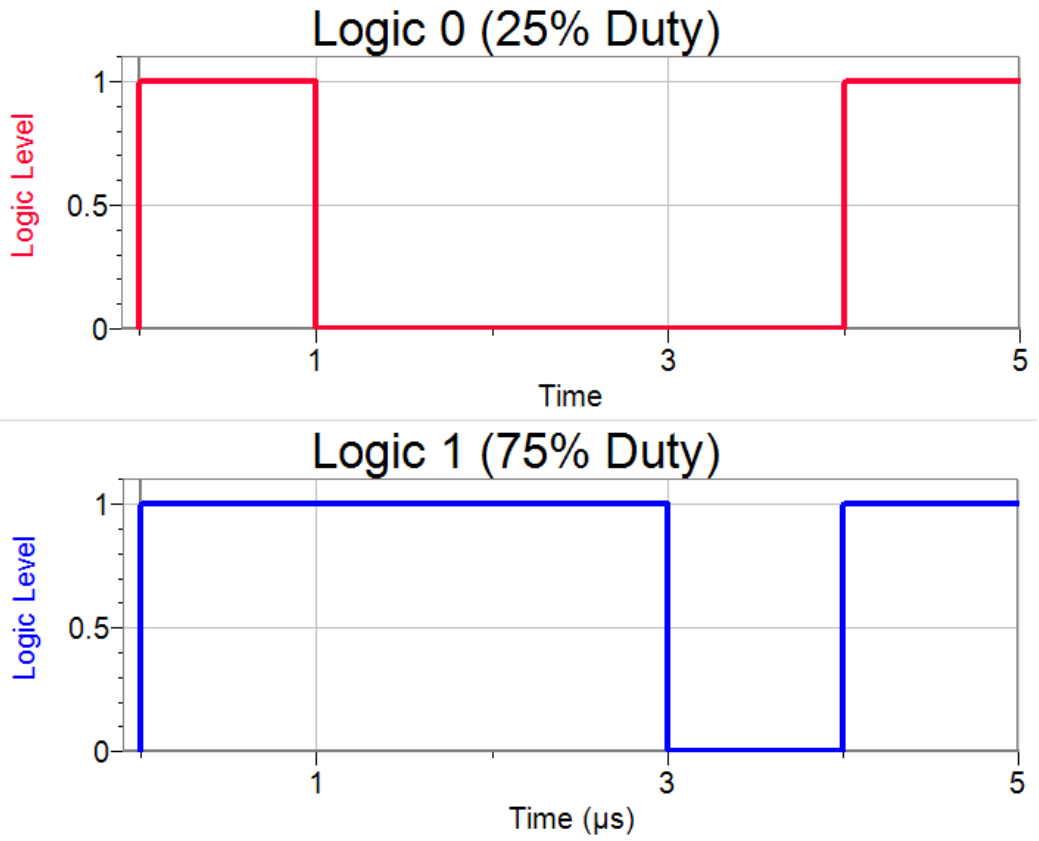


Figure 2-8: Idealized On-Off Keying Scheme. Top shows a logic 0, bottom shows a logic 1. Data is encoded between rising edges.

to a logic 0. This type of data-encoding scheme was selected because it embeds both the data and clock information into a single channel such that synchronized clocks between transmitter and receiver are not needed. Ideally, the duty cycle should be as close to 75% and 25% for logic 1 and 0 respectively in order to have the greatest immunity against timing errors and non-idealities such as clock jitter and skew. In order to decode duty cycle, the receiving system simply counts how many cycles a reference clock has 'ticked' between subsequent rising and falling edges corresponding to high and low times respectively. By simply comparing the two counts, a duty cycle estimation can be extracted. It should be noted that this reference clock needs to be several times the data rate in order to have sufficient resolution to compare high and low times as well as to have enough clock cycles to perform any calculation and transmit operation between data bits while not impacting subsequent bit counting. An example of the encoding scheme is shown in Figure 2-8.

In this design, OOK data transmission from the implant to external device is achieved by modulating the reflected impedance to the external device at approximately 62.5KBit/s. MOSFETs $M_1$, and $M_2$ in Figure 2-5, in conjunction with large value DC blocking capacitors $C_2$ and $C_3$, act as switches driven by $Uplink\_Out$, presenting an AC short across power coil $L_1$. This loading causes a drop in amplitude in the external coil voltage, thereby amplitude-modulating OOK data on the external device power coil. Detection and demodulation of this signal is discussed in Section 2.4.1. This topology for load modulation was chosen since power is delivered to the implant when the gates $M_1$ and $M_2$ are tied to ground. This is important for cold device startup since there is no bias voltage required to achieve resonance for power delivery - an issue in previous generation designs that caused a lockout fault and device failure after a period of time when not powered externally.

To pass data from the external device to implant in order to program the ASIC, OOK data is modulated onto the power link by turning on and off the external RF coil. Data is decoded by the implant device by monitoring the voltage level after the bridge rectifier as shown in figures 2-5 and 2-9 as $DC/Data$. Here, node $DC/Data$ is first voltage divided to a lower level so as to not damage the buffer, then low-pass

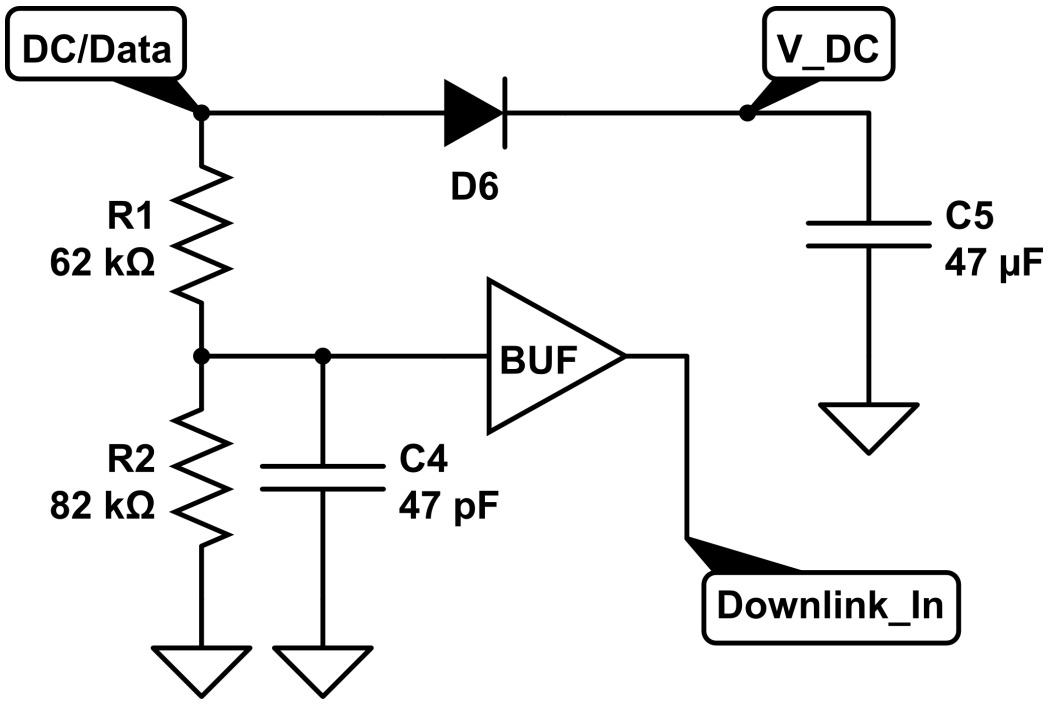


Figure 2-9: Schematic for Implant Decoder. $R_1, R_2$, and $C_4$ filter the carrier ripple and BUF is a Schmidt trigger input buffer for decoding.

filtered to remove carrier ripple while leaving the modulated data. Cutoff frequency for this filter is given by:

$$f_c = \frac{1}{2\pi(R_1||R_2)C_4} \tag{2.7}$$

the waveform is then fed into a Schmidt trigger input buffer in order to remove any additional noise and convert the waveform to a digital signal shown as $Downlink\_In$. This signal is sent directly to the ASIC. For our system, data is coming in at approximately 10kbit/s on a 2.5MHz carrier with a typical DC rectified level around 3V. This results in a cutoff frequency range requirement that is less than the 5MHz ripple due to the bridge rectifier, but greater than 10KHz, and a resistor divider ratio of .75 to have an output around 1.8V. A cutoff of approximately 100KHz is selected by setting $R_1$ to 62k$\Omega$, $R_2$ to 82k$\Omega$ and $C_4$ to 47pF.

Capacitor $C_5$ is the primary bulk storage capacitor for the system. Diode $D_6$ is a second rectification stage to block charge from flowing backwards into the detector from $C_5$ and other downstream energy storage elements thus providing a stable output voltage $V\_DC$.

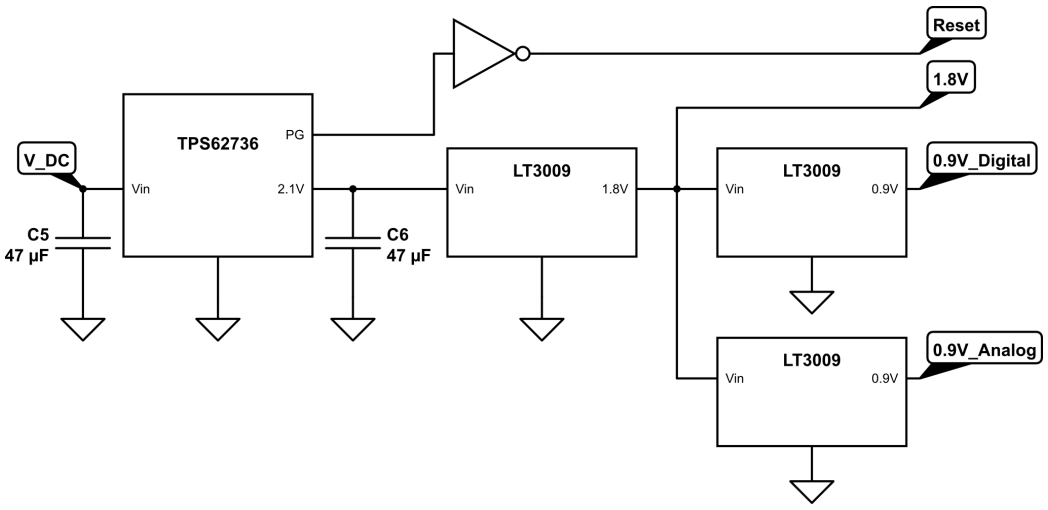


Figure 2-10: Schematic for Implant Power Management. $C_5$ and $C_6$ are bulk storage capacitors. TPS62736 is a buck converter, and LT3009 are low Iq linear regulators.

### Power Management

Further size reductions of the implant are realized through optimizations in on-board, off-chip power management. Through the elimination of Mode 2 from the ASIC, the device no longer is required to operate without an external power source as the power link also serves as the device's communication link to stream the EEG data. This allows for the removal of the built-in lithium ion and associated charging electronics, yielding a simplified power scheme. This also eliminates a charging fault condition in the previous design where the battery could not be recharged if charge fell below a certain, level rendering this device unusable.

Because the implant will only be operational when an external device is providing power, internal energy storage elements can be kept at a minimum. For this design, the primary energy storage element is a capacitor shown as $C_5$ in Figure 2-9 and 2-10. The only requirement for this capacitor is to ensure that the voltage at $V\_DC$ does not change appreciably when the external coil is off during ASIC programming. Assuming an extreme worst-case current consumption of 1mA and a worst-case duration that the external coil is off of 100$\mu$s, total charge consumed is 100nC. For an ideal 47$\mu$F capacitor, this results in a ripple of 2.1mV, or $<<$ 1% of the nominal 3V DC input level.

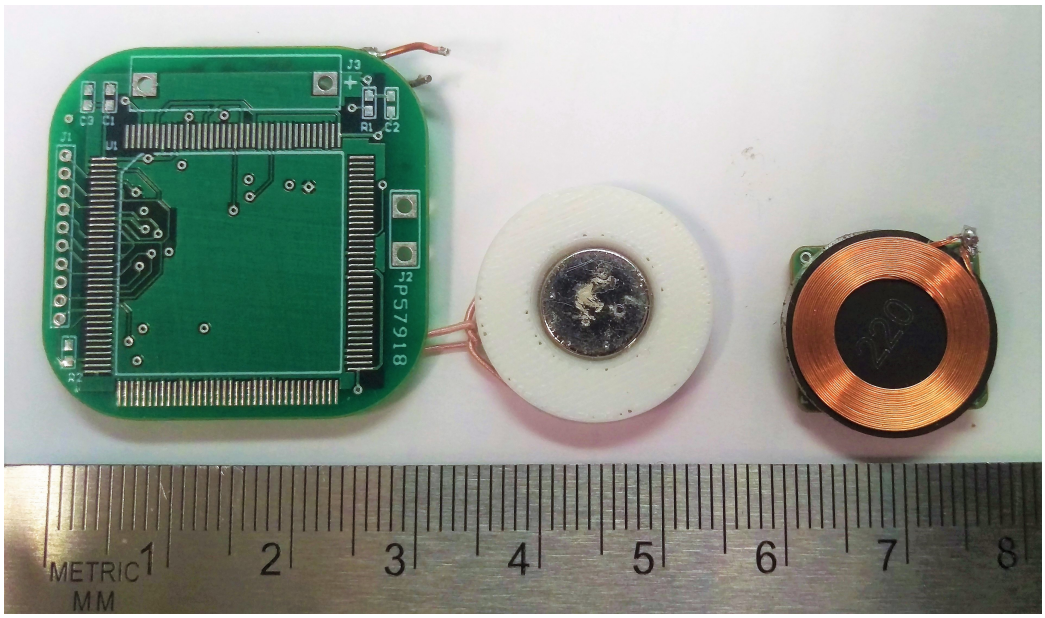


Figure 2-11: Implant PCB. Previous design with external coil attached (left). This work (right).

The ASIC itself requires 3 separate power supplies; a 1.8V supply and two 0.9V supplies for analog and digital. The unregulated DC voltage is first stepped down to 2.1V using a switching regulator. The TPS62736 is chosen for its light load efficiency and low quiescent current consumption as well as control outputs such as a "Power Good" (PG) signal that is used to hold the implant in reset until the output voltage has stabilized. To filter the voltage and remove any switching noise from the switching regulator and to step the voltage down further, a linear regulator first generates the 1.8V rail, then two more regulators generate the 0.9V. The LT3009 was selected for the linear regulators for its extremely low quiescent current.

### 2.3.2 Mechanical

#### PCB

The aforementioned circuits are designed in Altium Designer and printed and assembled by Sierra Circuits. The board measures 14.0mm x 15.5mm. Three sets of electrodes are attached to the device - two four-contact electrodes for the 8 channels of the EEG, and a third two-contact electrode containing a ground and reference

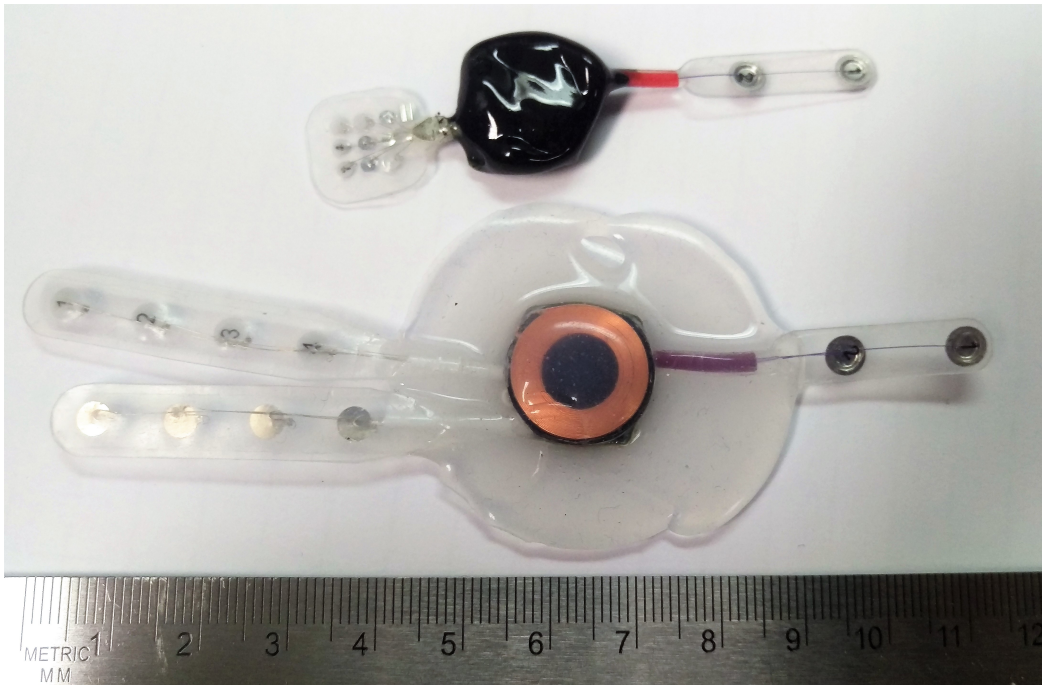


Figure 2-12: Coated Implant Devices. Epoxy coated[1] (top), Parylene/Silicone coated (bottom).

channel.

In the previous design, the power coil was a handmade coil wound around a plastic holder using litz wire resulting in a thick coil due to the mechanics of the coil. Because of the thick size, the coil had to be placed adjacent to the implanted PCB, resulting in a very large implant footprint size. By moving to pre-fabricated coils using higher gauge magnetic wire, coil thicknesses and diameter can be minimized to the point that the coil can be stacked on top of the PCB for further miniaturization.

Figure 2-11 shows the completed PCB on the right. The electrodes and coil are soldered in place and bonded to the PCB using superglue. The overall result is a device that is roughly 10% the original size when the coil is added.

### Fluid Barrier and Biocompatibility

In order to ensure the device remains functional while implanted, the device must be coated to prevent electrical shorting by body fluids. Two types of coating were

[1]Epoxy device shown has a 3x3 electrode array attached instead of the standard dual 1x4 electrode array.

Table 2.1: Electrical Properties of ParyleneN and MG Chemicals 832B

| *Parameter* | **Paryelene** | **MGC 832B** |
|---|---|---|
| Resistivity ($\Omega$/cm) | $1.4 \times 10^{13}$ | $5.3 \times 10^{12}$ |
| Dielectric Strength (V/$\mu$m) | 276 | 18.6 |
| Dielectric $\kappa$ @ 1MHz | 2.65 | 2.77 |
| Dissipation Factor @ 1MHz | $0.6 \times 10^{-}3$ | $17 \times 10^{-}3$ |

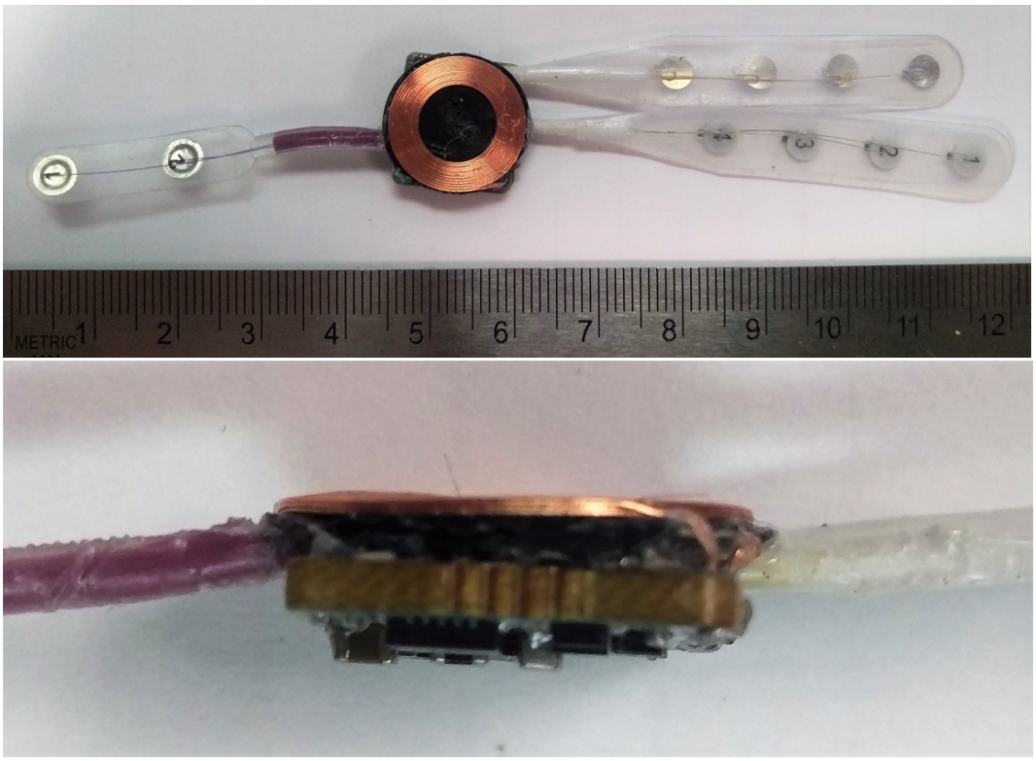


Figure 2-13: Parylene coated PCB - Top and Side Views. Features, ridges and overhangs of the board are still clearly defined and visible.

tested. In the first test, the device was coated in MG Chemicals 832B Black Epoxy and Potting Compound, as shown in the left of Figure 2-12. The device was coated by repeatedly dipping the implant in a large container of epoxy until the desired coating thickness was achieved - until there were no remaining exposed contacts. The coating process was verified by ensuring the device remained functional while submerged in a saline solution.

