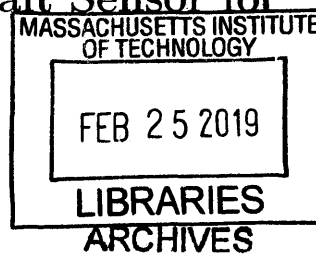


Design of Versatile and Low-Cost Shaft Sensor for
Health Monitoring

by
Erik M. Gest



B.S., Massachusetts Institute of Technology (2017)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2019

© Massachusetts Institute of Technology 2019. All rights reserved.

Signature redacted

Author

Department of Mechanical Engineering
January 22, 2019

Signature redacted

Certified by

Kamal Youcef-Toumi
Professor

Signature redacted Thesis Supervisor

Accepted by

Nicolas Hadjiconstantinou
Graduate Officer, Department of Mechanical Engineering

Design of Versatile and Low-Cost Shaft Sensor for Health Monitoring

by

Erik M. Gest

Submitted to the Department of Mechanical Engineering
on January 22, 2019, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

Virtually every mechanized form of transportation, power generation system, industrial equipment, and robotic system has rotating shafts. As the shaft is often the main means of mechanical power transmission, measuring the torque, speed, vibration, and bending of the shaft can be used in many cases to assess device performance and health and to implement controls. This thesis proposes a shaft sensor that measures all of these phenomena with reasonable accuracy while having a low cost and simple installation process. This sensor transfers strain from the shaft and amplifies it to increase sensitivity. Furthermore, this sensor requires no components to be in the stationary reference frame, allowing the entire device to rotate with the shaft. A prototype is presented. Experimental results illustrate the effectiveness of the proposed system.

Thesis Supervisor: Kamal Youcef-Toumi
Title: Professor

Acknowledgments

First and foremost I would like to thank my family: Mom, Dad, Daniel, Sarah, and Luke. My education would not have been possible without them and I owe everything I've accomplished to them. Thank you for the Skype meetings, care-packages, and support.

I would also like to thank Japan Steel Works, our project's sponsor. Their financial and technical help was invaluable to the project. I would like to specifically thank Mikio Furokawa and Takayuki Hirano for their assistance.

Furthermore, I would like to thank my fellow lab-mates in the Mechatronics Research Laboratory. Thank you for helping keep my days bright and for all the advice to further my research. It was a pleasure working with you.

Last but not least I would like to thank my advisor Professor Kamal Youcef-Toumi. You are one of the most genuinely caring people I have ever met and your constant support of the students you advise is incredible. I cannot imagine having a better advisor.

Contents

1	Background	17
1.1	Health Monitoring	17
1.2	The Internet of Things	18
1.3	Novel Sensor	19
1.4	Primary Application	19
2	Benchtop System	21
2.1	Background	21
2.2	Design	21
3	Torque and Bending Sensor	27
3.1	Background	27
3.2	Design	29
3.2.1	Torque Calculations	30
3.2.2	Bending	34
3.2.3	Instrumentation	34
3.2.4	Construction	37
3.3	Results	40
3.3.1	Stationary vs. Rotating Shafts	40
3.3.2	Stationary Results	40
3.3.3	Rotating and Bending Results	41
3.3.4	Long Term Drift	44

4	Acceleration	53
4.1	Theory	53
4.2	Selection	57
4.3	Results	58
5	Power Generation	61
5.1	Requirements	61
5.2	Background	62
5.3	Brushless Generator	64
5.3.1	Overview	64
5.3.2	Static Mechanical Model	65
5.3.3	Dynamic Mechanical Model	68
5.3.4	Implementation	69
5.3.5	Results	69
6	System Integration	73
6.1	Microphone	75
6.2	Wireless Communications	76
7	Conclusion	79
7.1	Summary	79
7.2	Future Work	79
A	Engineering Drawings	81
B	Wiring Diagrams	87
C	Operational Code	89
C.1	Sensor Code (C++)	89
C.2	Receiver Code (Python 3.6)	93
C.3	Benchtop Arduino Code (C++)	94
C.4	PWM Shift Code (C++)	98
C.5	Benchtop PC Code (Python 3.6)	98

D	Analysis Code	107
D.1	Data Processing Code (Python 3.6)	107
D.2	View Results Code (MATLAB)	108
D.3	FFT of Torque Signal Code (MATLAB)	111
D.4	Acceleration Simulation Code (MATLAB)	112
D.4.1	AccelMethods.m	112
D.4.2	plotSpeedsFromSignals.m	113
E	Bills of Materials	115
F	Finite Element Analysis Details	117
G	Instructions for Fabrication, Mounting, and Testing	127
G.1	Sensor Fabrication	127
G.2	Sensor Mounting	128
G.3	Testing	128
H	Version History	131

List of Figures

1-1	Vendors and customers agree on the value of many industrial IoT use cases [12]. Image modified from [12].	19
1-2	A schematic of a Twin Screw Plastic Extruder (TEX). Image modified from [36].	20
2-1	A Twin Screw Plastic Extruding (TEX) machine. Image from [44]. . .	22
2-2	A diagram of the Benchtop System used to test the sensor.	22
2-3	A rendering of the Benchtop System used to test the sensor.	23
2-4	By using relays and power resistors, a digitally controlled discrete variable resistor is made.	24
3-1	An example of a production torque sensor. Image from [35].	28
3-2	A simplified diagram of the designed torque sensor.	29
3-3	Finite element analysis on the sensor confirms nearly all strain occurs in the flexure zone of the bridge.	30
3-4	Definitions of variables used when sensor is not undergoing torque. . .	31
3-5	Definitions of variables used when sensor is undergoing torque.	31
3-6	Strain gauges can be used in the quarter, half, or full bridge configurations. Arrows facing apart represent a gauge undergoing tension while arrows facing each other represent a gauge undergoing compression. Image modified from [4].	35
3-7	A rendering of the novel torque sensor.	37
3-8	The new torque sensor as made.	38
3-9	Exactly half the parts required for the torque sensor.	38

3-10	The aluminum collars before (left) and after machining (right)	39
3-11	The experimental setup used to calibrate the torque sensor by varying the weight applied on the lever arm (not to scale).	41
3-12	Output of the torque sensor and applied torque over time.	42
3-13	A zoomed in view of Figure 3-12.	42
3-14	The output of the torque sensor versus the applied torque. The relationship is linear. The raw data is presented in Figure 3-12.	43
3-15	The experimental setup used to calibrate the torque sensor using a purchased force meter to measure applied on the lever arm (not to scale).	43
3-16	Output of the torque sensor and applied torque over time.	44
3-17	A burst reading from the torque sensor shows the torque as the mean reading and the amount of bending as the amplitude. The frequency of the signal is the angular speed of the shaft (600 RPM or 10 Hz).	45
3-18	The power spectrum of the signal shown in Figure 3-17 shows the clear dominant frequency of 10 Hz.	45
3-19	A power spectrum of the torque signal shows that the dominant frequency of the signal is the shaft speed (title of each graph).	46
3-20	Drift in the sensor reading over time with the HX711 chip.	47
3-21	Drift in the sensor reading over time with the LTC2440 chip.	48
3-22	Chip temperature versus strain reading for the HX711 chip. The correlation appears to be weak.	48
3-23	Chip temperature versus strain reading for the LTC2440 chip. There appears to be some very weak correlation.	49
3-24	Chip temperature versus strain reading for the LTC2440 chip for times 25 hours to 32 hours as shown in Figure 3-21. The correlation is strong and a linear fit is shown.	50
3-25	Temperature correction factors can be applied to improve the reading of the LTC2440.	50

3-26	The probability distribution of total strain drift over a minute. The vast majority of samples have less than 2 micro-strain of drift.	51
3-27	The probability distribution of total strain drift over an hour. The drift is on the order of micro-strains.	52
4-1	A simulation of the calculated speed from the accelerometer using three different methods assuming no noise in the signal.	55
4-2	A simulation of the calculated speed from the accelerometer using three different methods with white noise added to the signal. The noise was added at a signal-to-noise ratio of 5.	56
4-3	A simulation of the calculated speed from the accelerometer using three different methods with a static constant offset error of 0.5 g added to the signal.	56
4-4	A actual burst from the accelerometer is shown. It is close to a sine wave as predicted. The sampling rate was 750 Hz.	58
4-5	A comparison of methods used to determine angular speed. All are close.	59
5-1	An example of a brushless generator is shown. It is simply a brushless motor driven in reverse.	64
5-2	The generator can be considered to be just a pendulum.	65
5-3	The mechanical power exerted on the generator depends on the angle of the pendulum from vertical.	66
5-4	The brushless motor used as a generator. This motor has a through-hole allowing it to be placed on axis with the shaft. Image from [5]. .	70
5-5	The schematic of a three-phase rectifier is shown. Image modified from [3]. This rectifier was built using six 1N5817 diodes.	70
5-6	As expected, the voltage out of the motor is proportional to the speed. The slope of the fit is the motor constant, K_t	71
5-7	The electrical power output increases with the shaft speed.	72

6-1	Data is collected from the accelerometer, load-cell amplifier, and microphone, processed by the microcontroller, and is then sent over Wi-Fi to a receiver.	74
6-2	The old prototype case design completely enclosed all components. . .	74
6-3	A rendering of the new case design.	75
6-4	The new case design as made with components mounted.	76
6-5	The I2S microphone used on the prototype. Image from [1].	76
6-6	The trade-off between number of strain samples and sample frequency. The ratio of acceleration samples to strain samples is shown for each point.	78
7-1	Insetting the bridge into the shaft collar can reduce the size of the sensor.	80
B-1	This figure shows wiring of the sensor. The microcontroller (black, center) is wired to the load cell amplifier (red, left), the accelerometer (blue, top left), the microphone (blue, bottom right), and the LED (top right). The load cell amplifier is wired to the Wheatstone bridge (bottom left). This image was generated using Fritzing and using [43] and [2].	88
H-1	Comparison of the various sensor versions made over the project. . .	132

List of Tables

3.1	Comparison of two chips that can be used to read the strain. HX711 data is from [10] and LTC2440 data is from [21].	37
5.1	Device Power Requirements	63
E.1	Bill of materials for the designed sensor.	115
E.2	Bill of materials for the benchtop system.	116

Chapter 1

Background

1.1 Health Monitoring

Health monitoring and prediction of automatic systems is becoming crucial. Predicting failures can allow the system to adapt to avoid it, or at least alert a user to repair it. Furthermore, monitoring the system can help estimate wear and ensure all means of minimizing it are being utilized. Finally and most importantly, this can help increase the safety of systems by stopping dangerous situations before they happen.

Many automatic systems use shafts for power and information transmission. From industrial manufacturing equipment, transportation systems, to consumer products, rotating shafts are in many mechanical devices.

Understanding what phenomena a shaft is experiencing is useful for many reasons. During product testing, analyzing the shaft can ensure that the system is performing correctly and help identify issues in the design. Many devices utilize closed-loop feedback control and knowledge of the state of the shaft is often critical. For example, knowing the speed and torque on an actuator of a robotic system can be used to ensure precise manipulation and interactions with the environment. Many issues such as long term fatigue, wear-related issues, and acute failures can cause symptoms that are detectable on the shaft. Torque and speed, as the effort and flow variables of the shaft, are the most desirable state variables on a shaft to measure in nearly all these cases.

When a machine experiences an anomaly, it can manifest itself not only as a change in torque and speed on the shaft but also as bending on or vibration of the shaft. If all four of these phenomena can be measured, it is likely that issues can be detected before they become critical, reducing damage, increasing performance, and ensuring safety.

1.2 The Internet of Things

Many inexpensive microcontrollers and wireless communications are now available. This allows for nearly any device to be connected to the internet. Consumers can now buy Wi-Fi connected appliances, security systems, lights, and thermostats to build a "Smart Home" [14]. Distributed sensor networks can be used to monitor environmental conditions around the globe [19]. As mentioned above, similar systems can be used to monitor the health of industrial equipment. All these applications fall under the "Internet of Things" (IoT).

The Internet of Things has incredible economic potential. In fact, it is estimated that IoT revenues will increase from \$195B in 2015 to \$470B in 2020 [12]. Consumer applications make up about one-third of the 2020 revenue, while business applications take up the remaining two thirds.

The industrial IoT has several unique applications. Health monitoring uses collected data to determine the state of a machine in real time, allowing for both remote monitoring and quality control. Predictive maintenance uses data to estimate the future state of the system helping to prevent failures in the future. As shown in Figure 1-1, two of these are both very attractive to customers and ready to be adopted by vendors, making health monitoring for quality control and predictive maintenance some of the most immediately viable IoT applications.

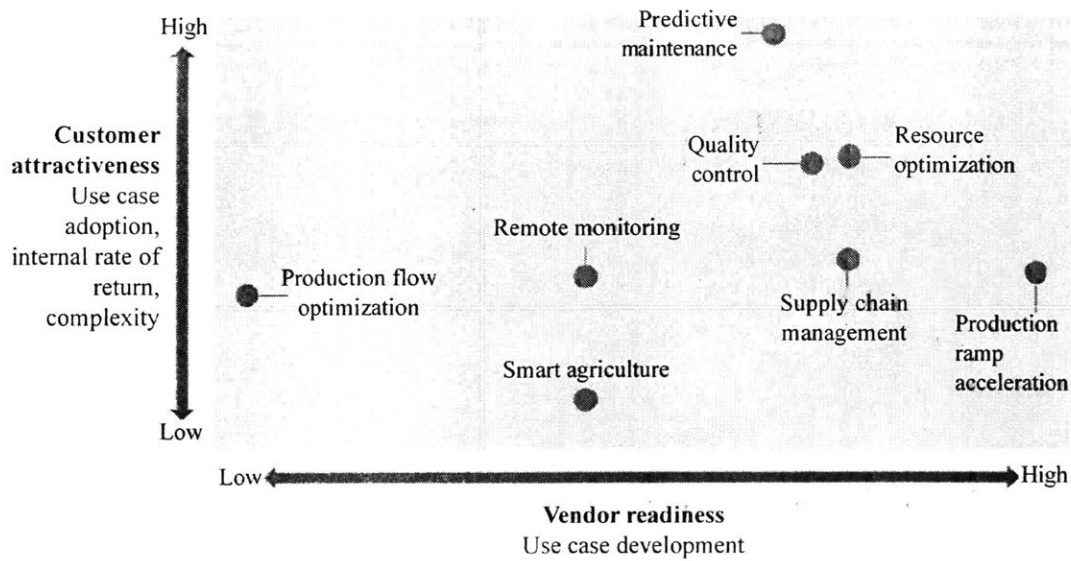


Figure 1-1: Vendors and customers agree on the value of many industrial IoT use cases [12]. Image modified from [12].

1.3 Novel Sensor

This paper presents the design of a new device for health monitoring of machinery with rotating shafts. In order to make use of the IoT and be effective at health monitoring, several constraints are placed on the device. First, it must be able to detect shaft speed, torque, vibration, and some modes of bending. Furthermore, it must be easy to install on existing systems. This extends to more tangible requirements: it cannot require modification of the shaft or adhesives on the shaft. It must also rotate with the shaft. Finally, it must be relatively low cost so it can be implemented on a large scale.

1.4 Primary Application

The chosen first application for this sensor is health monitoring of a Twin Screw Plastic Extruder (TEX). This was chosen by the project sponsor, Japan Steel Works (JSW). A diagram of one is shown in Figure 1-2.

A TEX melts, mixes, and forms plastic parts with a constant cross-section, like

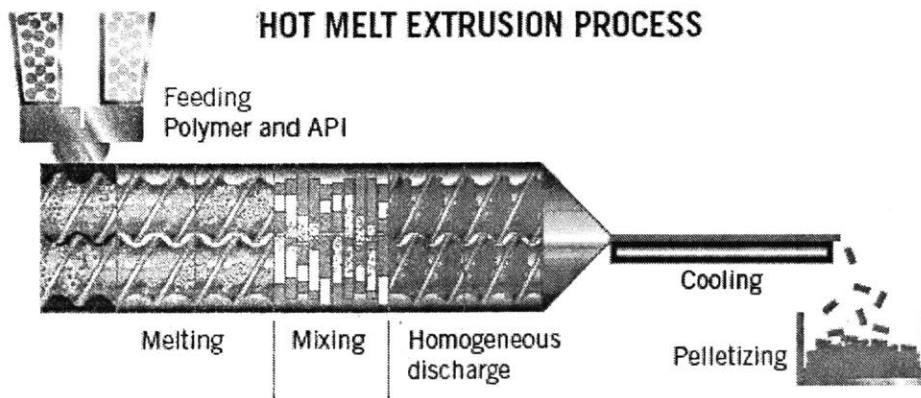


Figure 1-2: A schematic of a Twin Screw Plastic Extruder (TEX). Image modified from [36].

pping. First, plastic pellets (often of varied types) are fed into the auger. Heat is applied as the auger pushes the material forward. The liquid plastic is then mixed. Finally, the mixture is pushed through a die and cooled creating the final product.

Chapter 2

Benchtop System

2.1 Background

It is not practical for the team working at MIT to have direct access to a Twin Screw Extruder, an example of which is shown in Figure 2-1. The first issue is size: the extruders vary from 2.2 m to 11.5 m in length and 1200 kg to 18 000 kg [45]. MIT has very limited laboratory space and could not easily accommodate a TEX. Furthermore, a TEX is an expensive and valuable piece of equipment, and taking one out of production for this project would have dramatically increased the project cost. Finally, a skilled operator is required to use the TEX and it would be impractical for MIT to staff one.

2.2 Design

Since access to a TEX is not practical, a benchtop simulator was built to allow for testing of the sensor as shown in Figures 2-2 and 2-3. There are three critical components of the TEX to recreate: a driving electric motor, a shaft with a portion exposed, and a viscous damper. Like the TEX, the benchtop system uses an electric motor to spin a shaft. The shaft of the benchtop has an axially longer exposed than the TEX so sensor prototypes can be tested with less minimization. Furthermore, it is designed to spin at speeds similar to the TEX.

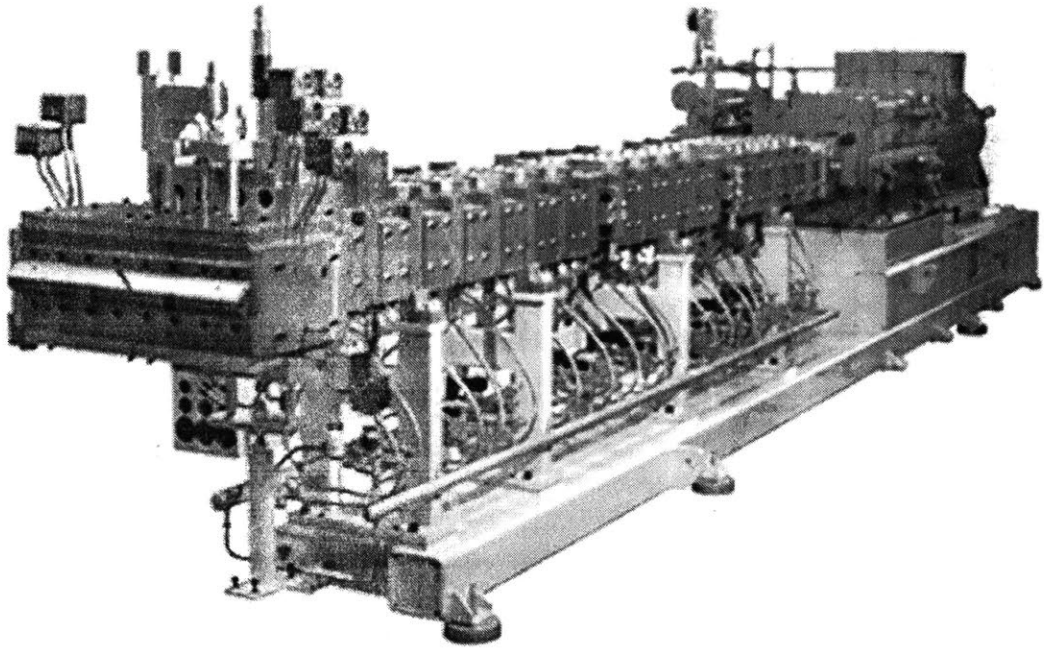


Figure 2-1: A Twin Screw Plastic Extruding (TEX) machine. Image from [44].

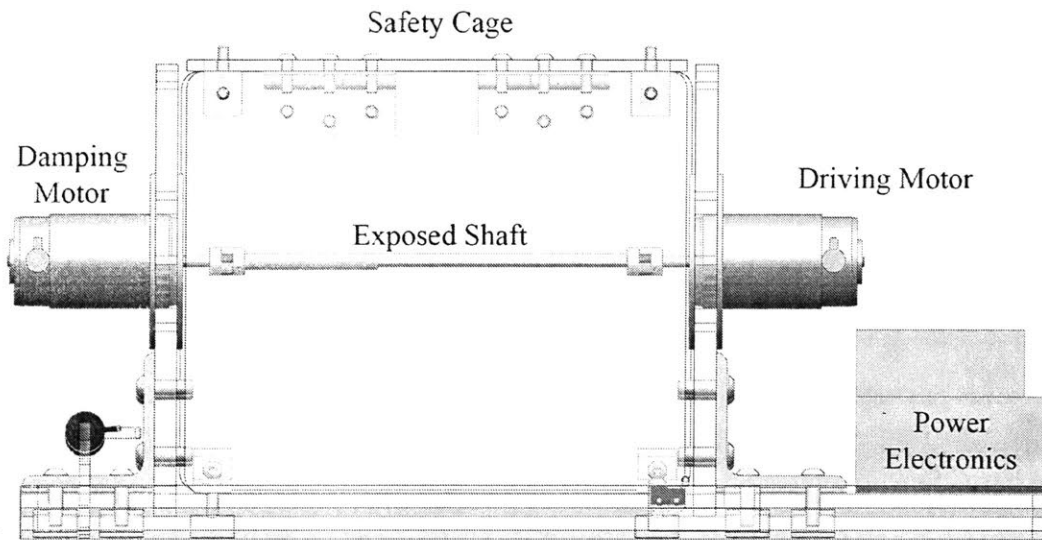


Figure 2-2: A diagram of the Benchtop System used to test the sensor.

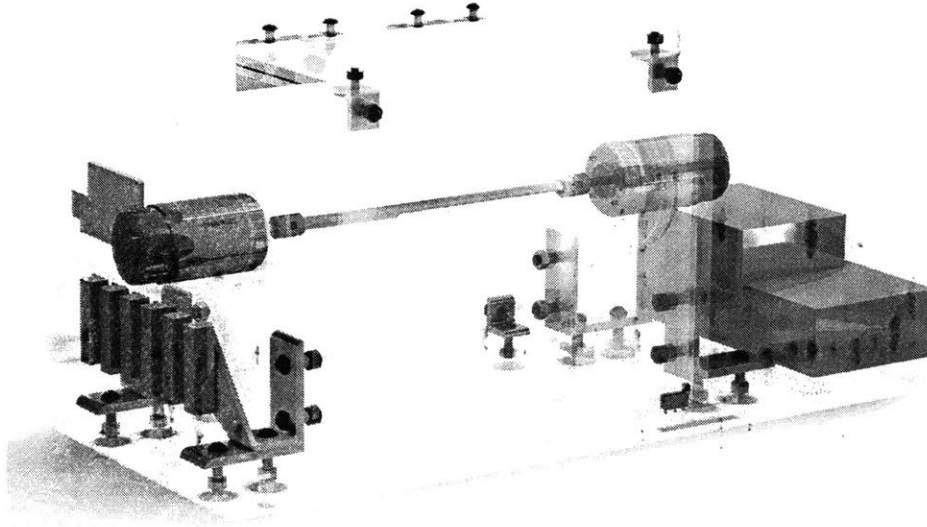


Figure 2-3: A rendering of the Benchtop System used to test the sensor.

The shaft of the TEX and that of the benchtop simulator is connected to a viscous damper. In the case of the TEX, this is caused by the rotating auger pushing the partially melted plastic through the chamber. The damping characteristics of the auger depend on the TEX model, type of plastic used, and operation settings. Therefore, it is not sufficient to choose a non-variable damper for the benchtop system. A variable viscous damper can be made a couple of ways. For example, a fan with variable pitch blades could be used. However, rotating fan blades are both loud and dangerous and are relatively difficult to reliably digitally manipulate.

Therefore, an electric motor with connected electrically to a resistor array was used to create a variable rotary damper. If a motor is modeled as an ideal gyrator with torque constant K_t , then the torque T on the shaft of the motor is proportional to the current i :

$$T = K_t i \quad (2.1)$$

Furthermore, the speed of the shaft ω is proportional to the back EMF voltage, V ,

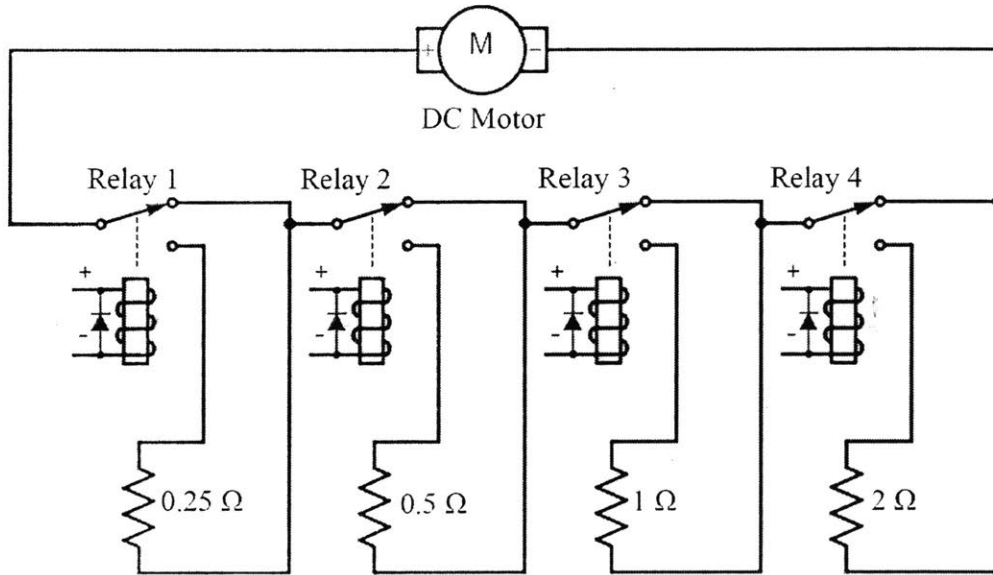


Figure 2-4: By using relays and power resistors, a digitally controlled discrete variable resistor is made.

across the leads of the motor:

$$V = K_t \omega \quad (2.2)$$

When the motor is attached to a resistor R , then Ohm's law must be satisfied:

$$V = iR \quad (2.3)$$

Equations (2.1), (2.2), and (2.3) can be combined into:

$$T = \frac{K_t^2}{R} \omega \quad (2.4)$$

This is the equation of a rotary damper with damping coefficient K_T^2/R . By modulating the resistance, the damping coefficient can be varied. A resistor array was built that allows for the effective resistance connected to the motor to be discretely set as shown in Figure 2-4. This system creates a viscous rotary damper that is safe, repeatable, and quiet.

The benchtop system is controlled by an Arduino microcontroller that can be

connected to a PC or Raspberry Pi. Commands are sent from the PC (or Raspberry Pi) to the Arduino via the USB serial connection. This allows for the motor speed and resistor array resistance to be set from the PC. The microcontroller also records the shaft speed via an encoder. This also allows the benchtop system to be speed controlled via a closed loop PI controller. This data is sent to the PC via the USB serial connection. A custom Python script was used to control the system from the PC which allowed for the speed and resistance settings to be automatically changed at given intervals for easy data collection. Both these pieces of code can be found in Appendix C.

Most of the benchtop model was made by laser cutting acrylic panels and bolting those pieces together. This includes the two base panels, the two vertical motor holding pieces, and the entire cage. This method allows for rapid and precise manufacturing. Furthermore, acrylic is clear and allows for viewing of the sensor during testing.

Several steps were taken to ensure that the benchtop system would be safe to operate. First, a large emergency stop switch is in line with the main power, allowing a user to rapidly de-energize the system. Second, an acrylic cage was built around the exposed shaft. This cage is strong enough to stop any sensor components that may come loose during rotation. It also prevents the user from getting hair, clothing, or jewelry tangled in the spinning shaft. Finally, a limit switch is used to ensure the cage is closed. If opened, the controller stops the motor.

To be a valid testing system, the benchtop system must have anomalies for the sensor to detect that are analogous to issues on the TEX. For example, the auger of the TEX can break, incorrect material can be used, or an operator can make an error. The chosen method was to add an off-center weight to the shaft, causing an imbalance. This leads to vibration in the shaft. This bending and vibration phenomena is far simpler than the coupled bending of the TEX shafts. However, this weight is enough to model a basic bending mode of the TEX. The project sponsor deemed this "issue" as sufficient for developing the anomaly detection algorithms. Non-complete engineering drawings of the benchtop system are in Appendix A. Wiring diagrams

for the system are in Appendix B. A bill of materials can be found in Appendix E.

Chapter 3

Torque and Bending Sensor

3.1 Background

Torque sensors are used for many functions. As mechanical power through a shaft is equal to angular speed times torque, they can be used in conjunction with a speed sensor to measure power usage or output. They are also often used for testing, especially when checking maximum strength or fatigue characteristics of parts or systems. They can be used in closed-loop feedback control as well as safety and health monitoring.

While there are many torque sensors already on the market, almost all are unsuitable for this application. An example is shown in Figure 3-1. Most do not work for rotating shafts. Of those that do, most are in line with the shaft. Of the remaining ones, most are too bulky, too expensive, and/or require modification of a shaft. This application requires a small, inexpensive, and minimally invasive sensor.

Nearly all torque sensors measure strain or displacement in order to estimate torque. The most common type of torque sensors for rotating shafts have their own axle that needs to be connected to the shaft on both ends. This means the shaft needs to be cut for the sensor to be installed, making installation a long process. If the device was not originally designed for the sensor, it may not be possible to install this type of torque sensor. For example, the shaft may not have a long enough exposed portion for the sensor to be added with couplers.

