

6-7-2017

Ambient Life

Ryan Scott

Santa Clara University, rwscott@scu.edu

Shaun Suezaki

Santa Clara University, ssuezaki@scu.edu

Follow this and additional works at: https://scholarcommons.scu.edu/elec_senior



Part of the [Electrical and Computer Engineering Commons](#)

Recommended Citation

Scott, Ryan and Suezaki, Shaun, "Ambient Life" (2017). *Electrical Engineering Senior Theses*. 31.

https://scholarcommons.scu.edu/elec_senior/31

This Thesis is brought to you for free and open access by the Engineering Senior Theses at Scholar Commons. It has been accepted for inclusion in Electrical Engineering Senior Theses by an authorized administrator of Scholar Commons. For more information, please contact rscroggin@scu.edu.

SANTA CLARA UNIVERSITY

Department of Electrical Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

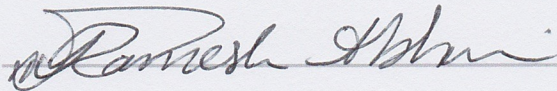
Ryan Scott, Shaun Suezaki

ENTITLED

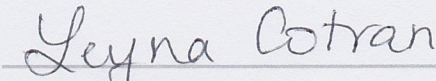
Ambient Life

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF SCIENCE
IN
ELECTRICAL ENGINEERING



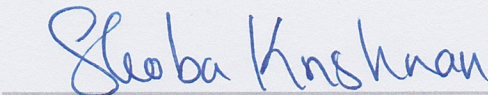
6/7/2017



6/8/2017

Thesis Advisor(s)

date



6/7/2017

Department Chair

date

Ambient Life

By

Ryan Scott, Shaun Suezaki

SENIOR DESIGN PROJECT REPORT

Submitted to

the Department of Electrical Engineering

of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements

for the degree of

Bachelor of Science in Electrical Engineering

Santa Clara, California

2017

ABSTRACT

Health Monitoring applications and devices are becoming very useful tools for people that want to take a more active role in their personal health. However, no current health-monitoring device is capable of harvesting its own power for operation. In our Senior Design project, we use an energy harvesting wireless sensor system that is designed to be easily worn on the wrist, or on the foot of an infant, in order to help people monitor their temperature and pulse rate. Our personal health-monitoring device operates using an Arduino pro mini microcontroller that reads in data from our sensors and passes them onto a Bluetooth module. The data that is recorded by the system is then sent over via Bluetooth to a laptop that will present the patient's results. The device is powered by a battery that can be charged by USB power, or through our RF energy harvesting system.

Acknowledgements

We would like to thank Dr. Abhari for working with us all year and for making sure that we never fell behind. We also want to thank Nick Mikstas for spending so much time with us in the lab helping us make all of the hard work. We would not have been successful without their help. Furthermore, we want thank Dr. Wood for leading our senior design class. Finally, we want to thank Santa Clara University for funding our project and allowing us to use their facilities.

Table of Contents

Chapter 1 Introduction.....	1
1.1 Project Background.....	1
1.2 Problem Statement	1
1.3 Project Objectives.....	2
Chapter 2 Project Requirements.....	3
2.1 Usability	3
2.2 Accuracy	3
2.3 Reliability	3
2.4 Safety.....	4
2.5 Art	4
2.6 Timeline	5
Chapter 3 Hardware Design	7
3.1 Wireless Sensor System Design	7
3.2 PCB Design.....	8
<i>3.2.1 Design Iterations</i>	<i>8</i>
<i>3.2.2 Overview of Features.....</i>	<i>11</i>
3.3 Calibration of Sensors	15
Chapter 4- Energy Harvesting.....	18
4.1 RF Energy Harvesting	18
4.2 RF to DC Converter	18
4.3 Antennas	19
4.4 Performance of Converter	20
Chapter 5 Constraints	23
5.1 Manufacturability.....	23
5.2 Environmental Impact	23
5.3 Social Sustainability	23
5.4 Ethics/Social Context.....	24
5.5 Economics.....	25
5.6 Health/Safety	25