However, during the first animal study, an electrical interaction with the pig tissue caused the power coil, $L_1$ in Figure 2-5, to shift in resonance slightly, resulting in reduced voltage and power transfer and inability to power on the device. This was likely caused by the high-frequency 2.5MHz carrier capacitively coupling into the pig's tissue as a result of a too thin coating and relatively high dielectric constant, $\kappa$, of the epoxy. Capacitive coupling into the pig shifts the impedance seen by the coil and moves the device away from resonance. In order to address capacitive coupling, the device was sent to VSIParylene to be professionally coated in 20$\mu$m of parylene. Parylene is a standard coating for many medical implants and offers a much higher resistance and $\kappa$ than the epoxy that was previously used. Table 2.1 compares the electrical specifications of MG Chemicals 832B epoxy with VSIParylene Parylene N coatings. Furthermore, parylene is applied using vapor deposition, which allows for a much more uniform coating on non-uniform surfaces than repeated dipping, thereby ensuring fewer openings for fluid ingress or capacitive leakage.

One pitfall of conformal coating via vapor deposition is that physical features or overhangs on the PCB will be covered, but still remain after coating, as shown in Figure 2-13. This means that any ridge or overhang will cause an air or fluid pocket when implanted, which heightens the risk of infection due to a lack of blood profusion in those pockets[2]. To address this, the device is coated in a silicone overmold in order to fill in any ridges, overhangs and other features. The outer silicone overmold provides a featureless and smooth surface that conforms to the surface of the skull to reduce air and fluid pockets, while the inner parylene coating ensures a strong electrical and fluid barrier.

[2]See Section 2.5.2.

It should be noted that the surface energy of parylene is very low making the coating highly hydrophobic. This prevents many overmoldings, including silicone, to not adhere very well to the device. In order to increase the surface energy to allow for increased bonding with our silicone, the device is placed in an argon plasma chamber for surface etching. Because the high electric fields introduced by the plasma may damage the exposed electrodes, the electrodes were all connected together using copper tape and connected to ground in order to ensure no high fields were placed across any two electrodes.

To coat the device, the device is placed in a small shallow bowl and quick setting silicone is poured over the device. The device is then flipped and other side coated. The final parylene/silicone coated device is shown on the right of Figure 2-12.

In vivo testing of this device is described in more detail in Section 2.5.

## 2.4 External Device Design

Similar to the implant revision, the main goal of the external device design revision is further miniaturization and mechanical survivability during in-vivo experiments. Previous animal tests showed that the large physical size of the external device, due to large power requirements, irritated the pig during the experiment giving more resolve to remove and destroy the test setup. Additionally, the bulky external device required the battery and electronics to be mounted away from the head and connect to the transmit coil on the pig's head via fragile tether cable. From the pitfalls of the previous animal experiments, the following design goals of the external device are derived:

- Miniaturized form factor that is self-contained entirely on the pig's head.
- Efficiency optimized RF amplifier for lower power consumption.
- Use of BTLE for lower power consumption.
- Operation for 100 hours on a single battery.

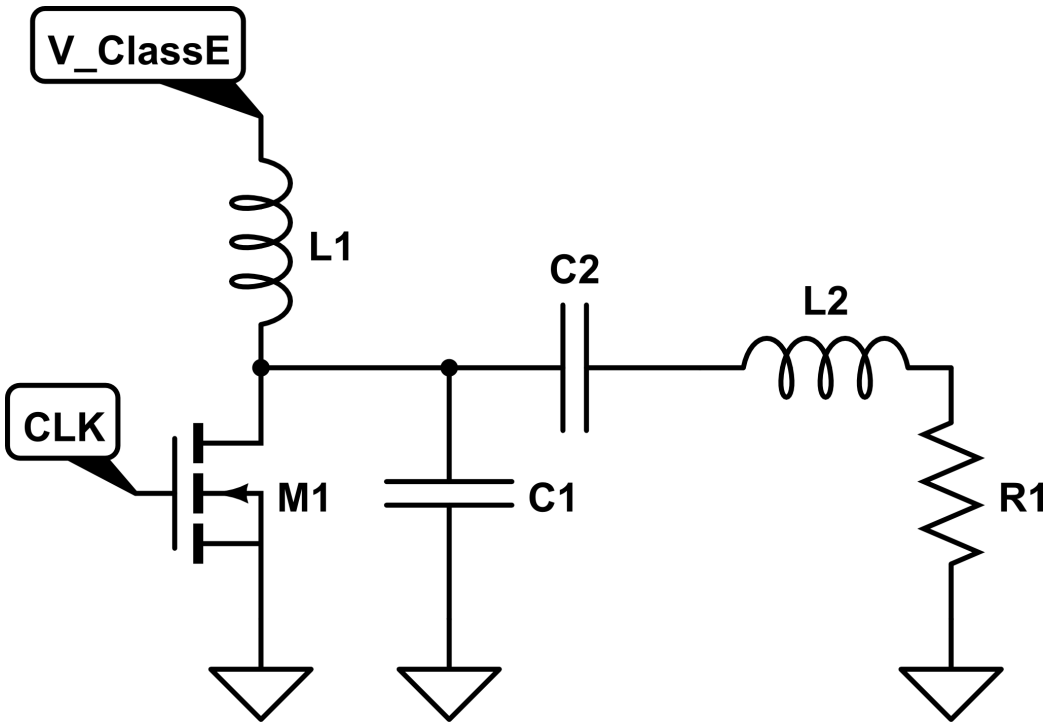


Figure 2-14: Generic Schematic for Class-E RF Power Amplifier.

- Optimized mechanical housing that is more easily serviceable by technicians but impact proof.

### 2.4.1 Electrical

#### RF Power Amplifier

The previous design of the external device relied on a class-E amplifier to transmit power and data to the implant via magnetically coupled coils. Power and data delivery is discussed in Section 2.3.1. Class-E was originally chosen due to its theoretical 100% efficiency and single phase clock requirement. However, in practice, the class-E power amplifier has a much more complex topology and relies heavily on matched impedances for optimal performance [21][22]. This becomes an issue when impedances are not tightly controlled. A generic class-E topology is shown in Figure 2-14. Equation 2.6 shows that the reflected impedance from the implant across the primary coil on the external device is a strong function of the coupling constant $k$. Since reflected impedance is changing not only for data modulation but, also coil

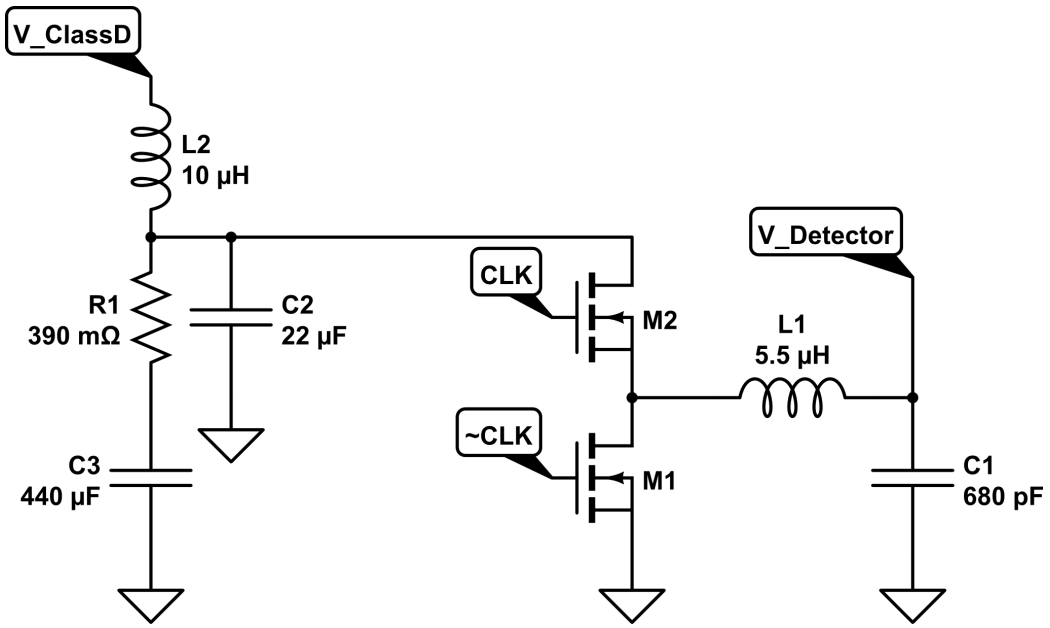


Figure 2-15: Schematic for Class-D RF Power Amplifier. $M_1$ and $M_2$ are the switching elements. $L_1$ is the external RF coil that resonates with capacitor $C_1$. $L_2, C_2, C_3$ and $R_1$ form an input power filter for noise filtering

positioning, efficient power transfer using a class-E amplifier is exceedingly difficult. Furthermore, the class E amplifier exhibits a very large peak-peak voltage for low DC input voltages requiring very high current and low voltage power rails on the order of 0.5V, which is difficult to generate efficiently from a 3V supply.

In order to address these issues in the previous design, a class-D topology is used as the power amplifier. The class-D is also a switching amplifier; hence, like the class-E, efficiencies $\eta > 90\%$ can be theoretically achieved. Here, the class-D amplifier trades simplicity of design and matching for complexity in driving the amplifier - the design has only four primary components but requires a two-phase-shifted clock for optimal performance.

Figure 2-15 shows the schematic for the class-D power amplifier used in the external design to power the implant. Inductor $L_1$ is a 5.5$\mu$H pre-fabricated transmit coil that is magnetically coupled to the coil in the implant device (shown as $L_1$ in 2-5). This inductor, combined with 680pF $C_1$, form a resonant output filter where the resonant frequency is 2.5MHz as determined by equation 2.3. As mentioned before, $C_1$ is slightly lower than the expected value for resonance in order to account for

parasitics due to inter-winding capacitance and downstream electronics.

This filter gains the fundamental of a square wave generated by two NMOS switches, $M_1$ and $M_2$, which are stacked to form a “totem pole" structure. These two switches are controlled by $CLK$ and its out-of-phase complement, $\tilde{}CLK$, generated by a micro-controller with a duty cycle slightly less than 50% in order to prevent shoot-through current - current from the power rail directly to ground when both switches are turned on simultaneously. The class-D can be turned off or on to signal data by stopping or starting the clocks that feed $M_1$ and $M_2$. It should be noted that gate drivers are not needed for this design as the class-D supply voltage, $V\_classD \approx 1.2V$, is significantly lower than the micro-controller $\approx 3.0V$ which is adequate to turn on high-side switch $M_2$. Furthermore, small switching MOSFETs are used that do not have significant gate capacitances as the total power dissipated by the class-D is only in the few 10's of mW.

Similar to the implant device, the experimental setup requires manual trimming to account for coil manufacturing variation. To achieve this, peak-peak voltage output of the class-D amplifier is maximized during no-load operation by adjusting resonance capacitor $C_1$.

### Input Filter

Elements $L_2, R_1, C_1$ and $C_2$ in Figure 2-15 form a damped $2^{nd}$ order filter to prevent current ripple injection back into the power supply from the high switching transients generated by the class-D. This is important, as the previous design showed that large current transients destabilized the feedback in upstream buck regulators that generate $V\_ClassD$, thus lowering overall expected efficiency of the system. Here, $L_2$ and $C_2$ form a filter with a cutoff frequency around 10KHz, which is 2 orders of magnitude lower than the expected 2.5MHz injected noise. This ideally results in at least 80dB of attenuation of current noise rejection at 2.5MHz. In actuality, attenuation is closer to -50dB due to the use of non-ideal components yielding parasitic poles and zeros in the transfer function. In order to reduce the resonant peak generated by the LC pair, $R_1$ is inserted to dampen the system. The value of $R_1$ was determined through

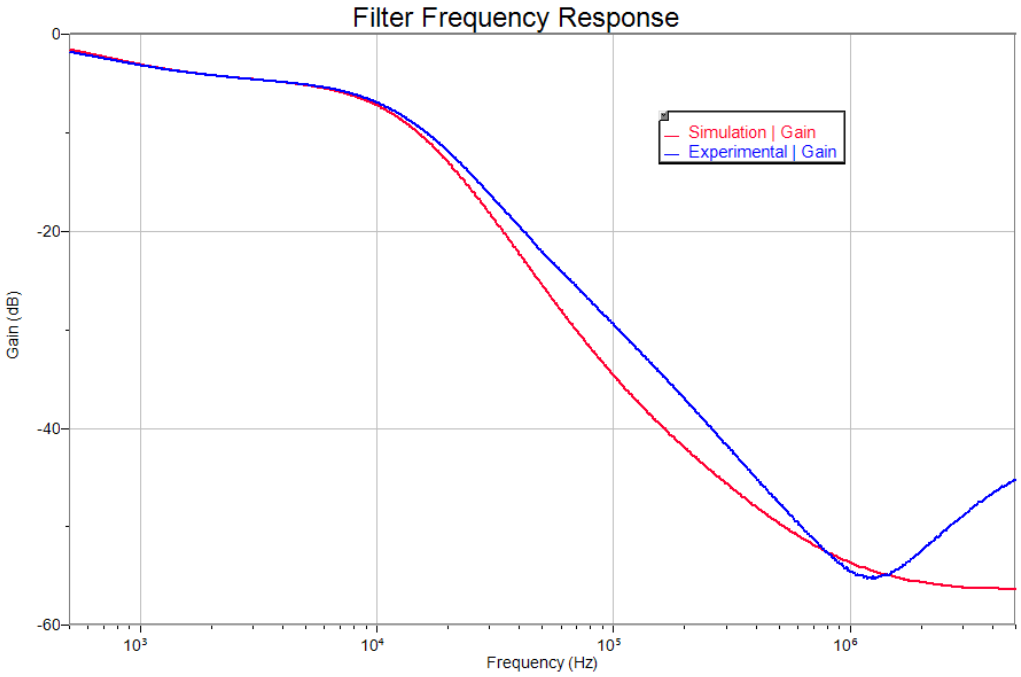


Figure 2-16: Power Filter Transfer Function. LTSpice gain and phase simulation is shown in red, measured results are shown in blue.

simulation. $C_3$ is placed as a large DC blocking capacitor to ensure no DC current is shorted to ground. Performance of the filter was simulated in LTSpice using device models derived by manufacturer data and individual device testing. The experimental setup was verified on the final device using a network analyzer measuring the voltage transfer function from the power rail to the top of the switches. In this filter topology, the voltage transfer function from the rail to the top of the switches is the current transfer function from the top of the switches back into the power rail. Both the simulated and experimental results are shown in the gain-phase plots in Figure 2-16.

### Detector

As described in Section 2.3.1, communication from the implant to the external device is achieved by impedance modulating OOK data across the implant coil, which is reflected across the external coil. This change in impedance amplitude-modulates (AM) the OOK data on the 2.5MHz RF signal and can be measured at node $V\_Detector$ referenced in figures 2-15 and 2-17. In order to demodulate the AM waveform, the

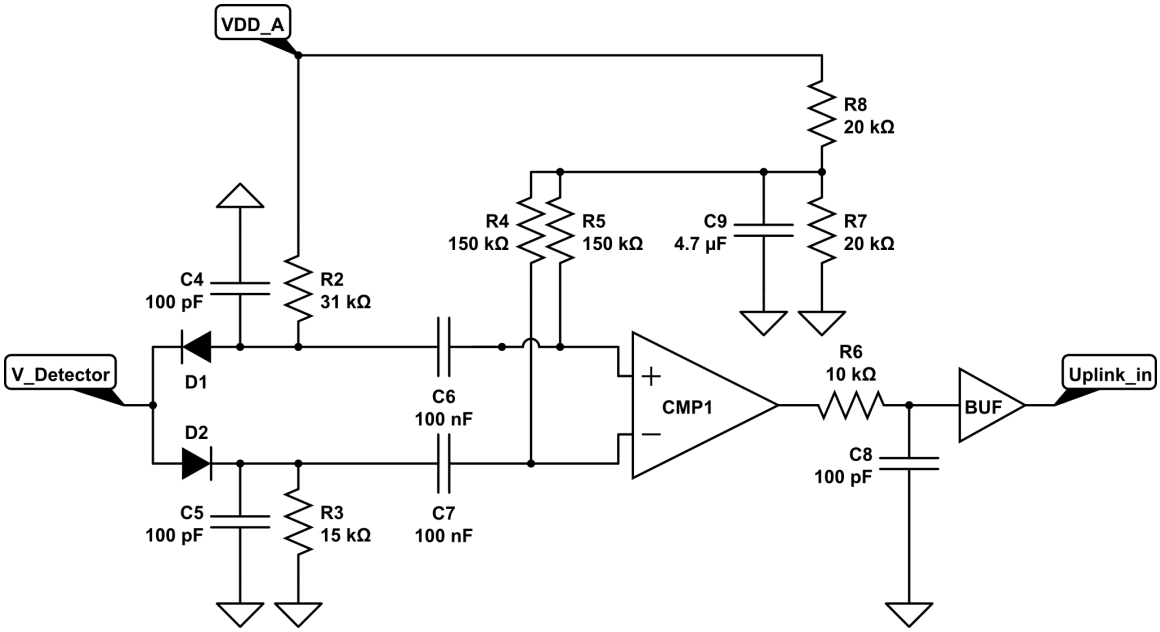


Figure 2-17: Schematic for Differential Envelope Detector.

signal is passed through the differential envelope detector. A differential topology was selected for detection in order to keep the system symmetrical and to increase the signal-noise ratio (SNR) as compared to a single-ended envelope detector. This differential detector is shown in Figure 2-17.

Opposite-facing diodes, $D_1$ and $D_2$ rectify the negative and positive cycles of the AM signal respectively. $C_4/C_5$ and $R_2/R_3$ form a filter to remove carrier ripple while passing the OOK data. The resulting waveforms are AC-coupled by $C_6/C_7$ to be centered around a mid-rail voltage generated by $R_7, R_8$ and $C_9$. Both differential signals are fed into a comparator to convert to a single-ended digital output. Finally, $R_6, C_8$ and $Buf$ form a glitch filter in order to remove any glitches caused by carrier ripple. The resulting digital signal is sent to an on board micro-controller for decoding. Waveforms from each step of the detector in the experimental setup is shown in Figure 2-18. It should be noted that Graph 5 demonstrates a glitch at $t \approx 16\mu s$ that is filtered out in Graph 6. It should also be noted that the output waveform pulse widths are slightly distorted when compared to the input OOK signal. This is due to the differential rectification step where the signal has an asymmetrical rise and fall time due to diodes $D_1$ and $D_2$.