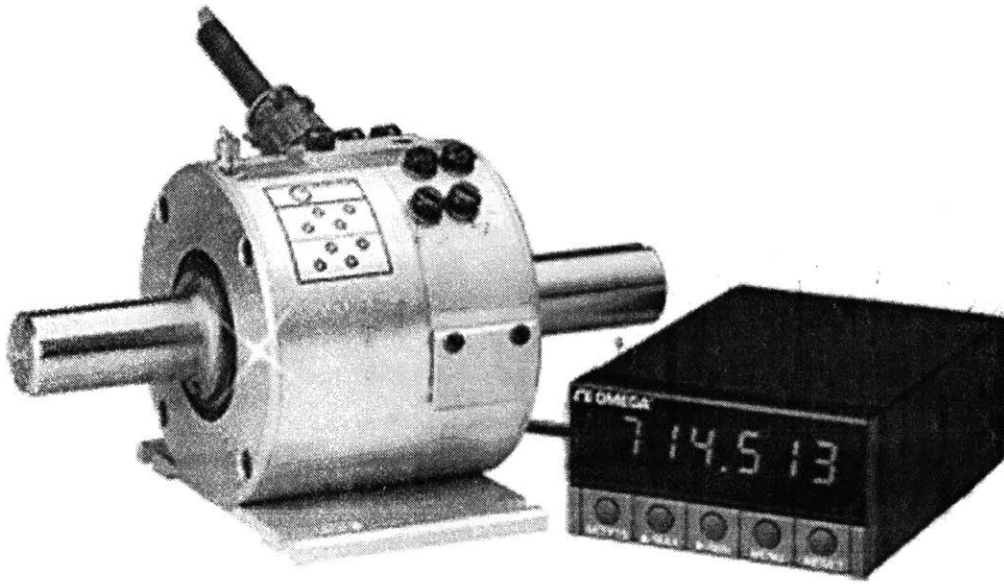


Figure 3-1: An example of a production torque sensor. Image from [35].

Clamp-on sensors solve this problem by allowing installation without modifying the shaft. These devices can use several different methods to determine strain. Surface acoustic wave (SAW) methods induce and measure waves that are affected by strain, allowing torque to be determined [16]. These require careful mounting of components on the surface of the shaft. Other optical devices measure the deflection of precise disks spread out axially over the shaft [42]. However, the simplest way is to mount strain gauges directly onto the shaft. Unfortunately, like the SAW methods, this requires precision and many steps in the installation process [32]. In addition, the shaft is often narrowed in the section where measurements are taken, further complicating the process and weakening the shaft.

One way to ease the installation process is to create a device that deforms with the shaft and measure the strain on that device [17][20]. This allows most of the calibration and precise work to be done prior to installation. Furthermore, strain gauges can be used to read the torque without direct mounting. In addition, these devices can be very compact compared to many standard torque transducers.

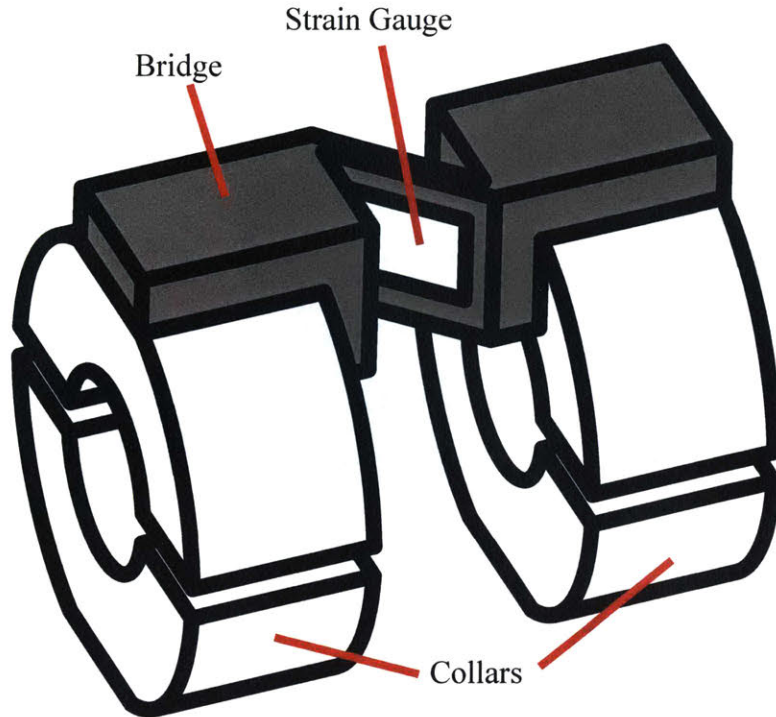


Figure 3-2: A simplified diagram of the designed torque sensor.

3.2 Design

Strain induced by torque or bending of the shaft is transferred to part of the sensor, where it is transduced into a signal. As shown in Figure 3-2, two aluminum collars are attached tightly to the shaft so there is a given distance between them, d_c . As the shaft is torqued, one collar will rotate slightly relative to the other one. As the shaft is bent, one side of the collars will move closer to each other while the opposite sides separate. A plastic bridge (or multiple bridges) are used to connect the collars. As the collars are thick and aluminum, they are far more rigid than the thin plastic pieces. This means that nearly all displacement and therefore strain is transferred to the bridge. Furthermore, the bridge is designed to transfer all strain to the flexure zone. This is confirmed with finite element analysis as shown in Figure 3-3. More details of this analysis are shown in Appendix F. A strain gauge (or gauges) is placed on the flexure zone, allowing for strain to be transduced. In addition, the plastic bridges are far less stiff than the shaft, so the sensor does not contribute significant stiffness to the shaft.

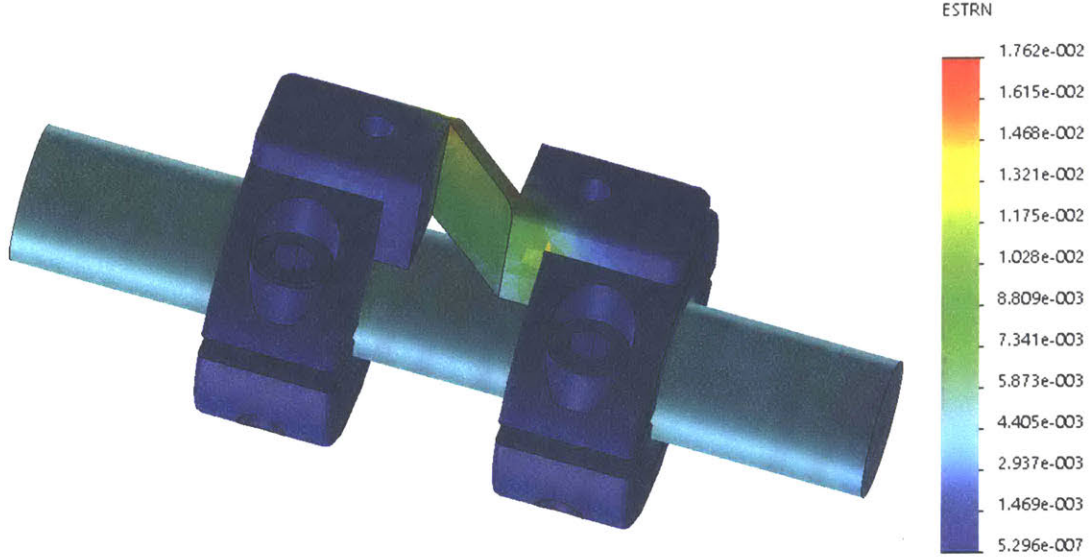


Figure 3-3: Finite element analysis on the sensor confirms nearly all strain occurs in the flexure zone of the bridge.

One advantage of this sensor over other ones is that it allows the strain induced by torque to be measured in tension instead of in shear. This allows for the use of less expensive and more common strain gauges.

3.2.1 Torque Calculations

With the assumptions, it can be shown that the strain experienced by the sensor is linear with the torque in the shaft. For a shaft of length L_{shaft} , polar moment of inertia J , and shear modulus G undergoing torque T , the angular displacement about the axial direction, θ_0 , is [40]:

$$\theta_0 = \frac{TL_{shaft}}{GJ} \quad (3.1)$$

If the collars are mounted such that the separation between them is d_c as shown in Figures 3-4 and 3-5, then the angle of deflection between them is simply:

$$\theta_0 = \frac{Td_c}{GJ} \quad (3.2)$$

$L_{flexure,z}$ is the initial length of the flexure in the axial direction as shown in Figure 3-4. Define x as the distance from a point to one end of the flexure in the axial

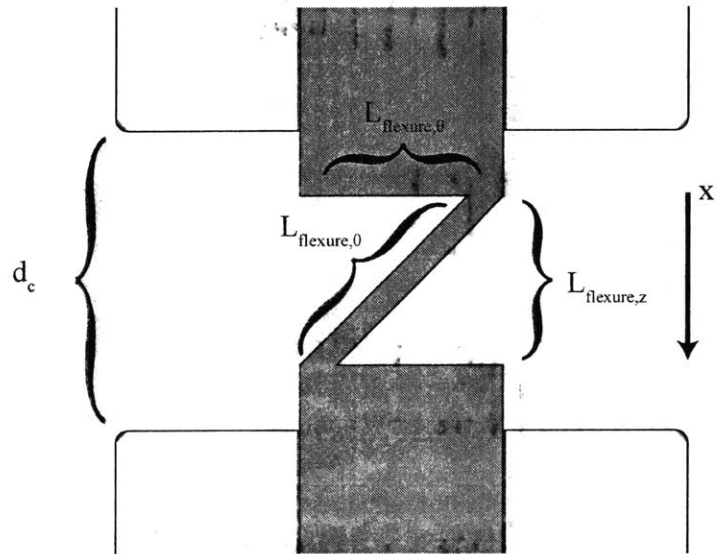


Figure 3-4: Definitions of variables used when sensor is not undergoing torque.

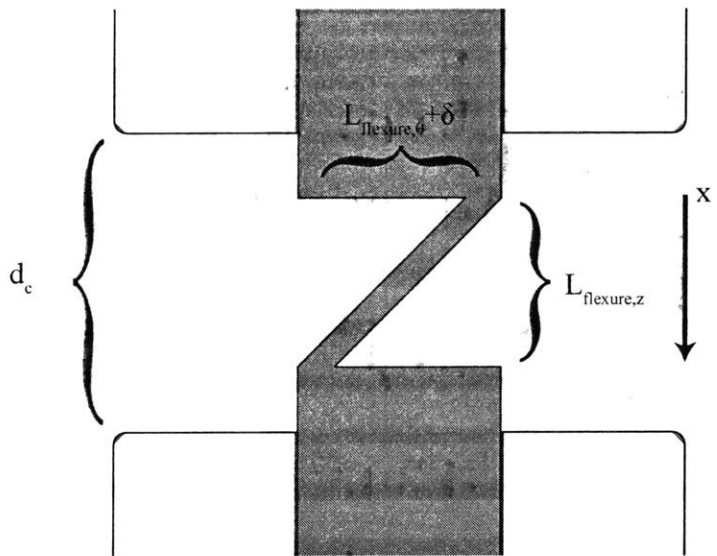


Figure 3-5: Definitions of variables used when sensor is undergoing torque.

direction as shown in Figures 3-4 and 3-5. The relative deformation in the angular direction, δ , at a distance r from the axis is simply:

$$\delta(x) = \frac{x}{L_{flexure,z}} \theta_0 r \quad (3.3)$$

Evaluating this at $x = L_{flexure,z}$ to show the total change over the bridge, δ_0 , is:

$$\delta_0 = \theta_0 r \quad (3.4)$$

The plane of the flexure is 45 deg offset from the axis of the shaft. The flexure's initial length in the axial direction, $L_{flexure,z}$, remains approximately constant. The flexure's length in the angular direction, $L_{flexure,\theta}$, changes by δ . Geometrically, $L_{flexure,\theta}$ and $L_{flexure,z}$ are equal and the overall initial flexure length, $L_{flexure,0}$ is simply:

$$L_{flexure,0} = \sqrt{(L_{flexure,\theta})^2 + (L_{flexure,z})^2} = \sqrt{2}L_{flexure,z} \quad (3.5)$$

The strain along the flexure, $\epsilon_{flexure}$, is simply the change in length over the original length. Using Equation (3.5), the strain is:

$$\epsilon_{flexure} = \frac{\sqrt{(L_{flexure,\theta} + \delta_0)^2 + (L_{flexure,z})^2} - \sqrt{2}L_{flexure,z}}{\sqrt{2}L_{flexure,z}} \quad (3.6)$$

Combining Equations (3.2), (3.4), and (3.6) and noting that $L_{flexure,\theta}$ and $L_{flexure,z}$ are equal:

$$\epsilon_{flexure} = \frac{\sqrt{(L_{flexure,z} + \frac{Td_c r}{GJ})^2 + (L_{flexure,z})^2} - \sqrt{2}L_{flexure,z}}{\sqrt{2}L_{flexure,z}} \quad (3.7)$$

Now, $\epsilon_{flexure}$ is a function of T and some geometric parameters. The following Taylor approximation can be made:

$$\epsilon_{flexure}(T) = \epsilon_{flexure}(T_0 + \Delta T) \approx \epsilon_{flexure}(T_0) + \left. \frac{d\epsilon_{flexure}}{dT} \right|_{T=T_0} \Delta T \quad (3.8)$$

Using, $T_0 = 0$ it is clear that $\epsilon_{flexure}(T_0 = 0)$ is simply 0. In this case, $\Delta T = T$. Now Equation (3.8) simplifies to:

$$\epsilon_{flexure}(T) \approx \left. \frac{d\epsilon_{flexure}}{dT} \right|_{T=0} T \quad (3.9)$$

Taking the derivative:

$$\frac{d\epsilon_{flexure}}{dT} = \frac{1}{\sqrt{2}L_{flexure,z}} \frac{2(L_{flexure,z} + \frac{Td_c}{GJ}r)\frac{rd_c}{GJ}}{2\sqrt{(L_{flexure,z} + \frac{Td_c}{GJ}r)^2 + (L_{flexure,z})^2}} \quad (3.10)$$

Evaluating Equation (3.10) for $T = 0$:

$$\left. \frac{d\epsilon_{flexure}}{dT} \right|_{T=0} = \frac{1}{\sqrt{2}L_{flexure,z}} \frac{2(L_{flexure,z})\frac{rd_c}{GJ}}{2\sqrt{(L_{flexure,z})^2 + (L_{flexure,z})^2}} = \frac{rd_c}{2GJL_{flexure,z}} \quad (3.11)$$

Combining Equations (3.9) and (3.11) yields:

$$\epsilon_{flexure}(T) \approx \frac{rd_c}{2GJL_{flexure,z}} T \quad (3.12)$$

For a shaft with a circular cross-section, the polar moment of inertia, J , is simply a function of the shaft diameter, D :

$$J = \frac{\pi D^4}{32} \quad (3.13)$$

Combining Equations (3.12) and (3.13) yields the final equation for strain on the flexure:

$$\epsilon_{flexure}(T) \approx \frac{16rd_c}{\pi GD^4 L_{flexure,z}} T \quad (3.14)$$

It should be noted that if the gauge is mounted on the surface, then $r = D/2$ and $d_c = L_{flexure,z}$. In this case, Equation (3.14) reduces to:

$$\epsilon_{flexure}(T) \approx \frac{8}{\pi GD^3} T \quad (3.15)$$

This matches what other sources say for strain gauge measured strain on a shaft [11].

3.2.2 Bending

Unlike most torque sensors, this sensor is also designed to detect bending of the shaft in certain conditions. This bending is in the same signal as the torque, as a bend causes the flexure to extend or compress. While in some cases this may be considered a disadvantage as bending can disrupt the signal, it is an advantage in others. There are two forms of bending on a shaft. The first is a force applied in a direction fixed to the stationary reference frame (the observer's point of view), which would appear to rotate in the rotating reference frame (the shaft's point of view). The second is the opposite, a force that appears stationary in the rotating reference frame but rotates in the stationary one. This sensor will detect the first as a fluctuation in torque: it will cause a positive error in one orientation and a negative in the opposite. The second will be a constant error in the torque reading. Calibration at zero torque can remove the effects of the second.

If the torque is relatively constant within a rotation, bending and torque can be easily extracted from the signal. The signal can simply be averaged over a rotation to find an accurate torque. Furthermore, the fluctuation of the signal in a cycle of the shaft can be used to find the bending. This allows the use of one sensor to detect both torque and bending of the shaft, which is useful in cost sensitive or volume constrained applications.

Unfortunately, the bending of the shaft is far more complex than the torque and depends greatly on the design of the machine. While torque can be measured by simply knowing the shaft diameter and material, for bending analysis the location and type of supports and loads on the shaft must also be known. However, if these are known, the calculations are quite simple.

3.2.3 Instrumentation

The change in resistance of the strain gauges is slight, so careful instrumentation is critical to measure an accurate torque. It is standard practice to use a Wheatstone bridge to turn the change in resistance to a measurable voltage chain. As shown in

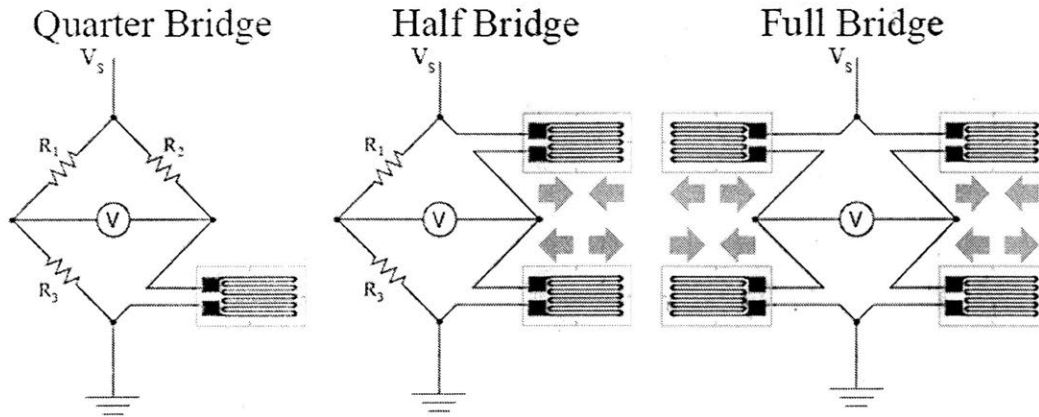


Figure 3-6: Strain gauges can be used in the quarter, half, or full bridge configurations. Arrows facing apart represent a gauge undergoing tension while arrows facing each other represent a gauge undergoing compression. Image modified from [4].

Figure 3-6, the Wheatstone bridge can be used in the quarter, half, or full configuration. Changes in temperature can change the resistance of the gauge, and, in the quarter bridge, this can cause significant error. However, in the half bridge and full bridge configurations, these changes are largely canceled out if all the gauges experience the same thermal effects. Furthermore, the signal strength of the half bridge is double that of the quarter as the half bridge is simply two quarter bridges superimposed. For the same reasons, the full bridge has double the signal strength of the half bridge and four times that of the quarter bridge.

A change in resistance of a strain gauge ΔR is related to the static resistance of the strain gauge R , the gauge factor G_F , and the strain it experiences, ϵ_{gauge} by the following equation [33]:

$$G_F = \frac{\Delta R}{R\epsilon_{gauge}} \quad (3.16)$$

Using simple voltage divider equations, if V_s is applied to the top of a bridge and the bottom is grounded, then the voltage across the bridge V is simply:

$$V = V_s \left(\frac{R_4}{R_2 + R_4} - \frac{R_3}{R_1 + R_3} \right) \quad (3.17)$$

If all resistances are equal, then V is zero. In the full bridge configuration, R_1 and R_4

are stretched and R_2 and R_3 are compressed. Using ΔR from Equation (3.14) and the fact that all resistances when unstretched are R , the voltage across the bridge will be:

$$V = V_s \left(\frac{(R + \Delta R)}{(R - \Delta R) + (R + \Delta R)} - \frac{(R - \Delta R)}{(R + \Delta R) + (R - \Delta R)} \right) \quad (3.18)$$

This equation can be simplified to:

$$V = V_s \frac{\Delta R}{R} \quad (3.19)$$

Combining Equations (3.16) and (3.19) yields:

$$\epsilon_{gauge} = \frac{1}{G_F} \frac{V}{V_s} \quad (3.20)$$

As the gauge is glued to the flexure, it can be assumed that their strains are approximately equal. Using this approximation and Equations (3.14) and (3.20):

$$T = \frac{\pi}{16} \frac{V}{V_s} \frac{L_{flexure,z}}{d_c} \frac{GD^4}{G_F r} \quad (3.21)$$

Most common strain gauges can only handle up to about 2% strain and have a gauge factor of approximately 2 [33]. Many microcontrollers operate at 3.3 V or 5 V [7]. Using this information and Equation (3.20), the maximum voltage V from the bridge would be 0.132 V and 0.20 V for the 3.3 V and 5 V microcontrollers, respectively.

Now that the maximum voltage is known, a chip to amplify and digitally read the strain gauges can be specified. It should be able to measure a voltage of ± 0.20 V before any safety factor. Furthermore, the lowest possible noise, highest possible resolution, accuracy, and highest sample rate are ideal. Two possible chips are compared in Table 3.2.3. While the LTC2440 has better specifications, it is much harder to integrate into the system. To utilize it, a custom printed circuit board (PCB) would have to be fabricated. Therefore it is much easier to use the HX711 which has already made breakout boards. However, some non-rotating results have been collected with the

	HX711	LTC2440
Resolution (bits)	24	24
Max Sample Rate (Hz)	80	3500
Voltage Range (V)	± 0.5	± 2.5

Table 3.1: Comparison of two chips that can be used to read the strain. HX711 data is from [10] and LTC2440 data is from [21].

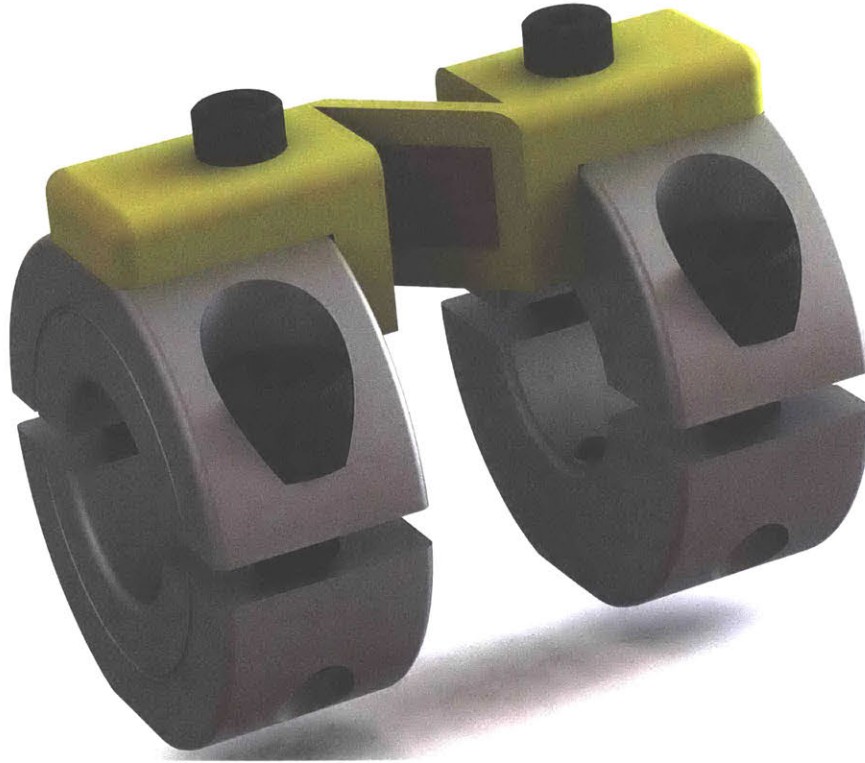


Figure 3-7: A rendering of the novel torque sensor.

LTC2240.

3.2.4 Construction

The sensor is relatively simple to produce. Furthermore, many of the parts can be made with relatively low precision without significantly affecting the accuracy of readings. Many of the tolerance issues can be corrected by calibration after the sensor is assembled. Engineering drawings of the prototype are presented in Appendix A and wiring diagrams are shown in Appendix B.

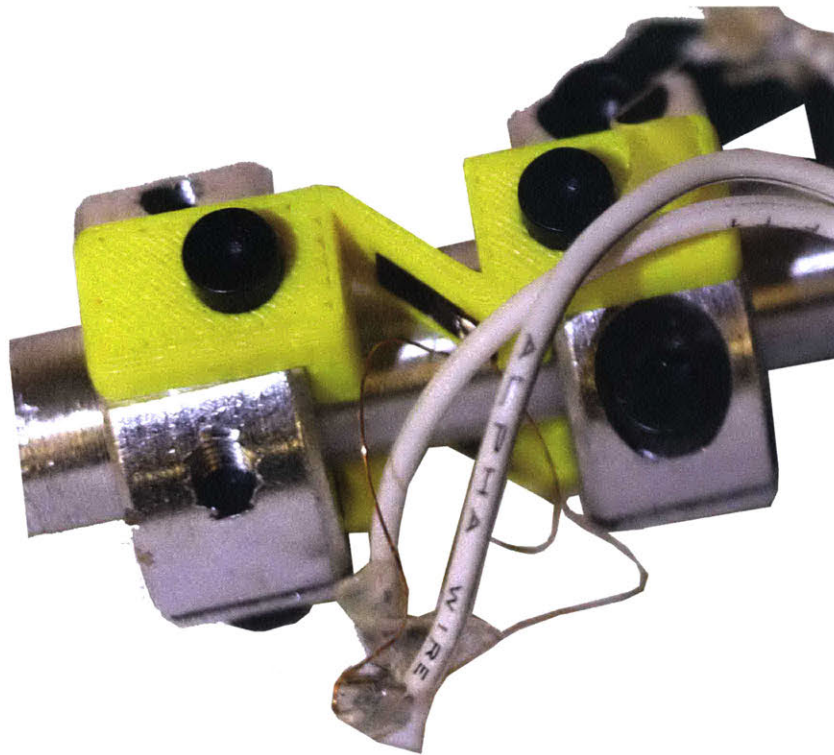


Figure 3-8: The new torque sensor as made.

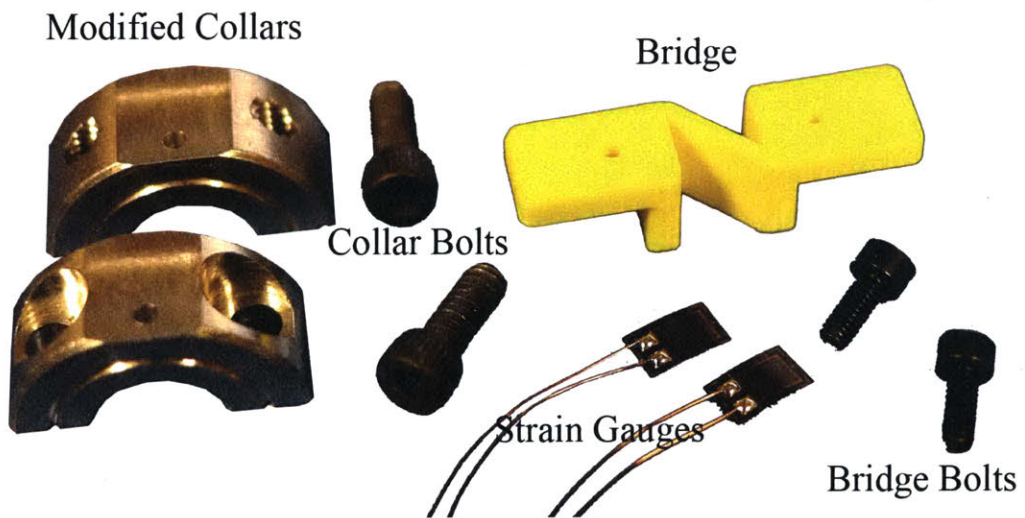


Figure 3-9: Exactly half the parts required for the torque sensor.

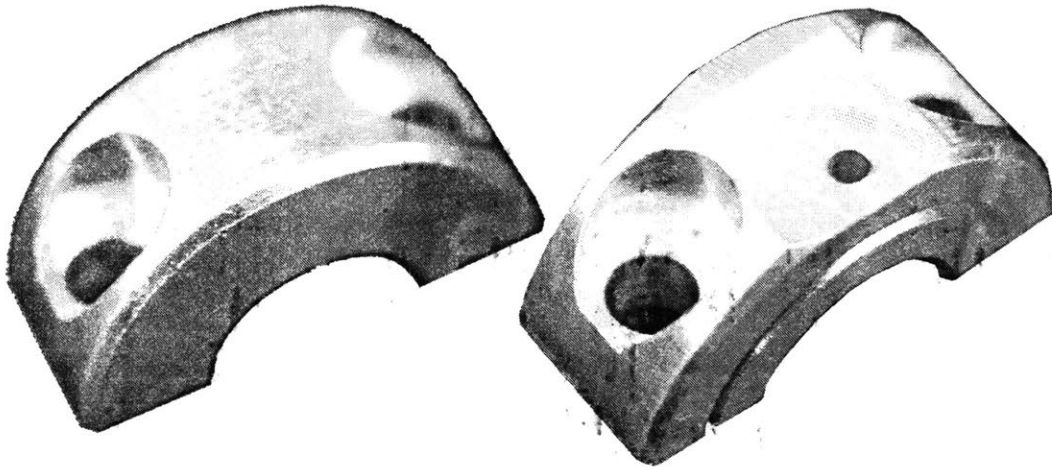


Figure 3-10: The aluminum collars before (left) and after machining (right)

The first step is to machine standard shaft collars into the collars used in the sensor. First, a facing cut is used to add a flat face on the cylindrical surface. Next, a hole is drilled and tapped in the flat face. This operation can be completed with fixing the part once in a manual mill. The initial and final collars are pictured in Figure 3-10.

Next, the bridge must be fabricated. The version in the prototype was 3D printed with acrylonitrile butadiene styrene (ABS) using the fused deposition method (FDM). The flexure zone thickness was limited by the resolution of the printer used (a Stratasys Fortus 250mc).

Final assembly is quite easy. Strain gauges are glued to the flexure zone of the bridge. Strain gauges should then be soldered to the ADC chip. The bridges are then bolted to the collars. The collars are then bolted to the shaft. It is helpful if the sensor is read while the collars are being bolted to ensure the sensor is not saturated by a misalignment. More details of this process are presented in Appendix G.

3.3 Results

3.3.1 Stationary vs. Rotating Shafts

Rotating the device requires significant development of other components of the sensor. First of all, wireless communications or power must be installed. In addition, all components must be secured to withstand the considerable centripetal forces experienced while spinning. Finally, it is difficult to induce known dynamic torques. While a known static torque can be easily applied by measuring a force on a lever, no such simple solution exists for spinning shafts.

Rotation can also add several phenomena that can interfere with calibration of the sensor. First, if the shaft is relatively thin, the weight of the sensor can cause bending (this is useful in Section 3.3.3) which interferes with the calibration. Vibrations caused by a number of sources, such as ball bearings or slight shaft imbalances, also cause noise in the torque signal.