5.7 Civic Engagement/Regulatory Standards	25
Chapter 6 Conclusion	27
6.1 Accomplishments	27
6.2 Future Work	27
6.3 Relevant Coursework	28
References	29
Appendix	30
Code	30
<i>Arduino Code:</i>	<i>30</i>
<i>Processing Code</i>	<i>42</i>
<i>Bill of Materials.....</i>	<i>53</i>

List of Figures/Tables

Figure 2.1: PCB Board Layout Design	5
Figure 2.2: Schedule	5
Figure 3.1: Original Device Design	7
Figure 3.2: Final Device Design	7
Figure 3.3: First Iteration Board Layout	8
Figure 3.4: First Iteration Physical Build	8
Figure 3.5: Second Iteration Board Layout	9
Figure 3.6: Second Iteration Physical Build	9
Figure 3.7: Final Iteration Layout	10
Figure 3.8: Final Iteration Physical Build	10
Figure 3.9: Final PCB Design	11
Figure 3.10: Charging Circuit	12
Figure 3.11: Switch Circuit	12
Figure 3.12: Pulse Rate Sensor	13
Figure 3.13: LMT70 Temperature Sensor	13
Figure 3.14: Temperature Sensor Circuit Ports	13
Figure 3.15: Arduino Pro Mini	14
Figure 3.16: FTDI Chip and Micro USB input	15
Figure 3.17: LMT70 Temperature Sensor	15
Figure 3.18: Pulse Sensor Front and Back	16
Figure 3.19: LMT70 Evaluation Board	16
Figure 3.20: Data Display of LMT70 Evaluation Board	17
Figure 3.21: Data Display of Wireless Sensor System	17
Figure 4.1: P2110B Evaluation Board	18
Figure 4.2: 915 MHz Patch Antenna	19
Figure 4.3: 915 MHz Dipole Antenna	19
Figure 4.4: Patch Antenna S11 Parameter	19
Figure 4.5: Dipole Antenna S11 Parameter	20
Figure 4.6: Signal Generator Settings	21
Figure 4.7: Antenna Placement	21
Figure 4.8: Antenna Placement	22
Figure 4.9: Oscilloscope Readout	22

Chapter 1 Introduction

1.1 Project Background

This project is to be used as a personal health monitoring device. Personal health monitoring is a way for people to use technology to take a more active role in their health and caring for themselves. [3] These devices track different parameters and display them to the user. Personal health monitoring is effective when the results from a person's device drive them to make healthier choices in their everyday life. Ideally, the more someone monitors their health and makes healthier choices because of it, they will go on to live a much healthier life. When a person is healthier, they require less frequent doctor visits, except for regular check-ups, as these are still very important.), which saves them a lot of time and money in health care. Furthermore, when they grow older, they will still require less unnecessary doctor visits, which means that they will retain their independence for longer than someone that did not choose to make healthier choices. We want people to be able to use our device as a way to monitor their health and live a healthy and long life. As well, this device can be used for infant, elderly and patient care by providing continuous monitoring of vital signs and serve as an automated aid to the nurse and care provider.

Therefore, it is also important that the health monitoring device transmits data wirelessly, as a person should not be tethered to a computer or phone to monitor their health. They should be able to go out and live their life without worrying if their health is being monitored

Many devices measure parameters such as steps, hours of exercise, calories, pulse rate, or glucose levels, among others. Parameters such as glucose, pulse rate, and temperature are used because they can clearly show whether a person is in need of immediate care. For example, if a person's pulse rate drops below a certain threshold, or if it rises above a certain threshold, it is a clear indication that a person is having cardiac issues and needs to be hospitalized immediately. Parameters such as calories, hours of exercise and steps are measured to give someone an idea of how healthy they are on a daily basis as well to help them reach personal goals. For example, if someone sets a goal of exercising for one hour every day and has a device that tells them exactly how many hours that they have logged, they will be more motivated and realize that their goal is closer than they might have thought.

When researching personal health monitoring devices, we found that most devices utilized a rechargeable battery to allow the user an extended period of time to be able to use their devices before plugging it back into power. This is an essential feature to any personal device. We did not find, however, any device that utilized any kind of energy harvesting technology. Our project advisor introduced this project to us as an RF energy harvesting project, so we wanted to implement RF energy harvesting into our device.