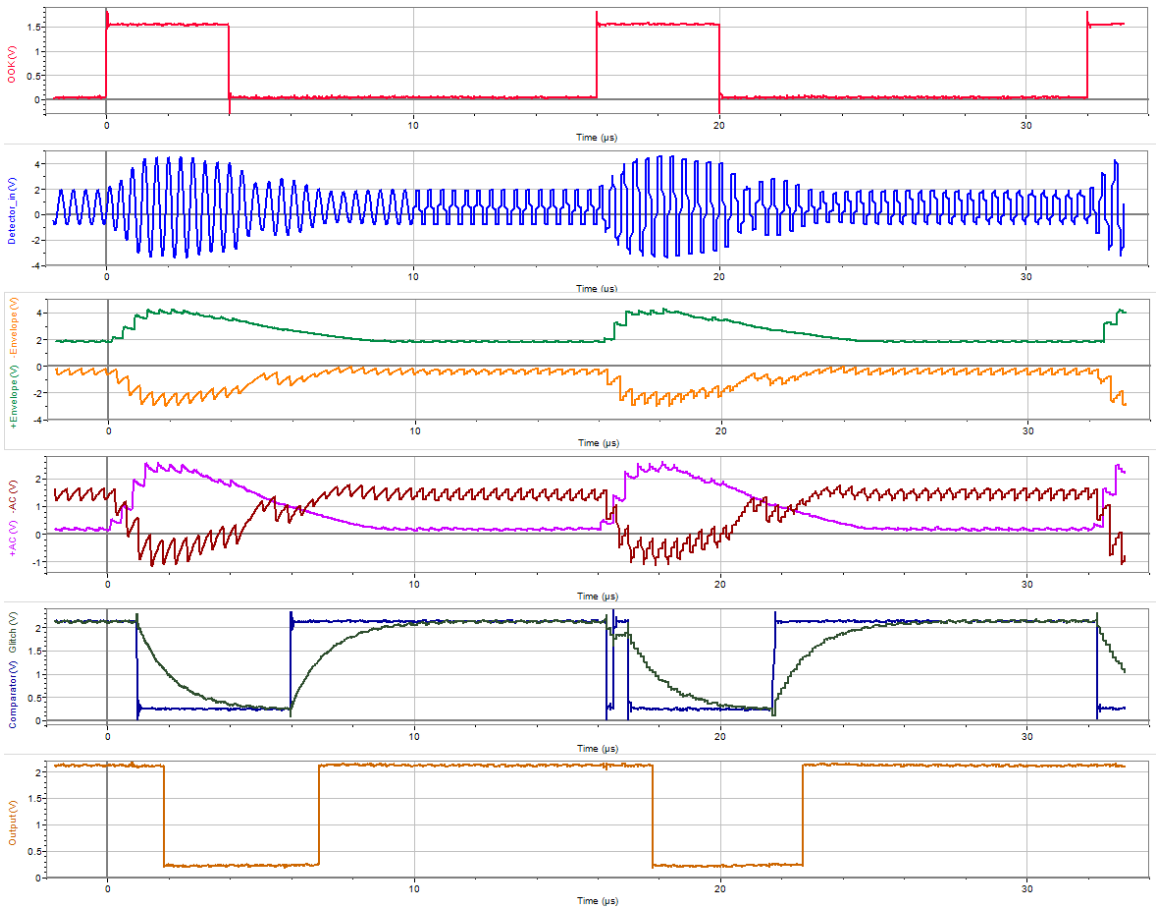


Figure 2-18: External Detector Waveforms. From top to bottom: OOK Data, Detector Input, Positive and Negative Envelope Detector, AC Coupled Envelope Detectors, Comparator and Glitch-Filter Outputs, Final Output. Note that the final output pulse width is slightly skewed compared to the input due to decay times presented by the rectification step.

Overall, this topology allows decoding of asynchronous AM OOK data where the carrier amplitude and modulation depth can change continuously due to coil placement.

### Digital Core and Communications

The external device contains a TI MSP430 micro-controller which handles the state-machine of the system and decoding of the data. The MSP430 series was selected due to their extremely low power consumption (<2.5mA typical at full load). In order to further optimize power, the MSP430 is kept in a sleep state for as long as possible. During operation, the micro-controller is held in sleep except for when an edge on the OOK data line triggers an interrupt. During the interrupt, the micro-controller only records the time since the last event and if the interrupt is a falling edge, compares the recorded times and writes the result to the UART. The micro-controller is placed in low-power sleep otherwise. C code is included in Appendix A.

The MSP430 communicates with a nearby laptop or RaspberryPi via a BTLE link connected via the UART lines. The previous design of the external device utilized a Bluetooth Classic radio for communication, which allowed for higher data rates, but required an order of magnitude more power to run and a much larger PCB footprint. By migrating to a DA14583 BTLE chip, power consumption and device size can be reduced. It should be noted that the DA14583 is using a custom, proprietary stack which only allows for phones or receivers running the same hardware and BTLE stack to communicate with it. This limitation is addressed in Section 4.1.1.

### Power Consumption

Due to optimizations in Bluetooth communication and RF power amplifier design, power consumption of the device has been decreased by one order of magnitude: from 205mW in the previous design, to 32.1mW when transmitting and running from a 3V supply and implant device 1cm away. Measurements were taken by a Keithley Sourcemeter powering the device at the power input. Table 2.2 compares relevant

Table 2.2: Power Consumption of External Device. Devices measured at 3V, 1.0cm coil distance, and transmitting to a receiver.

| **Block** | **Previous Design (mW)** | **This Work (mW)** |
|---|---|---|
| Micro-controller | 7.5 | 6.0 |
| Bluetooth | 87 | 8.3 |
| RF Power Amplifier | 42 | 16.6 |
| Analog Components + PMIC | 70[3] | 1.2 |
| **Total** | 205 | 32.1 |

sub-blocks of the previous design with this work.

#### Battery Selection

Previously, the high power consumption of the external device required a large battery to be used which made the device overly large and bulky. For this design, we aim for a runtime of approximately 100 hours using a 3Wh battery. It should be noted that the external device requires a constant current draw of approximately 10mA if powered by a 3V source. Although the largest 3V commercial button battery, CR2477, has the capacity needed and would provide an optimal shape and size to power this device, Lithium Manganese Dioxide (Li-$MnO_2$) cells have a nominal current draw in the 100's of $\mu$A and cannot be used for this application. A Tadiran 1/2AA lithium ion battery was selected to power this device. The battery has a nominal capacity of 3.9Wh while suppling a continuous current in excess of 10mA. The cylindrical battery measures measures 14.5mm dia x 25.0mm and should last for approximately 120 hours.

### 2.4.2 Mechanical

#### PCB

The electronics were printed on a PCB shown in Figure 2-19 on the right. The board measures 22mm in diameter. Similar to the implant device, the transmit coil is a

[3]Power consumption is high due to noise injection into power supply. See Section 2.4.1, subSection *Filters*.

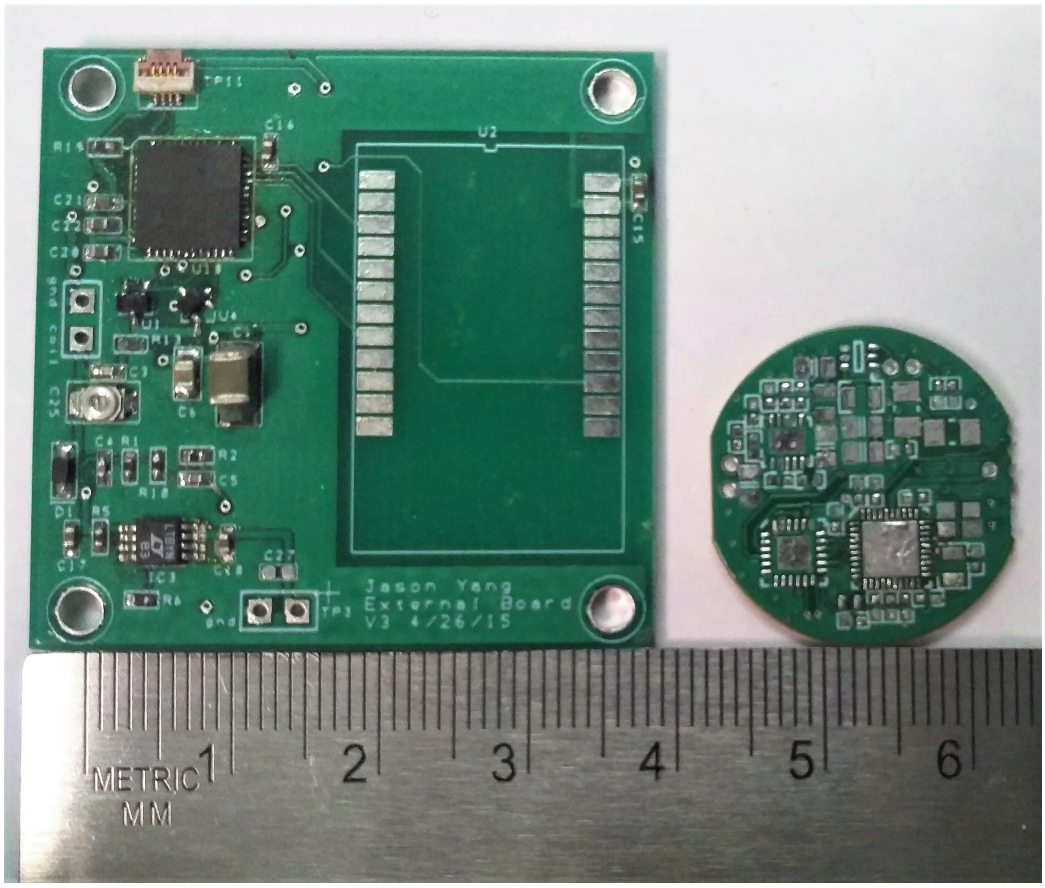


Figure 2-19: External Device PCB. Previous design (left). This work (right).

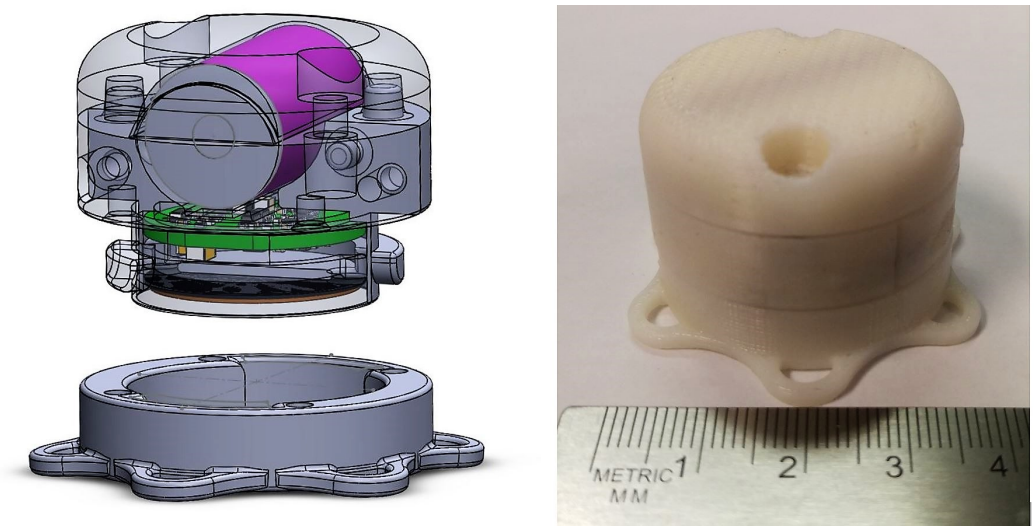


Figure 2-20: External Device Enclosure. 3D Render (left), 3D Printed (right).

prefabricated coil using high gauge wire resulting in a very thin package that can be stacked on top of the PCB to reduce device footprint.

### Enclosure

Previous designs used in in-vivo animal tests relied on a backpack-mounted enclosure with a tether wire from the electronics to the transmit coil located pig's head. This tether was frequently damaged, causing the experiment to fail. Because of the reduced board size and power consumption allowing the use of smaller batteries, cumbersome enclosures to encompass the external electronics are no longer needed and the battery, board, and transmit coil can all reside on the pig's head eliminating the need of a tether wire.

In order to achieve this, an enclosure was designed with the help of Nevan Hanumara, Ph.D and Daniel Teo, Ph.D. 3D renders of the enclosure showing battery (purple, top), PCB (green, middle) and transmit coil (black, bottom) with mounting plate is shown in the left of Figure 2-20. 3D prints of the device are shown on the right. A baseplate is mounted on the pig and the electronics package is mounted on the baseplate via a quarter turn-lock secured by magnets for quick removal and replacement for servicing. All the electronics are contained in the upper portion of the device that can be serviced on a bench with a hex Allen wrench away from the

Figure 2-21: Operating Room Theater for Pig Implantation.

pig. Figure 2-24 shows the device mounted on the pig. The enclosed external device measures 30mm in diameter and 22mm in height.

## 2.5 Animal Testing

The implant was tested in-vivo using 50kg swine at Massachusetts General Hospital (MGH). Tests were performed on two different pigs named Snowball and Kevin. Figure 2-21 shows the operating theater for implantation.

### 2.5.1 Pig 1: *Snowball*

The pig is first anesthetized and an approximately 6cm x 9cm U-shaped incision is made right above the nuchal crest on the head of the pig. An epoxied device is placed down on top of the skull with the eight electrodes pointing towards the nose of the pig and ground and reference electrodes folded to the rear. The device and electrodes are glued down to the periosteum of the skull. The skin flap is replaced over the

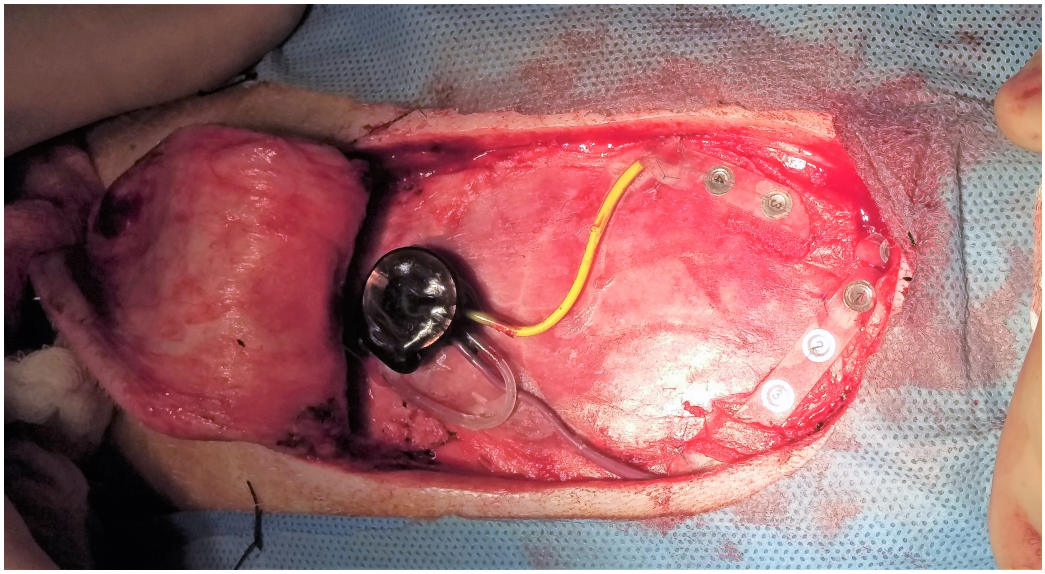

Figure 2-22: Pig 1 implantation surgery showing electronics package (black) and 8 electrodes (yellow, brown).

surgical area and device is tested a final time before closing the incision. Finally, the external device baseplate is sutured down to the skin right above the implant, and the external device mounted in the baseplate and pig allowed to wake up.

Unfortunately, shortly after closing up the pig, the external device stopped being able to communicate with the implant. This was caused by the minimum gap between the coils exceeding the experimentally validated 5mm. By placing the coils too close together, coupling constant $k$ becomes much larger than expected and reflected impedance across the external coil drops as shown in equation 2.6. This places a larger load across the external coil and causes the amplitude to drop. Furthermore, equation 2.4 suggests a higher $k$ also decreases voltage gain across the coils which holds the implant in a reset state due to too low of a rectified voltage. This change in minimum operating distance is likely caused by capacitive interactions with the pig's tissue which shifted the resonant frequency of the implant coil; however, this cannot be verified as the epoxy coated device cannot be probed once coated. In order to resurrect the device, the external device's peak-peak voltage level across the transmit coil was increased from the original 7Vpp to 10Vpp in order to overcome the reduced implant voltage. With this modified device, EEG could be recorded from the pig for

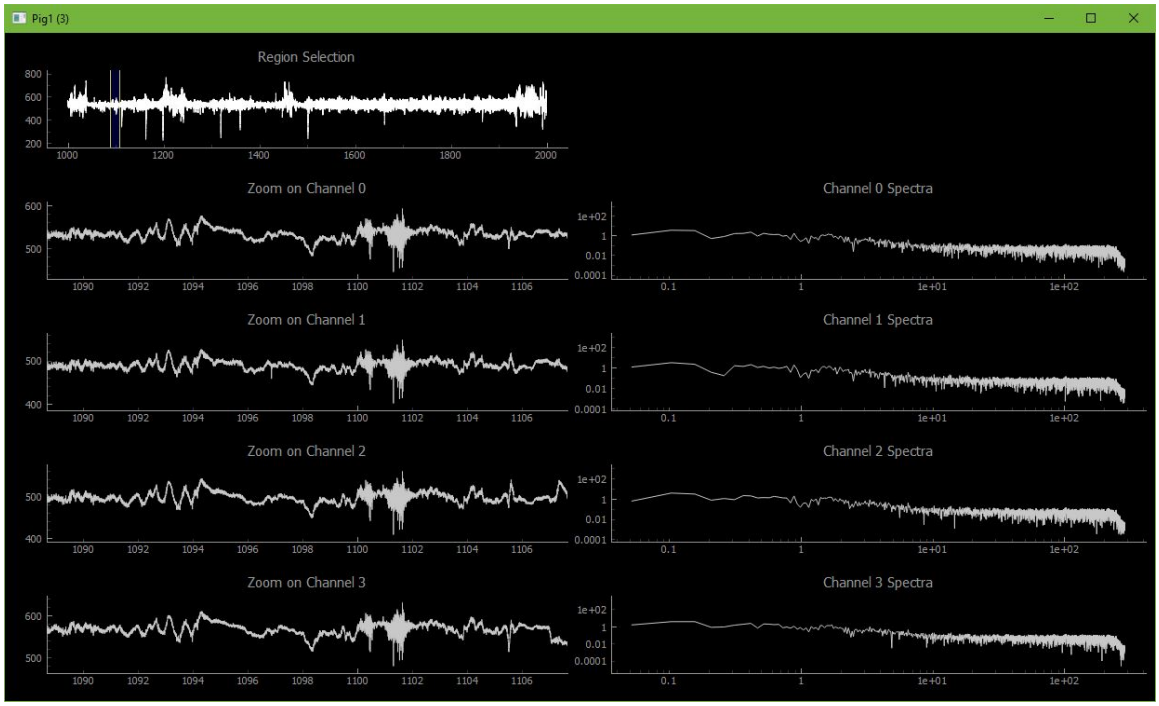


Figure 2-23: EEG Data from Pig 1. Only channels 0-3 are shown. Time domain is shown on the left, power spectrum is shown on the right.

about an hour and half. A sample of this data is shown in Figure 2-23.

Figure 2-24 shows the external device on top of the pig. Because the external device is now on the pig's head, the device becomes more prone to impact and liquids, as the pig has more freedom to hit the device against the pen's walls as well as soak the device when drinking water from an overhead spigot. This caused the device to fail again after an hour and half into testing due to water ingress into the external device, shorting out the electronics. The device was then removed from the pig and inner chamber, which holds the electronics, filled with epoxy to waterproof.

The device was again placed on the pig's head but failed after 30 minutes. It was initially thought that the failure stemmed from the sutures that held the baseplate to the pig's head coming lose and moving significantly, but after reattaching the external device, the implant still remained unresponsive. Connecting a spectrum analyzer to an external coil showed that resonance of the implant device had not changed significantly and that power transfer was being achieved, but device not responding indicating that the implant itself was electrically dead and could not be recovered.