3.3.2 Stationary Results

For these reasons, the sensor was initially tested while stationary. One end of the shaft was fixed while a torque was applied to the other end. All tests in this section were done with a 3/8" (9.5 mm) aluminum shaft.

In the first experiment, different weights were applied to a lever arm to induce a known torque on the shaft as shown in Figure 3-11. The lever arm used was 142.5 mm in length and the weights varied from 2.4 N to 12.7 N. This results in torques from 0.34 N m to 1.81 N m. The output from the torque sensor, T_S , and the calculated applied torque over time are shown in Figure 3-12 and Figure 3-13. As shown in Figure 3-14, these results are very linear. Furthermore, the scaling factor is consistent while the zero offset does drift some.

This experiment shows several important characteristics of the sensor. As shown in Figure 3-14, these results are very linear. In the figure, 750 to 1000 points were averaged for each experimental data point. As the mass was manually changed,

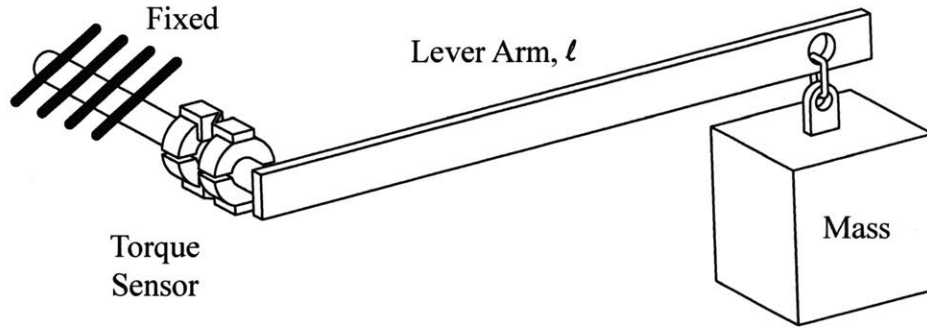


Figure 3-11: The experimental setup used to calibrate the torque sensor by varying the weight applied on the lever arm (not to scale).

the quickness of the sensor response cannot be accurately calculated from this data. However, it is clearly less than one second. Some of the noise at steady state can be attributed to the nature of the setup: the force exerted on the lever arm by the weight is subject to airflows on the weight in the room where it was tested. Finally, some of the offset drift can be seen at the first torque reading and the first two zero readings in Figure 3-12. This is part of the warm-up of the sensor.

In the second experiment shown in Figure 3-15, a force meter was used to measure a varying force, F , applied on a lever arm of ℓ . The results of this experiment are shown in Figure 3-16. It is clear that the sensor tracks the changing torque fairly accurately. The drift can be seen towards the end of the sample. In addition, small errors can be seen at high torques, indicating possible saturation.

3.3.3 Rotating and Bending Results

In order to collect data from a rotating the shaft, the sensor and shaft were inserted into the benchtop system described in Section 2 and shown in Figure 2-2. As the shaft is supported on both ends, the weight of the sensor causes the shaft to bend downwards. When the shaft rotates, the bending force rotates from the shaft and sensor's perspective. This creates a fluctuating bending. Therefore, at a constant angular speed and torque, the signal should be a sine wave. The frequency is the angular speed of the shaft, the amplitude is the amount of bending, and the offset is the torque. As the bending remains stationary while the shaft rotates, this should

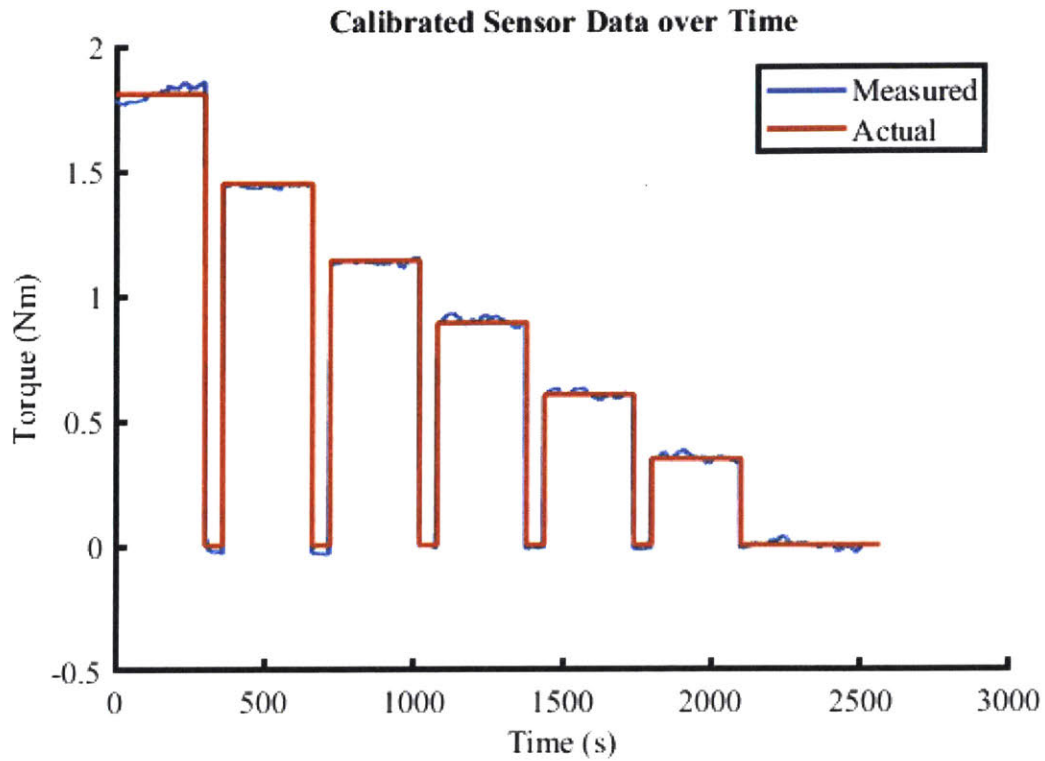


Figure 3-12: Output of the torque sensor and applied torque over time.

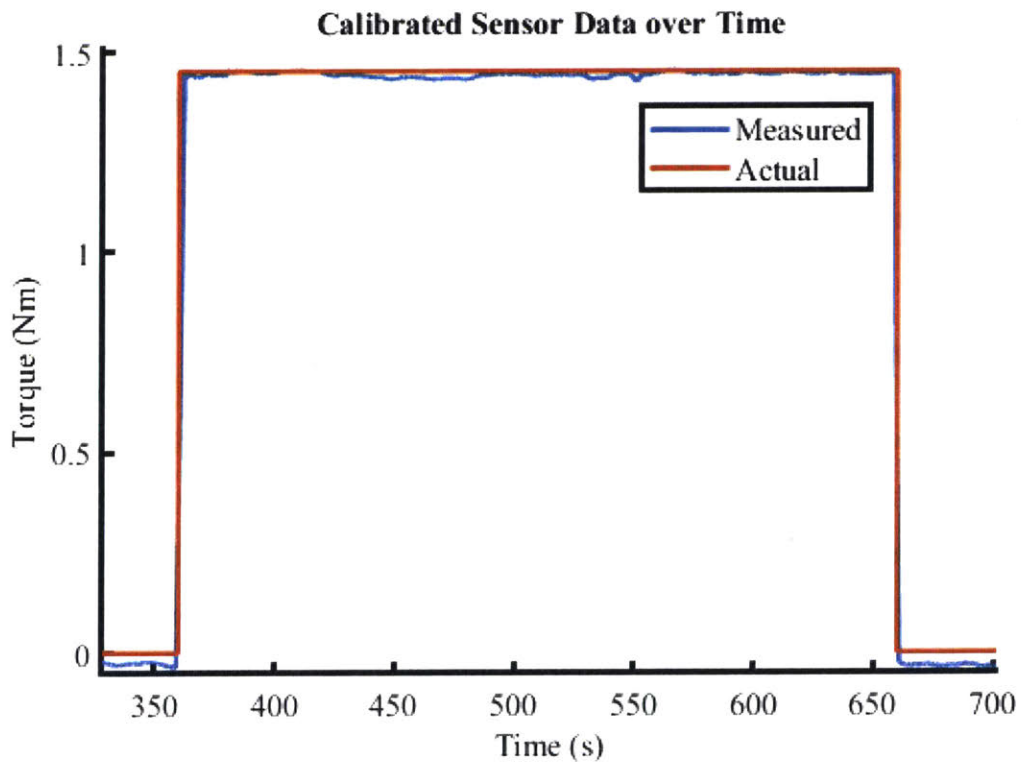


Figure 3-13: A zoomed in view of Figure 3-12.

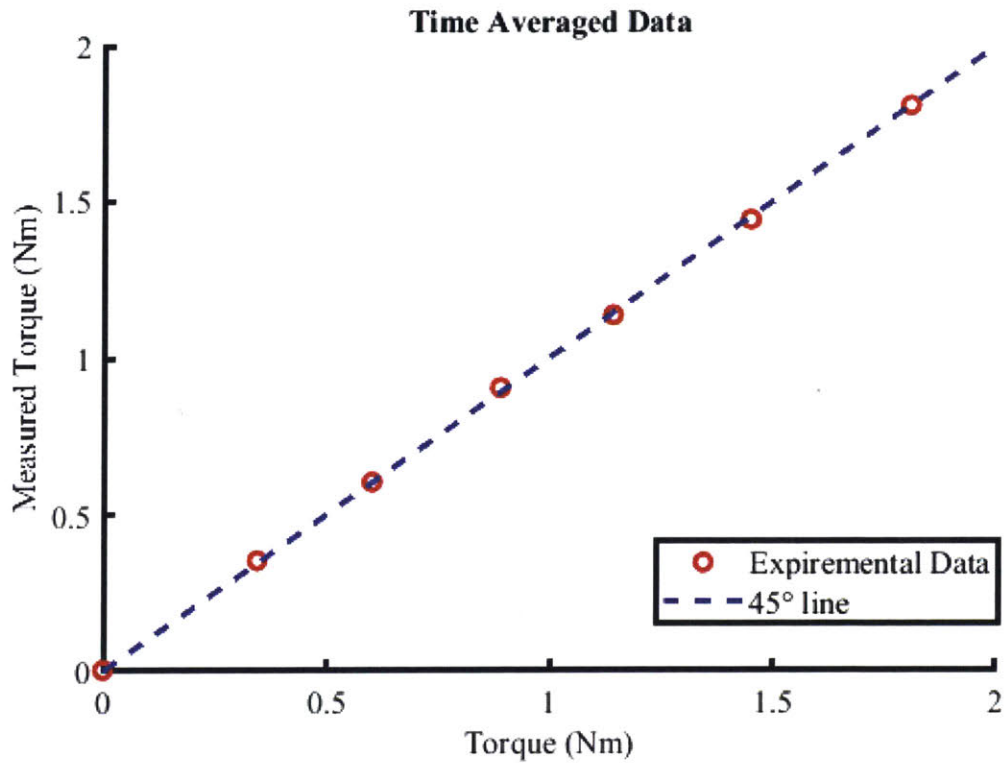


Figure 3-14: The output of the torque sensor versus the applied torque. The relationship is linear. The raw data is presented in Figure 3-12.

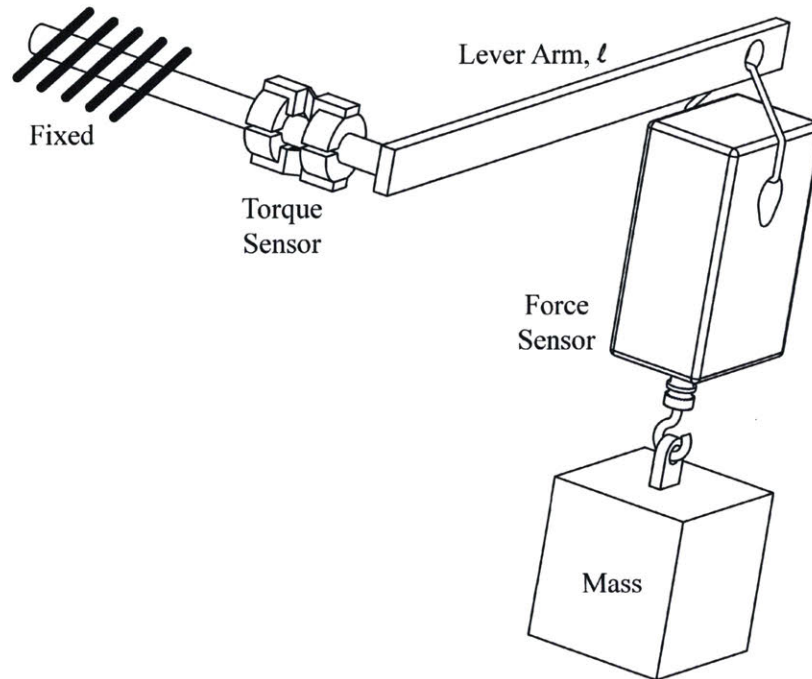


Figure 3-15: The experimental setup used to calibrate the torque sensor using a purchased force meter to measure applied on the lever arm (not to scale).

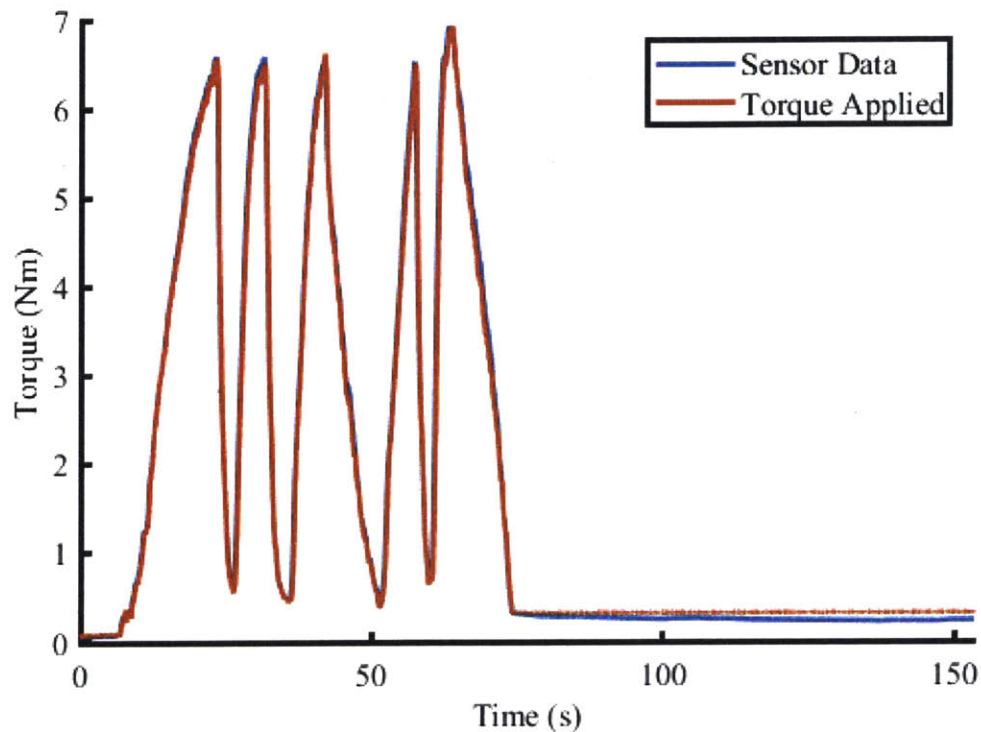


Figure 3-16: Output of the torque sensor and applied torque over time.

cause the signal to fluctuate between the positive and negative amount of strain induced by bending plus the strain caused by torque.

Due to the reasons explained in Section 6.2, the sensor samples in bursts rather than continuously. One of these bursts is pictured in Figure 3-17. As expected, it appears to be a sine wave. As apparent in the figure, there are cyclical anomalies every other cycle. The cause of this is unknown. However, the power spectrum of this signal, shown in Figure 3-18, indicates that occurs at around 4 Hz.

In a separate but similar experiment, the sensor was run at various speeds and the power spectrum was taken of the signal at each speed. As shown in Figure 3-19, the dominant frequency is at the shaft speed.

3.3.4 Long Term Drift

The most significant issue is the drift in the reading over long periods of time. This occurs with both the HX711 and LTC2440 chips as shown in Figures 3-20 and 3-21,

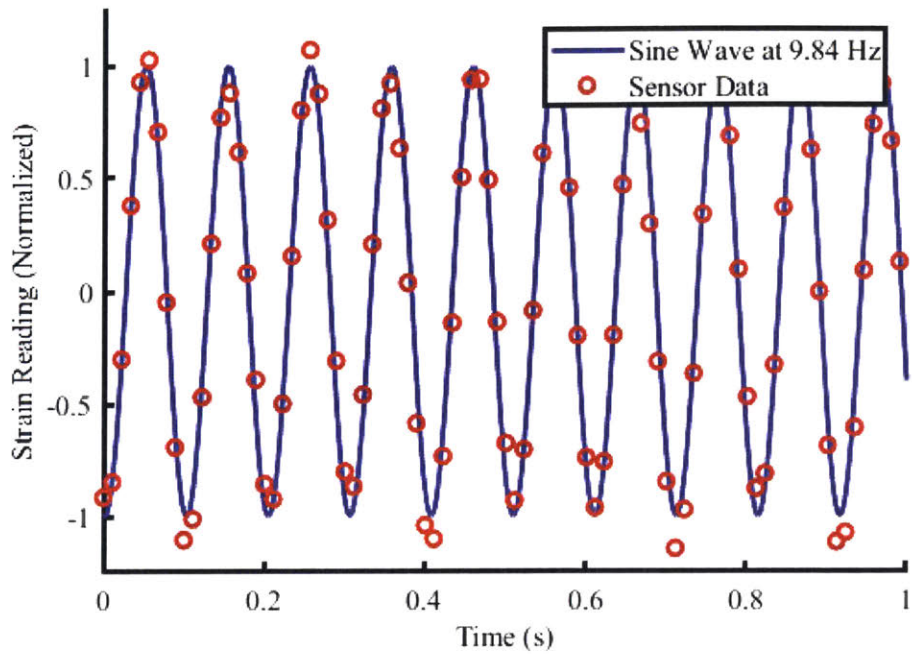


Figure 3-17: A burst reading from the torque sensor shows the torque as the mean reading and the amount of bending as the amplitude. The frequency of the signal is the angular speed of the shaft (600 RPM or 10 Hz).

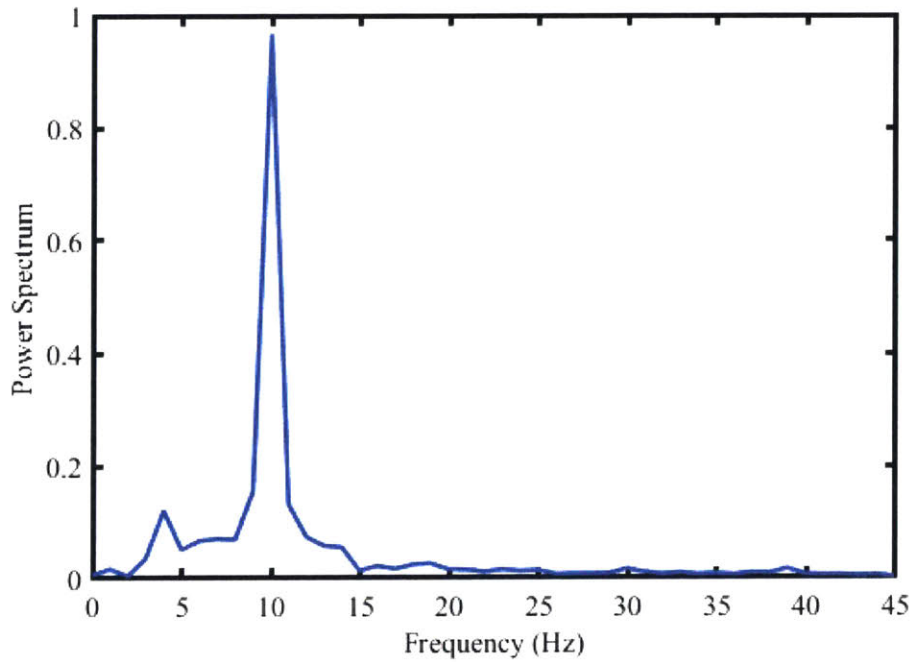


Figure 3-18: The power spectrum of the signal shown in Figure 3-17 shows the clear dominant frequency of 10 Hz.

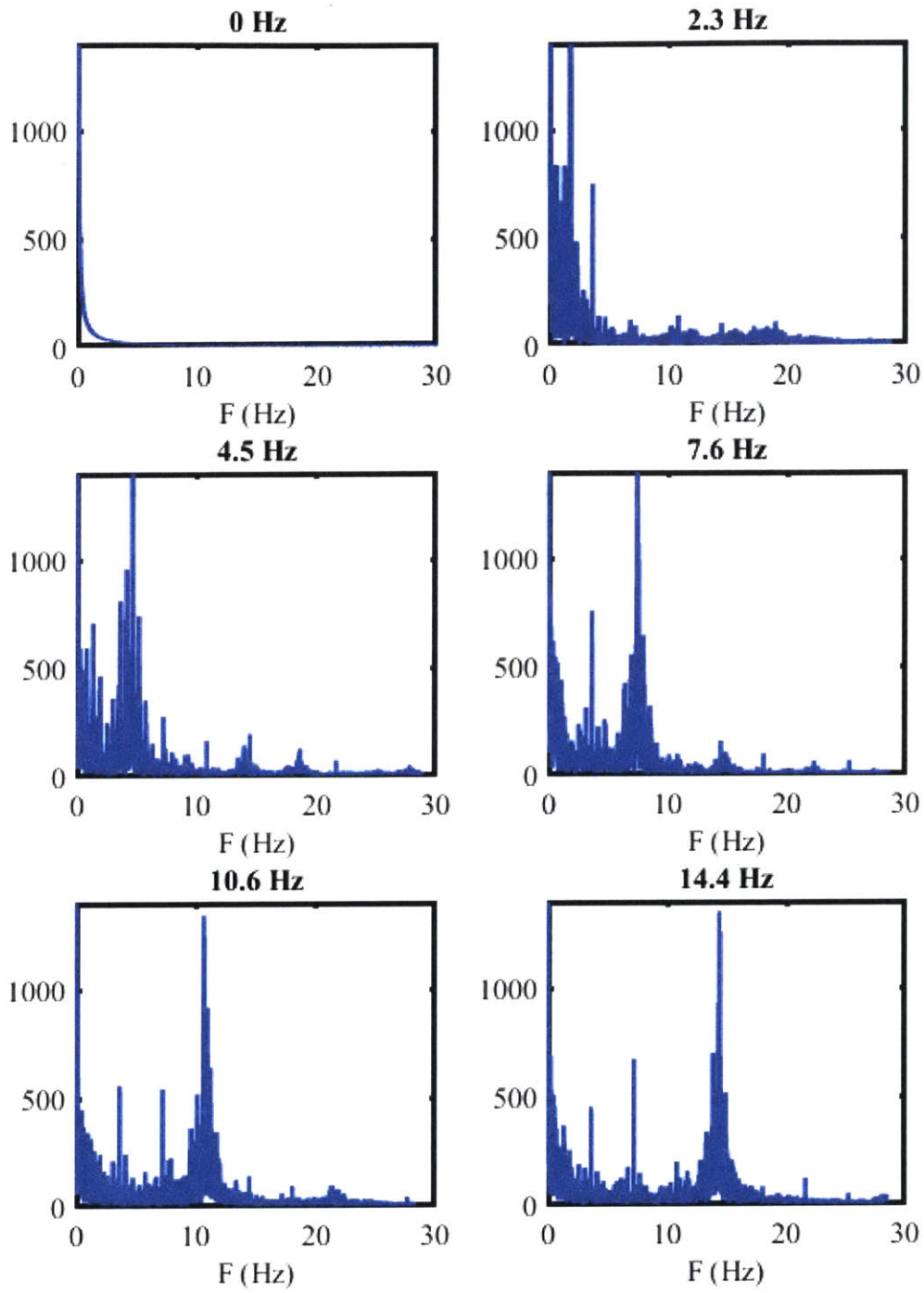


Figure 3-19: A power spectrum of the torque signal shows that the dominant frequency of the signal is the shaft speed (title of each graph).

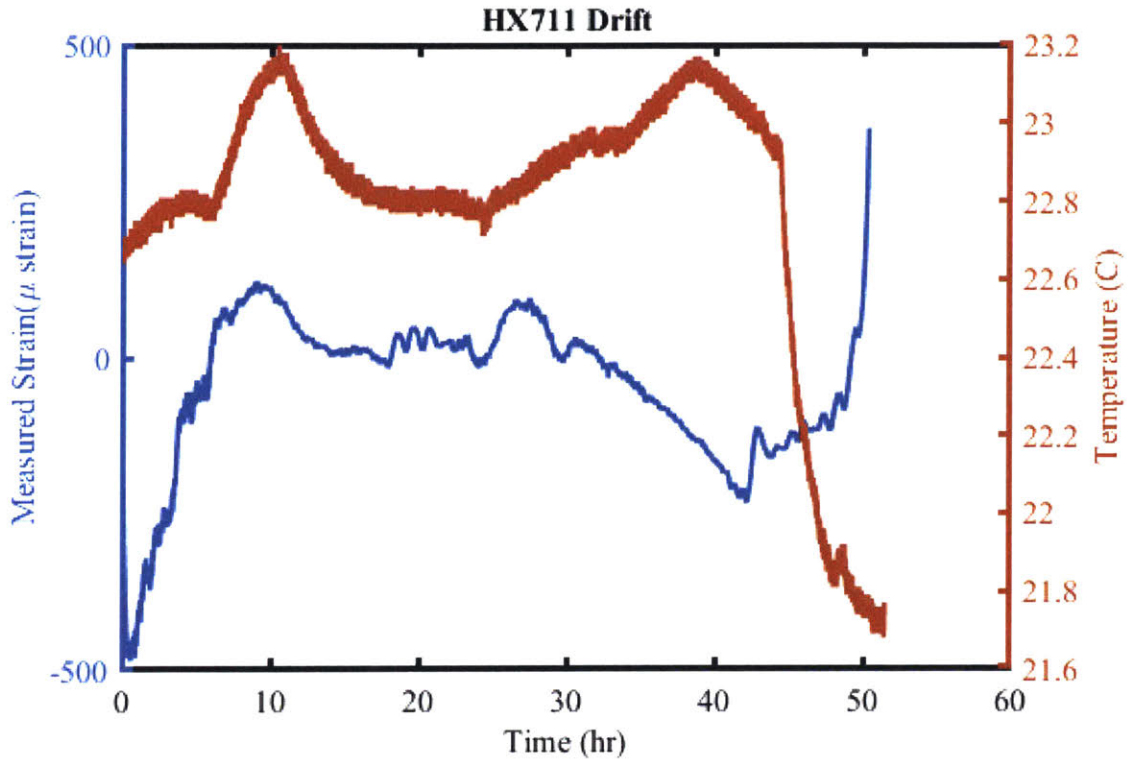


Figure 3-20: Drift in the sensor reading over time with the HX711 chip.

respectively. In these experiments, the sensor was stationary with no applied torque on the shaft. This is common with strain gauges and is often attributed to various thermal effects [18].

Therefore, it may be possible to measure temperature and use it to correct the strain reading. As shown in Figure 3-20, measured strain with the HX711 and temperature appear to sometimes be correlated and other times not. For the first 20-25 hours, they seem to be correlated. The large spike in measured strain at the end can be attributed to the battery dying. Similar results are found with the already lower-drift LTC2440, shown in Figure 3-23. There appears to be a correlation between strain and temperature after about 27 hours, but not before. The temperature versus measured strain is shown for each chip in Figures 3-22 and 3-23.

Using a linear fit, a correction can be applied. For example, if only the data from times between 25 hours and 32 hours are taken from Figure 3-21, then the correlation is very strong as seen in Figure 3-24. The linear fit can then be used to correct the

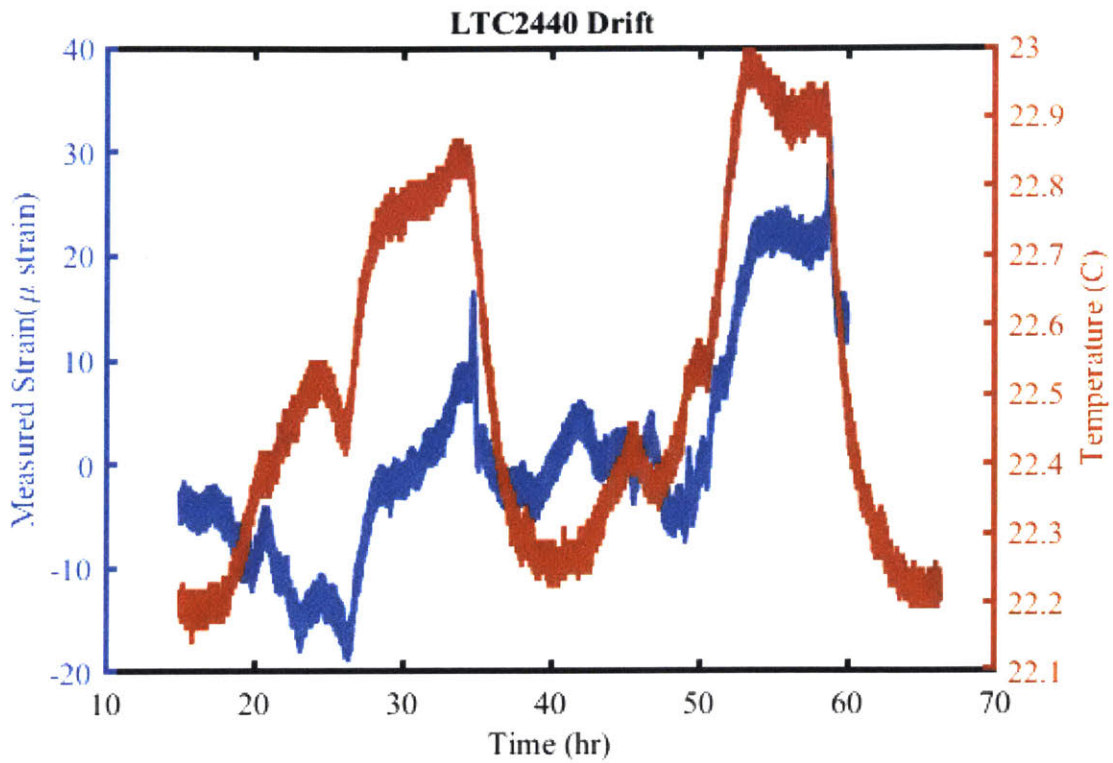


Figure 3-21: Drift in the sensor reading over time with the LTC2440 chip.

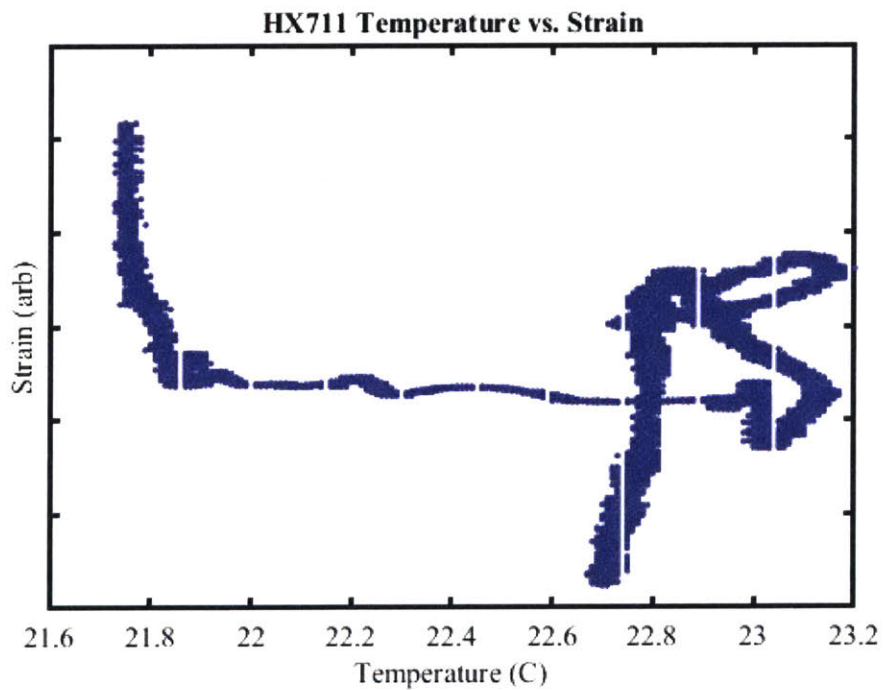


Figure 3-22: Chip temperature versus strain reading for the HX711 chip. The correlation appears to be weak.

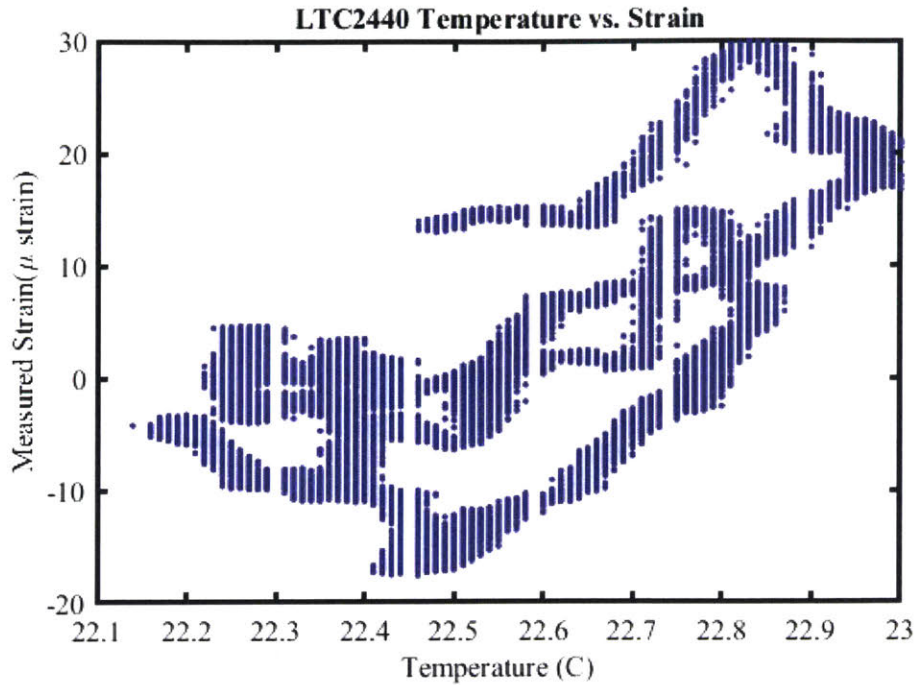


Figure 3-23: Chip temperature versus strain reading for the LTC2440 chip. There appears to be some very weak correlation.

signal. If x_t is the temperature reading and y_{zero} is the strain reading at zero torque, regression can be used to find linear constants a_r and b_r :

$$y_{zero} \approx a_r * x_t + b_r \quad (3.22)$$

Using this analysis, the corrected strain, $y_{corrected}$, from the strain input y_{raw} and temperature input x_t is approximately:

$$y_{corrected} = y_{raw} - a_r * x_t - b_r \quad (3.23)$$

This correction is shown in Figure 3-25. While this methodology was not implemented into the sensor, it could easily be done before any production.

Even without corrections and with the LTC2440 chip, the sensor has reasonable stability over a given minute or hour. As shown in Figures 3-26 and 3-27, the drift in the short term is quite minimal. This means that the sensor can retain reasonable accuracy as described in the figures if calibrated every hour (or minute). It should

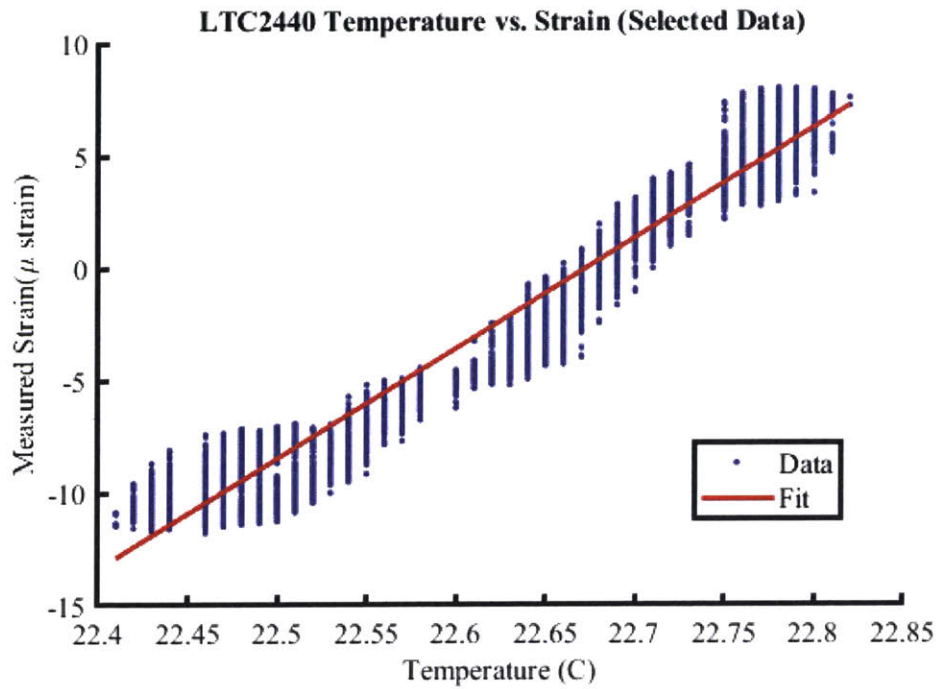


Figure 3-24: Chip temperature versus strain reading for the LTC2440 chip for times 25 hours to 32 hours as shown in Figure 3-21. The correlation is strong and a linear fit is shown.

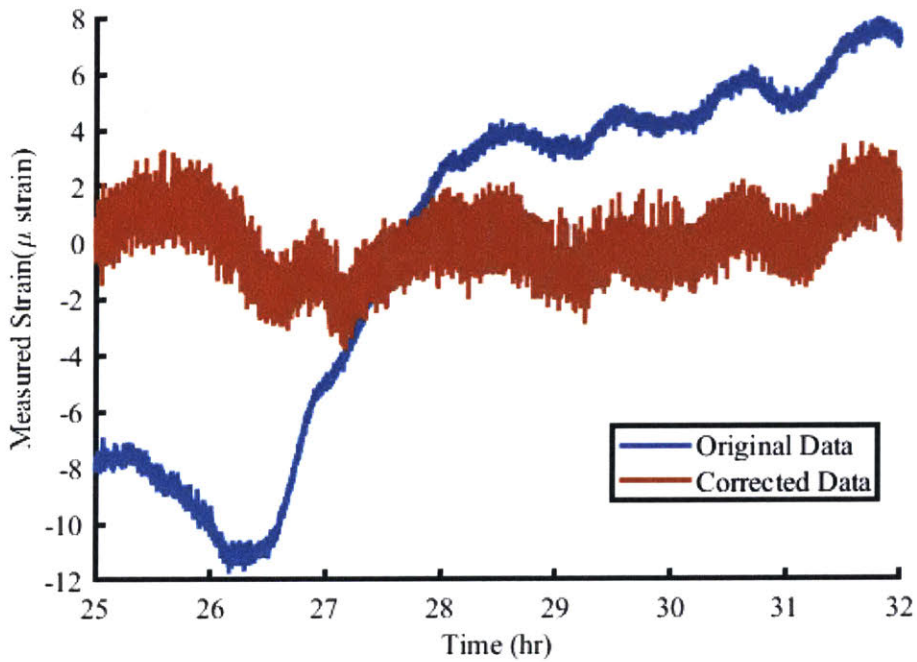


Figure 3-25: Temperature correction factors can be applied to improve the reading of the LTC2440.

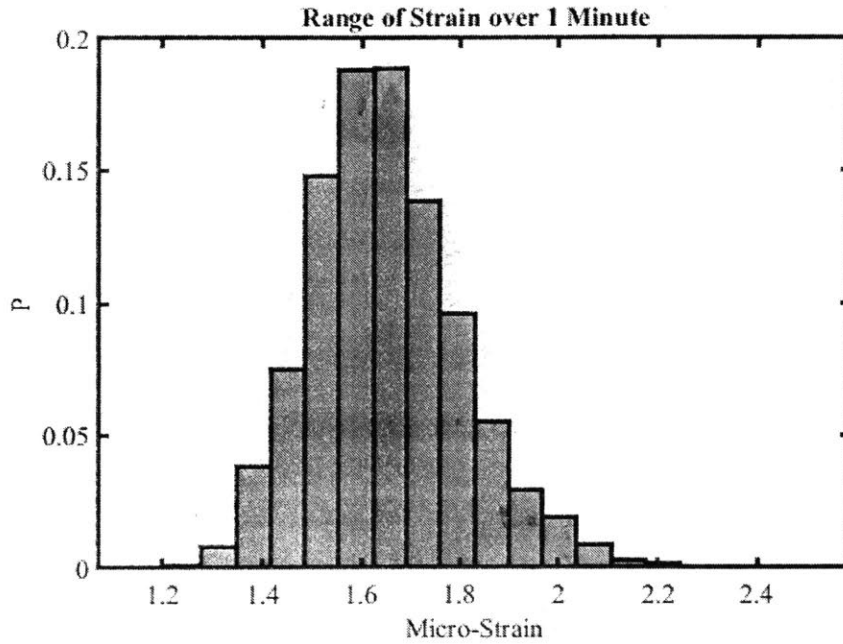


Figure 3-26: The probability distribution of total strain drift over a minute. The vast majority of samples have less than 2 micro-strain of drift.

be noted that the samples used here were truncated from earlier figures. The drift at the very beginning of some of the earlier samples is attributed to "warm-up" and some at the end is due to the battery voltage dropping. In most cases, the warm-up was found to be around 20 to 30 minutes.

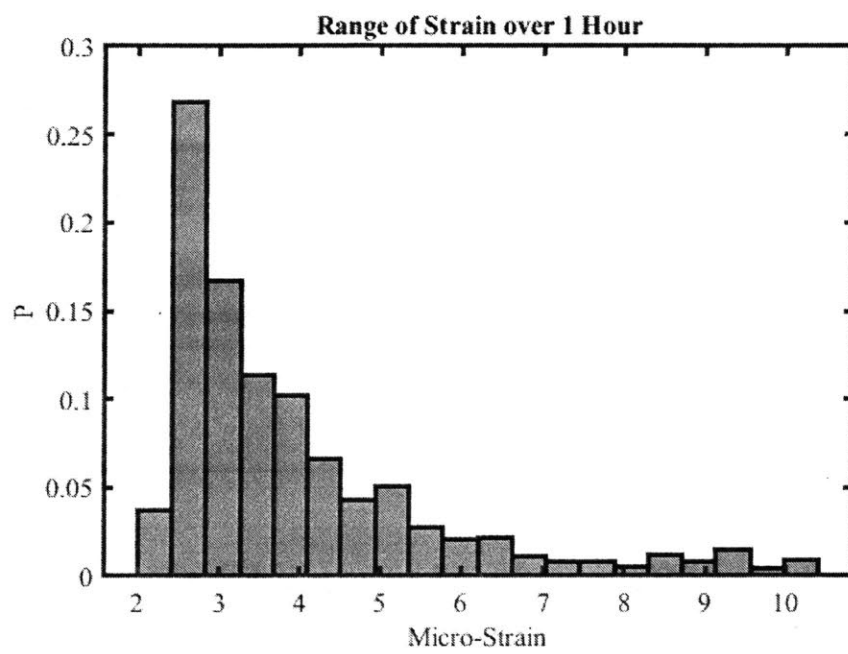


Figure 3-27: The probability distribution of total strain drift over an hour. The drift is on the order of micro-strains.

Chapter 4

Acceleration

Measuring the speed and vibration of the shaft is critical for the sensor. Speed is the flow variable of the shaft system and can be used to determine power. It is also one of the critical parameters of the TEX. In addition, vibrations are common symptoms of problems in machinery. By utilizing an accelerometer and associated methods as described in this section, both speed and vibrations can be measured.

4.1 Theory

As with torque, many solutions exist for measuring shaft speed. Hall effect sensors can be used to determine when a point on the shaft passes by. Differentiating this signal produces the angular speed [31]. Similarly, encoders and photo tachometers track when parts of the shaft pass and differentiate for speed [15]. In addition, motors can be driven by the shaft and the output voltage is approximately proportional to the speed [13].

However, most of these methods require fixed components that do not rotate with the shaft. An accelerometer, however, can rotate with the shaft and is relatively inexpensive. There are at least three ways to estimate rotational speed from an accelerometer attached to the shaft: angular integration, centripetal acceleration, and frequency analysis. Furthermore, an accelerometer can be used to detect the frequency and amplitude of vibrations present on the shaft [34]. This frequency data

can be useful in detecting problems in the system.

Integration of the angular acceleration causes significant error. This is similar to how accelerometers are used to estimate position and has the same issues. If there is any calibration error, then speed will be read as constantly increasing or decreasing even if the angular speed is constant. Therefore, this method is not used.

Determining angular speed precisely from the magnitude of radial acceleration requires accurate calibration. If the radius of the shaft r_s is known and the magnitude of the centripetal acceleration (radial direction) is a_c , then the angular speed ω is as given by Equation 4.1. The downside of this approach is that it requires the accelerometer to be calibrated accurately, although it is not as sensitive to calibration as the integration method. Any error in a_c directly affects the estimate of ω as shown in Equation (4.1).

$$\omega = \sqrt{\frac{a_c}{r_s}} \quad (4.1)$$

Gravitational effects affect the readings of radial and angular acceleration in most non-vertical and rotating shafts. While this effect may be insignificant relative to the centripetal acceleration, it can also simply be averaged out if the sample rate is high enough. If enough data points from the signal are averaged through a cycle or across cycles, the gravitational effects in each individual data point cancel. In fact, these effects complement each other: at high speeds the centripetal acceleration is high so it minimizes the gravitational effect in the signal while at low speeds it is easier to have faster sampling relative to the shaft speed.

Analyzing the frequency of the radial or angular acceleration signals is this most resilient of the methods in some conditions. If the shaft is not vertical, at the radial and angular signals will fluctuate within a rotation due to gravity. In the case of a horizontal shaft at constant angular speed, one would expect the angular acceleration to vary from positive g to negative g each rotation, where g is the acceleration of gravity. When the accelerometer is on the top of the shaft, it will read zero angular acceleration. A quarter-turn later when it is on the side, it will read plus or minus

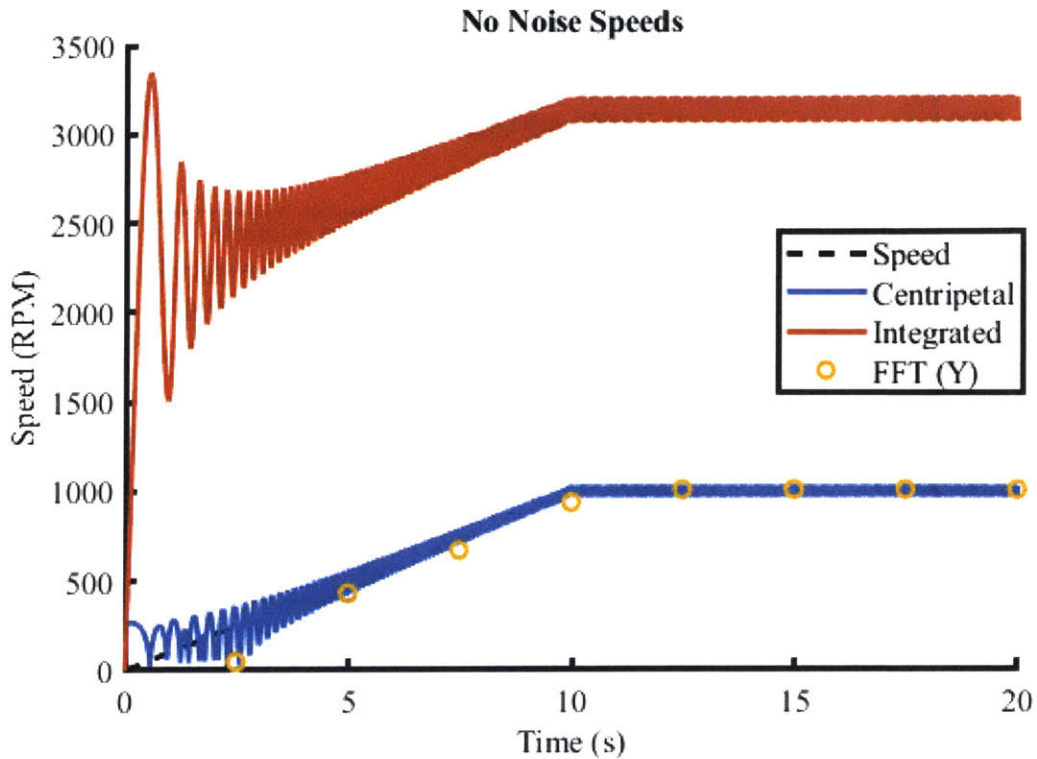


Figure 4-1: A simulation of the calculated speed from the accelerometer using three different methods assuming no noise in the signal.

g , depending on the orientation. Another quarter-turn so it is on the bottom of the shaft will make it read zero once again. Finally, one more quarter turn will put it on the opposite with a reading of minus or plus g . Similarly, the radial acceleration would vary from $a_c + g$ to $a_c - g$. As long as the sampling rate is sufficiently high (at least twice the angular frequency) and the angular speed does not change rapidly, a power spectrum will clearly show the angular speed as the dominant frequency in the signal.

In the simulation shown in Figures 4-1, 4-2, and 4-3, a shaft starts at rest and is constantly accelerated to 1000 RPM (16.7 Hz) over 10 seconds. It then spins at a constant speed for 10 more seconds. The accelerometer is simulated as being 12.5 mm from the axis of rotation. The code used to run this simulation is included in Appendix D. All the signals oscillate as the accelerometer is rotating relative to gravity. It is clear that even with perfect signals as shown in Figure 4-1, the integration method has a substantial error. This is due to the effects of gravity. Whenever the accelerometer

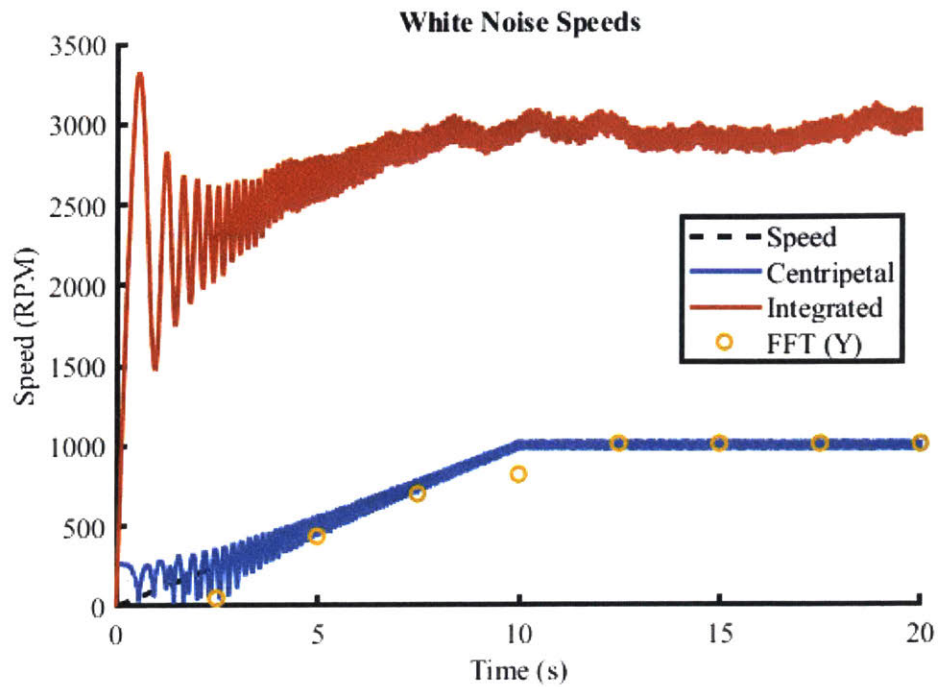


Figure 4-2: A simulation of the calculated speed from the accelerometer using three different methods with white noise added to the signal. The noise was added at a signal-to-noise ratio of 5.

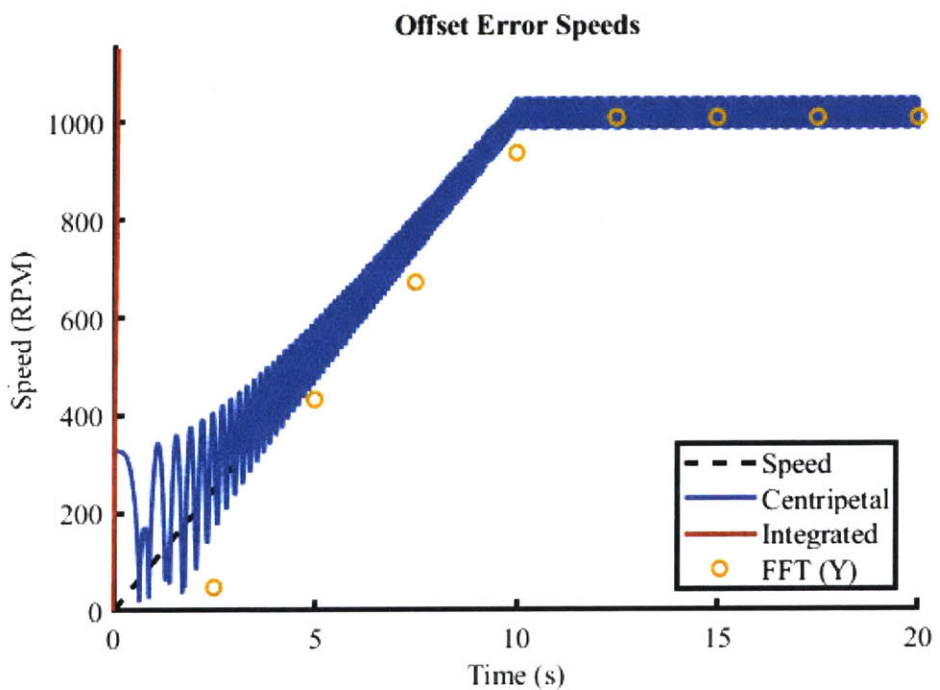


Figure 4-3: A simulation of the calculated speed from the accelerometer using three different methods with a static constant offset error of 0.5 g added to the signal.

is on the side of the shaft, even if the shaft is not rotating, the integration method integrates the gravity reading treating it as an angular acceleration. This creates the large offset seen in Figure 4-1.

In all the simulations, the frequency method does not work well in the transient. This is because it averages over a time period. However, it performs well in steady state. As shown in Figure 4-2, both the frequency and centripetal acceleration methods track well in the steady state with white noise in the signal. If there is a static offset error as shown in Figure 4-3, then the frequency method is more accurate than the centripetal acceleration method. Ideally, these two methods could be used in combination: the frequency method for steady speed accuracy and the centripetal acceleration method for measuring transient states.

Additionally, other frequencies in the accelerometer power spectrum are likely the results of vibration. The frequency and amplitudes of these vibrations are additional useful information this device can collect.

4.2 Selection

The accelerometer only has a few requirements: size, sampling rate, accuracy, resolution, and range. The accelerometer must be able to fit on the sensor. In the case of the prototype built, most accelerometers fit making this requirement arbitrary. If the frequency method is used, then the sampling rate must be at least twice the angular speed of the shaft to meet the Nyquist criterion. If the shaft is spinning at 1000 RPM (16.7 Hz), then the sampling rate must be greater than 34 Hz. Ideally, this should be considerably higher, and in most accelerometers it is. The accuracy depends on the required accuracy of the speed and vibration data, which varies by application. The accelerometer will experience significant acceleration as shown in Equation (4.1). In the case of 1000 RPM (104.7 rad s^{-1}) and a r_s of 25 mm, this acceleration is about 274 m s^{-2} or 28 g. Clearly, this should be within the maximum acceleration of the accelerometer.

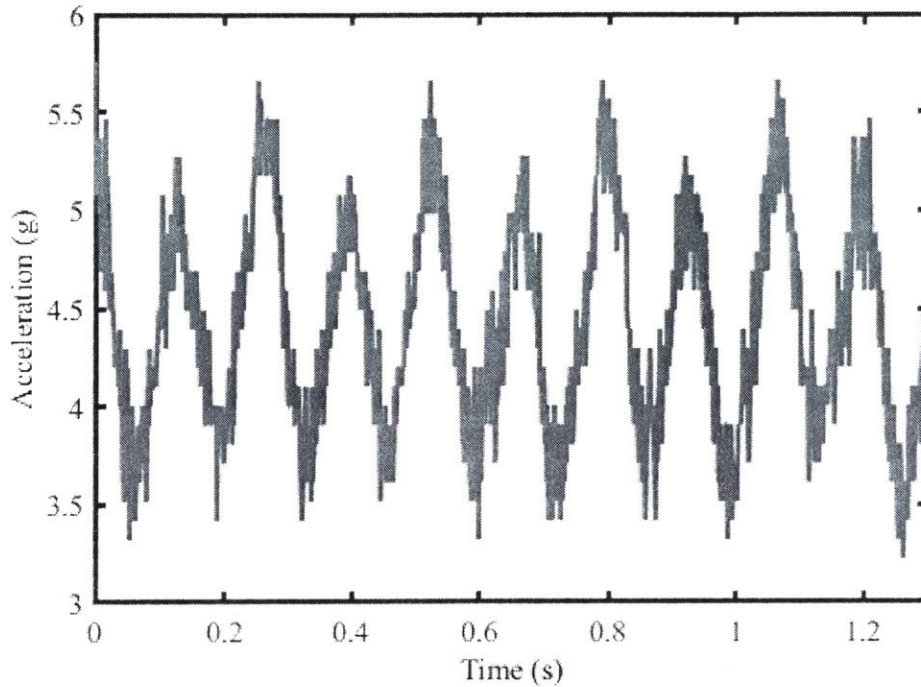


Figure 4-4: A actual burst from the accelerometer is shown. It is close to a sine wave as predicted. The sampling rate was 750 Hz.

4.3 Results

As shown in Figure 4-4, the signal from the accelerometer is close to what is predicted in Section 4.1. The acceleration in the radial direction is approximately a sine wave with a frequency of the angular speed of the shaft and an offset of the centripetal acceleration.

To ensure the proper method was chosen, an encoder was used to test the methods. The different signal outputs are shown in Figure 4-5. It is clear that the radial acceleration method and the frequency method both track the encoder well. Either of these methods can be used depending on the application. In fact, both can be used if only the radial acceleration is transmitted, meaning both the angular and axial acceleration can be ignored.

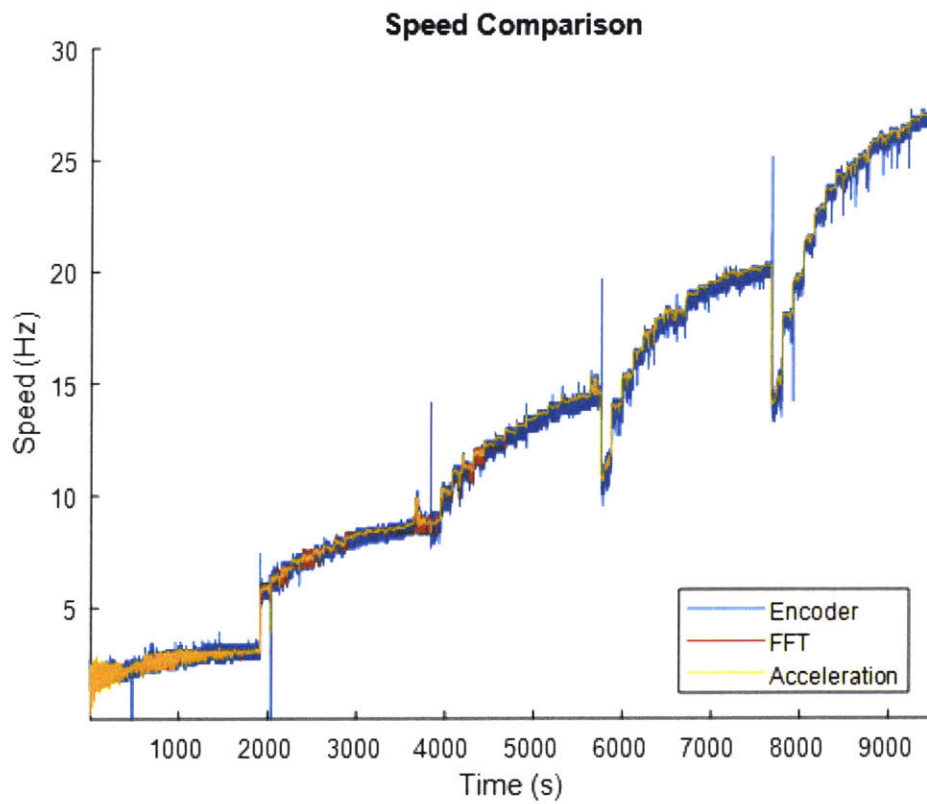


Figure 4-5: A comparison of methods used to determine angular speed. All are close.