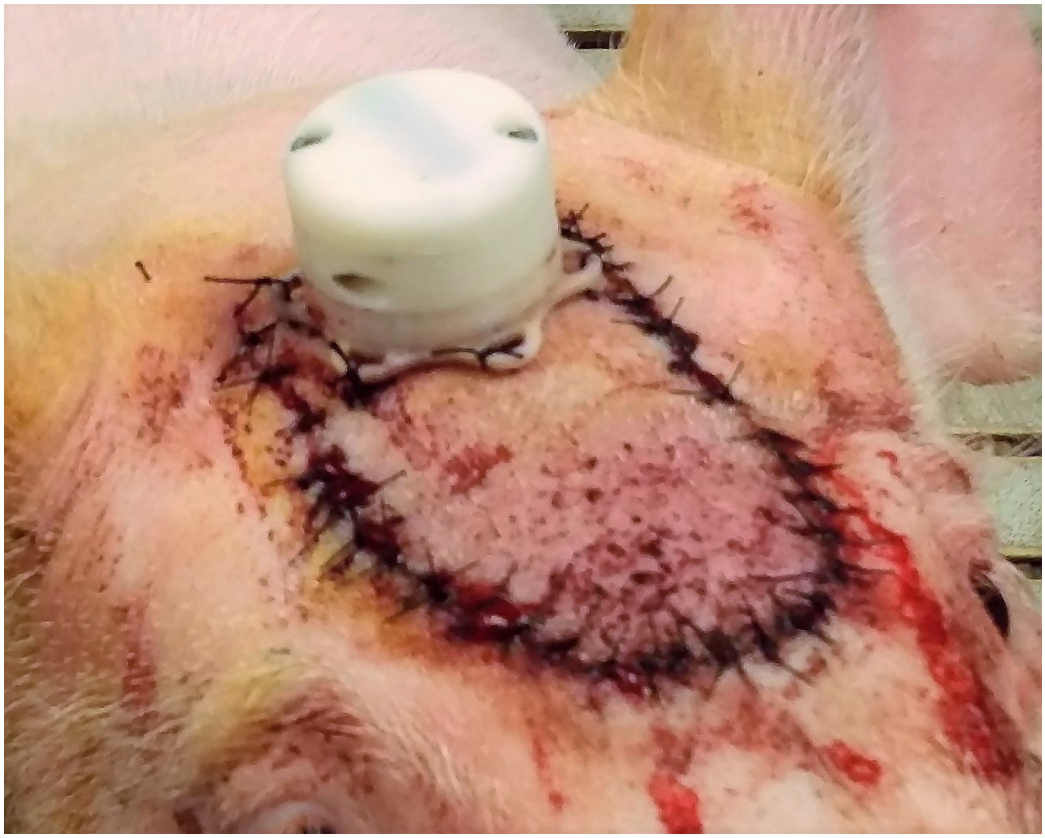

Figure 2-24: Pig 1 with External Device.

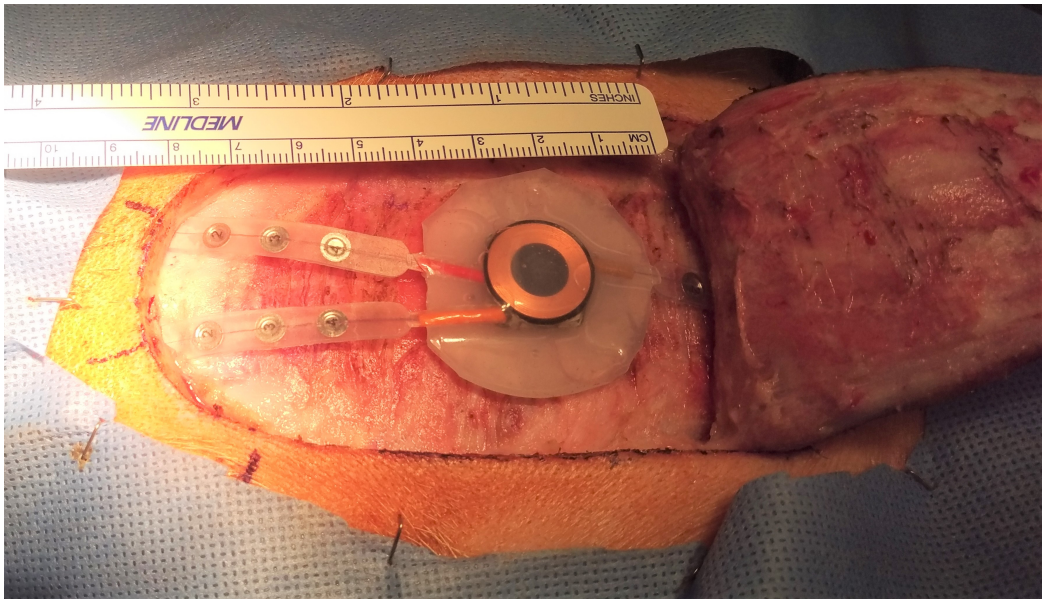


Figure 2-25: Pig 2 implantation surgery showing the silicone encapsulated device.

The pig was euthanized and device explanted. It should be noted that this particular implant did not have the protective Zener diode shown as $D_5$ in Figure 2-5. Because of this, it is likely that an overvoltage condition damaged the buffer in Figure 2-9. This overvoltage condition was likely caused by the loose device causing the two coils to move significantly relative to each other. This movement changes the $k$ between the implant and external coil which, in addition to the increased peak-peak voltage of the external device, can cause large voltages in the implant device as suggested by equation 2.4.

### 2.5.2 Pig 2: *Kevin B.*

From analysis of the failure mode from the first pig, the system was modified to be more electrically robust to the in-vivo environment. In order to address the possible capacitive tissue interaction causing changes in the resonant properties of the implant device, the implant was sent to be professionally coated in parylene as described in Section 2.3.2. Additionally, Zener diode $D_5$ in Figure 2-5 was added to protect against possible overvoltage conditions. Finally, resonance on the implant side was adjusted slightly to compensate for thinner pig skins. The effective range of the device now is

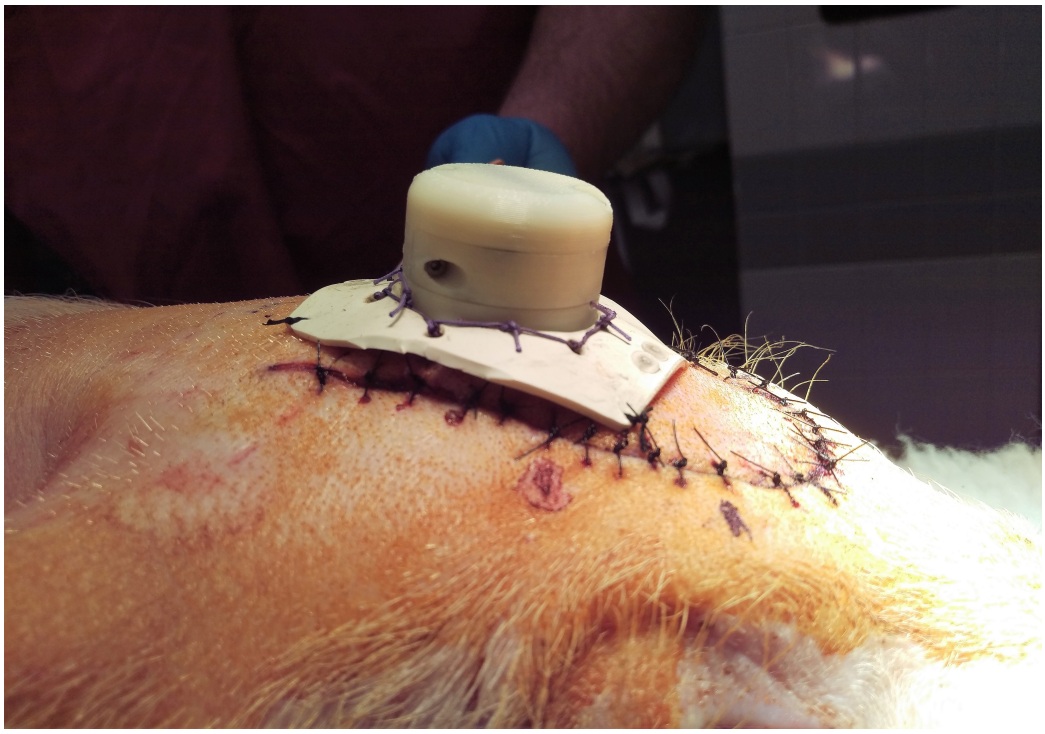

Figure 2-26: Pig 2 with External Device.

3mm to 12mm from the original 5mm to 15mm.

The implantation procedure remained largely the same as the previous experiment. Figure 2-25 shows the new device being implanted. In order to fortify the sutures such that the external device does not fall off like in the previous experiment, sutures were drawn through the entire skin flap instead of just through the first few epidermal layers. Additionally, a rubber sheet was placed over the external device and sutured down in order to reduce the strain on the primary sutures. The external device is shown in Figure 2-26.

In this trial, the system was able to collect data for approximately 2 and half hours, with a gap in data due to a battery change. Unfortunately, communication to the implant was lost afterward due to not enough voltage reaching the implant. It is hard to determine the exact root cause of this failure, but one possibility is that swelling of the skin caused the coils to move further apart, beyond the 12mm range. In order to overcome this, output voltage was increased yet again from 10Vpp to 13Vpp. This workaround successfully repaired the system; however, it was found that the

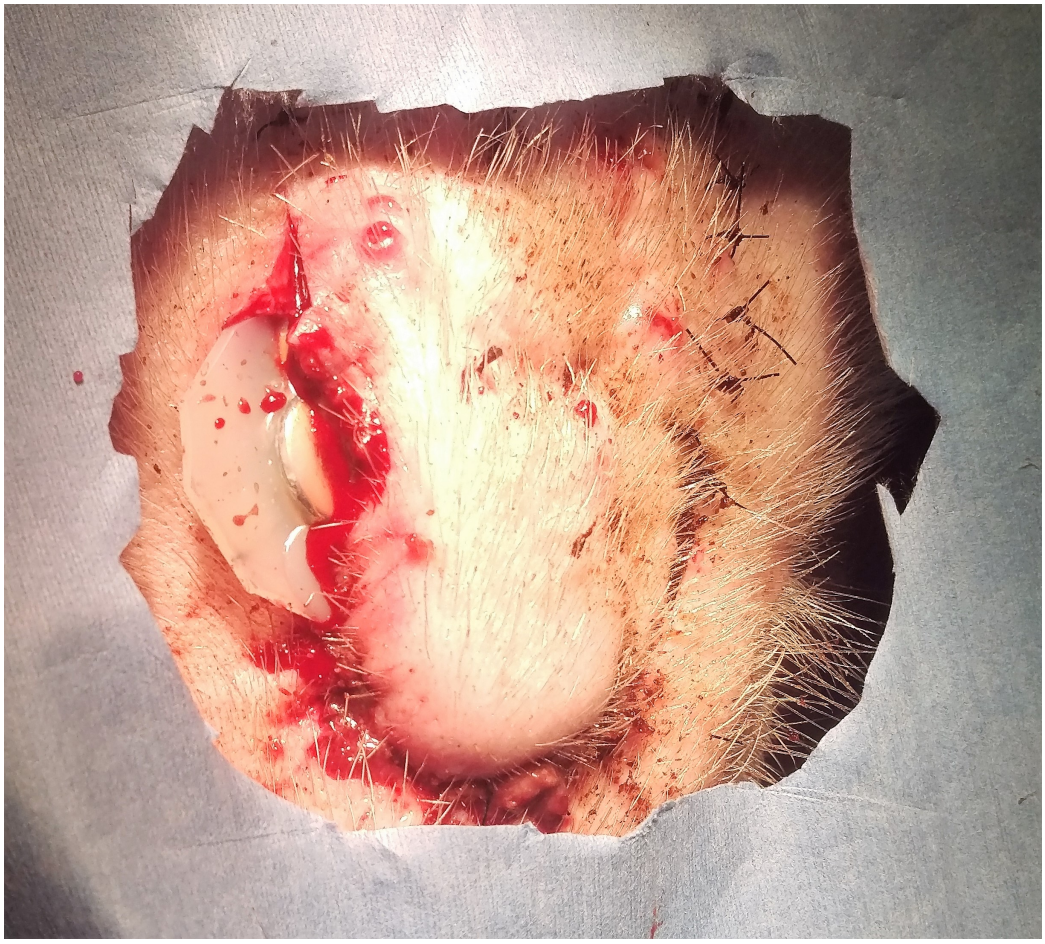

Figure 2-27: Pig 2 Infection in Head.

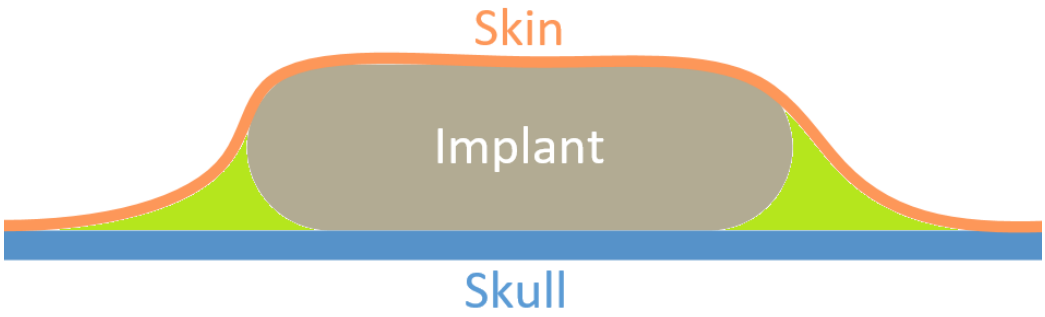


Figure 2-28: Cross section view of the implant under the skin. Skin is shown in orange, device in grey, skull in blue. Pockets are formed on the left and right of the implant underneath the skin shown in green.

pig had developed a severe infection and had to be euthanized thereby terminating the experiment. Figure 2-27 shows the infected area on the pig. The discolored skin is narcotizing tissue and pus with the implant being exposed on the left side of the incision.

It is likely that this infection is due primarily to the shape of the implant. Because the pig's skin is not extremely malleable, placing a semi-rigid object under an otherwise flat surface and covering it with a skin will form a lump on the surface of the skin. This deformation causes air and fluid pockets to form around the implant between the skin and skull as the device pushes up on the skin. The formation of the air pockets are shown in Figure 2-28. Such air and fluid pockets are not ideal for long term implants, as these regions are void of any vascular systems and therefore have no blood supply. Without blood perfusion, antibodies cannot reach these areas and fight any infection should they occur. By modifying the external shape of the implant to conform more closely to the skull and skin, these void pockets can be avoided thereby minimizing the risk of infection.

## 2.6 Data Analysis

In order to determine if the EEG is functioning as expected and producing plausible waveforms, the data was analyzed with a neurologist, Dr. Sydney Cash. Data is validated by looking at feature landmarks in both the time domain and frequency

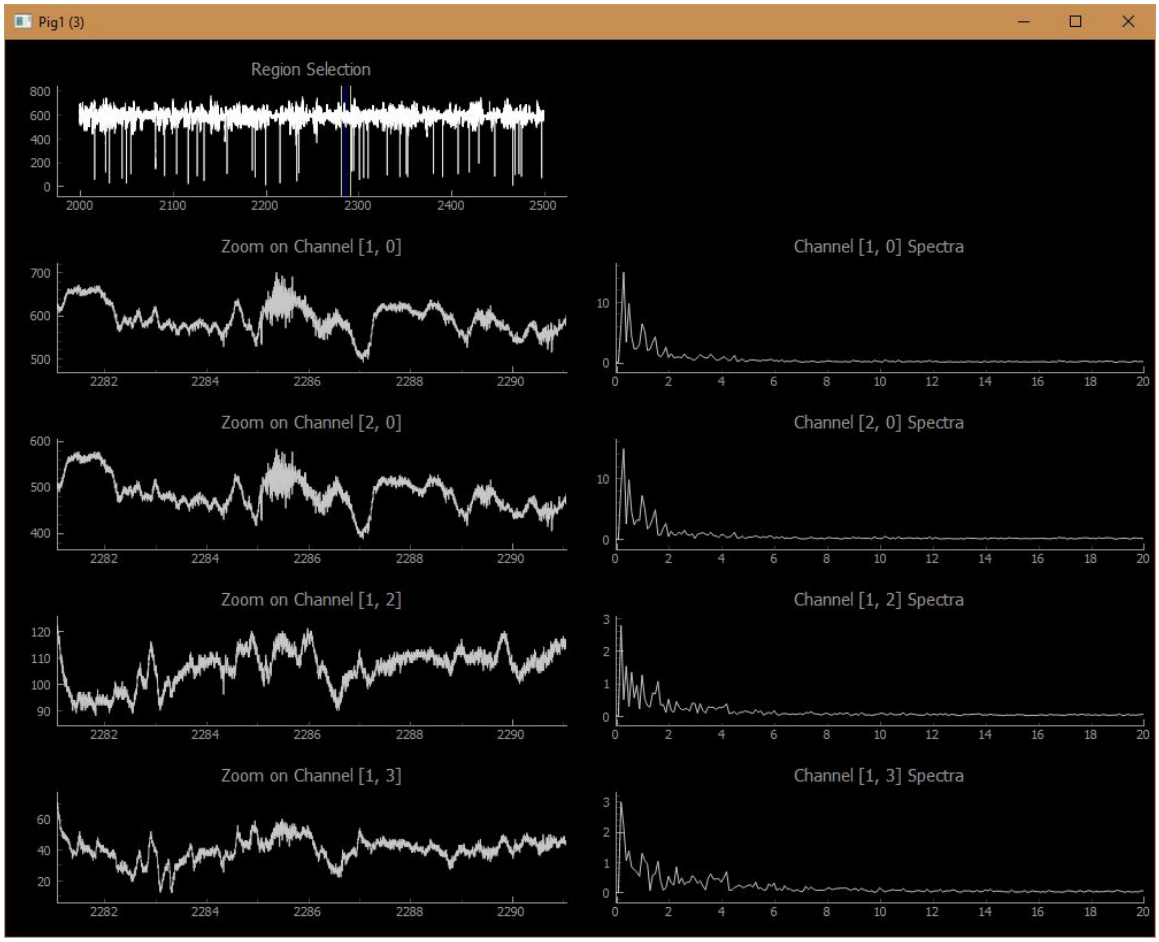


Figure 2-29: Bipolar Pig Waveforms. Top graph is a region selection. Middle two graphs are unipolar channel 1 and channel 2 plots and spectrums. Bottom two plots shows bipolar plots and spectrums between channel 1 and 2 and channel 1 and 3.

domain waveforms. For example, a pig under or just waking up from anesthesia is likely to have predominately low frequency ($<$ 8Hz) waveforms though, a precise correlation is difficult without a very detailed record or video of the pig's actions while being recorded. Because the device records EEG as a unipolar measurement (referential montage), the data must be first converted to a bipolar measurement by subtracting EEG channels from each other. This aids in waveform analysis and landmark identification by removing noise caused by the reference channel. Since unipolar measurements are all referenced to a single reference electrode, any noise from the reference electrode such, as muscle or eye movements, will produce artifacts in all the channels. Figure 2-29 shows both unipolar and bipolar plots. The middle two plots are unipolar plots for channel 1 and 2 and show almost identical waveforms and spectrums. The bottom two plots are bipolar plots between channels 1 and 2, and 1 and 3 respectively. These bipolar plots are computed by taking the difference between the unipolar measurements of the respective channels. It should be noted that the high frequency oscillation in the unipolar plot near time 2285s is likely caused by muscle interference and appears across all unipolar measurements; however, bipolar plots 1-2 and 1-3 eliminate this interference and look markedly different from each other and from the unipolar measurements.

# Chapter 3

# Wearable System

This section details the design of major components that comprise the wearable system. The design of this wearable EEG patch has the following functional requirements:

- Rapid application of sensors to access EEG from the frontal and prefrontal cortexes (forehead).
- Comfortable recording for at least 72 hours.
- No dependence with surrounding environment (e.g wires to a large module, dependence on outlets, sockets, etc.)
- Real time and historical data monitoring.
- Minimal EEG noise and interference from the environment.

## 3.1 System Overview

Figure 3-1 shows the system block diagram of the wearable EEG system. Prefabricated electrodes are connected to the ASIC described in Section 1.4.1 with the modifications detailed in Section 2.3.1. A microprocessor is directly connected to the ASIC in order to program and read data from the device and to transmit EEG data via BTLE to a receiving device described in Chapter 4.


Figure 3-1: Wearable System Block Diagram showing the main functional components of the patch EEG.