Chapter 5

Power Generation

The sensor requires electrical power to read and transmit the data. In order to simplify the installation process, wireless power methods are preferred over wired ones. One method that is promising is to use a weight and brushless motor to harvest energy from the rotating shaft. Analysis of this method and some experimental results are presents.

5.1 Requirements

The power consumption of most components is shown in Table 5.1. One clearly lacking element is wireless communications. Wi-Fi uses about 300 to 850 mW to transmit data [37] while Bluetooth and Bluetooth LE (Low Energy) use 2.5 mW and 0.025 mW, respectively [38]. The chosen technology will depend on a specific applications sample rate and range requirements. The prototype uses Wi-Fi, the fastest and most power intensive of the options. Wi-Fi, in fact, dominates the energy budget, while a slower technology like Bluetooth LE would comprise only a small part of it. If Wi-Fi is used, the total power requirement approaches 1 W.

The Wheatstone bridge consists of strain gauges (resistors) that consume significant power. As shown in Figure 3-6, the bridge consists of two sets of resistors in series. These sets are in parallel. If each strain gauge has resistance R , then each set has effective resistance R_{set} :

$$R_{set} = R + R = 2R \quad (5.1)$$

As each set is in parallel, the total effective resistance of the bridge R_{bridge} is given by:

$$\frac{1}{R_{bridge}} = \frac{1}{R_{set}} + \frac{1}{R_{set}} = \frac{2}{R_{set}} \quad (5.2)$$

Combining Equations (5.1) and (5.2) yields:

$$R_{bridge} = \frac{1}{2}R_{set} = R \quad (5.3)$$

The power dissipated P_{dis} through any circuit element is simply the product of the voltage V and the current i :

$$P_{dis} = iV \quad (5.4)$$

For a resistor, Ohm's law must be satisfied:

$$V = iR \quad (5.5)$$

In the Wheatstone bridge, the applied voltage V to the resistors is the supply voltage V_s . Finally, combining Equations (5.3), (5.4), and (5.5) yields that the power dissipated in the Wheatstone bridge is:

$$P_{dis} = \frac{V_s^2}{R} \quad (5.6)$$

Using R of $350\ \Omega$ and V_s of $3.3\ \text{V}$, the power dissipated is found to be about $31\ \text{mW}$.

5.2 Background

As stated in Section 1.3, the sensor must rotate with the shaft. This means that traditional wires cannot be used to send power to the sensor as they would be tangled. Therefore, another method needs to be used.

Component	Power (mW)	References
Microcontroller	11-22	[9]
Accelerometer	1	[6]
Strain Amplifier	5	[10]
Wheatstone Bridge	31	
Microphone	2	[1]
Bluetooth	2.5	[38]
Wi-Fi	300-850	[37]

Table 5.1: Device Power Requirements

One possible solution is a slip ring [46]. Utilizing brushes, this device allows wires to be run to the shaft and to be electrically connected to wires that rotate with the shaft. However, these devices have several limitations. First, the mechanical components wear and can fail, requiring significant maintenance [30]. Furthermore, they can be difficult to install as most need to be slid onto the shaft axially. Last but not least, they require wires to be routed to near the sensor, further complicating installation.

Another option is induction charging, similar to how wireless charging works on modern cellphones. This technology creates oscillating electromagnetic waves that induce an electrical current on the receiver. This technology can also be used at very high powers, up to at least 22 kW [47]. While this could clearly meet the power requirements of the sensor, it does require routing wires to near the sensor, making it less than ideal.

The sensor could also gather energy from the environment using solar panels or vibrational energy harvesting. However, the size of the sensor is small and the ambient light available is extremely application dependent. In most situations, solar is estimated to not be able to provide sufficient power. If the shaft is well balanced as it should be, vibrations will be minimal. Furthermore, the amplitude and frequency of vibrations are very application dependent. Therefore, both vibrational and solar harvesting are likely not sufficient for the sensor.

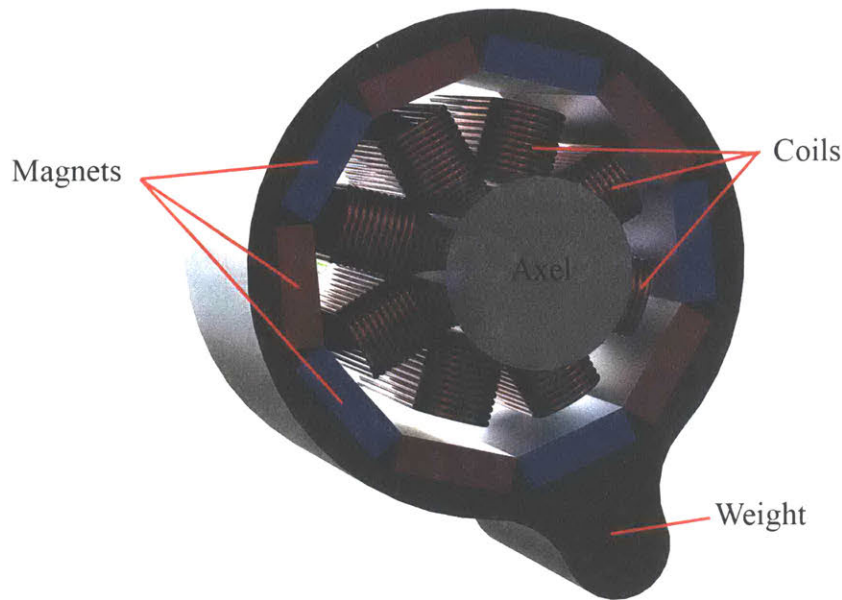


Figure 5-1: An example of a brushless generator is shown. It is simply a brushless motor driven in reverse.

5.3 Brushless Generator

5.3.1 Overview

The best solution is to use the shaft's energy to power the sensor by implementing a brushless generator. As shown in Figure 5-1, coils of conductive wire are fixed to the shaft. Then, permanent magnets are mounted around the shaft. These magnets should be stationary (not rotating with the shaft). As the shaft spins, the magnetic flux through each coil changes. This induces a current on the coil. Rectifying this current and boosting it to the needed voltage can provide the direct current power for the sensor. This is essentially an electric brushless motor being mechanically driven.

This method can provide large amounts of power. As the power required for the sensor is small relative to the mechanical power transferred through a shaft, the impact on the machinery should be acceptable. Furthermore, this system is easy to install and independent of most environmental factors.

In the case described above, the magnets are fixed by some apparatus to the stationary reference frame. However, in some applications, gravity could be used to

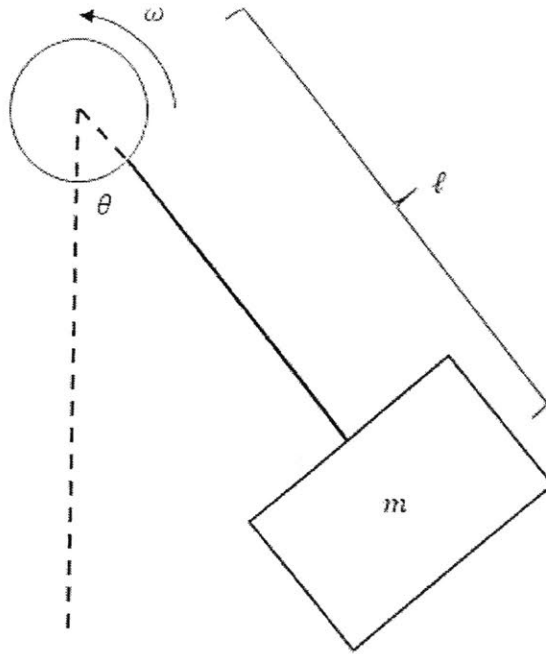


Figure 5-2: The generator can be considered to be just a pendulum.

hold the magnets stationary. If the magnets are free spinning about the shaft, then a weight can be added to hold the magnets in place. This may lead to a simpler installation process. Similar work is presented in [39].

5.3.2 Static Mechanical Model

This coil-magnet system can be modeled as a motor. A simplified diagram is shown in Figure 5-2. A mass m is suspended a distance ℓ from the axis of rotation. g is the acceleration of gravity and θ is the absolute angular displacement from vertical. The torque exerted by the pendulum on the motor shaft, τ_g is simply the radius at which it is applied, r , cross the force of gravity, F_g :

$$|\vec{\tau}_g| = |\vec{r} \times \vec{F}_g| = \ell m g \cdot \sin(\theta) \quad (5.7)$$

If the shaft spins with a speed ω , then the mechanical power exerted on the shaft, P_m is:

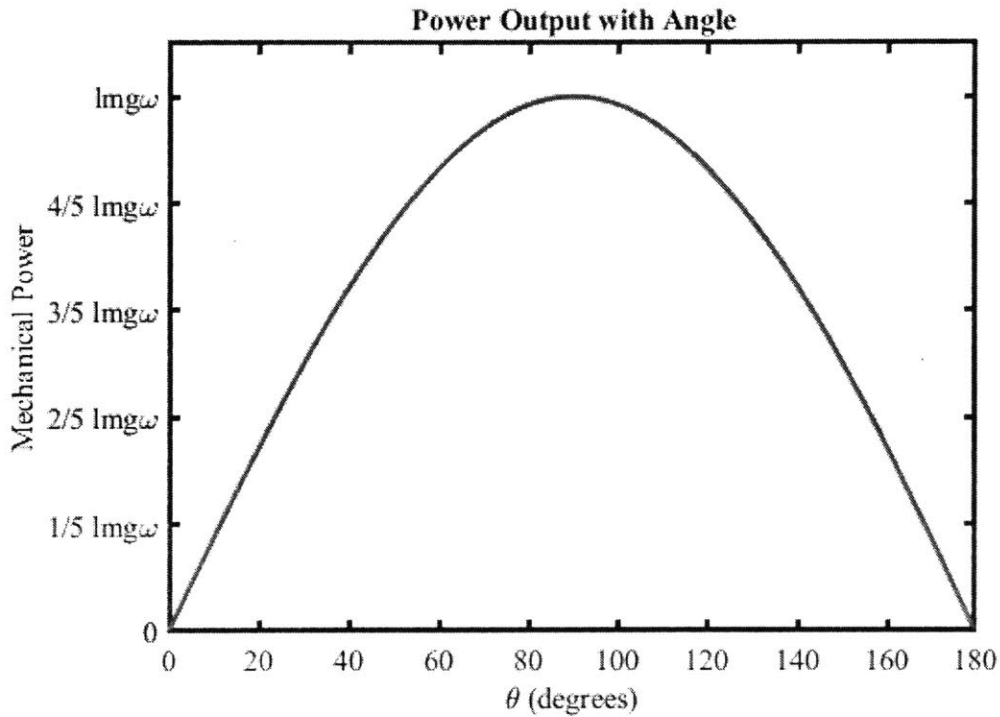


Figure 5-3: The mechanical power exerted on the generator depends on the angle of the pendulum from vertical.

$$P_m = \tau_g \omega = \ell m g \omega \cdot \sin(\theta) \quad (5.8)$$

Clearly, maximizing power is important for this system. Length and mass of the pendulum are design choices limited by the size of the sensor. Furthermore, the angular velocity is a given for the shaft and gravitational acceleration is constant. Therefore, the angle, θ , is the only operational parameter available. As shown in Figure 5-3, the maximum power point is achieved when $\theta = 90$ deg. At this point, power is simply $P_m = \ell m g \omega$.

Now, consider the motor modeled as an ideal gyrator with motor torque constant K_t attached to a resistor of resistance R (to model the load). The gyrator equations are:

$$\tau_m = K_t i \quad e = K_t \omega \quad (5.9)$$

Where τ_m is the torque exerted by the motor on the shaft, K_t is the motor torque constant, i is the current through the motor, e is the back EMF on the motor, and ω is the shaft speed. When the resistor is added, the equations simplify to:

$$\tau_m = \frac{K_t^2}{R}\omega \quad (5.10)$$

For the pendulum to be held at 90 deg in steady state, the torques in Equation (5.7) and Equation (5.10) must be equal. If length, angular speed, and mass have already been determined, then the ratio of K_t^2 to R is determined to achieve the 90 deg steady-state.

$$\tau_m = -\tau_g \quad (5.11)$$

$$\frac{K_t^2}{R}\omega = \ell mg \cdot \sin(\theta) \quad (5.12)$$

As θ is 90 deg, $\sin(\theta)$ is simply 1:

$$\frac{K_t^2}{R} = \frac{\ell mg}{\omega} \quad (5.13)$$

This system has a substantial vulnerability: instability. If a perturbation caused the pendulum to move up slightly (increase θ), then the gravitational torque is slightly reduced while the motor torque is constant. This will cause an upward acceleration. At this point, the pendulum will accelerate to spin with the shaft. It will not stop the next time $\theta = 90$ deg as it will carry inertia which will cause the cycle to repeat. One solution would be to have the pendulum stay at an angle of less than 90 deg, which sacrifices power for stability.

This also does not maximize power at sub-optimal speeds. If the parameters are set so the system achieves 90 deg at maximum speed but then the speed is lowered, power falls far worse than if the resistance could be optimized for any speed. If the angle is always optimized (90 deg), Equation (5.8) shows that power increases linearly with speed. If it is not optimized, then Equation (5.12) shows that the sine of the angle is proportional to angular velocity (up to 90 deg). When this is factored into

Equation (5.8), it is clear to see that now power is proportional to angular speed squared. The solution to both the instability and non-ideal power output is to add active controls to maintain the optimal angle. In order to do this, a dynamic model of the system is needed.

5.3.3 Dynamic Mechanical Model

For the dynamic model, it is important to note that the motor "sees" a different speed, ω_m than the shaft, defined as:

$$\omega_m = \omega - \dot{\theta} \quad (5.14)$$

where ω is the shaft speed and θ is the angle of the pendulum from vertical, both as shown in Figure 5-2. This new assumption can be combined with Equation (5.10) to show that now:

$$\tau_m = \frac{K_t^2}{R}(\omega - \dot{\theta}) \quad (5.15)$$

Where $\dot{\theta}$ is the time derivative of θ . A torque balance around the pendulum pictured in Figure 5-2 shows that:

$$\ddot{\theta}I = \tau_m + \tau_g \quad (5.16)$$

where I is the pendulum's inertia, equal to $m\ell^2$. Substituting using Equations (5.15) and (5.7) yields:

$$\ddot{\theta}I = \frac{K_t^2}{R}(\omega - \dot{\theta}) - \ell mg * \sin(\theta) \quad (5.17)$$

This equation is non-linear, and while this non-linearity maybe important during large changes of speed, if the shaft speed is relatively constant, Equation (5.17) can be linearized for small displacements ψ around $\theta = 90$ deg to:

$$\ddot{\psi}I = -\frac{K_t^2}{R}\dot{\psi} - \ell mg\psi \quad (5.18)$$

In order to control this system, some parameter needs to be varied by the controller. Varying m , adding mass to the pendulum while in motion, would be difficult. Altering g in most cases is impossible. Changing ℓ would require a mechanical system, which could hurt the reliability of the system over time. K_t is fixed based on the magnets and coils chosen. This leaves R , the resistance attached to the motor, as the parameter of choice.

If the motor is ideal (no inductance or internal resistance), then it yields interesting results for the limits of R . If R goes to zero (short circuit across motor leads), then $\omega = \dot{\theta}$ which indicates the motor is "locked" and the pendulum and shaft will rotate at the same speed. Alternatively, if R goes to infinity (open circuit), then the motor will exert no torque on the pendulum.

This model makes several assumptions. Namely, motor impedance and internal resistance are neglected. Furthermore, no consideration is given to air resistance on the pendulum or friction between the pendulum and motor.

5.3.4 Implementation

While the modeling required for an active system was developed, it was not implemented. Instead, a brushless motor was driven through a rectifier, shown in Figure 5-5, to produce DC power. The motor chosen has a through hole as shown in Figure 5-4. This allows the motor axis to be coincident with the shaft axis without the shaft being modified. The stator was then fixed in the stationary reference frame.

5.3.5 Results

The motor constant, K_t , was tested. In the experiment shown in Figure 5-6, a $1\ \Omega$ resistor was attached to the motor while the shaft was spun at a known speed and the voltage across the resistor, V , was measured. The slope of the best fit line is the approximate motor constant K_t . In this case, it was found to be $1.518 * 10^{-3}$ volts per RPM. Using Equation (5.4), the power dissipated through the resistor can be found. This is shown against speed in Figure 5-7. It is a parabola as expected:



Figure 5-4: The brushless motor used as a generator. This motor has a through-hole allowing it to be placed on axis with the shaft. Image from [5].

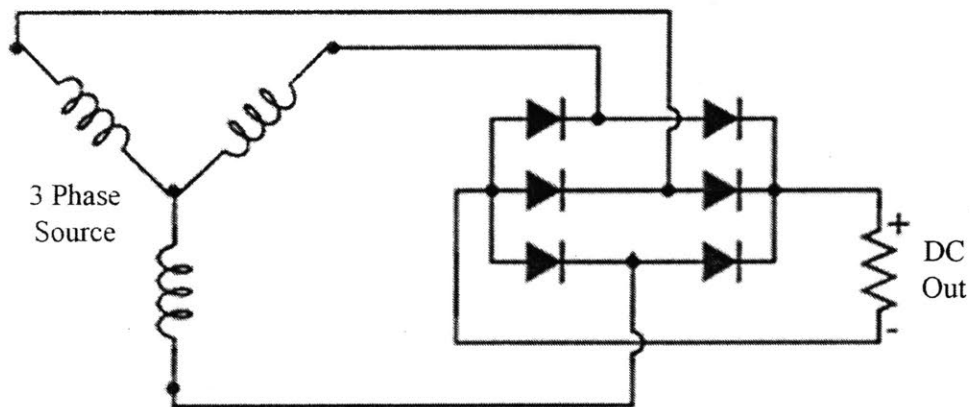


Figure 5-5: The schematic of a three-phase rectifier is shown. Image modified from [3]. This rectifier was built using six 1N5817 diodes.

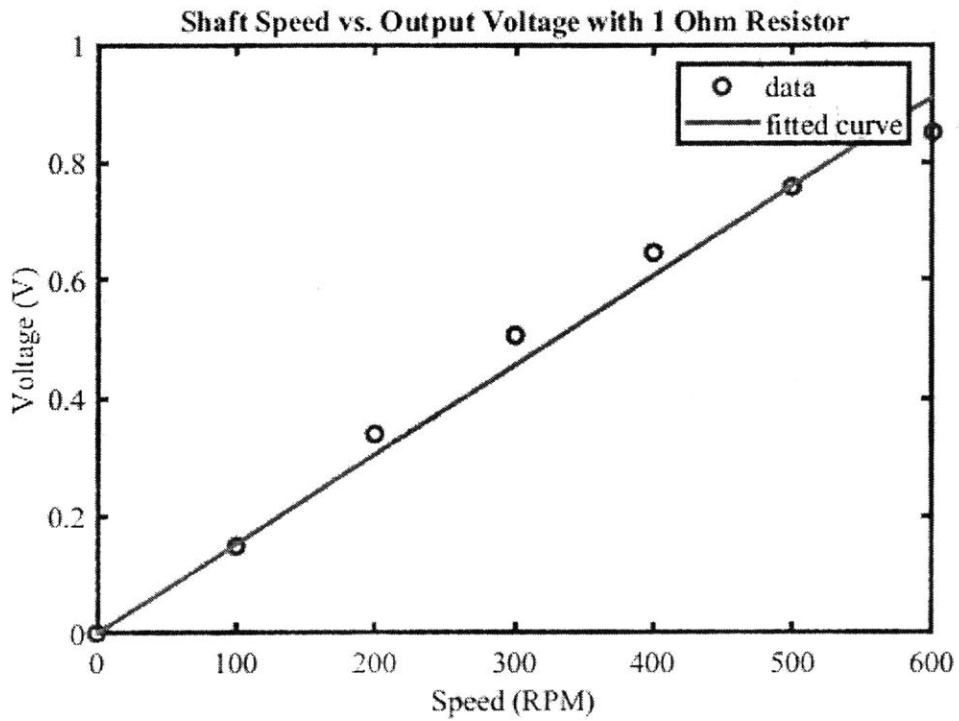


Figure 5-6: As expected, the voltage out of the motor is proportional to the speed. The slope of the fit is the motor constant, K_t .

voltage increases linearly with speed and power increases with voltage squared. It should be noted that for a resistance of $1\ \Omega$ the motors began to overheat at speeds above 600 RPM.

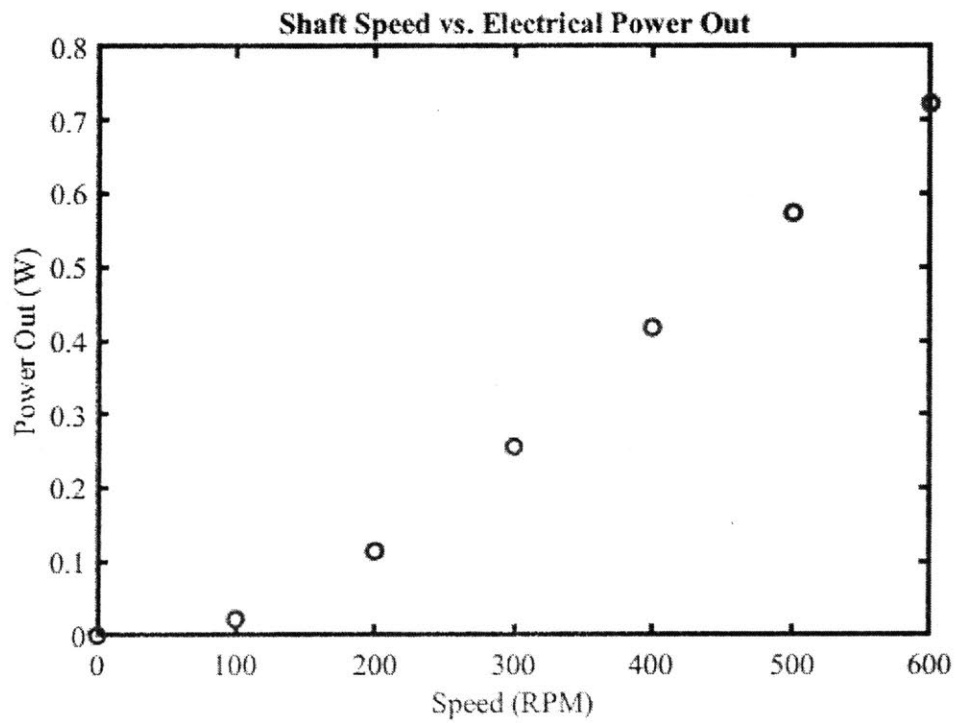


Figure 5-7: The electrical power output increases with the shaft speed.

Chapter 6

System Integration

In order for the sensor to function, all the data from the various sensors must be compiled and sent off the device. This section describes how a microcontroller was chosen to connect and read the sensors. It also explains the microphone selection and finally the wireless communications implementation.

A microcontroller is used to gather the data from the sensors and prepare it for transmission, as shown in Figure 6-1. Initially, a Cortex M0 microcontroller was used. However, later iterations used an ESP32 chip as it has Wi-Fi and Bluetooth already integrated. The code used for the microcontroller is included in Appendix C.

The prototype must hold several components to the shaft. While the torque sensor collars are attached directly, other components are not. Specifically, the microcontroller, load-cell amplifier, accelerometer, microphone, and a battery. A 3D printer was used to create the structure required to hold the components. A full bill of materials is provided in Appendix E.

In the early versions shown in Figure 6-2, the components were enclosed in a case. Slots in the case held the accelerometer in place and the other components fit into spaces in the case. However, this meant that all the components needed to be removed from the case to be modified, which could lead to wires breaking. Furthermore, the case was large. This consumed considerable 3D printer filament, took significant time to print, and added weight to the shaft. Furthermore, it required the shaft to be machined to have a "D" profile machined in it.

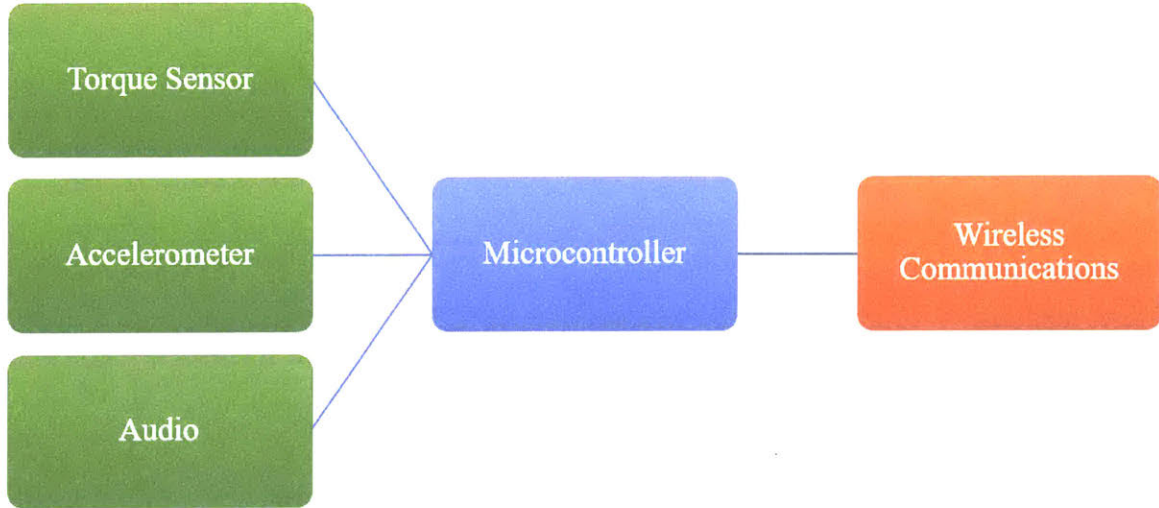


Figure 6-1: Data is collected from the accelerometer, load-cell amplifier, and microphone, processed by the microcontroller, and is then sent over Wi-Fi to a receiver.



Figure 6-2: The old prototype case design completely enclosed all components.

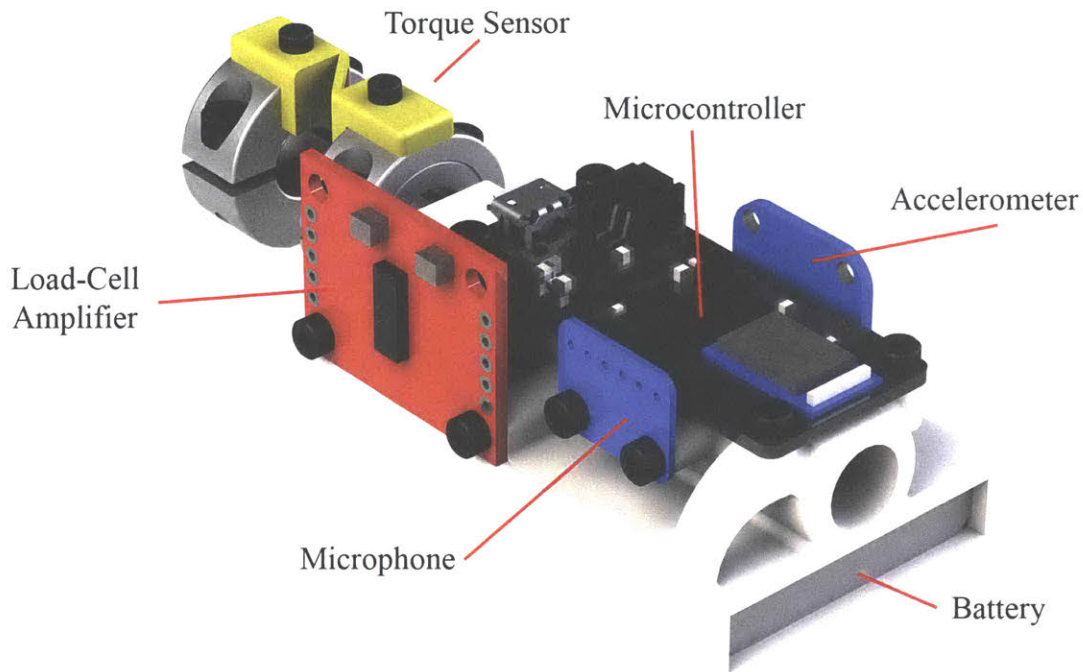


Figure 6-3: A rendering of the new case design.

Due to the problems with the old design, a new "open" design was used. In this version, as shown in Figures 6-3 and 6-4, the 3D printed structure is fixed to the shaft with a set screw. The holes are drilled and tapped so that the components can simply be bolted to the shaft with M2.5x6 bolts. The battery simply slides into the slot and is held in by friction. This allows the sensor to be easily modified and monitored. In some cases, a counterweight is used to keep it balanced about the shaft.

6.1 Microphone

In order to detect vibrations, a microphone was added to the sensor. Initially, a microphone was placed in the benchtop cage next to the shaft, but in later versions, a different microphone was integrated onto the sensor. The chosen microphone was an I2S MEMS Microphone due to its small form factor, simple interface, and availability of existing libraries. It can sample frequencies up to 20 kHz [1]. However, it was determined that most vibrations can be extracted from an audio signal sampled at

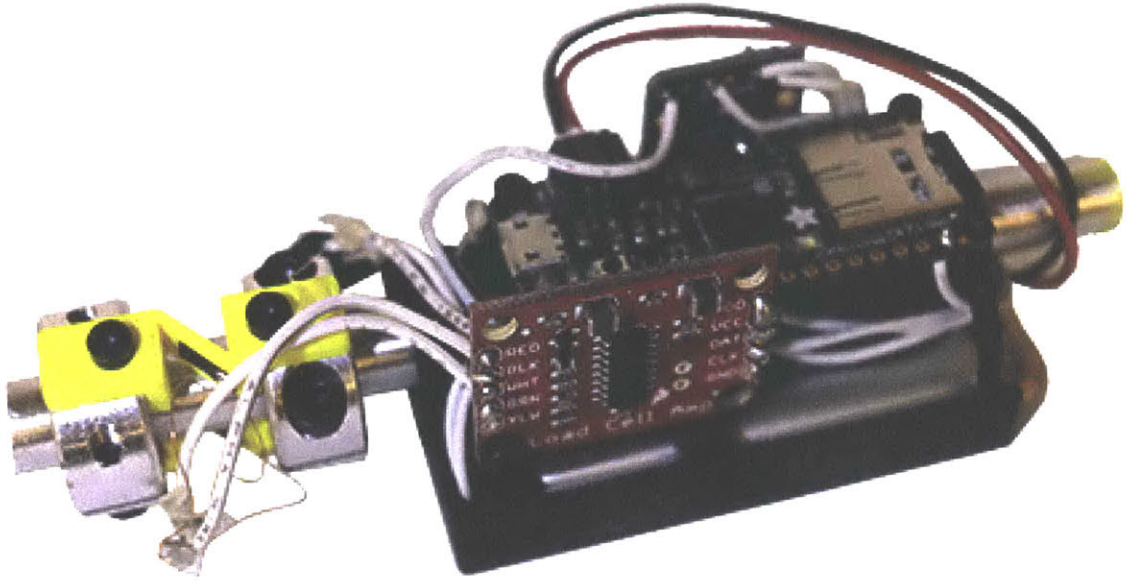


Figure 6-4: The new case design as made with components mounted.

3 kHz. Furthermore, the burst length was chosen to be 1.5 s so 3 full waves could be captured at 120 RPM.

6.2 Wireless Communications

Initially, the sensor was built using a Micro SD card to record data instead of wireless communications as it is simpler. Wireless communications are much more difficult to implement as both the transmitter and receiver have to function reliably. For an

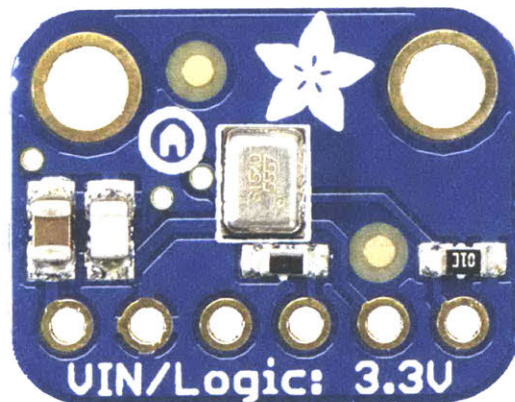


Figure 6-5: The I2S microphone used on the prototype. Image from [1].

initial prototype, using an SD card allowed for the torque sensor and acceleration methods to be tested much more quickly.

The sensor, using wireless or an SD card, utilizes "burst sensing". The SD card and wireless both consume power when on and take some time to initiate. By sampling data for a certain period of time, stopping sampling, and then transmitting data, the sensor can achieve a higher sample frequency within the burst than possible if the data written continuously. Furthermore, it is more power efficient in practice than live streaming.

Several parameters need to be defined for the burst sensing. First, the sample rate of each signal (acceleration, audio, and strain) needs to be determined. Secondly, the sample length needs to be determined. Clearly, the Nyquist criterion defines the minimum acceptable sample rate as twice the highest frequency of interest. In the system of interest, the maximum shaft speed is around 1000 RPM or 16.6 Hz. This means all signals should be sampled at least 33 Hz. Furthermore, the length of a signal is limited by the memory available on the microcontroller. The higher the sampling rate, the shorter the signal must be. In order to capture the slowest speeds of interest, the signal length must be equal to at least the period of the slowest frequency in the signal.

In addition, all sensor hardware limitations must also be met. For example, as explained in Section 3.2.3, the HX711 ADC can only sample at 80 Hz. If acceleration and then strain was cycled in alternating order, then the acceleration sample rate would be limited to 80 Hz. Therefore, it is advantageous to sample the acceleration multiple times and then sample strain once. The trade-off between the number of acceleration samples to each strain sample is shown in Figure 6-6. Clearly, the ideal ratio is 9 acceleration samples to 1 strain sample in the current hardware configuration.

The selected wireless method for the sensor is Wi-Fi. Wi-Fi has a higher bandwidth than Bluetooth which allows it to be used for more applications. Wi-Fi is still easy to implement and uses common, inexpensive hardware. Furthermore, it can communicate on existing network infrastructures.

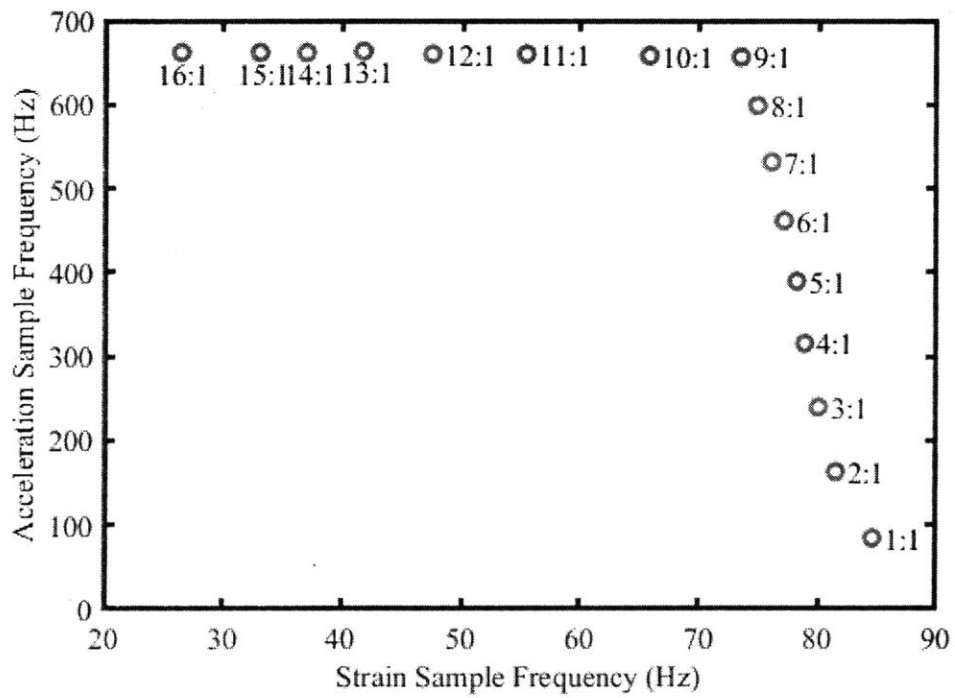


Figure 6-6: The trade-off between number of strain samples and sample frequency. The ratio of acceleration samples to strain samples is shown for each point.

Chapter 7

Conclusion

7.1 Summary

The sensor demonstrated in this paper measures torque, speed, vibration, and bending without any stationary components. Furthermore, it is very low cost. A possible method of power is presented that can be used without any stationary components. All data collected can be transmitted wirelessly.

This sensor is designed for use in health monitoring of industrial machinery. However, the components and system could have a wide range of applications. Nearly every mechanical system uses a spinning shaft, allowing versions of this sensor to be utilized in cars, planes, and robotics. Beyond health monitoring, it could be used for feedback control, system testing, and end-of-life analysis.

7.2 Future Work

Future work on this sensor would involve largely miniaturization. The torque sensor can be made smaller as the limiting factor is the height and length of the strain gauges. Furthermore, the accelerometer, microphone, analog to digital converter, and microcontroller should be integrated onto one printed circuit board to save space and remove unnecessary components. The tested motor for powering the system could also be custom made to fit the sensor needs and therefore much smaller.

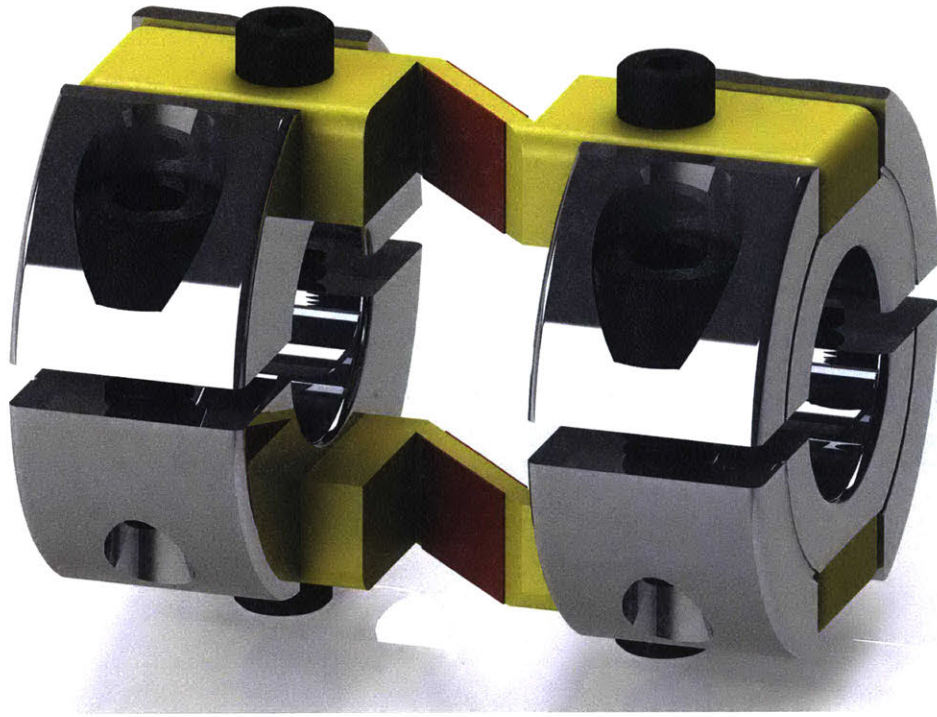


Figure 7-1: Insetting the bridge into the shaft collar can reduce the size of the sensor.

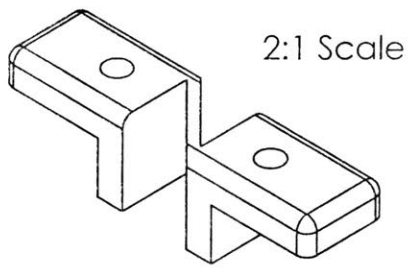
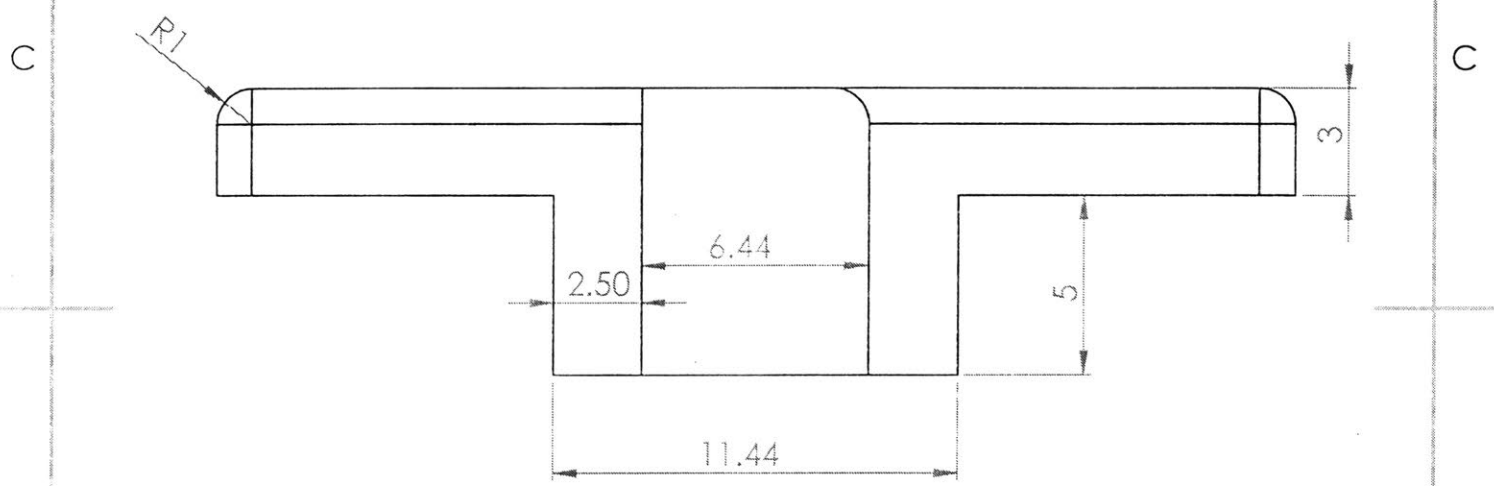
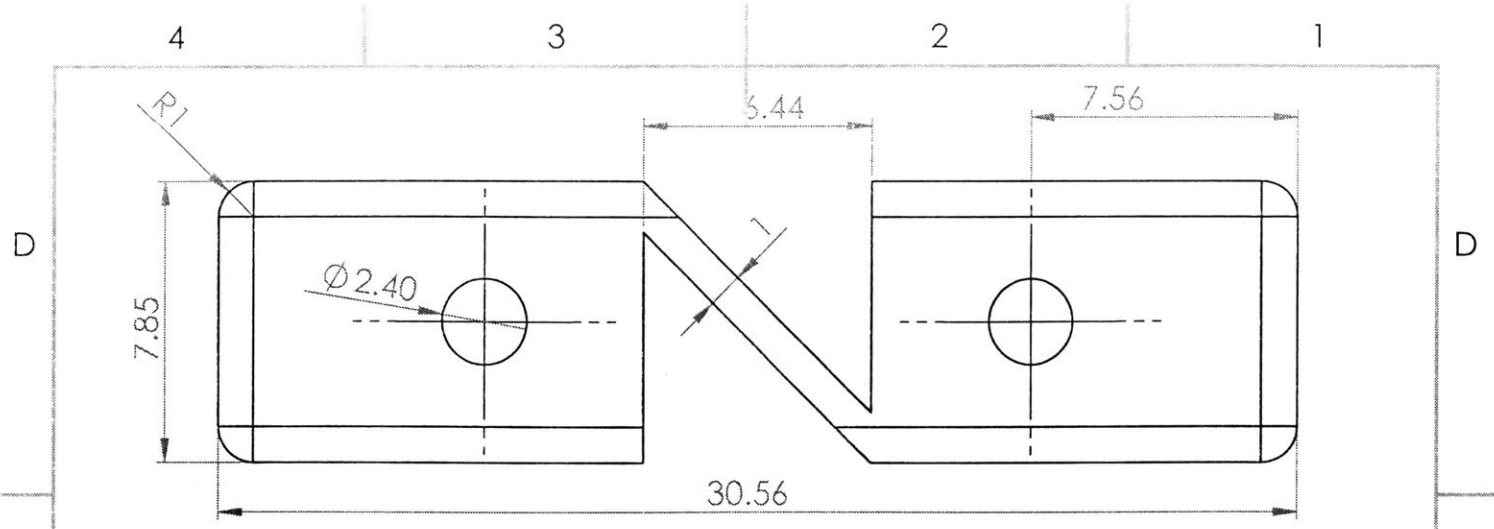
Finally, some more work needs to be done for commercialization. Machining pre-fabricated collars and using additive manufacturing for the bridge are likely not the most effective mass production methods. Furthermore, it may be possible to embed the strain gauges in the bridge flexure zone which may increase reliability. It may also be more efficient to use adhesives to attach the bridge to the collars than the current bolt system.

One way to miniaturize the torque sensor is to inset the bridge into the shaft collar as shown in Figure 7-1. As prototyped, the torque sensor requires 10.73 mm of clearance off the shaft. If inset, it only requires 7.96 mm of clearance. The limiting factor in this case is the bolts on the shaft collar. If these are ground, the limiting factor is the bolt on the bridge and the clearance is 7.12 mm. If this bolt is replaced with adhesives or ground down, then the clearance is limited by the shaft collars and is 6.35 mm.

Appendix A

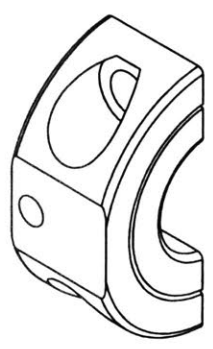
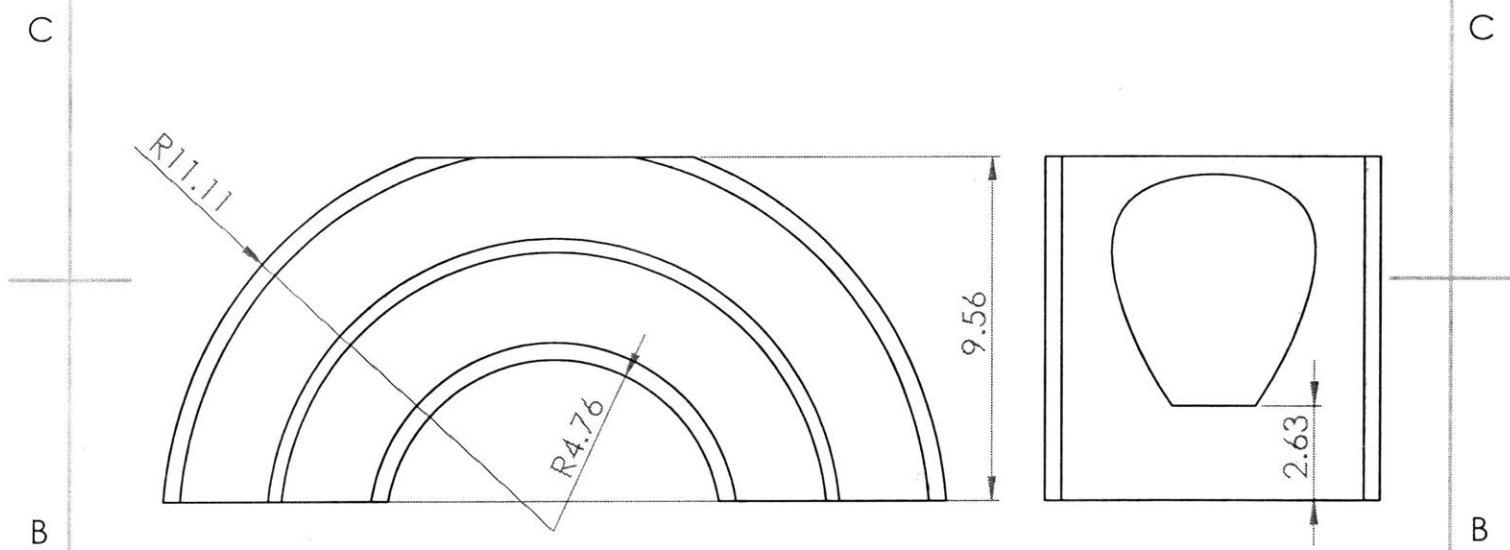
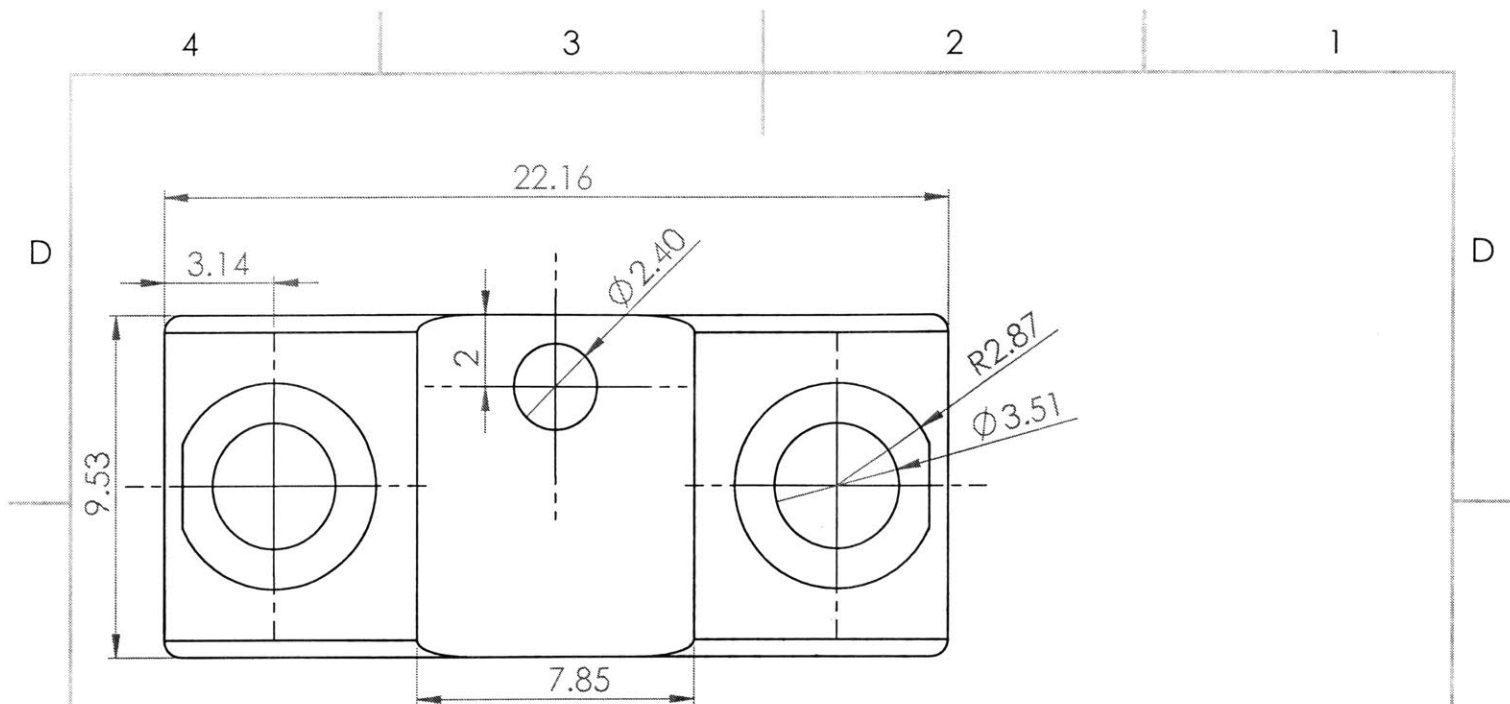
Engineering Drawings

Note: the collar CAD files are modified from [24] and the benchtop system CAD contains files copied and/or modified from [29, 26, 28, 25, 27, 23].



Part Name	Bridge
Created By	Erik Gest
Drawing Made	11/20/2018
Material	ABS Plastic
Manufacturing Method	3D Printed (FDM)
Weight	0.81 grams
Volume	794 mm ³

Units	millimeter
Scale (unless otherwise specified)	5:1

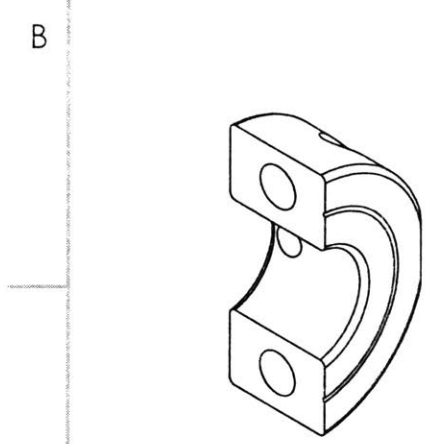
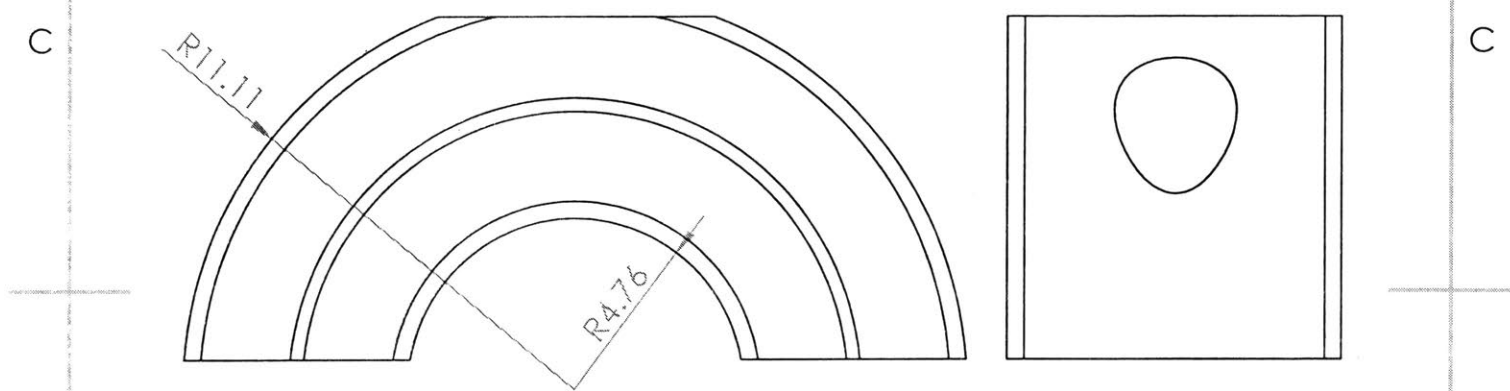
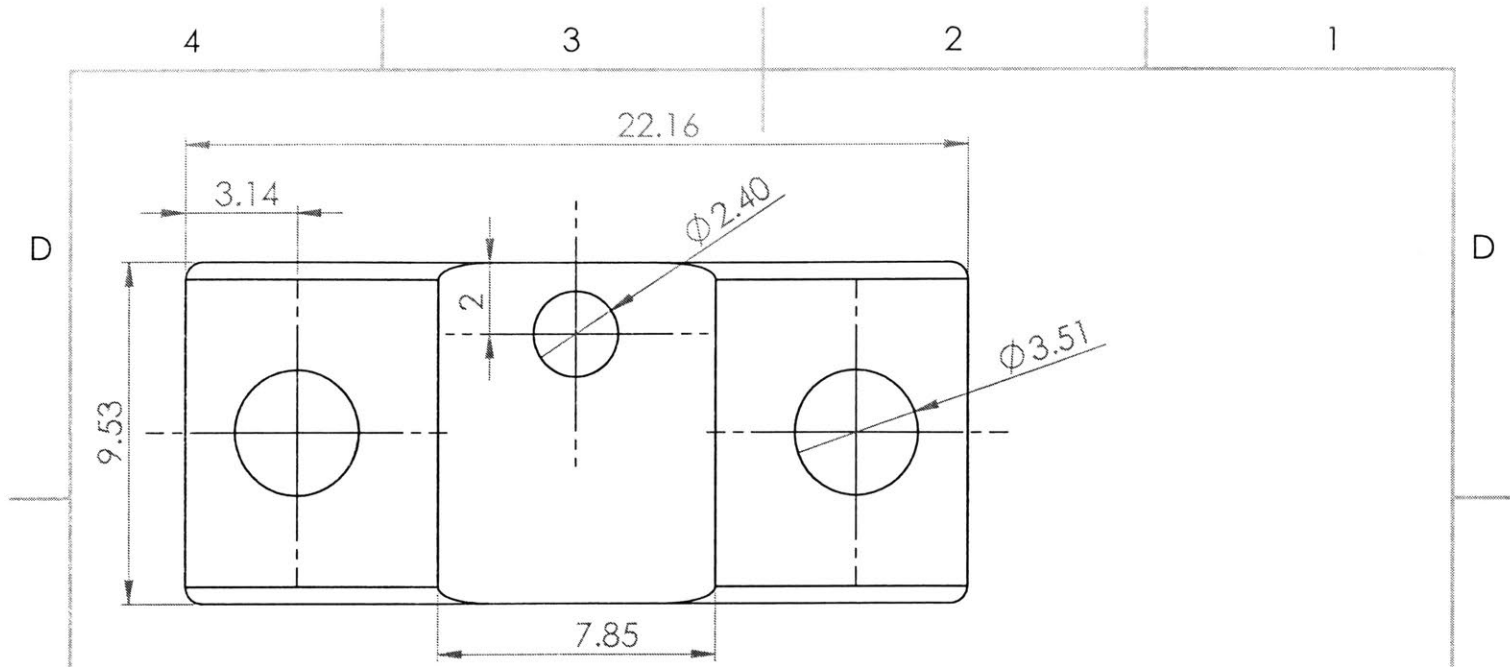


Scale 2:1

Part Name	Collar A
Created By	Erik Gest
Drawing Made	11/20/2018
Material	Aluminum
Manufacturing Method	Mill
Weight	2.92 grams
Volume	1082 mm ³