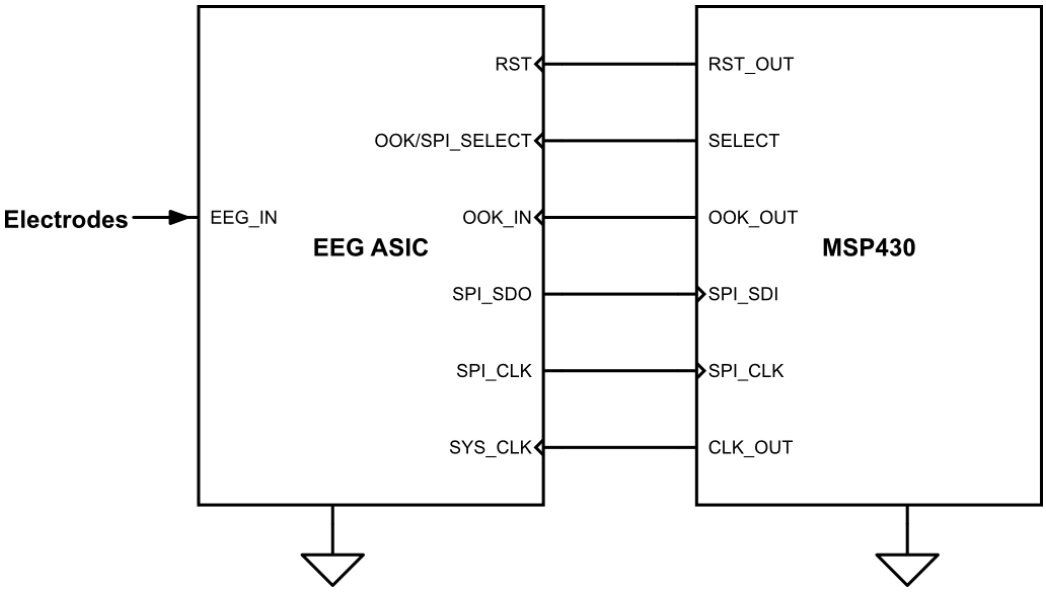


Figure 3-2: Schematic for Wearable ASIC Communication. ASIC and MSP430 are now electrically connected together and share common clocks.

## 3.2 Design

At a high level, the design of this system is remarkably similar to the subdermal design without the coupled coil communication; the same ASIC originally intended for the aforementioned subdermal EEG system is repurposed for this wearable design.

### 3.2.1 Electrical

#### Communication and Microprocessor

In order to program and receive data from the ASIC, the same MSP430 series microprocessor is used. Figure 3-2 shows the primary control signals between the ASIC and microprocessor. Because the ASIC is no longer physically separate from the mi-

croprocessor, it is not necessary to rely on the OOK communication scheme described in Chapter 2.3.1. Moving away from OOK has advantages in power consumption as detailed in Section 3.2.1.

The ASIC allows for SPI communication originally intended for bench-top testing in order to read and write data directly to the chip without using the OOK pins. This communication can be leveraged in order to reduce the processor requirements and simplify the overall system. It was originally intended that SPI or OOK communication could be selected between via an external digital pin on powerup shown as $OOK/SPI\_SELECT$ in Figure 3-2. However, an error in the chip disabled the SPI interface regardless of the state of the select pin. In order to circumvent this issue, it was discovered that data could be received using the SPI interface if the select pin was set after the ASIC is first programed using OOK via direct electrical connection to the pins. After SPI data is received by the microprocessor, the data is streamed over BTLE to the same base station used in the subdermal system.

In order to increase flexibility of the wearable system and reduce power consumption dynamically, the ASIC's main clock is sourced from the microprocessor directly instead of being set by an external resistor. By allowing the microprocessor to control the system clock of the ASIC, the microprocessor can change the sample rate of the ASIC in order to achieve higher bandwidth recordings or minimize power consumption depending on the needs of the user. The upper sapling rate of this system is only limited by the maximum data rate of the BTLE module, which for this particular radio is 88.9kBit/s. At 56 bits per sample for this 4-channel device, the maximum achievable sample rate is 1.58KHz.

### Power Consumption and Optimization

Figure 3-3 shows the power consumption of relevant blocks of the wearable EEG system. The device was measured using a benchtop power supply connected to an Agilent Technologies benchtop digital multimeter. The supply was set to 1.4V and connected to the input power terminal of the system. The system is sampling at 542Hz and consumes a total of 6.8mW.

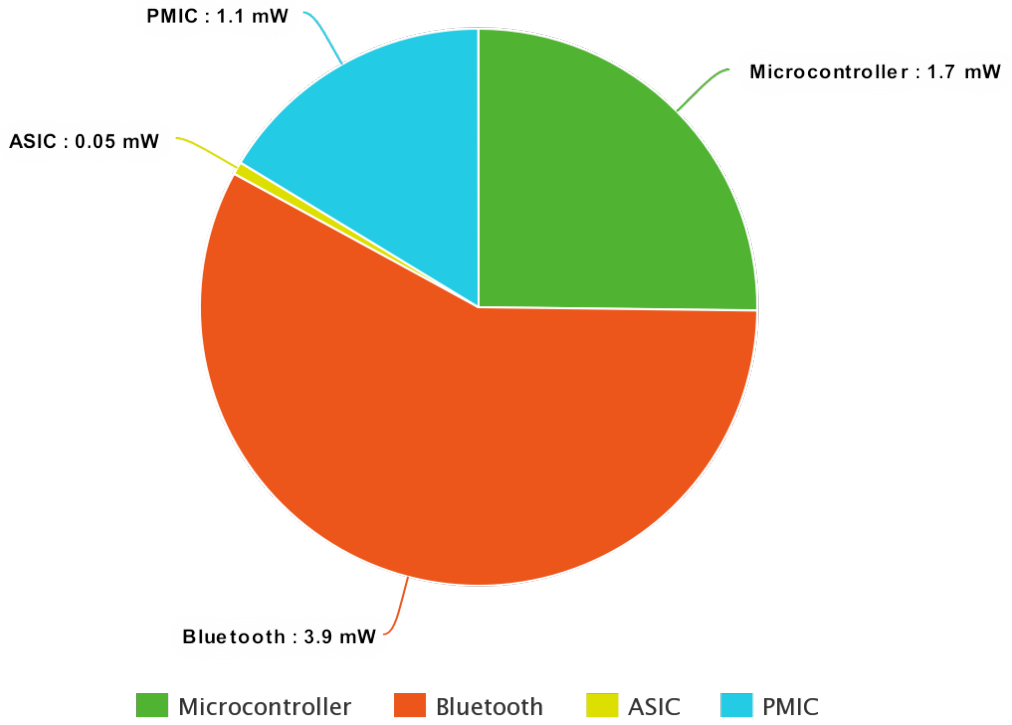


Figure 3-3: Wearable Power Consumption Breakdown. Measured at 1.4V at input to the system sampling at 542Hz. Total power consumption is 6.8mW.

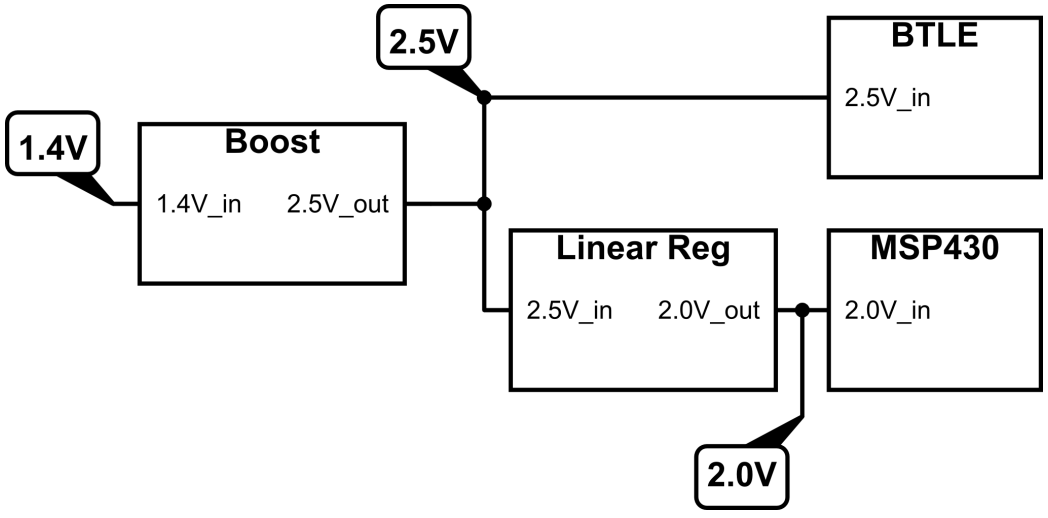


Figure 3-4: Diagram for power management on the wearable device showing locations and generation of all the relevant voltage domains.

It should be noted that when compared to table 2.2, the power consumption for analogous blocks are lower for the wearable design. This is due, in part, to the reduced supply voltage to the microcontroller and BTLE chips. For the subdermal design, the microcontroller and BTLE module in the external system ran primarily off of 3V. In this design, a 1.4V supply is boosted to 2.5V for the BTLE chip then regulated down to 2.0V using a linear regulator for the microcontroller. This setup is shown in Figure 3-4. The reduced supply voltage to these components reduces power consumption by approximately 15%.

Energy savings for the microcontroller is also due to the primary mode of communication to the ASIC no longer being OOK, but rather SPI. OOK communication is expensive in terms of processor resources as it requires core high clock frequencies to ensure adequate timing resolution for accurate decoding. By switching to SPI from OOK, the core clock frequency can be decreased from 20MHz to 6.67 MHz resulting in a proportional power savings.

Bluetooth communication power consumption was also decreased due to fewer bits being transferred. In the subdermal design, 8 channels of EEG were sampled at approximately 580Hz with 10 bits per channel with an 8-bit header totaling 88 bits per sample. For the wearable design, only 4 channels are needed; however, for this ASIC, SPI communications return 11 bits per channel per sample instead of 10 plus an 8-bit header giving 52 bits. The 52 bits must be rounded up to the next byte boundary to be packed evenly into 7 bytes, or 56 bits of data per sample. This reduction in data rate allows for the BTLE radio to remain in an idle state for longer durations, further reducing power consumption of the system.

### Battery Selection

Because of the reduced power requirements compared to the subdermal system, a wider range of battery sizes and types are able to be used for this application. In order to achieve a battery life greater than 72 hours, the battery must have a minimum capacity of approximately 0.5Wh. Maximum current consumption of this device is approximately 5mA, which is still too great for Li-$MnO_2$ based coin cells, but within

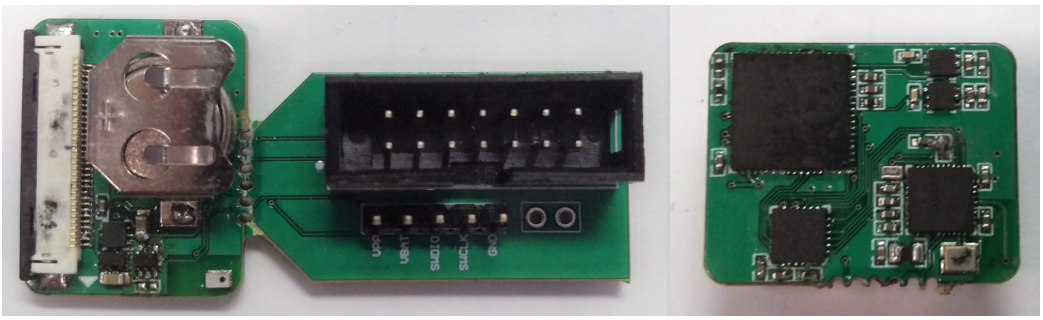

Figure 3-5: Wearable System PCBs. Programmer attached (left). Reverse side with no programmer (right).

the operating range of Zinc-Air based cells [23]. For this design, an A675 coin cell battery was selected. The battery measures 11.6mm in diameter and 5.4mm in height and as a capacity of 0.84Wh. It should be noted that compared to the lithium battery in the subdermal design, this battery requires a boost converter in order to be useful as its nominal output is only 1.4V.

### 3.2.2 Mechanical

#### PCB

In order to keep the board as small as possible, the board was designed with perforations to break off unused components. One half of the board contains all the relevant electronics (microcontroller, ASIC, BTLE, etc.), and the other half contains a programming interface in order to load firmware to the microcontroller and BTLE chips. In practice, programming only needs to happen during device assembly, thus the programmer can be safely removed by breaking the PCB along the perforation to minimize the overall size of the PCB. Figure 3-5 shows the final PCB for the wearable design - the left shows a board with the programming section still attached and the right is the reversed side a finished board showing the ASIC, BTLE module and microcontroller. Without the programmer, the final board measures 20mm x 24mm with a height of 9mm. Because of the relatively small size of the fabricated PCB, the device can be mounted entirely on the user's head without external wiring when recording EEG.

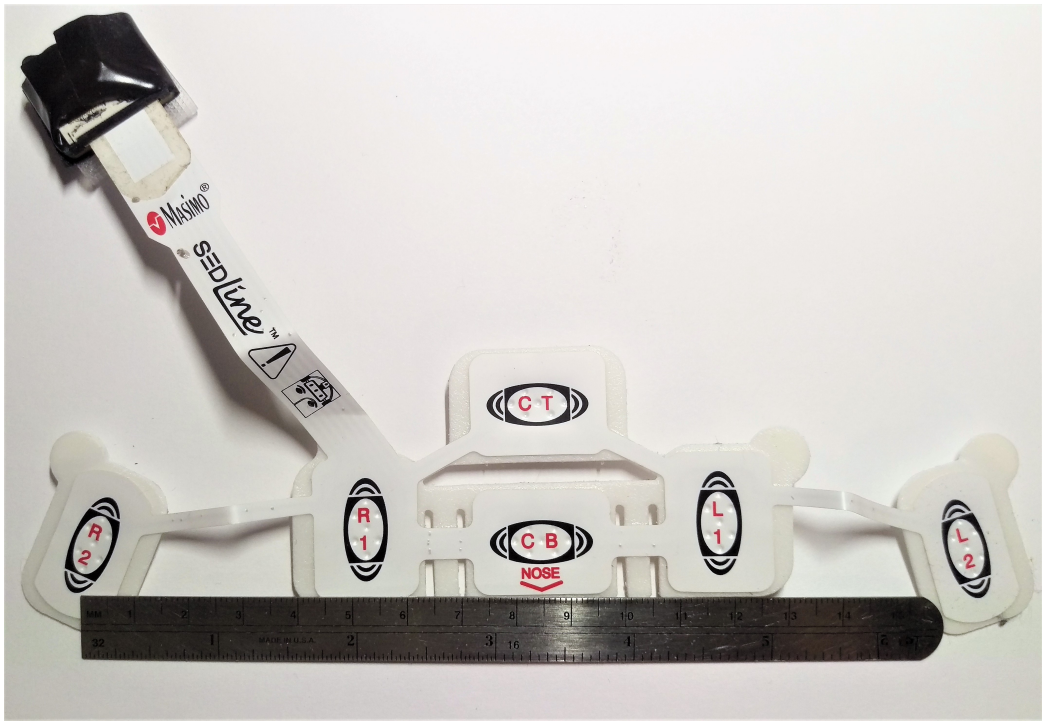


Figure 3-6: Masimo SedLine Patch.

### Electrodes

In order to interface with the body, a commercially available EEG adhesive electrode array is used. The SedLine patch by Masimo is a 4 channel (with ground and reference) EEG patch that is specifically designed to monitor sedation levels in the ICU when combined with their acquisition system. The electrodes are designed to adhere to the patient's forehead and give access to the F7, F8, Fp1 and Fp2 electrode locations shown in Figure 1-3. The patch is shown with electronics package attached in Figure 3-6. The two electrodes on each side of the patch form the 4 electrodes; ground and reference are the contacts in the middle.

## 3.3 Testing

### 3.3.1 Sleep

In order to verify the functionality of the wearable design, the device was worn by the author for 24 hours. Figure 3-7 shows the wearable device mounted on the author's

Figure 3-7: Wearable System Testing. Electronics package in black is adhered on the temple.

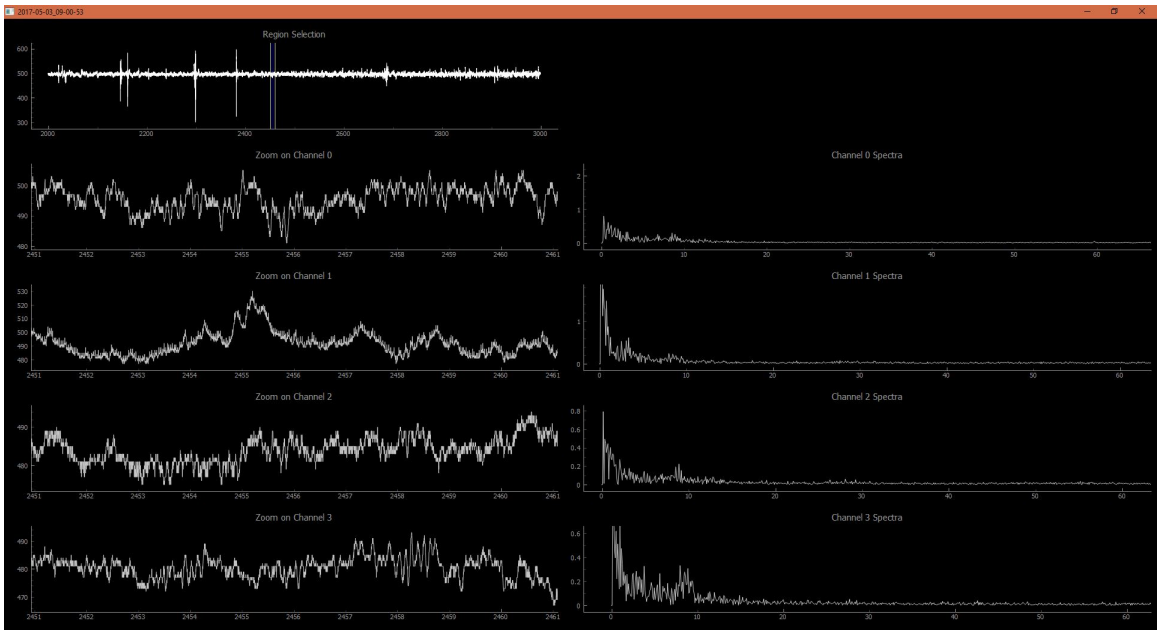


Figure 3-8: Wearable System Unipolar Plots. Left column shows unipolar time domain data from 4 channels. Right column shows power spectrum per channel.

forehead with the electronics package mounted on at the temple with adhesives. The device is self-contained without any wires. In order to capture and read the data, the wearable device is connected to a battery powered RaspberryPi base station via BTLE. The RaspberryPi both stores and uploads the data to the web which allows the user to roam freely. This base station is discussed in more detail in Chapter 4. The device was worn throughout the day while going through daily activities as well as sleeping through the night. EEG measurements are taken as a unipolar measurement (in relation to the reference electrode in the center) rather than a bipolar measurement. Figure 3-8 shows a sample EEG waveform from the wearable device. By leaving the device on and connected to the base station but not worn, the device battery lasted 97 hours.

### 3.3.2 Comparison to Conventional EEG Systems

To determine the quality of data that the wearable design produces, the device was tested alongside a conventional EEG system. The wearable device was placed on the author's head and conventional EEG electrodes placed adjacent to the adhesive

Figure 3-9: Simultaneous Recording of Wearable System with Conventional EEG Systems

electrodes of the patch. Figure 3-9 shows the test setup. It should be noted that although the electrode placement between the conventional and wearable EEGs are not precisely the same, correlation data is still valid due to EEG being an aggregate measurement of brain activity in that region with relatively low spatial resolution. Simultaneous EEG was recorded for approximately 30 minutes with the author wearing both four channels of conventional EEG and the wearable EEG described in this work. The resulting waveforms were visually inspected and found to share many of the same features.