Units	millimeter
Scale (unless otherwise specified)	5:1

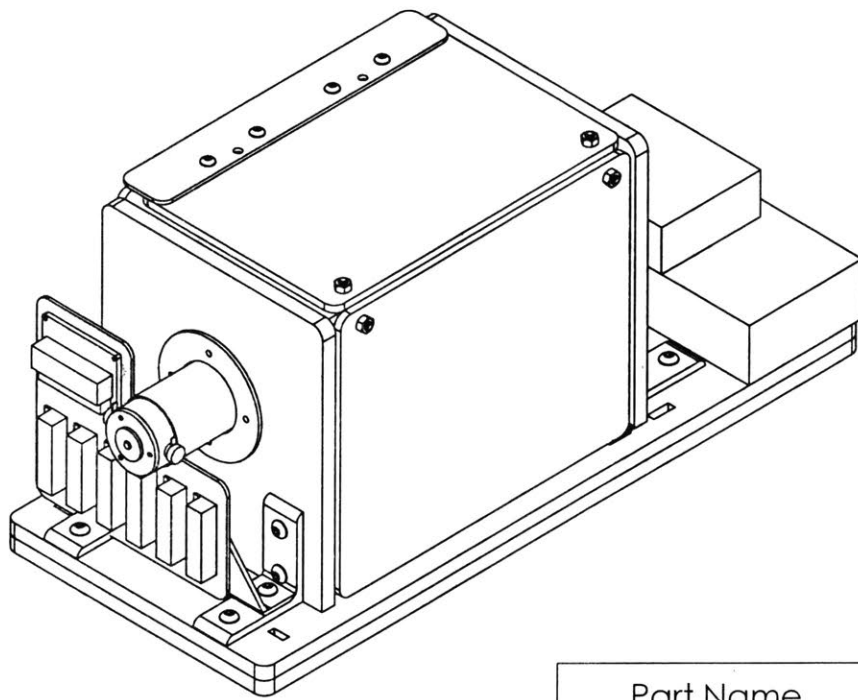
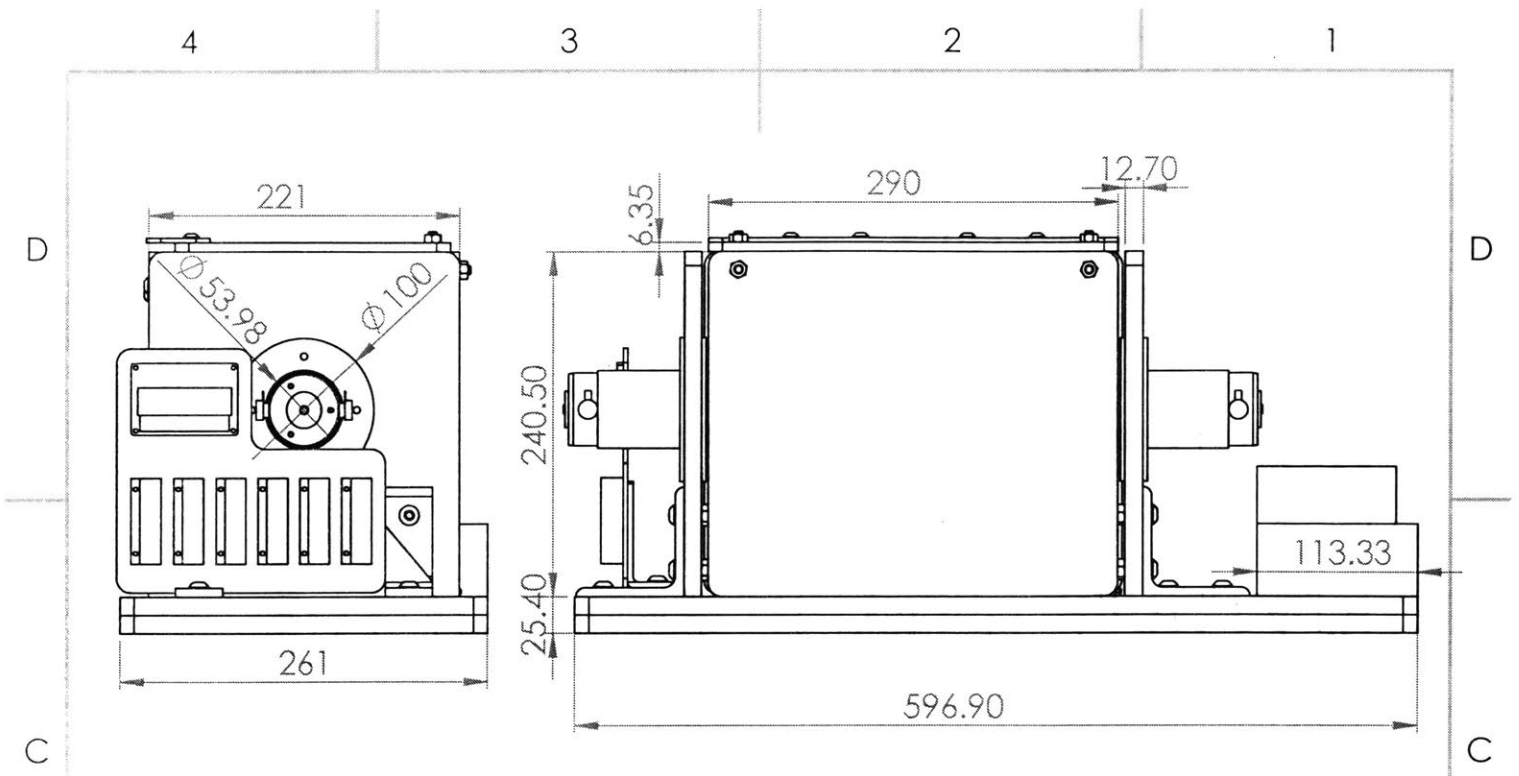
83



Scale 2:1

Part Name	Collar B
Created By	Erik Gest
Drawing Made	11/20/2018
Material	Aluminum
Manufacturing Method	Mill
Weight	3.26 grams
Volume	1208 mm ³

Units	millimeter
Scale (unless otherwise specified)	5:1



Part Name	Benchtop System
Created By	Erik Gest
Drawing Made	11/20/2018
Main Material	Acrylic
Manufacturing Method (Main)	Laser Cut
Weight	10.9 kilograms

Units	millimeter
Scale (unless otherwise specified)	1:5

85

Appendix B

Wiring Diagrams

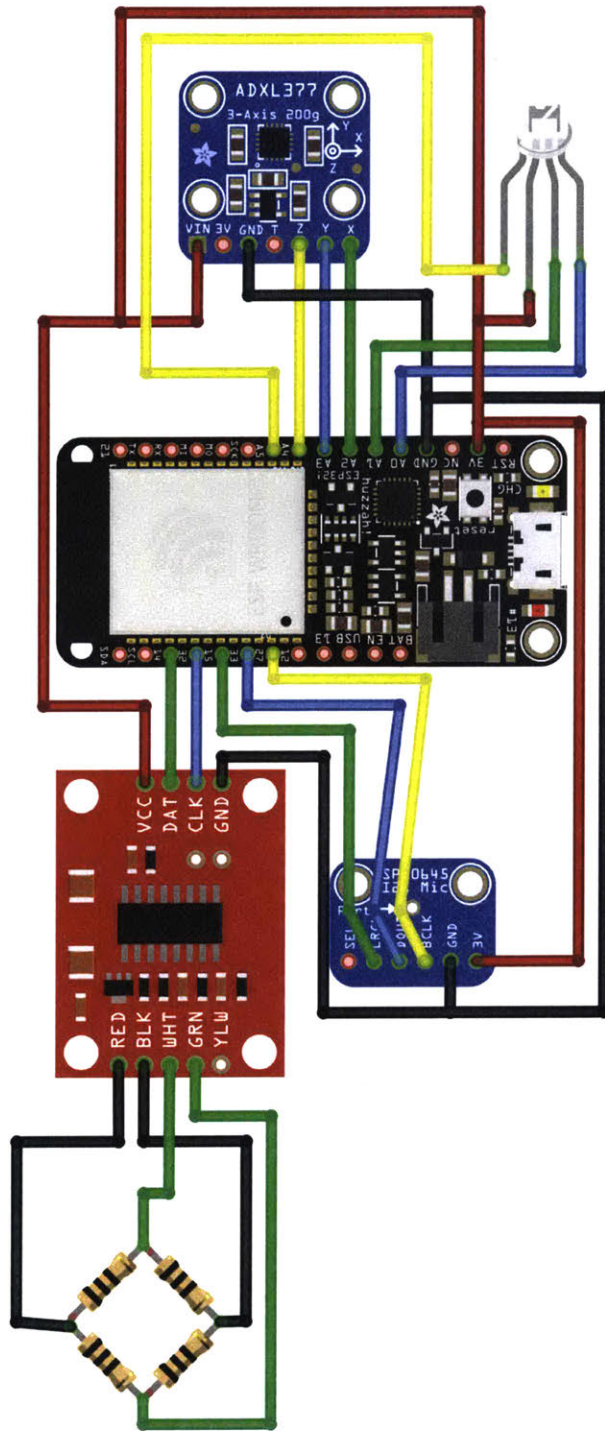


Figure B-1: This figure shows wiring of the sensor. The microcontroller (black, center) is wired to the load cell amplifier (red, left), the accelerometer (blue, top left), the microphone (blue, bottom right), and the LED (top right). The load cell amplifier is wired to the Wheatstone bridge (bottom left). This image was generated using Fritzing and using [43] and [2].

Appendix C

Operational Code

C.1 Sensor Code (C++)

The following code should be uploaded to the ESP32 microcontroller via the Arduino software. This code utilizes portions of code from the Adafruit and Sparkfun example libraries for the sensors and microcontroller as well as [8],

```
// WIFI Parameters
#include <WiFi.h>
const char* ssid = "MIT";
const char* password = "";
const uint16_t port = 8090;
const char * host = "";

// Acceleration Parameters
const int xPin=34;
const int yPin=39;
const int zPin=36;
const int accelBurstLength=1200;

// Strain Parameters
#include "HX711.h"
const byte hx711_data_pin = 14; //A5
const byte hx711_clock_pin = 32;
HX711 scale(hx711_data_pin, hx711_clock_pin);
```

```

const int strainInterval=9;

// Audio Parameters
#include <driver/I2S.h>
const int audioLength=4500;
const i2s_port_t I2S_PORT = I2S_NUM_0;
int32_t audioData[audioLength];

// Storage
int accelX[accelBurstLength];
int accelY[accelBurstLength];
int accelZ[accelBurstLength];
int times[accelBurstLength];
const int strainLength=(int)(accelBurstLength/strainInterval);
int strain[strainLength];
int packetCounter=1;

//COLOR!
const int red=4; //A5
const int green=25; //A1
const int blue=26; //A0

void setup() {
// WIFI Setup
pinMode(red,OUTPUT);
pinMode(green,OUTPUT);
pinMode(blue,OUTPUT);
digitalWrite(red,LOW);
digitalWrite(green,HIGH);
digitalWrite(blue,HIGH);
Serial.begin(115200);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.println("...");}
Serial.print("WiFi_connected_with_IP:_");

```

```

Serial.println(WiFi.localIP());
digitalWrite(green,LOW); //turn yellow

// Audio Setup
esp_err_t err;
const i2s_config_t i2s_config = {
    .mode = i2s_mode_t(I2S_MODE_MASTER | I2S_MODE_RX),
    .sample_rate = 3000,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_32BIT,
    .channel_format = I2S_CHANNEL_FMT_ONLY_RIGHT,
    .communication_format = i2s_comm_format_t(I2S_COMM_FORMAT_I2S |
        ↪ I2S_COMM_FORMAT_I2S_MSB),
    .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
    .dma_buf_count = 4,
    .dma_buf_len = 8};
const i2s_pin_config_t pin_config = {
    .bck_io_num = 27, // BCKL
    .ws_io_num = 15, // LRCL
    .data_out_num = -1, // not used (only for speakers)
    .data_in_num = 33}; // DOUT
err = i2s_driver_install(I2S_PORT, &i2s_config, 0, NULL);
if (err != ESP_OK) {
    Serial.printf("Failed_installing_driver:_%d\n", err);
    digitalWrite(green,HIGH);
    digitalWrite(blue,LOW); //turn purple
    while (true);}
err = i2s_set_pin(I2S_PORT, &pin_config);
if (err != ESP_OK) {
    Serial.printf("Failed_setting_pin:_%d\n", err);
    digitalWrite(green,HIGH);
    digitalWrite(blue,LOW); //turn purple
    while (true);}
Serial.println("I2S_driver_installed.");

// Other Setup
analogReadResolution(12); }

```

```

void loop() {
  // Begin WiFi
  WiFiClient client;
  if (!client.connect(host, port)) {
    Serial.println("Connection_to_host_failed");
    delay(1000);
    return;}
  else{
    digitalWrite(red,HIGH); //turn green from yellow
    digitalWrite(green,LOW);
    digitalWrite(blue,HIGH);
  }
  // Burst Sample Acceleration and Strain
  int i=0;
  while(i<accelBurstLength) {
    accelX[i]=analogRead(xPin);
    accelY[i]=analogRead(yPin);
    accelZ[i]=analogRead(zPin);
    times[i]=micros();
    if (i%strainInterval==0) {
      strain[i/strainInterval]=scale.read();}
    else { delayMicroseconds(1200);}
    i+=1;}
  // Burst Sample Audio
  i=0;
  int32_t sample = 0;
  while(i<audioLength){
    int bytes_read = i2s_pop_sample(I2S_PORT, (char *)&sample, portMAX_DELAY);
    audioData[i]=sample;
    i+=1;}
  digitalWrite(green,HIGH);
  digitalWrite(blue,LOW); //turn from green to blue
  // Transmit Data
  client.println("D");
  i=0;

```

```

while(i<accelBurstLength) {
  if (i%strainInterval==0) {
    client.println("@" + String(times[i]) + "," + String(accelX[i]) + "," + String(accelY[i]) + ","
      ↵ + String(accelZ[i]) + "," + strain[i/strainInterval] + "$"); }
  else {
    client.println("@" + String(times[i]) + "," + String(accelX[i]) + "," + String(accelY[i]) + ","
      ↵ + String(accelZ[i])+"$");}
  i+=1;}
i=0;
client.println("A");
while(i<audioLength){
  client.println(audioData[i]>>15);
  i+=1;}
// End WiFi
client.stop();
Serial.println("Data_sent_at_" + String(round(millis()/1000)) + "_Packet_" +String(
  ↵ packetCounter));
packetCounter++;}

```

C.2 Receiver Code (Python 3.6)

This code is run on a PC or Raspberry Pi to receive the data from the sensor. It stores the data in "data.csv". Uses code modified from [41].

```

import socket
import csv
import time
start_time = time.time()
s = socket.socket()
s.bind(('0.0.0.0', 8090 ))
s.listen(0)

run=True
myFile=open('data.csv', 'w')
myFile.writer=csv.writer(myFile,lineterminator='\n')
print('Started...')

```

while run:

```
print(time.time()-start_time)
myFile.writer.writerow([time.time()-start_time])
client, addr = s.accept()
myData=""
while True:
    content = client.recv(32)
    if len(content) ==0: break
    else:
        content=content.decode('ASCII') #convert from bytes to string
        content=content.replace('\n','')
        myData+=content
myFile.writer.writerow([myData])
print("Data_Recieved")
client.close()
```

C.3 Benchtop Arduino Code (C++)

This code should be uploaded to the Arduino Nano that runs the benchtop system using the Arduino IDE.

```
#include <Encoder.h>
Encoder myEnc(2,3);

//Parameters
float delayTime = 10;
bool stopped=true;
float KI=0.05;
float KP=0.05;
float maxI=100000;

//Other Pins
const int speedSetPin=11;
const int safetyPin=4;
```

```

const int S1=6;
const int S2=7;
const int S3=10;
const int S4=9;

float shaftSpeed=0;
float setValue=0;
float R=15;
float sumError=0;
float writeSpeedV=0;

bool out=false;
float output=0;
int k=0;
long oldPosition = -999;
unsigned long oldTime=0;

void setup() {
  Serial.begin(115200);
  pinMode(S1,OUTPUT); pinMode(S2,OUTPUT); pinMode(S3,OUTPUT); pinMode(S4,
    ↪ OUTPUT);
  updateR(R);}

void loop() {
  checkForInputs();
  shaftSpeed=readSpeed();
  if (k>=10) {
    sendData();
    k=0;}
  else{
    k++;}
  if (!stopped){writeSpeed();}
  delay(delayTime);}

float readSpeed(){
  long newPos=myEnc.read();

```

```

    long dx=newPos-oldPosition;
    long timeChange=micros()-oldTime;
    float tempSpeed=29644*(1.0*dx/timeChange);
    oldTime=micros();
    oldPosition=newPos;
    return tempSpeed;}

void writeSpeed(){
    if (digitalRead(safetyPin)==HIGH){
        float error=setValue-shaftSpeed;
        sumError+=error;
        if (sumError>maxI) {sumError=maxI;}
        if (sumError<-maxI) {sumError=-maxI;}
        writeSpeedV=(setValue+error*KP+sumError*KI)/15.4 ;
        if (writeSpeedV<0){writeSpeedV=0;}
        analogWrite(speedSetPin,writeSpeedV); }
    else {
        Serial.println("CAGEOPEN");
        digitalWrite(speedSetPin,LOW);}}

void sendData(){
    Serial.print(millis());
    Serial.print('@');
    Serial.print(shaftSpeed);
    Serial.print('@');
    if (stopped) { Serial.print(0); }
    else { Serial.print(setValue); }
    Serial.print('@');
    Serial.println(R);}

void checkForInputs(){
    String data;
    if (Serial.available() > 0) {
        byte input=Serial.read();
        if (input==48){ //update set speed 0
            data=getNumber();

```



```

    setValue=data.toFloat();}
if (input==49){ //update resistance
    data=getNumber();
    R=data.toInt();
    updateR(R);}
else if (input==53){ //stop 5
    stopped=true;
    digitalWrite(speedSetPin,LOW);}
else if (input==54){ //start 6
    stopped=false;}
sumError=0;}}

float getNumber(){
    String output="";
    while(Serial.available() > 0) {
        byte input=Serial.read();
        if (input==46){output=output+".";}
        else { output=output+String(input-48); }}
    return output.toFloat(); }

void updateR(int setting) {
    //setting can be 0-15, R=setting*0.25 ohm
    float tempR=0;
    if (setting % 2==0) {disable(S1); }
    else {enable(S1); tempR=tempR+0.25; }
    if (setting==2 or setting==3 or setting==6 or setting==7 or setting==10 or setting==11 or
        ↔ setting==14 or setting==15){enable(S2); tempR=tempR+0.5;}
    else { disable(S2); }
    if ((setting>3 and setting<8) or (setting>11)) {enable(S3); tempR=tempR+1; }
    else {disable(S3);}
    if (setting>7) { enable(S4); tempR=tempR+2;}
    else { disable(S4);}
    R=setting;//tempR;}

void enable(int pin){digitalWrite(pin,HIGH);}
void disable(int pin){digitalWrite(pin,LOW);}

```

C.4 PWM Shift Code (C++)

This is for the secondary Arduino Nano on the benchtop that shifts the PWM frequency for the speed controller.

```
const int inPin=2;
const int gndPin=5;
const int outPin=4;

int pwm_value;

void setup() {
pinMode(inPin,INPUT);
pinMode(outPin,OUTPUT);
pinMode(gndPin,OUTPUT);
digitalWrite(outPin,LOW);
digitalWrite(gndPin,LOW);}

void loop() {
pwm_value=pulseIn(inPin,HIGH);
digitalWrite(outPin,HIGH);
delayMicroseconds(1500+pwm_value/3);
digitalWrite(outPin,LOW);
delayMicroseconds(1500-pwm_value/3);}
```

C.5 Benchtop PC Code (Python 3.6)

This code displays and records data from the benchtop system on a PC or Raspberry Pi. It also allows the benchtop system to be controlled through a UI.

```
from tkinter import *
import random,time,datetime,csv,serial,os,math
from tkinter import messagebox

class Application(Frame):
    def __init__(self,master=None):
```

```

super().__init__(master)
self.pack()
self.create_widgets()
self.speed=-1
self.time=-1
self.data=[[[],[],[],[]]]
self.running=False
self.data_line=""
self.ser=self.attemptSerial()
self.lastTime=0
self.speedVals=[]
self.freqVals=[]
self.fileNum=1
self.RVals=[]
self.setValue=0
self.output=0
self.readFromCSV=False
self.startTime=0
self.Rset=15
self.correctCount=0;
self.initial=True
self.count=0

if os.path.isfile(os.getcwd() + '\inputs.csv'):
    print("File_Found")
    temp=messagebox.askokcancel("Inputs","A_data_sheet_for_input_speeds_has_been_found._
        ↪ Would_you_like_to_use_this?")
    self.readFromCSV=temp
    if temp: self.importCSVdata()
    else: print("No_File_Found")

def quit(self):
    self.ser.write(b'5')
    root.destroy()

def create_widgets(self):

```

```

self.fileName=StringVar(self)
tempString=str(datetime.datetime.now())
tempString=tempString.split(".")[0]
tempString=tempString.replace(":", "-")
self.fileName.set(tempString)
self.fileNameBox=Entry(self,textvariable=self.fileName).grid(row=0,column=0,columnspan=2)
self.start = Button(self, text="START", bg="green",command=self.toggle)
self.start.grid(row=1,column=0,sticky=W)
self.quit = Button(self, text="QUIT", fg="red",command=root.destroy).grid(row=1,column
    ↪ =1,sticky=W)
self.label1=Label(self, text="READINGS",font='Helvetica_12_bold').grid(row=2,column=0,
    ↪ colspan=2)
self.label2=Label(self, text="TIME_(s)").grid(row=3,column=0)
self.label3=Label(self, text="SPEED_(RPM)").grid(row=4,column=0)
self.label5=Label(self, text="SET_SPEED").grid(row=5,column=0)
self.label6=Label(self, text="RESISTANCE").grid(row=6,column=0)
self.label11=Label(self,text="Sample_Frequency_(Hz)").grid(row=7,column=0)
self.time_label=Label(self, text="-")
self.time_label.grid(row=3,column=1)
self.speed_label=Label(self, text="-")
self.speed_label.grid(row=4,column=1)
self.setLabel=Label(self, text="-")
self.setLabel.grid(row=5,column=1)
self.Rlabel=Label(self, text="-")
self.Rlabel.grid(row=6,column=1)
self.freq_label=Label(self,text="-")
self.freq_label.grid(row=7,column=1)
self.outputParam=StringVar(self)
self.outputParam.set("Set_Value")
self.outR=StringVar(self)
self.outR.set("15")
self.label11=Label(self, text="CONTROLS",font='Helvetica_12_bold').grid(row=8,column=0,
    ↪ colspan=2)
self.selector=Label(self, text="Change_Speed").grid(row=9,column=0)
self.inputValueButton=Button(self,text="UPDATE_SPEED",command=self.sendCommand).
    ↪ grid(row=10,column=0,columnspan=2)

```

```

self.userInput=StringVar(self)
self.userInput.set("0")
self.inputValueBox=Entry(self,textvariable=self.userInput).grid(row=9,column=1)
self.rsetlabel=Label(self, text="Change_R").grid(row=11,column=0)
self.inputRBox=Entry(self,textvariable=self.outR).grid(row=11,column=1)
self.inputRbutton=Button(self,text="UPDATE_R",command=self.sendR).grid(row=12,
    ↪ column=0,columnspan=2)
self.nextStep=Label(self)
self.nextStep.grid(row=13,column=0,columnspan=2)
self.messageBox=Label(self)
self.messageBox.grid(row=14,column=0,columnspan=2)

def toggle(self):
if self.running: #turn off
    self.ser.write(b'5')
    self.start.config(text="START",bg="green")
    self.running=False
    self.writeData()
else:
    self.ser.write(b'6')
    self.start.config(text="STOP",bg="red")
    self.running=True
    self.startTime=self.time
    if self.initial:
        self.initial=False
        maxTime=-1
        for i in self.autoTimes:
            if (maxTime==-1 or i[0]>maxTime): maxTime=i[0]

def sendCommand(self):
    self.ser.write(b'0')
    self.setValue=self.userInput.get()
    self.sendString(str(self.setValue))

def sendR(self):
    self.ser.write(b'1')

```

```

self.Rset=self.outR.get()
self.sendString(str(self.Rset))

def sendString(self,inString):
    for c in inString: self.ser.write(bytes(c, 'UTF-8'))

def readData(self):
    s=str(self.ser.readline())
    s=s[2:]
    s=s[:-5]
    self.data_line=s
    inputs=s.split('@')
    print(s)
    try:
        if len(inputs)==4:
            self.speed=float(inputs[1])
            self.speedVals.append(self.speed)
            self.time=float(inputs[0])
            self.output=float(inputs[2])
            self.R=(float(inputs[3]))
            if self.running:
                if self.count>=10:
                    self.data[0].append(self.time)
                    self.data[1].append(self.speed)
                    self.data[2].append(self.output)
                    self.data[3].append(self.R)
                    self.count=0
                else: self.count+=1
            return True
        else: return False
    except: return False

def writeData(self):
    myFile=open(self.fileName.get() + '_' +str(self.fileNum) + '.csv', 'w')
    self.fileNum+=1
    with myFile:

```

```

writer=csv.writer(myFile,lineterminator='\n')
writer.writerow(['Time','Speed','output'])
for i in range(0,len(self.data[0])): writer.writerow([self.data[0][i],self.data[1][i],self.data[2][i],
↔ self.data[3][i]])
self.data=[[],[],[],[]]

```

```

def attemptSerial(self):
    print("Finding_serial...")
    found=0
    for i in range(1,31):
        if found==0:
            try:
                ser=serial.Serial("COM" + str(i),115200)
                found=1
                print("Serial_Found_on_COM_" +str(i))
            except:
                try:
                    ser = serial.Serial('/dev/ttyACM1',115200)
                    found=1
                    print("Serial_Found_on_ACM1")
                except:
                    try:
                        ser = serial.Serial('/dev/ttyACM0',115200)
                        found=1
                        print("Serial_Found_on_ACM0")
                    except:
                        try:
                            ser = serial.Serial('/dev/ttyACM2',115200)
                            found=1
                            print("Serial_Found_on_ACM2")
                        except: pass
        if found==0:
            print('No_serial_found,_retrying_in_2s')
            time.sleep(2)
            ser=self.attemptSerial()
    return ser

```

```

def importCSVdata(self):
    self.autoTimes=[]
    with open('inputs.csv','rt') as csvfile:
        r=csv.reader(csvfile,delimiter=',')
        for row in r: self.autoTimes.append((row[0],row[1],row[2]))
    print(self.autoTimes)

def step(self):
    if self.running and self.readFromCSV:
        minTime=-1
        minTimeIndex=-1
        for i in self.autoTimes:
            if int(i[0])<minTime or minTime==--1:
                minTime=int(i[0])
                minTimeIndex=i
            if int(i[0])<=(self.time-self.startTime):
                self.correctCount=0
                self.setValue=str(i[1])
                self.Rset=str(i[2])
                self.ser.write(b'0')
                self.sendString(str(i[1]))
                time.sleep(0.2)
                self.ser.write(b'1')
                self.sendString(str(self.Rset))
                print("Speed_Set_to_" + i[1] + "_and_R_set_to_" + i[2])
                self.autoTimes.remove(i)
        try: self.nextStep.config(text="Next_Step_at_" + self.processTime(int(minTimeIndex[0])+
            ↳ self.startTime) + "_Speed:" + str(minTimeIndex[1]) + "_R:" + str(
            ↳ minTimeIndex[2]))
        except: pass
    if self.readData():
        self.time_label.config(text=self.processTime(self.time))
        if len(self.speedVals)>=1:
            self.speed_label.config(text=str(round(sum(self.speedVals),1)))
            self.speedVals=[]

```



```

self.setLabel.config(text=str(self.output))
self.Rlabel.config(text=str(self.R))
try: self.freqVals.append(1000.0/(self.time-self.lastTime))
except: pass
if len(self.freqVals)>=1:
    self.freq_label.config(text=str(round(sum(self.freqVals),1)))
    self.freqVals=[]
self.lastTime=self.time
else:
    if len(self.data_line)>0 and len(self.data_line)<20: self.messageBox.config(text=self.
        ↔ data_line)
if self.correctCount>15 and False:
    if (self.running) and not (float(self.output)==float(self.set Value)):
        self.ser.write(b'0')
        self.sendString(str(self.set Value))
        print("Speed_Corrected")
        time.sleep(0.2)
    if (self.running) and not (int(self.Rset)==int(self.R)):
        self.ser.write(b'1')
        self.sendString(str(self.Rset))
        print("R_Corrected_from_" + str(self.R) + "_to_" + str(self.Rset))
        self.correctCount=0
else: self.correctCount+=1
self.master.after(10,self.step)

def processTime(self,inTime):
    hours=math.floor((inTime)/3600000.0)
    remain=(inTime)-hours*3600000.0
    minutes=math.floor(remain/(60000.0))
    remain=remain-minutes*60000.0
    if minutes<10: minutes="0"+str(minutes)
    seconds=round(remain/1000.0,1)
    if seconds<10: seconds="0" + str(seconds)
    out= str(hours) + ":" + str(minutes) + ":" + str(seconds)
return out

```

```
root=Tk(className='Benchtop_Model_Control_Software')
app=Application(master=root)
root.after(10,app.step)
root.mainloop()
```

Appendix D

Analysis Code

D.1 Data Processing Code (Python 3.6)

The code described in C.2 outputs the file "data.csv" containing strain, audio, and acceleration data. This code breaks that file into two: "audio.txt" containing just the audio data and "other.txt" containing strain and acceleration data. It also removes some errors.

```
data_file = open("data.csv", "r")
audio_file = open("audio.txt", "w")
other_file = open("other.txt", "w")
print("Started compiling data...")
```

```
line: str
mode='n'
lastTime='nullTime'
first=True
for line in data_file:
    line = line.rstrip('\n')
    line=line.replace('@','')
    line=line.replace('$','')
    if 'A' in line: mode='a'
    elif 'D' in line:
        mode='d'
```

```

    first=True
elif "." in line: pass
else:
    if mode=='a':
        try: audio_file.write(lastTime + "," + str(int(line)) + "\n")
        except: print(line)
    elif mode=='d':
        if not first: other_file.write(line + "\n")
        else: first=False
        lastTime=line.split(',')[0]
    else: print(line)

```

D.2 View Results Code (MATLAB)

The following code will import the audio, acceleration, and strain data from the sensor into MATLAB. The files from the section before, "other.txt" and "audio.txt" should be in the MATLAB path.

```

clear all
close all

%% Audio Data
Fs=3000;
M=csvread('audio.txt');

% Linearize Time
while sum(diff(M(:,1))<0)>0
for i=2:length(M)
if M(i,1)<M(i-1,1)
M(i,1)=M(i,1)+2^32;
end
end
end

% Adjust for Fs

```

```

M=M(1:(end-1),:);
M(:,1)=M(:,1)/10^6;
tAudio=zeros(length(M),1);
yAudio=zeros(length(M),1);
lastTime=0;
j=0;
for i=1:length(M)
if lastTime==M(i,1)
j=j+1;
tAudio(i)=lastTime+j/Fs;

else
tAudio(i)=M(i,1);
j=0;
end
lastTime=M(i,1);
yAudio(i)=M(i,2);
end

figure
plot(tAudio,yAudio)
xlabel('Time_(s)')
ylabel('Audio')
title('Audio_Bursts')
clear i j lastTime M

%% Acceleration and Strain
M=csvread('other.txt');
times=M(:,1);
accels=M(:,2);
strains=M(:,3);

% Filter Time jumps
dTimes=diff(times);
remove=zeros(length(times),1);
for i=2:(length(times)-1)

```

```

if and(abs(dTimes(i-1))>15*10^6,abs(dTimes(i))>15*10^6)
remove(i)=1;
end
end
times=times(~remove);
accels=accels(~remove);
strains=strains(~remove);

% Linearize Time
while sum(diff(times)<0)>0
for i=2:length(times)
if times(i)<times(i-1)
times(i)=times(i)+2^32;
end
end
end
times=times/10^6;

% Filter Bad Accels
times=times(accels<4098);
strains=strains(accels<4098);
accels=accels(accels<4098);

figure
plot(times,accels/4098*3.3)
xlabel('Time_(s)')
ylabel('Acceleration_Voltage_(V)')
title('Accereleration')

% Filter Strain
strainTime=times(strains~=0);
strains=strains(strains~=0);
filter=and(strains>-3*10^6,strains~=-1);
strainTime=strainTime(filter);
strains=strains(filter);

```

```

figure
plot(strainTime,straains)
xlabel('Time_(s)')
ylabel('Strain_(arb)')
title('Strain')

```

D.3 FFT of Torque Signal Code (MATLAB)

The following code was used to generate Figure 3-19. This code uses excerpts from [22].

```

clear all
close all
load matlab.mat

useMiddle=0.5;
speeds=[0, 2.3,4.5,7.6,10.6,14.4]; %Hz
duration=60*5;
times=0:duration:(1800-duration);
figure(5)

segStraains=cell(length(speeds),2);
for i=1:length(speeds)
    segStraains{i,1}=strain(and(strainTime>times(i),strainTime<(times(i)+duration)));
    segStraains{i,2}=strainTime(and(strainTime>times(i),strainTime<(times(i)+duration)));
    low=round(useMiddle/2*length(segStraains{i,1}));
    high=round((1-useMiddle/2)*length(segStraains{i,1}));
    segStraains{i,1}=segStraains{i,1}(low:high);
    segStraains{i,2}=segStraains{i,2}(low:high);

    figure(5)
    subplot(3,2,i)
    Fs = 1/mean(abs(diff(segStraains{i,2}))); % Sampling frequency
    L = length(segStraains{i,2}); % Length of signal
    X=resample(segStraains{i,1},segStraains{i,2});
    Y = fft(X);

```

```

P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);

f = Fs*(0:(L/2))/L;
plot(f,P1)
axis([0 30 0 1400])
xlabel('F_(Hz)')
title(string(speeds(i)) + ' Hz')
end

```

D.4 Acceleration Simulation Code (MATLAB)

The following two pieces of code were used to generate the simulation results shown in Section 4.1, specifically Figures 4-1, 4-2, and 4-3. "AccelMethods.m" calls the function "plotSpeedsFromSignals.m" when it runs. This code uses modified excerpts from [22].