## 3.4 Data Analysis

Similar to the subdermal system, the EEG data gathered during sleep for the wearable EEG was analyzed by Dr. Sydney Cash. Data is first converted to bipolar measurements and resulting waveforms are plotted in 10 second segments. This is

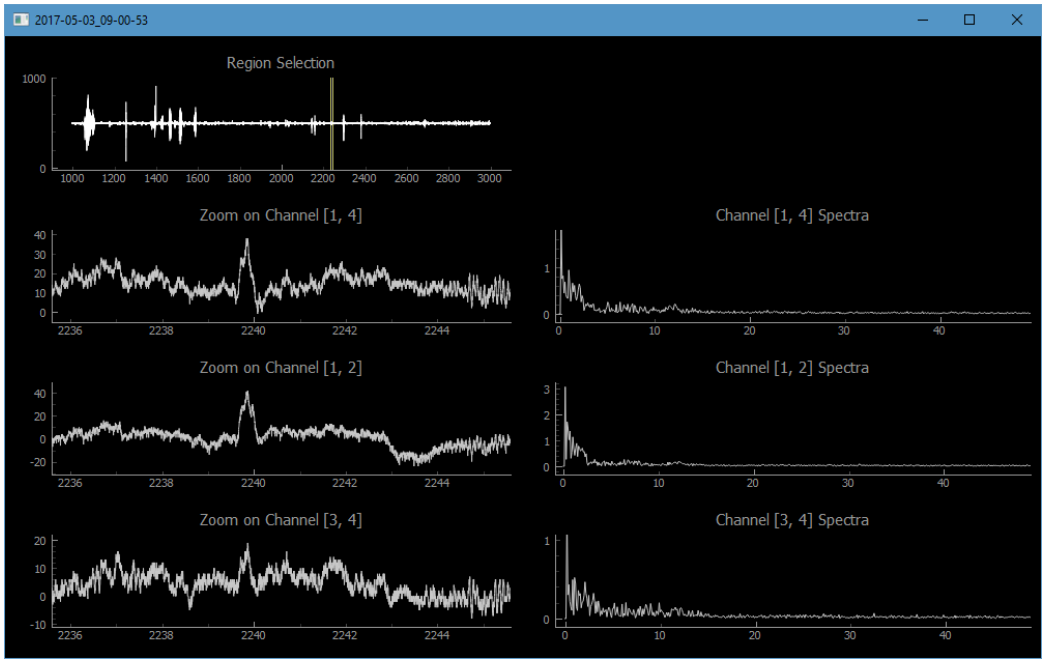


Figure 3-10: Wearable System Bipolar Plots. Plots shows a possible K-complex at $t = 2240s$ indicating sleep.

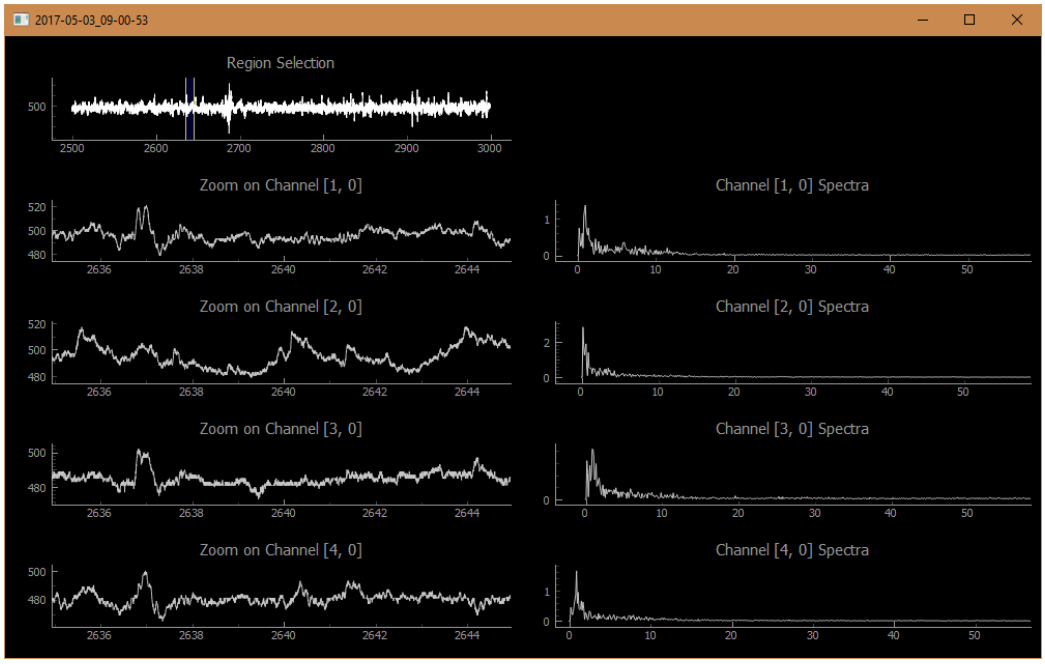


Figure 3-11: Wearable System Unipolar Plots. Plots shows a possible K-complex at $t = 2637s$ and a possible sleep spindle at $t = 2641s$.

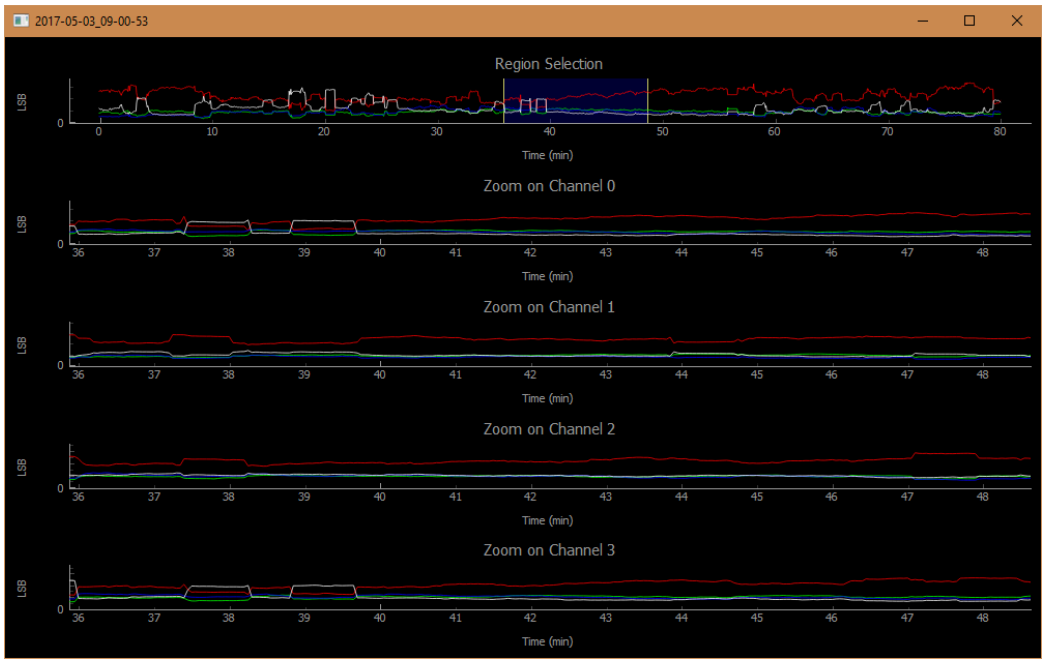


Figure 3-12: Normalized EEG Power Bins Over Time. 50 second FFT was taken every 2 seconds of data. The red, green, blue, and white traces correspond to the power in Delta, Theta, Alpha, and Beta EEG frequency bands respectively.

shown on the left side plots of Figure 3-10. Here, the plot shows a large transient at $t = 2240s$ which could be a K-complex landmark of non-REM sleep. Data was also inspected in a referential montage, or unipolar measurements. Figure 3-11 shows a unipolar plot with possible non-REM K-complexes and sleep spindles at $t = 2637s$ and $t = 2641s$ respectively.

It is known that EEG frequency decreases when entering sleep and then cycles from low to high frequencies as the patient traverses through sleep states. This idea can be used as an additional method to validate the performance of the wearable design. In order to extract EEG frequency band information, an FFT of the EEG data was taken over a 50 second rectangular window. The resulting curve is integrated between the frequencies corresponding to each EEG band in order to get the power in each EEG band. This FFT and integration process is repeated every 2 seconds of data and the resulting power bins are normalized to the total power and plotted over time. Figure 3-12 plots the normalized relative power in each of the EEG frequency bands over time. Code for this analysis is included in Appendix B.

It is difficult to draw conclusions from this analysis since the wearable EEG system does not include any motion artifact rejection; hence, has a high level of contamination in the time domain waveforms. These contaminants make the FFT conversion unreliable and difficult to observe frequency changes as evidenced by the step transitions in the normalized power bin plots.

# Chapter 4

# Base Station

This section discusses the hardware and software design pertaining to the EEG base station. This base station acts as a mobile bridge between the BTLE enabled EEG system (both the subdermal and the wearable designs) and the outside world. The base station has the following functional requirements:

- Communicate with either the subdermal or wearable EEG system wirelessly via BTLE.
- Stream EEG data live to multiple remote clients simultaneously.
- Store historical EEG data either locally or on the cloud such that it can be retrieved at a later time for analysis.
- Ability to run completely wirelessly without a power source for at least an hour while recording data, and stream data if an Internet connection is available.
- Require little or no intervention to set up and deploy.

## 4.1 Hardware

### 4.1.1 Wireless Receiver

Although both the subdermal and wearable devices utilize BTLE to exchange data, the BTLE device used in both EEG systems utilize a custom stack and protocol. This

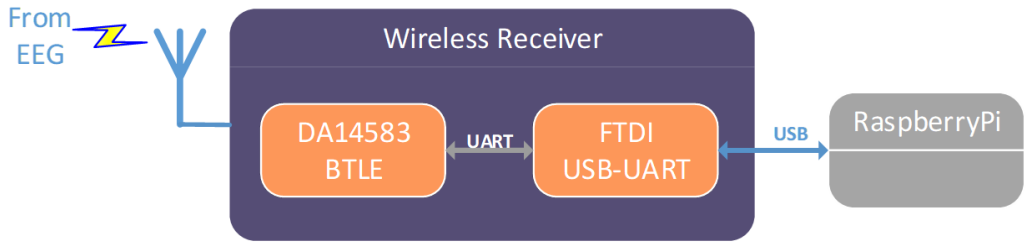


Figure 4-1: Wireless Receiver System Block Diagram. EEG is wirelessly received by BTLE that is connected to a RaspberryPi via a FTDI USB-UART bridge.

custom stack allows for only BTLE devices running the same stack and protocol to be able to communicate with the EEG systems, such as phones or computers with specialized hardware, unlike the previous generation EEG systems where any generic Bluetooth receiver could pair with the EEG. In order to increase future flexibility to add features later and decrease development times, a custom hardware 'dongle' was developed to allow any computer to directly connect to the EEG systems. The dongle utilizes the same DA14583 BTLE chip running the analogous EEG hardware stack in order to communicate with the EEG systems. Communication with the BTLE chip is achieved with UART using a FTDI USB-UART bridge. This allows for any computer with a USB port to access and address the EEG systems via virtual serial port emulation over USB. Figure 4-1 depicts the aforementioned data flow of the dongle.

Figure 4-2 shows the assembled dongle. The dongle fits directly into a full size USB port without external cabling and draws power directly from the USB port. The dongle measures 30mm x 15mm with a height of 4.5mm. When attached to a USB port, the dongle extends 17mm out of the port.

### 4.1.2 Raspberry Pi

Single board computers (SBC) are simple, power efficient computers on a single PCB, typically geared for fulfilling a specific task with limited hardware resources. For this design, a RaspberryPi 3 is used in order to easily store and stream EEG data in a compact form factor. The RaspberryPi 3 runs a Unix environment which allows

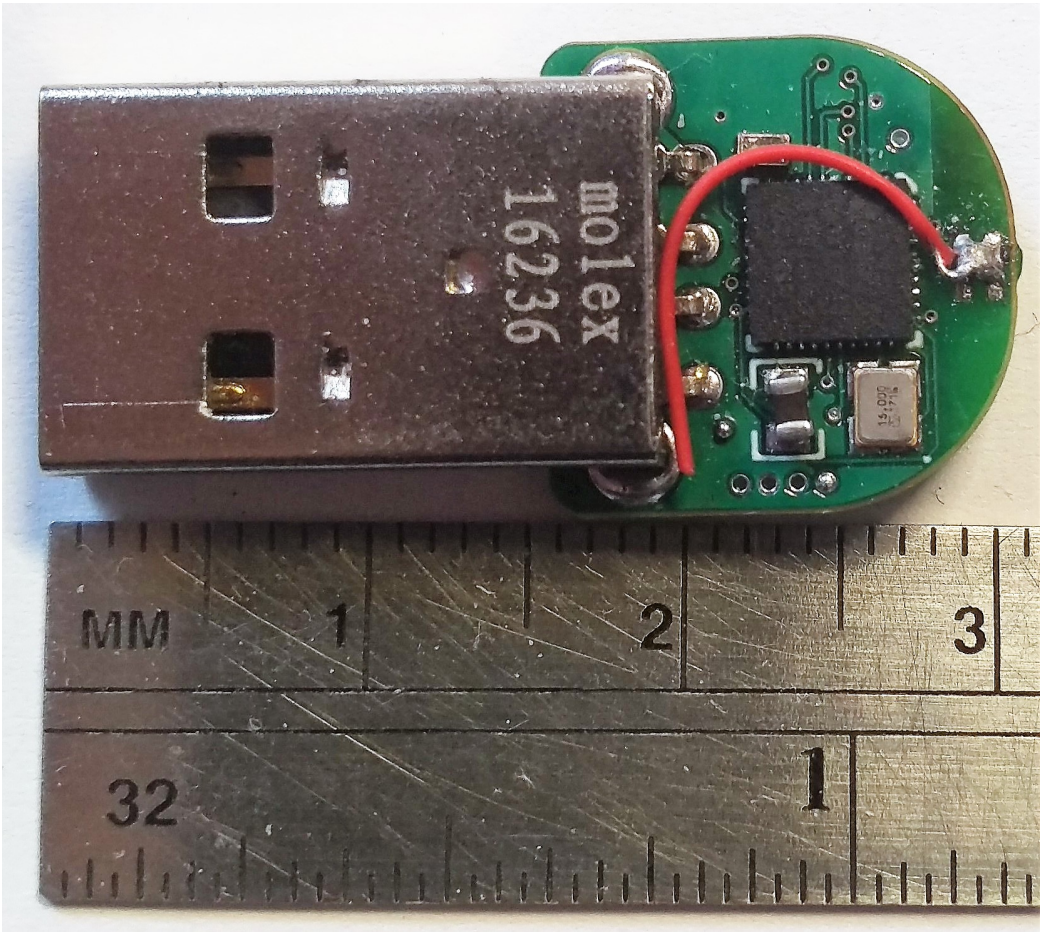


Figure 4-2: Dongle PCB.

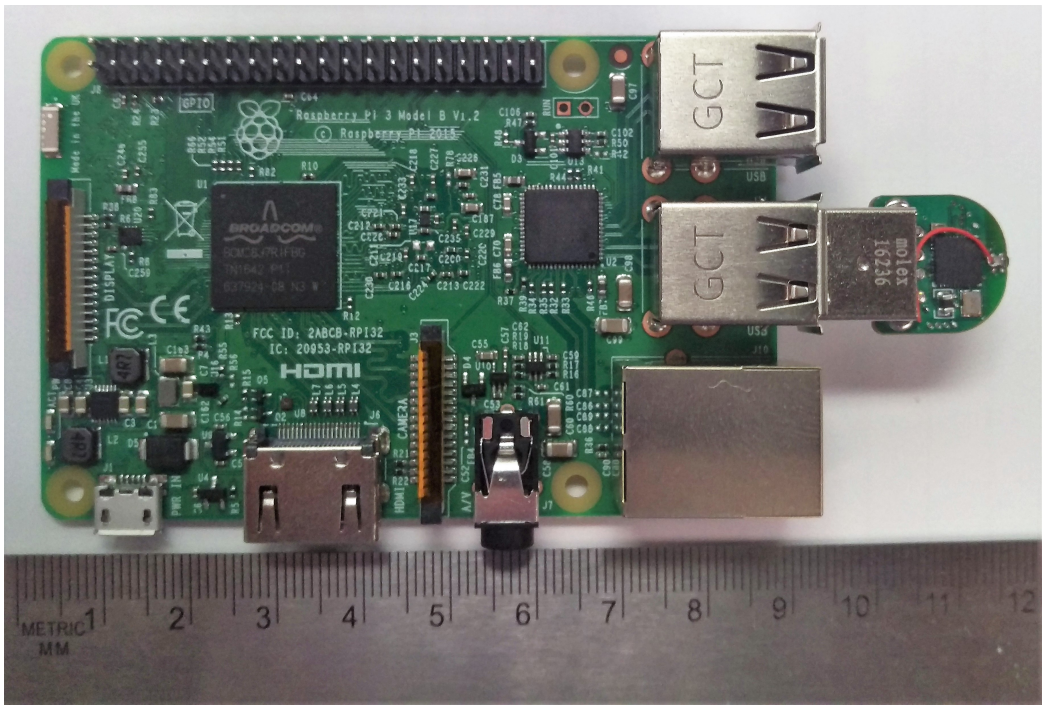


Figure 4-3: RaspberryPi with Dongle. Dongle is attached to the USB port on the right.

for programming in a wide range of languages and scripting to automate setup of the EEG system. It also has onboard Wi-Fi to allow for remote access as well as data streaming and upload without being tethered in place. For local storage, a 16GB microSD card is used, which, combined with cloud storage, gives a virtually infinite space to store and recall historical EEG data. The RaspberryPi consumes an average of 3W which can be powered off of a battery for portability. An 11.8Wh 18600 lithium-ion rechargeable battery is used; powering the RaspberryPi for almost 4 hours. Figure 4-3 shows the RaspberryPi, battery, and dongle attached to the USB port. With this dongle, the EEG system - either subdermal or wearable - must only remain within 5 meters of this portable base station in order to ensure data is being recorded. The base station itself can remain portable as long as the batteries have charge.

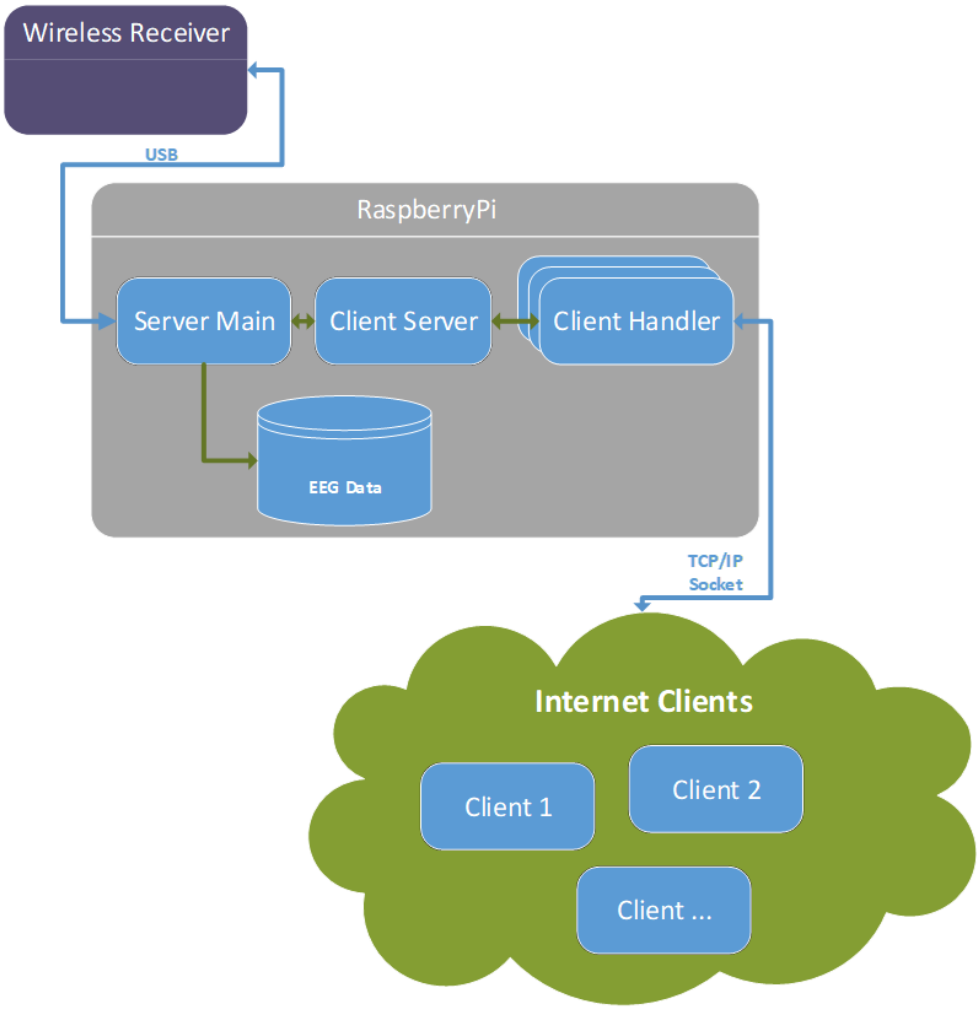


Figure 4-4: Base Station Software System Diagram.

## 4.2 Software

The RaspberryPi software and the remote clients are written in Python for operating system independence such that any computer can access the data, or any computer can take the place of the RaspberryPi as the base station. Software is split into two parts - a single 'server' instance that runs on the SBC that handles direct communication with the EEG system via the BTLE dongle, and any number of independent 'client' instances that allows for remote viewing of the real-time data. These parts are shown in the grey and green sections in Figure 4-4 respectively. The following code can be found in Appendices C, D, E, and F.

### 4.2.1 Server - RaspberryPi

The RaspberryPi is set up to run the Python server program on boot without any intervention after running several initialization scripts. The main Python program is broken up into three threads. These threads allow for asynchronous events to occur without interrupting data flow. This program is diagrammed in the grey section of Figure 4-4.

#### Main

The main thread is responsible for all of the low-level data transfers between the BTLE hardware dongle and the client handling software instances. This thread spawns the web handling thread as well as any support threads such as an error logging thread as well as the thread that records data to the disk. The main loop of this thread operates by reading in raw EEG data from the UART buffer then forwards the data into two queues: one queue to be committed to disk and another queue that feeds into all the client threads when clients are connected. This is the first thread to be spawned and typically last thread to be closed. After all the child threads and the main thread has exited, the program will upload all the data collected to Dropbox for storage and analysis.

#### Client Listener

This thread is started by the Main thread and is responsible for monitoring the TCP/IP socket it is bound to (port 10000 in our case). When a connection is initiated, this program will instantiate and hand-off control to a new instance of the Client Handler thread in order to start transmitting data. This thread also contains a list of all currently connected clients such that data provided by the main thread is copied to all of the Client Handler's queues.

#### Client Handler

The client handler object is instantiated by the Client Listener thread each time a new connection is initialized by a remote computer. The handler is comprised by two separate threads: an outbound thread that monitors a queue and forwards any data placed in the queue by the main thread to the remote computer, and an inbound thread that monitors the socket for any data, parse, and performs any necessary action. This scheme is used since any incoming or outgoing transmission can happen at any moment asynchronously from each other and the main thread. This allows for any number of clients to be communicating with the Python server without dropping data or causing unnecessary delays in transmission.

### 4.2.2 Networking

It was found that due to hospital firewalls and security, incoming TCP/IP socket connections could not connect to the base station; thus, real-time viewing of the data and verification of the EEG system operation was impossible. In order to circumnavigate the network address translation (NAT) firewall, the RasperryPi employs a persistent reverse-SSH tunnel to a non firewalled computer on the MIT campus. Using this scheme, the base station initiates an outgoing connection and creates a tunnel to the MIT computer for network traffic to be passed through back to the RaspberryPi. The MIT computer is set to forward network traffic through the tunnel and the outside world. Thus, in order to communicate with the base station, a user need only to

access the MIT based computer, which has a fixed address, and a connection will be established to the RaspberryPi regardless of network location of the RaspberryPi.

### 4.2.3 Client - Remote Computers

In order to remotely view the real-time data from the EEG system, a client program is written in Python, again, for portability and operating system independence. The client viewer initiates a socket connection with the MIT computer which forwards the request to the base station via the reverse-SSH tunnel. Once the socket connection is established, the client can remotely send commands to the EEG system and receive EEG data. In order to plot the data, the client program must first identify the packet header, 0xAA in our case, then parse the following 80 bits of data for the subdermal device, or 48 bits for wearable device, into the individual channels of EEG. The program then plots the data using the PyQTGraph Python package.

### 4.2.4 Summary

Overall, this base station achieves a flexible platform to easily record EEG data from any EEG system that utilizes the DA14583 custom stack for BTLE communication. The base station is set to automatically connect to both the Internet when available and any nearby EEG system without any intervention. This design allows for real time EEG data collection and monitoring from anywhere in the world while maintaining a mobile form-factor without any wires or cables to the EEG system.

# Chapter 5

# Conclusions

This thesis demonstrates the design, fabrication, and testing of an improved system for two ambulatory EEG applications. The first consists of electrical and mechanical optimizations to a subdermal EEG system, allowing for easier deployment and survival of the subdermal device. The second is a design that repurposes the EEG ASIC for a miniaturized wearable patch for sleep quality measurement. These two systems are one step closer in a design for an optimal miniaturized ambulatory EEG system. They offer clinicians and researchers more insight into the functionality of the brain as well as streamlined tools to provide more accurate diagnoses.

## 5.1 Summary of Contributions

In summary, this work presents the following contributions:

- **A subdermal EEG design.** This device includes an optimized energy transfer and communication scheme as well as further miniaturization of the device. the implanted device is roughly a factor of 5 smaller than the previous implantable device. Animal testing using a swine model was performed at MGH.

- **An improved device to communicate with the subdermal EEG.** This device includes optimized circuits to more efficiently provide power and decode data from the implant. Improvements to the mechanical stability of the device

were also achieved. Power consumption is reduced by almost a factor of 10 and PCB area reduced by a factor of 4 when compared to the previous design.

- **A wireless, low power EEG patch for sleep studies.** This device communicates with a mobile base station to give doctors a streamlined way of accessing EEG data in real time.

## 5.2 Future Work

Several other optimizations are worth exploring:

- This design revolves around the AFE of a previously published ASIC; however, this ASIC contains a seizure detection algorithm that is not needed. Reworking the silicon in order to remove this could reduce the ASIC size and further system miniaturizations can be realized.

- Although the bond diagram of the ASIC could be made to fit in a 64-pin QFN, additional silicon changes could be done in order to balance the bond diagram even further.

- The digital core of the ASIC simply serializes the EEG data from the channels and appends a header to the data stream in order to transmit it to the external device via OOK. Communication to and from the ASIC when implanted could be improved through some error detection or correction coding scheme.

- The ASIC requires three separate power sources; two 0.9V sources, and a 1.8V source. However, most off chip, on board regulators to generate these low voltages have a quiescent current that is roughly the same or exceeds the current consumption of the entire ASIC. By integrating the power management on the die of the ASIC, power efficiency could increase significantly.

- The implanted design relies on capacitors for its backup power source when the external power coil is removed. These capacitors can power the device for a few milliseconds, which is enough for slight movements between the implant

and external device, but animal testing revealed that longer durations where the implant is self-powered without triggering a reset is needed. Placing larger capacitors or optimizing the power consumption of the downstream electronics to allow for the implant to be self-powered for a few seconds would improve stability of the implanted system.

- The external device that powers the implant has no feedback that measures the power output of the RF field. As the coils of the subdermal system are moved closer together, the voltage at the implant decreases due to the higher coupling coefficient. However, the voltage also drops due to the larger reflected load shorting the external side RF field. This gives the external device limited range from the implant. By employing feedback in order to make sure the external device RF amplitude is constant or at least pseudo-constant, the system can be made more robust to external device positioning fluctuations. This can be done through envelope detection of the class-D amplifier to generate a control voltage, then feeding this control voltage back into the buck converter to increase or decrease the voltage to the class-D switching elements. OOK data from the implant can be recovered by monitoring this control voltage rather than the class-D amplitude directly.

- The EEG systems do not include an algorithm to eliminate motion artifact or other types of interference. By designing such an algorithm or method of post-processing the data, more accurate EEG interpretations can be extracted.

# Appendix A

# ExternalDeviceMain.c

Firmware for External device.

Associated libraries can provided by emailing jasony@mit.edu

```
#include "deviceConfig.h"
#include "OOK.H"
#include "sleep.h"
#include "UCS.h"
#include "UART.h"

#define COIL_RX_TIMEOUT 1000u
#define TIMER_OFFSET 0
#define DEVICE_ID 0x03

static volatile char coilRXByte;
static volatile char coilRXShiftCounter;
static volatile char coilRXByteCounter;
static volatile char statusRegister;
static volatile char TXPayload[5];
static volatile char TXPayloadCounter;

static char ECHO_CMD[4] = {0xBB, 0XBB, 0XBB, DEVICE_ID };
static char WRITE_LNA_CURR[5] = {0xAA, 0xE8, 0xC0, 0x00, 0x03};
static char READ_LNA_CURR[5] = {0xAA, 0x68, 0xC0, 0x00, 0x03};
static char WRITE_LOAD_CAP[5] = {0xAA, 0xE9, 0x00, 0x00, 0x03};
static char READ_LOAD_CAP[5] = {0xAA, 0x69, 0x00, 0x00, 0x03};
static char WRITE_VGA_GAIN[5] = {0xAA, 0xE9, 0x40, 0x00, 0x02};
static char READ_VGA_GAIN[5] = {0xAA, 0x69, 0x40, 0x00, 0x02};
static char WRITE_VGA_CURR[5] = {0xAA, 0xE9, 0x80, 0x00, 0x03};
static char READ_VGA_CURR[5] = {0xAA, 0x69, 0x80, 0x00, 0x03};
static char WRITE_LPF_CURR[5] = {0xAA, 0xE9, 0xC0, 0x00, 0x03};
static char READ_LPF_CURR[5] = {0xAA, 0x69, 0xC0, 0x00, 0x03};
static char EEG_START_TRAN[5] = {0xAA, 0xFF, 0xC0, 0x00, 0x00};

void EEGSetup();
void coilRXSetup();
void DA14583Setup();
void classDSetup();
void startD();
void startEEG();
void startCoilRX();
void stopCoilRX();

/*
```

```
 * main.c
 */
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;    // Stop watchdog timer

        setDCO();                                           // Init DCO
        DA14583Setup();                                     // Init DA14583
        UARTSetup();                                        // Init UART
        OOKSetup(90, 30, LPM0_bits);                        // Init OOK
        coilRXSetup();                                      // Init CoilRx
        classDSetup();                                      // Init ClassD

        _bis_SR_register(LPM0_bits + GIE);       //enter LPMx
        // Wait here, system runs on interrupts

        return 0;
}

void DA14583Setup()      // Init BTLE
{
        PJDIR |= BIT3;  // Reset DA14583
        PJOUT |= BIT3;
        longSleep(100, LPM0_bits);
        PJOUT &= ~BIT3;

        P1DIR &= ~BIT7;
        longSleep(1000, LPM0_bits);

}

// setup eeg ASIC
void EEGSetup()
{
        longSleep(1000, LPM0_bits);              //device stabilization

        OOKSendArray(WRITE_LNA_CURR, 5);
        longSleep(600, LPM0_bits);

        OOKSendArray(WRITE_LOAD_CAP, 5);
        longSleep(600, LPM0_bits);

        OOKSendArray(WRITE_VGA_GAIN, 5);
        longSleep(600, LPM0_bits);

        OOKSendArray(WRITE_VGA_CURR, 5);
        longSleep(600, LPM0_bits);

        OOKSendArray(WRITE_LPF_CURR, 5);
        longSleep(600, LPM0_bits);
}

// Start ASIC trans
void startEEG()
{
        OOKSendArray(EEG_START_TRAN, 5);
}

// Setup class D
void classDSetup()
{
        TA1CCR0 = 4;
        TA1CCTL1 = OUTMOD_2;
        TA1CCTL2 = OUTMOD_6;
        TA1CCR1  = 2;
        TA1CCR2 = 2;
        TA1CTL |= TASSEL_2 + MC_3;        //TA1 SMCLK, updown
```

```
        P1DIR |= BIT2 + BIT3;
        P1OUT &= ~(BIT2 + BIT3);
        // P1SEL0 |= BIT2 + BIT3;
}

// Start RF output
void startD()
{
        P1SEL0 |= BIT2 + BIT3;
}

// Stop RF output
void stopD()
{
        P1SEL0 &= ~(BIT2 + BIT3);
}

// Setup ISR for data input
void coilRXSetup()
{
        TA0CCTL1 = CM_2 + SCS + CAP + CCIE;
        TA0CCTL2 = CM_1 + SCS + CAP + CCIE;
        TA0CCTL0 = CCIE;
        TA0CCR0 = COIL_RX_TIMEOUT;
}

// Start ISR for data
void startCoilRX()
{
        coilRXByte = 0x00;              // reset shift reg
        coilRXShiftCounter = 9;         // reset SM

        P1DIR &= ~(BIT0 + BIT1);        // set inputs
        P1SEL0 |= BIT0 + BIT1;          // set secondary func.
        // setup timer, SMCLK, continuous, reset, interrupt
        TA0CTL = TASSEL_2 + MC_2 + TACLR + TAIE;
}

// Stop ISR for data
void stopCoilRX()
{
        TA0CTL = 0;
        P1SEL0 &= ~(BIT0 + BIT1);
}

// OOK actions
void OOKHighAction()
{
        P1SEL0 |= BIT2 + BIT3;
}

void OOKLowAction()
{
        P1SEL0 &= ~(BIT2 + BIT3);
}

void OOKPreAction()
{
        if(TA0CTL != 0)
                statusRegister |= BIT7;
        stopCoilRX();
        OOKLowAction();
        sleep(90, LPM0_bits);
}

void OOKPostAction()
{
        OOKHighAction();
```

```
        if(statusRegister & BIT7)
                startCoilRX();
}

// Executed each time a command comes from BTLE
void UARTReceiveAction(unsigned char byte)
{
        _no_operation();
//      UARTSendByte(byte);
//      UARTSendByte(statusRegister);

        if(statusRegister & BIT1)
        {
                if(statusRegister & BIT0)
                {
                        TXPayload[TXPayloadCounter] = byte;
                        TXPayloadCounter ++;
                        if(TXPayloadCounter == 5)
                        {
                                TXPayloadCounter = 0;
                                OOKSendArray(TXPayload, 5);
                                statusRegister &= ~(BIT0 + BIT1);
                        }
                }

                switch(byte)
                {
                        case 0xBB:      //Repeat Header
                                return;
                        case 0xF0:      // Echo
                                UARTSendArray(ECHO_CMD, 4);
                                break;
                        case 0xF1:      // Turn on Class D
                                startD();
                                break;
                        case 0xF2:      // Turn off Class D
                                stopD();
                                break;
                        case 0xF3:      // Enable coil RX ISR
                                startCoilRX();
                                break;
                        case 0xF4:      // Disable coil RX ISR
                                stopCoilRX();
                                statusRegister  &= ~BIT7;
                                break;
                        case 0xF5:      // Reset System (turns on class D)
                                stopCoilRX();
                                stopD();
                                longSleep(2000, LPM0_bits);
                                startD();
                                statusRegister  &= ~BIT7;
                                break;
                        case 0xF6:      // sends default cfg to EEG
                                EEGSetup();
                                break;
                        case 0xF7:      // starts EEG
                                startEEG();
                                break;
                        case 0xF8:      // send cmd to EEG (next 5 bytes)
                                statusRegister |= BIT0;
                                break;
                }
                UARTSendByte(byte);

                if(!(statusRegister & BIT0))
                        statusRegister &= ~BIT1;
        }
```

```
        else
                if(byte == 0xBB)
                        statusRegister |= BIT1;
}

// ISR to decode OOK data
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer0_A0(void)
{
        // no data for a long time, reset SM
        coilRXByte = 0x00;
        coilRXShiftCounter = 9;
}

#pragma vector=TIMER0_A1_VECTOR
__interrupt void Timer0_A1(void)
{
        switch(TA0IV)
        {
                case 0:
                        break;

                case 2:         // TA0CCR1 - rising edge capture OOK

                        _bis_SR_register(GIE);
                        TA0CTL |= TACLR;                // reset timer
                        coilRXByte <<= 1;               // rotate

                        if (TA0CCR2 > TA0CCR1 - TA0CCR2 - TIMER_OFFSET)
                                coilRXByte |= BIT0;

                        coilRXShiftCounter--;
                        if (coilRXShiftCounter == 0)
                        {
                                UARTSendByte(coilRXByte);
                                coilRXByteCounter++;
                                coilRXByte = 0x00;
                                coilRXShiftCounter = 8;
                        }
                        break;

                case 4:         // TA0CCR2 - falling capture
                        break;  // Dont need to do anything

                default:        // no data for a long time, reset SM
                        coilRXByte = 0x00;
                        coilRXShiftCounter = 9;
                        coilRXByteCounter = 0;
                        break;
        }
        return;
}
```

# Appendix B

# FFTPowerExtractor.py

Python program to parse and extract EEG power data from stored EEG waveforms.

Associated libraries can provided by emailing jasony@mit.edu

```
# program to extract FFT power bins from waveforms
#saves data as a CSV for future analysis

from pyqtgraph.Qt import QtGui, QtCore
import numpy as np
import pyqtgraph as pg
import scipy.fftpack as fft
import csv
import sys
import os

passcounter = 0
datacounter = 0
data = [[[]],[[]],[[]],[[]]]
buf = [[],[],[],[]]
datalength = 0

rate = 542.0
FFTSize = 50 #seconds
FFTRate = 2 #seconds
bins = [[0.5, 4], [4, 8], [8, 14], [14, 30]] #Hz

def init():
    print "starting"
    if (len(sys.argv) != 2):
        raise ValueError("invalid arguments provided")

    try:
        os.chdir(sys.argv[1])
    except:
        raise ValueError("Cannot open directory " + sys.argv[1])

    if (len(os.listdir('.')) == 0):
        raise ValueError("no files found")
    print "Found Files: "
    for x in os.listdir('.'):
        print "\t" + x


# extracts power bins (delta, theta etc.) for
```

```
# arbitrary data set
def getPowerBins(data):
    dlength = len(data)
    fdata = fft.fft(data)
    fdata = np.abs(fdata[0:dlength/2])/float(dlength)
    l = len(fdata)
    fxdata = np.linspace(0, rate/2, l)

    # integrate between boundaries
    output = []
    for b in bins:
        l = int((b[0] * dlength)/rate)
        u = int((b[1] * dlength)/rate)
        output.append(np.sum(fdata[l:u]))
    return output

def main():
    global buf
    print "Calculating"
    #open input file
    with open('output.csv') as f:
        d = csv.reader(f)
        d.next()
        #open output file
        with open('fftOutput.csv', 'wb') as csvfile:
            csvwriter = csv.writer(csvfile, dialect='excel')

            for row in d:
                if (len(row) == 4):
                    for i in range(4):
                        buf[i].append(int(row[i]))
                    # take a chunk for FFT
                    if (len(buf[0]) == rate*FFTSize):
                        payload = []
                        for i in range(4):
                            pBins = getPowerBins(buf[i])
                            stringLoad = ""
                            for j in pBins:
                                stringLoad = stringLoad + str(j) + "#"
                            # print stringLoad
                            payload.append(stringLoad[:-1])

                            #shift by time specified by FFTRate
                            buf[i] = buf[i][int(rate*FFTRate):]
                        csvwriter.writerow(payload)
                        # break

# do things
init()
main()
```

# Appendix C

# BaseStationServerMain.py

Python main server code for base stations and dongles.