D.4.1 AccelMethods.m

```

clear all
close all

%% Inputs
shaft_diameter=25; % mm
shaft_speed=1000; % RPM
signal_to_noise=5;
sampleRate=80; % Hz
time_length=20; % seconds
rampTime=10; % seconds
gravity=9.8; % m/s^2
fft_evaluation_points=200;
offset_error=0.5; % g

```



```

%% Generate Accelerometer Signals
r=shaft_diamter/2/1000; % in meters
t=0:(1/sampleRate):time_length;
speed=shaft_speed/60*2*pi*ones(length(t),1); % in rad/s
speed(t<rampTime)=shaft_speed/60*2*pi.*(0:(length(t(t<rampTime))-1))/length(t(t<
    ↪ rampTime));
angle=tril(ones(length(t)),-1)*speed*1/sampleRate; %rads
gravityFactor=cos(angle)*gravity; % in m/s^2
az=speed.^2*r+gravityFactor;
ay=zeros(length(t),1);
ay(t<rampTime)=shaft_speed/60*2*pi/rampTime*r;
ay=ay+gravityFactor;
figure
hold on
plot(t,ay/9.81)
plot(t,az/9.81)
title('No_Noise_Signals')
xlabel('Time_(s)')
ylabel('Acceleration_(g)')
legend('Y','Z')

%% Gen Samples
plotSpeedsFromSignals(t, speed, ay, az, sampleRate, r, fft_evaluation_points,'No_Noise_Speeds')
plotSpeedsFromSignals(t, speed, awgn(ay,signal_to_noise), awgn(az,signal_to_noise),
    ↪ sampleRate, r, fft_evaluation_points,'White_Noise_Speeds')
plotSpeedsFromSignals(t, speed, ay+offset_error*9.81, az+offset_error*9.81, sampleRate, r,
    ↪ fft_evaluation_points,'Offset_Error_Speeds')

```

D.4.2 plotSpeedsFromSignals.m

```

function []= plotSpeedsFromSignals(t, speed, ay, az, sampleRate, r, fft_evaluation_points,intitle
    ↪ )
int_speed=tril(ones(length(t)),-1)*ay*1/sampleRate/r;
f=sampleRate*(0:(fft_evaluation_points/2))/fft_evaluation_points;
fftspeed=zeros(floor(length(t)/fft_evaluation_points),1);
fftTimes=fftspeed;
for i=1:floor(length(t)/fft_evaluation_points)

```

```

tempSignal=ay(((i-1)*fft_evaluation_points+1):(i*fft_evaluation_points));
Y=fft(tempSignal);
P2 = abs(Y/fft_evaluation_points);
P1 = P2(1:fft_evaluation_points/2+1);
P1(2:end-1) = 2*P1(2:end-1);
P1(1)=0;
[I,I]=max(P1);
fftspeed(i)=f(I);
fftTimes(i)=max(t(((i-1)*fft_evaluation_points+1):(i*fft_evaluation_points)));
end
figure
hold on
plot(t,speed/2/pi*60,'k--')
plot(t,sqrt(abs(az/r))/2/pi*60)
plot(t,int_speed/2/pi*60)
plot(fftTimes,fftspeed*60,'o')
title(intitle)
xlabel('Time_(s)')
ylabel('Speed_(RPM)')
legend('Speed','Centripetal','Integrated','FFT_(Y)','location','southeast')
end

```

Appendix E

Bills of Materials

Note: many of the items listed are sold in packs containing multiple items. The quantities are the number of parts needed, not the number of packs. The part numbers may refer to packs. Some of the part numbers may not correspond to the actual parts used, but should be valid substitutes.

Part Name	Quantity	Supplier	Part Number
M2.5x6 Socket Head Bolts	13	McMaster-Carr	91290A101
Two Piece 3/8" Shaft Collar	2	McMaster-Carr	6436K133
Strain Gauge	4	Amazon	059164
3D Printed Bridge Side 1	1		
3D Printed Bridge Side 2	1		
Load Cell Amplifier	1	Sparkfun	13879
High-G Accelerometer	1	Adafruit	1413
I2S MEMS Microphone	1	Adafruit	3421
HUZZAH32 ESP32 Feather	1	Adafruit	3405
Battery	1	Adafruit	2011
3D Printed Case	1		
Wire			

Table E.1: Bill of materials for the designed sensor.

Part Name	Quantity	Supplier	Part Number
1/2" Acrylic Sheet, 12"x24"	3	McMaster-Carr	8560K266
1/4" Acrylic Sheet, 12"x24"	2	McMaster-Carr	8560K355
1/8" Acrylic Sheet, 12"x12"	1	McMaster-Carr	8560K239
Power Supply	1	Amazon	S-360-12
Speed Control	1	Digilent	410-334-1-RET
Motor	2	McMaster-Carr	6331K13
Buna-N Rubber Spider	2	McMaster-Carr	6408K610
Coupler for 1/4" Shaft	2	McMaster-Carr	6408K913
Coupler for 3/8" Shaft	2	McMaster-Carr	6408K915
3/8" by 12" Aluminum Rod	1	McMaster-Carr	8974K24
Hinges	2	McMaster-Carr	1798A25
Large T-Slotted Bracket	4	McMaster-Carr	47065T241
Small T-Slotted Bracket	4	McMaster-Carr	47065T236
5/16"-18 Bolts, 1.25" Long	16	McMaster-Carr	91306A404
5/16"-18 Nuts	16	McMaster-Carr	94575A230
5/16"-18 Washers	32	McMaster-Carr	92141A030
1/4"-20 Bolts, 3/4" Long	16	McMaster-Carr	91255A540
1/4"-20 Nuts	16	McMaster-Carr	94575A110
1/4"-20 Washers	32	McMaster-Carr	92141A029
10-32 Bolts, 1.5" Long	10	McMaster-Carr	92949A237
10-32 Nuts	10	McMaster-Carr	91841A195
10-32 Washers	16	McMaster-Carr	92131A011
6-32 Bolts, 1/2" Long	10	McMaster-Carr	91251A148
M3 Bolts, 12mm Long	16	McMaster-Carr	92000A122
M3 Nuts	16	McMaster-Carr	90685A037
Arduino Nano	2	Amazon	EL-CB-004
Encoder	1	Digikey	102-1307-ND
Relay Module	1	Amazon	4450182
0.5 Ohm Power Resistor	3	Amazon	a12050500ux0003
1.0 Ohm Power Resistor	3	Amazon	a12042100ux0268
Emergency Stop Button	1		
Wire			

Table E.2: Bill of materials for the benchtop system.

Appendix F

Finite Element Analysis Details

This is the modified version of the report generated by SOLIDWORKS Simulation.

Assumptions



Original Model



Model Analyzed

Study Properties

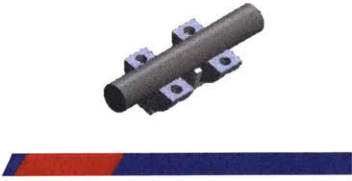


Analysis type	Static
Mesh type	Solid Mesh
Thermal Effect:	On
Thermal option	Include temperature loads
Zero strain temperature	298 Kelvin
Include fluid pressure effects from SOLIDWORKS Flow Simulation	Off
Solver type	FFEPlus
Inplane Effect:	Off
Soft Spring:	Off
Inertial Relief:	Off
Incompatible bonding options	Automatic
Large displacement	Off
Compute free body forces	On
Friction	Off
Use Adaptive Method:	Off

Units

Unit system:	SI (MKS)
Length/Displacement	mm
Temperature	Kelvin
Angular velocity	Rad/sec
Pressure/Stress	N/m ²

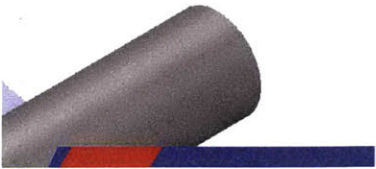


Material Properties

Model Reference	Properties	Components
	<p> Name: 1060 Alloy Model type: Linear Elastic Isotropic Default failure criterion: Unknown Yield strength: 2.75742e+007 N/m² Tensile strength: 6.89356e+007 N/m² Elastic modulus: 6.9e+010 N/m² Poisson's ratio: 0.33 Mass density: 2700 kg/m³ Shear modulus: 2.7e+010 N/m² Thermal expansion coefficient: 2.4e-005 /Kelvin </p>	<p> SolidBody 3(Cut-Extrude5[1])(6436K133_TWOPIECE CLAMP-ON SHAFT COLLAR-1), SolidBody 3(Cut-Extrude5[1])(6436K133_TWOPIECE CLAMP-ON SHAFT COLLAR-2) </p>
Curve Data:N/A		
	<p> Name: ABS Model type: Linear Elastic Isotropic Default failure criterion: Unknown Tensile strength: 3e+007 N/m² Elastic modulus: 2e+009 N/m² Poisson's ratio: 0.394 Mass density: 1020 kg/m³ Shear modulus: 3.189e+008 N/m² </p>	<p> SolidBody 1(Fillet1)(Bridge-1), SolidBody 2(Mirror3)(Bridge-1) </p>
Curve Data:N/A		
	<p> Name: Cast Alloy Steel Model type: Linear Elastic Isotropic Default failure criterion: Unknown Yield strength: 2.41275e+008 N/m² Tensile strength: 4.48082e+008 N/m² Elastic modulus: 1.9e+011 N/m² Poisson's ratio: 0.26 Mass density: 7300 kg/m³ Shear modulus: 7.8e+010 N/m² Thermal expansion coefficient: 1.5e-005 /Kelvin </p>	<p> SolidBody 1(Cut-Extrude1)(POC Shaft-1) </p>
Curve Data:N/A		

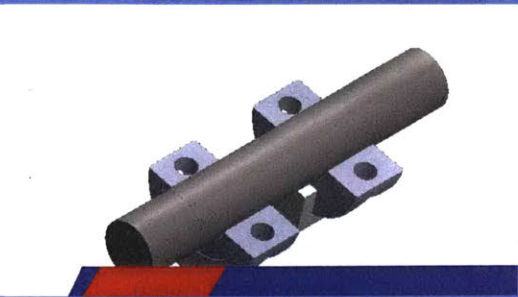
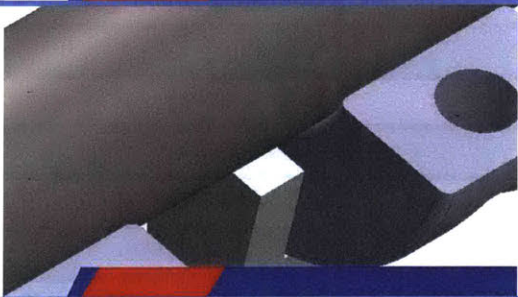

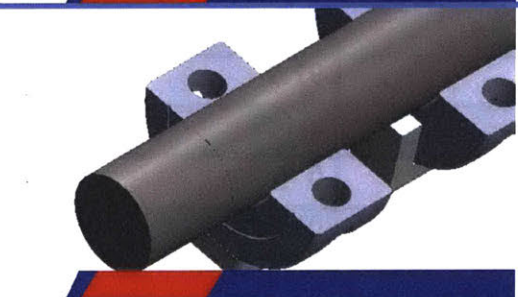



Loads and Fixtures

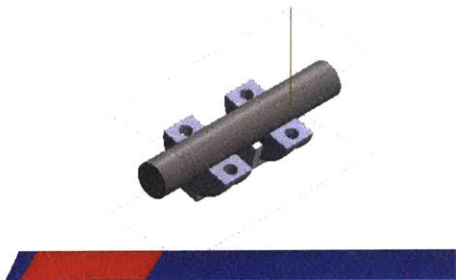
Fixture name	Fixture Image	Fixture Details		
Fixed-1		Entities: 1 face(s) Type: Fixed Geometry		
Resultant Forces				
Components	X	Y	Z	Resultant
Reaction force(N)	21.733	88.7678	-0.0201473	91.3896
Reaction Moment(N.m)	0	0	0	0

Load name	Load Image	Load Details
Torque-1		Entities: 1 face(s) Reference: Face< 1 > Type: Apply torque Value: 100 N.m

Contact Information

Contact	Contact Image	Contact Properties
Contact Set-5		Type: Bonded contact pair Entites: 3 face(s)
Contact Set-6		Type: Bonded contact pair Entites: 2 face(s)
Contact Set-7		Type: Bonded contact pair Entites: 2 face(s)
Contact Set-8		Type: Bonded contact pair Entites: 2 face(s)
Contact Set-9		Type: Bonded contact pair Entites: 2 face(s)



Global Contact		Type: Bonded Components: 1 component(s) Options: Compatible mesh
----------------	---	---

Mesh information

Mesh type	Solid Mesh
Mesher Used:	Standard mesh
Automatic Transition:	Off
Include Mesh Auto Loops:	Off
Jacobian points	4 Points
Element Size	1.16749 mm
Tolerance	0.0583743 mm
Mesh Quality Plot	High
Remesh failed parts with incompatible mesh	Off

Mesh information - Details

Total Nodes	61317
Total Elements	39685
Maximum Aspect Ratio	38.574
% of elements with Aspect Ratio < 3	97.9
% of elements with Aspect Ratio > 10	0.169
% of distorted elements(Jacobian)	0
Time to complete mesh(hh:mm:ss):	00:00:03



Model name: Acrem1
Study name: (Static 2)-Default-1
Mesh type: Solid Mesh



SOLIDWORKS Educational Product. For Instructional Use Only.

Resultant Forces

Reaction forces

Selection set	Units	Sum X	Sum Y	Sum Z	Resultant
Entire Model	N	21.733	88.7678	-0.0201473	91.3896

Reaction Moments

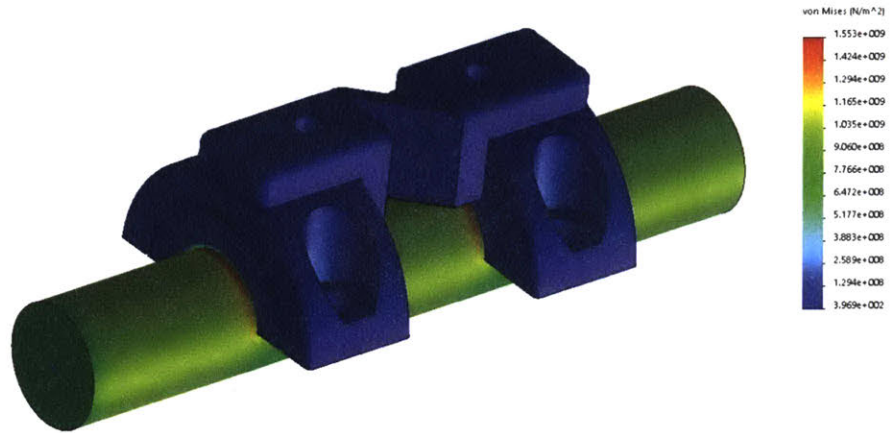
Selection set	Units	Sum X	Sum Y	Sum Z	Resultant
Entire Model	N.m	0	0	0	0



Study Results

Name	Type	Min	Max
Stress1	VON: von Mises Stress	3.969e+002N/m ² Node: 26997	1.553e+009N/m ² Node: 60985

Model name: Assem1
Study name: Static 2 (Default)
Plot type: Static nodal stress Stress1
Deformation scale: 1

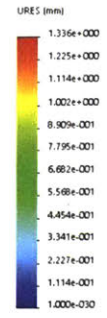
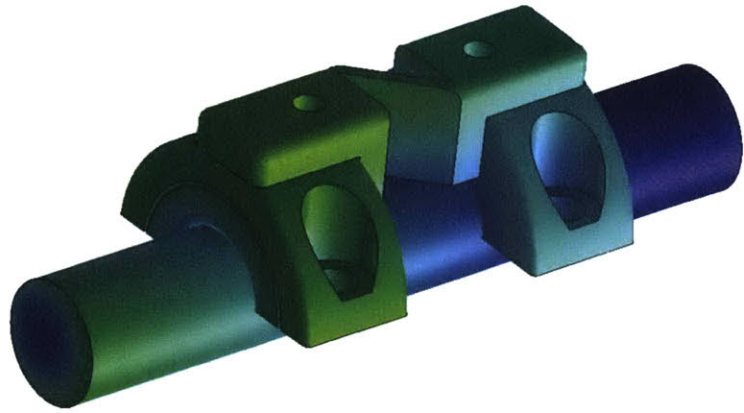


SOLIDWORKS Educational Product. For Instructional Use Only.

Name	Type	Min	Max
Displacement1	URES: Resultant Displacement	0.000e+000mm Node: 33715	1.336e+000mm Node: 29195



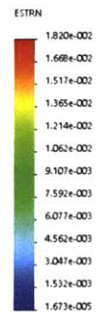
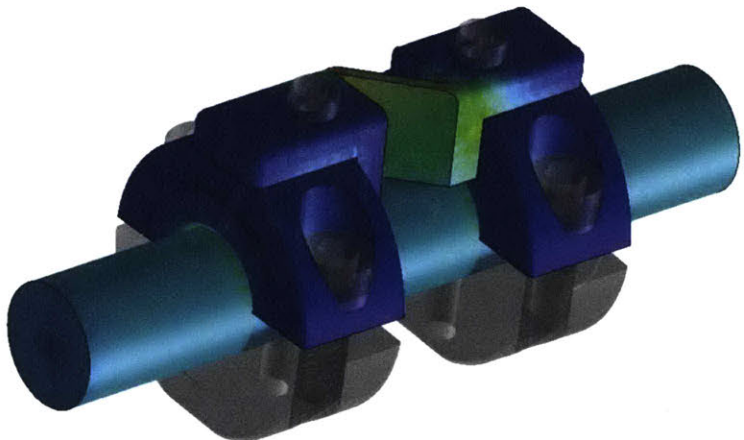
Model name: Assem1
 Study name: Static 2: Default...
 Plot type: Static displacement Displacement1
 Deformation scale: 1



SOLIDWORKS Educational Product. For Instructional Use Only.

Name	Type	Min	Max
Strain1	ESTRN: Equivalent Strain	4.139e-007 Element: 19345	1.820e-002 Element: 14747

Model name: Assem1
 Study name: Static 2: Default...
 Plot type: Static strain Strain1
 Deformation scale: 1



SOLIDWORKS Educational Product. For Instructional Use Only.



Appendix G

Instructions for Fabrication, Mounting, and Testing

G.1 Sensor Fabrication

The following steps should be taken to fabricate the sensor. Figures 3-7 and 6-3 may be useful in this process.

1. Use a mill to machine the collars as described in Appendix A
2. Tap the holes in the collars to add M2.5 threading
3. 3D print the bridge and a mirrored version as described in Appendix A
4. Use glue, such as cyanoacrylate, to attach gauges to each side of the bridges (4 total)
5. Use 4 M2.5x6 bolts to attach the bridges to the collars
6. 3D print the open case (pictured in Figure 6-3)
7. Drill and tap the mounting holes in the case to M2.5
8. Use M2.5x6 bolts to attach the accelerometer, microphone, load-cell amplifier, and microcontroller to the case

9. Wire and solder all connections as shown in Figure 3-6 and Appendix B
10. Insert the battery into the case

G.2 Sensor Mounting

1. Press fit the sensor case onto the shaft
2. Bolt one collar to the shaft
3. Monitor strain reading while tightening the other collar as evenly as possible to ensure it is not saturated. The strain reading should be approximately in the middle of the sensor range when the shaft is not experiencing torque. If a 24-bit sensor is used like HX711, then the digital reading should be about 8388608 on the scale from 0 to 16777215

G.3 Testing

The following steps should be taken to run a test:

1. Press the emergency stop (red button)
2. Open the cage
3. Make any modifications inside the cage that need to be made, such as changing the sensor or axle.
4. Plug sensor battery into sensor microcontroller.
5. Launch Python code to receive the sensor readings over Wi-Fi
6. Wait until a burst is received indicating successful connection
7. Close the cage, ensuring that the safety switch is now depressed.
8. Plug in one end of a mini-USB cable to the main benchtop Arduino and the other into a PC (if not already done)

9. Press the green button on the emergency stop to allow power to the system
10. Launch the Python code that runs the benchtop simulator
11. Use the GUI to run the experiment
12. Click "STOP" and then "QUIT" on the GUI
13. End receiving Python code
14. Press the emergency stop
15. Open cage and unplug microcontroller from battery

Appendix H

Version History

Various versions of the sensor have been built, most using all new components. The differences between these are described in Figure H-1. This thesis uses version 5 for most of the content. The code in Appendix C.1 is for version 6 but is backward compatible with version 5 (the only difference is the LED support). Likewise, the diagram in Appendix B is also for version 6 but is backward compatible with version 5 if the LED is simply removed. The version 6 torque sensor is pictured in Figure 7-1, but all other images of the torque sensor are version 4 or 5 (the torque sensor remained unchanged between versions).

Version	1	2	3	4	5	6
Color	Yellow	Red	Gray	Black	White	Yellow
Shaft Material	Plastic	Aluminum				
Strain Method	On Shaft		Novel Sensor			
Enclosure Type	Closed			Open		
Communication Method	Micro SD Card				Wi-Fi	
Microphone	No				Yes	
Torque Sensor Diameter (mm)	N/A		10.7			8.0
Wheatstone Bridge Configuration	Full		Quarter	Full		
Battery Type	Alkaline	Lithium Ion				
LED Indicator	No					Yes
Microcontroller	Arduino Nano		Adafruit Feather Adalogger		Addafruit Feather Huzzah ESP32	

Figure H-1: Comparison of the various sensor versions made over the project.

Bibliography

- [1] Adafruit Industries. Adafruit i2s mems microphone breakout. URL: <https://learn.adafruit.com/adafruit-i2s-mems-microphone-breakout/>.
- [2] Adafruit Industries. Fritzing library, 2018. URL: <https://github.com/adafruit/Fritzing-Library>.
- [3] All About Circuits. Rectifier circuits. URL: <https://www.allaboutcircuits.com/textbook/semiconductors/chpt-3/rectifier-circuits/>.
- [4] All About Circuits. Strain gauges. URL: <https://www.allaboutcircuits.com/textbook/direct-current/chpt-9/strain-gauges/>.
- [5] Amazon.com. Tiger gb54-1 brushless gimbal motor. URL: https://www.amazon.com/gp/product/B07BFJ4RQ1/ref=oh_aui_detailpage_o03_s00?ie=UTF8&psc=1.
- [6] Analog Devices, Inc. Small, low power, 3-axis 200 g accelerometer, 2012. URL: <http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL377.pdf>.
- [7] Arduino. Compare board specs. URL: <https://www.arduino.cc/en/Products/Compare>.
- [8] Michael Aspetsberger. Esp32 i2s mems microphone arduino ide example, 2018. URL: <https://github.com/maspetsberger/esp32-i2s-mems>.
- [9] Atmel Corporation. Smart arm-based microcontroller, 2015. URL: <https://cdn-shop.adafruit.com/product-files/2772/atmel-42181-sam-d21-datasheet.pdf>.
- [10] AVIA Semiconductor Co. LTD. 24-bit analog-to-digital converter (adc) for weigh scales. URL: https://www.mouser.com/ds/2/813/hx711_english-1022875.pdf.
- [11] Alok Barua. Torque measurement, December 2009. URL: <https://nptel.ac.in/courses/108105064/6>.

- [12] Peter Bowen, Asit Goel, Michael Schallehn, and Michael Schertler. Choosing the right platform for the industrial iot. *Bain and Company*, September 2017. URL: <https://www.bain.com/insights/choosing-the-right-platform-for-the-industrial-iot>.
- [13] Joseph Carlstein. Tachometer generator, July 1960. US Patent 3,018,395.
- [14] Alex Colon and Eric Griffith. The best smart home devices for the holidays. *PCMag*, November 2018. URL: <https://www.pcmag.com/article2/0,2817,2410889,00.asp>.
- [15] Exttech Instruments. Exttech 461893: Photo tachometer. URL: <http://www.exttech.com/display/?id=14283>.
- [16] Joseph Gierut and Dr. Ray Lohr. Automotive powertrain and chassis torque sensor technology. Technical report, Transense Technologies and Honeywell, 2005. URL: <http://www.transense.co.uk/downloads/articles/Poster%20Paper%20Nuremberg%202005.pdf>.
- [17] John Harbour. Torque sensing apparatus, March 1987. US Patent 4,649,758.
- [18] HBM Inc. Temperature compensation of strain gauges. Technical report. URL: <https://www.hbm.com/en/6725/article-temperature-compensation-of-strain-gauges/>.
- [19] International Journal of Sensor Networks and Data Communications. Distributed sensor networks. URL: <https://www.omicsonline.org/scholarly/distributed-sensor-networks-journals-articles-ppts-list.php>.
- [20] Ulrich Kindler. Deformation measuring device for measuring the torque of a cylindrical shaft, December 1996. US Patent 5,585,572.
- [21] Linear Technology Corporation. 24-bit high speed differential adc with selectable speed/resolution. URL: <https://www.mouser.com/datasheet/2/609/2440fe-1270455.pdf>.
- [22] Mathworks, Inc. Fast fourier transform. URL: <https://www.mathworks.com/help/matlab/ref/fft.html>.
- [23] McMaster-Carr Supply Company. 31000 rpm buna-n rubber spider for 5/8 od flexible shaft coupling iron hub. 3D CAD File. URL: <https://www.mcmaster.com/6408k61>.
- [24] McMaster-Carr Supply Company. Clamping two-piece shaft collar. 3D CAD File. URL: <https://www.mcmaster.com/6436k133>.
- [25] McMaster-Carr Supply Company. Compact round-face dc motor. 3D CAD File. URL: <https://www.mcmaster.com/6331K13>.

- [26] McMaster-Carr Supply Company. Flexible shaft coupling iron hub. 3D CAD File. URL: <https://www.mcmaster.com/6408k9>.
- [27] McMaster-Carr Supply Company. Silver corner bracket, 1 long for 1 high rail t-slotted framing. 3D CAD File. URL: <https://www.mcmaster.com/47065T236>.
- [28] McMaster-Carr Supply Company. Silver corner bracket, 3 long for 1-1/2 high rail t-slotted framing. 3D CAD File. URL: <https://www.mcmaster.com/47065T241>.
- [29] McMaster-Carr Supply Company. Surface-mount hinge with holes. 3D CAD File. URL: <https://www.mcmaster.com/1798A25>.
- [30] Mersen. How to maintain carbon brushes, brush-holders, commutators and slip rings. URL: <https://www.mersen.com/sites/default/files/publications-media/30-ptt-motor-maintenance-mersen.pdf>.
- [31] Midtronics. Midtronics magnetic pick-up sensors. URL: <http://www.midtronics.com/shop/products-1/integrated-solutions/midtronics-magnetic-pick-up-sensors>.
- [32] D. R. Myers and A. P. Pisano. Torque measurements of an automotive halfshaft utilizing a mems resonant strain gauge. In *TRANSDUCERS 2009 - 2009 International Solid-State Sensors, Actuators and Microsystems Conference*, pages 1726–1729, June 2009.
- [33] National Instruments Corporation. Measuring strain with strain gages, May 2016. URL: <http://www.ni.com/white-paper/3642/en/>.
- [34] National Instruments Corporation. Measuring vibration with accelerometers, December 2017. URL: <http://www.ni.com/white-paper/3807/en/>.
- [35] Omega Engineering. Tq501 rugged rotary torque sensor. URL: <https://www.omega.com/pptst/TQ501.html>.
- [36] Particle Sciences Inc. Hot melt extrusion. Technical report, 2011. URL: <https://www.particlesciences.com/news/technical-briefs/2011/hot-melt-extrusion.html>.
- [37] U. Perez. Low power wifi: A study on power consumption for internet of things. MS Thesis, Facultat d'Informàtica de Barcelona, February 2015. URL: <https://upcommons.upc.edu/bitstream/handle/2099.1/25583/104901.pdf?sequence=1&isAllowed=y>.
- [38] G. D. Putra, A. R. Pratama, A. Lazovik, and M. Aiello. Comparison of energy consumption in wi-fi and bluetooth communication in a smart building. In *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 1–6, January 2017.

- [39] Falvio Renga and Mario Rossi. System for harvesting energy including a counterweight and a system for controlling the angular position of the counterweight, November 2013. European Patent EP2738934B1.
- [40] David Roylance. Shear and torsion, June 2000. URL: <http://web.mit.edu/course/3/3.11/www/modules/torsion.pdf>.
- [41] Nuno Santos. Python: socket server, January 2018. URL: <https://techtutorialsx.com/2018/01/19/python-socket-server/>.
- [42] Sarmad Shams, Ji Yeong Lee, and Changsoo Han. Compact and lightweight optical torque sensor for robots with increased range. *Sensors and Actuators A: Physical*, 173(1):81–89, January 2012. URL: <https://doi.org/10.1016/j.sna.2011.10.019>.
- [43] SparkFun Electronics. Fritzing parts, 2018. URL: https://github.com/sparkfun/Fritzing_Parts.
- [44] The Japan Steel Works. Compounding extruders. URL: <https://www.jsw.co.jp/en/products/compounding/index.html>.
- [45] The Japan Steel Works. Specification. URL: <https://www.jsw.co.jp/en/products/compounding/specification.html>.
- [46] United Equipment Accessories, Inc. Slip rings. URL: <https://www.uea-inc.com/products/slip-rings/>.
- [47] WiTricity Corporation. Why magnetic resonance. URL: <http://witricity.com/products/why-magnetic-resonance/>.