Associated libraries can provided by emailing jasony@mit.edu

```
# main server instance for BaseStation

from threading import Lock
import threading
import serial
from ClientServer import ClientServer
from Logger import Logger
import ServerHelper
import time

class ServerMain(threading.Thread):

    #initialization
    def __init__ (self, CLIENT_PORT=10000, ADMIN_PORT=10001,
                                            UART='/dev/ttyAMA0', BAUD=115200):
        threading.Thread.__init__(self)

        startTime = time.strftime("%Y-%m-%d_%H-%M-%S", time.localtime())
        ServerHelper.printHeader(startTime)
        ServerHelper.printOut(self, "Init System")

        #connect to the EEG
        self.com = serial.Serial(port=None, baudrate=BAUD, timeout=1, rtscts=True)
        self.com.port=UART
        #create a new client server
        self.clientServer = ClientServer(CLIENT_PORT, self)
        self.logger = Logger(startTime)
        self.IS_ALIVE = True
        self.testCounter = 0

        ServerHelper.printOut(self, "Init Done")

    def __str__(self):
        return ""

    def run(self):
        #connect to the EEG
        try:
            self.com.open()
        except Exception:
```

```
            self.cleanup();
            return;

        #start the client server to accept connections
        self.clientServer.start()
        self.logger.start()

        ServerHelper.printOut(self, "Starting Mainloop")

        #mainloop
        while(self.IS_ALIVE):
            try:
                #read data from dongle
                data = self.readAllCOM()

                if(self.IS_ALIVE == False):
                    break;

                if(data == ""):
                    continue

                #write data to the connected clients and log
                self.logger.writeData(data)
                self.clientServer.writeClients(data)

            #bad things happned
            except Exception as e:
                print e
                break
            except KeyboardInterrupt:
                ServerHelper.printOut(self, "Interrupt Signal Caught")
                break


        self.cleanup()

    #cleanup
    def shutdown(self):
        ServerHelper.printOut(self, "Stopping Mainloop")
        self.IS_ALIVE = False

    def cleanup(self):
        try:
            ServerHelper.printOut(self, "Stopping Serial")
            self.com.close()
        except:
            ServerHelper.printError(self, "Cannot close Serial")

        try:
            ServerHelper.printOut(self, "Stopping Logger")
            self.logger.shutdown()
            self.logger.join()
        except:
            ServerHelper.printError(self, "Cannot close Logger")

        try:
            ServerHelper.printOut(self, "Stopping Clients")
            self.clientServer.shutdown()
            self.clientServer.join()
        except Exception as e:
            print e
            ServerHelper.printError(self, "Cannot close Clients")

        ServerHelper.printShutdown()

    #read all data from dongle
    def readAllCOM(self):
        size = self.com.inWaiting()
```

```
    if(size < 100):
        size = 100
    data = self.com.read(size)

    return data

#send data to dongle
def sendToCOM(self, data):
    self.com.write(data)
    self.com.flush()


def getDataTest(self, size):
    data = ""
    for i in range(500):
        data = data + str(self.testCounter)
        self.testCounter = self.testCounter + 1
        if self.testCounter > 9:
            self.testCounter = 0
    time.sleep(0.2)

    return data
```

# Appendix D

# ClientListener.py

Client Listener object for Base station server.

Associated libraries can provided by emailing jasony@mit.edu

```
# code to accept new socket connections over the Inet

import socket
import ServerHelper
from ClientHandler import ClientHandler
import threading
from threading import Lock
import copy

class ClientServer(threading.Thread):

        #Class Initialization
    def __init__(self, PORT, MAIN):
        threading.Thread.__init__(self)
        self.port = PORT
        self.masterThread = MAIN

        ServerHelper.printOut(self, "Initializing Server on %s" % self.port)
        #bind the port to the address
        self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_address = ('', PORT)
        self.socket.bind(server_address)
        self.socket.listen(10)

        #Structure to hold current users
        self.userList = {} #[port : connection]

        #concurrency locks
        self.userListLock = Lock()
        self.comLock = Lock()

        self.IS_ALIVE = True

    def __str__(self):
        return("[Client Server - %s]" % self.port)

    def run(self):
        ServerHelper.printOut(self, "Starting Server")

        #mainloop
```

```
    while(self.IS_ALIVE):

        connection = None
        clientAddress = None
        try:
            connection, clientAddress = self.socket.accept()
            # connection.settimeout(5)
        except socket.timeout:
            continue;

        if(self.IS_ALIVE == False):
            try:
                connection.close()
            except:
                pass
            break;

        #connection received
        ServerHelper.printOut(self, "Connection from %s" % str(clientAddress))

        try:
                # create a new client handler
            handler = ClientHandler(connection, clientAddress[1], self)
            handler.run()
            self.userList[clientAddress[1]] = handler
        except Exception as e:
                #bad things happned here
            print e
            ServerHelper.printError(self,"%s Connection Failed"%clientAddress[1])
            try:
                connection.close()
            except:
                ServerHelper.printError(self, "Could not close connection")


# removes a user from the server
def removeUser(self, key):
    self.userListLock.acquire()
    ServerHelper.printOut(self, "Kicking User %s" % key)
    try:
        self.userList.pop(key)
        # ServerHelper.printOut(self, "Removed User %s " % key)
    except Exception as e:
        print e
        ServerHelper.printError(self, "Cannot Remove User")
    finally:
        self.userListLock.release()

# writes data to all the connected clients
def writeClients(self, data):
    self.userListLock.acquire()
    try:
        for c in self.userList.values():
            c.putData(data)
    except Exception as e:
        print e
        ServerHelper.printError(self, "Write to Clients Failed")
    finally:
        self.userListLock.release()

# called when data needs to be sent to the EEG device
def readAction(self, data):
    self.comLock.acquire()
    try:
        self.masterThread.sendToCOM(data)
    finally:
        self.comLock.release()
#cleanup
```

```
def shutdown(self):
    ServerHelper.printOut(self, "Shutting down Server (10s)...")
    self.IS_ALIVE = False
    socket.socket(socket.AF_INET, socket.SOCK_STREAM)
              .connect(('localhost', self.port))

    self.socket.close()
    self.join()

    ServerHelper.printOut(self, "Stopping Clients...")
    try:
        self.userListLock.acquire()
        users = copy.copy(self.userList)
    finally:
        self.userListLock.release()

    for u in users:
        try:
            users[u].shutdown()
            # users[u].join()
        except Exception as e:
            print e
            ServerHelper.printError(self, "Cannot kill %s" % u)

    ServerHelper.printOut(self, "Shutdown Complete")
```

# Appendix E

# ClientHandler.py

Client Handler object for base station server.

Associated libraries can provided by emailing jasony@mit.edu

```
# Code to service a connected client to the server

import threading
import Queue as queue
import ServerHelper
import socket
# import ClientServer

class ClientHandler():

    def __init__(self, SOCKET, PORT, SERVER):

        # self.userList = USERLIST
        self.port = PORT
        ServerHelper.printOut(self, "Initializing Handler")
        self.server = SERVER
        self.dataQueue = queue.Queue()
        self.socket = SOCKET
        self.socketFile = self.socket.makefile(mode='rw', bufsize=-1)
        self.socketWriterThread = ClientWriter(self, self.socketFile, self.dataQueue)
        self.socketReaderThread = ClientReader(self, self.socket)
        self.IS_ALIVE = True

    def __str__(self):
        return("[Handler - " + str(self.port) + "]")

    def run(self):
        ServerHelper.printOut(self, "Starting Handler")
        self.socketWriterThread.start()
        self.socketReaderThread.start()

    def putData(self, data):
        try:
            self.dataQueue.put(data, False)
        except:
            ServerHelper.printError(self, "Cant Put Data")

    # cleanup
    def shutdown(self):
        try:
```

```
            ServerHelper.printOut(self, "Stopping Handler")
            self.IS_ALIVE = False
            self.dataQueue.put("killPill", False)
            try:
                self.socketWriterThread.join()
            except:
                pass

            try:
                self.socket.close()
                self.socketReaderThread.join()
            except:
                pass

            self.server.removeUser(self.port)
            # self.socketFile.close()
            ServerHelper.printOut(self, "Stopped Handler")

        except Exception as e:
            print e
            ServerHelper.printError(self, "SHUTDOWN FAILURE")

    # when getting a command from network...
    def receiveAction(self, data):
        try:
            self.server.readAction(data)
        except:
            ServerHelper.printError(self, "Cannot process: " + str(data))



# Class to read from buffer and write to network socket
class ClientWriter(threading.Thread):

    def __init__(self, handle, sockFile, dQueue):
        threading.Thread.__init__(self)

        self.handler = handle
        self.socketFile = sockFile
        self.dataQueue = dQueue

    def __str__(self):
        return str(self.handler) + " (Writer)"

    def run(self):
        while(self.handler.IS_ALIVE):

            data = ""
            try:
                data = self.dataQueue.get(True)
            except queue.Empty:
                pass

            if(self.handler.IS_ALIVE == False):
                break

            if (len(data) != 0):
                try:
                    self.socketFile.write(data)
                    self.socketFile.flush()
                except:
                    ServerHelper.printError(self, "Connection Lost")
                    self.handler.shutdown()
                    break

        ServerHelper.printOut(self, "OutputStream closed")
```

```
# class to read from network buffer and write to EEG
class ClientReader(threading.Thread):

    def __init__ (self, handle, sock):
        threading.Thread.__init__(self)

        self.handler = handle
        self.socket = sock

    def __str__ (self):
        return str(self.handler) + " (Reader)"

    def run(self):
        self.socket.settimeout(2)

        while(self.handler.IS_ALIVE):
            data = ""

            try:
                data = self.socket.recv(4096)
            except:
                # print 'fuck'
                continue

            if(data == ""):
                ServerHelper.printError(self, "Connection Lost")
                self.handler.shutdown()
                break

            if(self.handler.IS_ALIVE == False):
                break;

            self.handler.receiveAction(data)

        ServerHelper.printOut(self, "InputStream closed")
```

# Appendix F

# RemotePlotter.py

Python program to remotely connect and plot data from EEG devices.

Associated libraries can provided by emailing jasony@mit.edu

```
# code to remotely access an EEG server and plot data

import socket
import sys
import time

try:

    import pyqtgraph as pg
    from pyqtgraph.Qt import QtCore, QtGui
    import numpy as np
    import scipy.fftpack
    import serial

except Exception as e:
    print e
    print "please install the prerequisite packages"
    input("press enter to exit")
    raise Exception()

# User Variables
PORT = 'com9'
BAUD = 460800
PACKET_HEADER = 170
PACKET_SIZE = 11
BUFFER_DEPTH = 6400
LSB_OFFSET = 540
SAMPLE_FREQ = 640

# Calculated Values
CHUNK_SIZE = 1
SAMPLE_PERIOD = 1.0/SAMPLE_FREQ

# Globals
sock = None
win = None
win2 = None
curve1 = None
curve2 = None
curve3 = None
```

```
curve4 = None
curve5 = None
curve6 = None
curve7 = None
curve8 = None

data1 = None
data2 = None
data3 = None
data4 = None
data5 = None
data6 = None
data7 = None
data8 = None

def init():
    global sock, win, curve1, curve2, curve3, curve4, curve5, curve6, curve7,
     curve8, data1, data2, data3, data4, data5, data6, data7, data8, xVals

    address = ""
    with open('serverAddress', 'r') as f:
        address = f.readline().strip().split(" ")
        address = address[0]

    sock = socket.create_connection((address, 10000))
    sock.settimeout(5)
    print "connected"

    win = pg.GraphicsWindow()
    win.setWindowTitle('EEG Data')

    p1 = win.addPlot(title = "Channel 1")
    p1.setLabel('bottom', 'Time', 's')
    p1.setLabel('left', 'LSB')
    win.nextColumn()

    p2 = win.addPlot(title = "Channel 2")
    p2.setLabel('bottom', 'Time', 's')
    p2.setLabel('left', 'LSB')
    win.nextRow()

    p3 = win.addPlot(title = "Channel 3")
    p3.setLabel('bottom', 'Time', 's')
    p3.setLabel('left', 'LSB')
    win.nextColumn()

    p4 = win.addPlot(title = "Channel 4")
    p4.setLabel('bottom', 'Time', 's')
    p4.setLabel('left', 'LSB')
    win.nextRow()

    p5 = win.addPlot(title = "Channel 5")
    p5.setLabel('bottom', 'Time', 's')
    p5.setLabel('left', 'LSB')
    win.nextColumn()

    p6 = win.addPlot(title = "Channel 6")
    p6.setLabel('bottom', 'Time', 's')
    p6.setLabel('left', 'LSB')
    win.nextRow()

    p7 = win.addPlot(title = "Channel 7")
    p7.setLabel('bottom', 'Time', 's')
    p7.setLabel('left', 'LSB')
    win.nextColumn()

    p8 = win.addPlot(title = "Channel 8")
    p8.setLabel('bottom', 'Time', 's')
```

```
    p8.setLabel('left', 'LSB')


    data1 = [0]
    data2 = [0]
    data3 = [0]
    data4 = [0]
    data5 = [0]
    data6 = [0]
    data7 = [0]
    data8 = [0]
    updateChunk(1)
    data1 = data1 * BUFFER_DEPTH
    data2 = data2 * BUFFER_DEPTH
    data3 = data3 * BUFFER_DEPTH
    data4 = data4 * BUFFER_DEPTH
    data5 = data5 * BUFFER_DEPTH
    data6 = data6 * BUFFER_DEPTH
    data7 = data7 * BUFFER_DEPTH
    data8 = data8 * BUFFER_DEPTH

    xVals = np.linspace(-1*SAMPLE_PERIOD*BUFFER_DEPTH, 0,BUFFER_DEPTH).tolist()

    curve1 = p1.plot(x=xVals,y=data1)
    curve2 = p2.plot(x=xVals,y=data2)
    curve3 = p3.plot(x=xVals,y=data3)
    curve4 = p4.plot(x=xVals,y=data4)
    curve5 = p5.plot(x=xVals,y=data5)
    curve6 = p6.plot(x=xVals,y=data6)
    curve7 = p7.plot(x=xVals,y=data7)
    curve8 = p8.plot(x=xVals,y=data8)

    print("filled buffer")
    return



def updateChunk(size):
    data1[:(-1*size)] = data1[size:]
    data2[:(-1*size)] = data2[size:]
    data3[:(-1*size)] = data3[size:]
    data4[:(-1*size)] = data4[size:]
    data5[:(-1*size)] = data5[size:]
    data6[:(-1*size)] = data6[size:]
    data7[:(-1*size)] = data7[size:]
    data8[:(-1*size)] = data8[size:]

    for pos in range((-1*size) , 0):
        while(ord(sock.recv(1)) != PACKET_HEADER):
            continue


        packet = (sock.recv(PACKET_SIZE-1))
        if(len(packet) < PACKET_SIZE-1):
            # print len(packet)
            packet = packet + (sock.recv(PACKET_SIZE-1-len(packet)))

        packet = processData(packet)
        data1[pos]=packet[0]
        data2[pos]=packet[1]
        data3[pos]=packet[2]
        data4[pos]=packet[3]
        data5[pos]=packet[4]
        data6[pos]=packet[5]
        data7[pos]=packet[6]
        data8[pos]=packet[7]
```

```
def update():
    updateChunk(60)
    curve1.setData(xVals, data1)
    curve2.setData(xVals, data2)
    curve3.setData(xVals, data3)
    curve4.setData(xVals, data4)
    curve5.setData(xVals, data5)
    curve6.setData(xVals, data6)
    curve7.setData(xVals, data7)
    curve8.setData(xVals, data8)

def processData(raw):
    output = []

    raw = toIntArray(raw)
    output.append((raw[0]<<2) + (raw[1]>>6))
    output.append(((raw[1]&63)<<4) + (raw[2]>>4))
    output.append(((raw[2]&15)<<6) + (raw[3]>>2))
    output.append(((raw[3]&3)<<8) + (raw[4]))

    output.append((raw[5]<<2) + (raw[6]>>6))
    output.append(((raw[6]&63)<<4) + (raw[7]>>4))
    output.append(((raw[7]&15)<<6) + (raw[8]>>2))
    output.append(((raw[8]&3)<<8) + (raw[9]))
    return output

def findStart():
    print("finding start")
    while(ord(sock.recv(1)) != PACKET_HEADER):
        continue
    sock.recv(PACKET_SIZE-1)

    sock.recv(PACKET_SIZE-1)
    print("done")
    return




def toIntArray(charArray):
    out = []
    for x in charArray:
        out.append(ord(x))
    return out

def run():
    findStart()
    timer = pg.QtCore.QTimer()
    timer.timeout.connect(update)
    timer.start(1)

    ## Start Qt event loop unless running in interactive mode or using pyside.
    if __name__ == '__main__':
        import sys
        if (sys.flags.interactive != 1) or not hasattr(QtCore, 'PYQT_VERSION'):
            QtGui.QApplication.instance().exec_()


#main Program
init()
run()
# raw_input("test")
print >>sys.stderr, 'closing socket'
sock.close()
```

# Bibliography


[1] Ernst Niedermeyer and FH Lopes da Silva. *Electroencephalography: basic principles, clinical applications, and related fields*. Lippincott Williams & Wilkins, 2005.

[2] William O Tatum IV. *Handbook of EEG interpretation*. Demos Medical Publishing, 2014.

[3] Alexander J Casson, David C Yates, Shelagh JM Smith, John S Duncan, and Esther Rodriguez-Villegas. Wearable electroencephalography. *IEEE engineering in medicine and biology magazine*, 29(3):44–56, 2010.

[4] Paul L Nunez and Ramesh Srinivasan. *Electric fields of the brain: the neurophysics of EEG*. Oxford University Press, USA, 2006.

[5] Mark F Bear, Barry W Connors, and Michael A Paradiso. *Neuroscience*, volume 2. Lippincott Williams & Wilkins, 2007.

[6] Tao Le, Vikas Bhushan, Juliana Tolles, and J Hofmann. *First aid for the USMLE step 1 2013*. McGraw-Hill Medical, 2010.

[7] Luigi De Gennaro and Michele Ferrara. Sleep spindles: an overview, 2003.

[8] Kimberly A Cote, Duncan R De Lugt, Susan D Langley, and Kenneth B Campbell. Scalp topography of the auditory evoked k-complex in stage 2 and slow wave sleep. *Journal of Sleep Research*, 8(4):263–272, 1999.

[9] Joachim H Nagel. *Medical Instruments and Devices: Principles and Practices*, chapter Biopotential amplifiers, pages 1–16. CRC Press, 2015.

[10] M. Teplan. Fundamentals of eeg measurement. *Measurement Science Review*, Vol. 2(Section 2), 2002.

[11] American Clinical Neurophysiology Society et al. Guideline twelve: guidelines for long-term monitoring for epilepsy. *American journal of electroneurodiagnostic technology*, 48(4):265, 2008.

[12] Herbert H Jasper. The ten twenty electrode system of the international federation. *Electroencephalography and clinical neurophysiology*, 10:371–375, 1958.

[13] John R Hughes. *EEG in clinical practice*. Butterworth-Heinemann, 1994.

[14] Robert S Fisher, Walter van Emde Boas, Warren Blume, Christian Elger, Pierre Genton, Phillip Lee, and Jerome Engel. Epileptic seizures and epilepsy: definitions proposed by the international league against epilepsy (ilae) and the international bureau for epilepsy (ibe). *Epilepsia*, 46(4):470–472, 2005.

[15] Nih epilepsy website. http://www.ninds.nih.gov/disorders/epilepsy/epilepsy.htm.

[16] H Jasper. Report of the committee on methods of clinical examination in electroencephalography. *Electroencephalography and Clinical Neurophysiology*, 10:370–375, 1958.

[17] Sacred Heart HealthCare. Ambulatory electroencephalography. *online reference*, April 2017. http://www.shh.org/outpatient-services/neurodiagnostics/ambulatory-electroencephalography.asp.

[18] Bruno Guimaraes Do Valle. *Long-Term, Subdermal Implantable EEG Recording and Seizure Detection*. PhD thesis, Massachusetts Institute of Technology, 2016.

[19] Fritz Langford-Smith. *Radio designer's handbook*. Butterworth-Heinemann, 1997.

[20] Xun Liu, WM Ng, CK Lee, and SY Hui. Optimal operation of contactless transformers with resonance in secondary circuits. In *Applied Power Electronics Conference and Exposition, 2008. APEC 2008. Twenty-Third Annual IEEE*, pages 645–650. IEEE, 2008.

[21] Nathan O Sokal and Alan D Sokal. Class ea new class of high-efficiency tuned single-ended switching power amplifiers. *IEEE Journal of solid-state circuits*, 10(3):168–176, 1975.

[22] Weili Dai, Wei Tang, Yuanxiu Xiao, and Juntao Fei. A wireless power transfer system based on class e amplifier. In *Control, Automation and Systems (ICCAS), 2013 13th International Conference on*, pages 427–430. IEEE, 2013.

[23] Duracell-Gillete. Zinc-air technical bulletin. *online reference*, May 2017. https://d2ei442zrkqy2u.cloudfront.net/wp-content/uploads/2016/03/Zinc-Air-Tech-Bulletin.pdf.