1.2 Problem Statement

In today's busy world, people often don't have time for unnecessary doctor visits¹. An unnecessary visit is a doctor visit that could have been prevented by a person making healthier

¹ Regular check-ups are essential to good health and should in no way be avoided.

choices, or one that could be replaced by a physician viewing a person's tracked vital signs since their last visit. Our device attempts to solve this problem by monitoring a person's vital signs, including temperature and pulse rate, and transmit them to a computer over Bluetooth [7]. Using our device a person can monitor their own health and make choices to improve it. Furthermore, the data is transmitted to a computer where it can be transmitted to a doctor or other health care professional.

SIDS (Sudden Infant Death Syndrome) is an unexpected death of an infant, usually during sleep. SIDS accounted for 39.4 infant deaths per 100,000 born in 2015 [8]. While this is not a large percentage of infants, it is still problematic simply because it can be prevented with proper monitoring. Our device can also be used to prevent SIDS when an infant is wearing our device on its foot using a sock. Vital signs can be monitored this way and any irregularities can be quickly seen and proper action can be taken.

As non-renewable resources dwindle in the world, we look more and more to renewable resources to power our everyday devices. Many devices are powered by solar power, or thermoelectric power, but none that we saw were powered using RF energy harvesting. Since the sun is not always out, and it is not always easy to find a good heat source, RF energy harvesting will become more popular as the technology improves. Our device attempts to use RF energy harvesting at 915 MHz to charge and power our device.

1.3 Project Objectives

1. Miniaturize the project into a more mobile and simpler prototype that will allow the user to comfortably wear the device, easily learn the functionality of the product, and use it for their own individual needs and benefits.
2. Improve sensors to more accurately record and display vital signs.
3. Create a PCB design to be powered mostly through energy harvesting power sources rather than using a traditional outlet charger to power the system.
4. Create an RF harvesting system that harvests power to charge our device using 915 MHz waves.

Chapter 2 Project Requirements

2.1 Usability

Usability is a very important requirement that pertains to our senior design project because we want our device to be user friendly in order for everyone to easily use our product regardless of a person's technical background. A problem we saw with our initial iteration was that our device was setup using a breadboard where the components had to be connected in the right manner in order for the system to properly work. The first steps we took to accomplish better usability within our project was by designing our own PCB board layout so that we could interconnect our sensors and means to power the device into a single wireless sensor system. We also included extra features onto the board such as a switch to turn the device ON and OFF to save power as well as pin connectors that allow for the user to easily interchange parts of the system such as sensors and battery if certain components are not working properly. Our wireless sensor system helps improve usability with our project because it takes away a lot of the steps needed to setting up our personal health monitoring device such as soldering components on the board or connecting devices using wires. Another aspect of the project that helps implement this idea of usability for our device is with a data display we have created for the user to view his or her results. Our data display consists of temperature readings at Celsius and Fahrenheit, Pulse readings, and BPM readings. Once the user turns on their personal health monitoring device, they will be able to view the results of their body temperature and pulse rate at real time through the data display. By implementing a data display that is connected with our wireless sensor system, this allows the user to easily monitor their own vital signs by themselves without consistently having to go to hospitals in order to detect changes in their health.

2.2 Accuracy

Accuracy is a parameter that measures the quality or degree at which a calculation must match a standard correct value. With that in mind, accuracy is another important requirement within in our senior design project due to the fact that components such as our vital sign sensors need to have a very precise and accurate margin of error. The difference in the accuracy of sensors can be the difference between life and death when measuring basic vital signs of a user since our device will need to be able to detect these changes within a user's health. On a similar note, the means to power our device must be able to consistently receive the proper amount of energy to power the rechargeable battery and the whole system. That is why we have included two power sources on our wireless sensor system which include an RF energy harvesting port and a Micro USB input. If our RF energy harvesting port is not properly receiving enough power within the system, the user can use the Micro USB as a fail-safe feature to help charge the remainder of our device's battery.

2.3 Reliability

Reliability of a product is another key requirement when dealing with our personal health monitoring device since it must consistently work and be able to monitor the basic vital signs of a user. When designing the PCB board for our wireless sensor system, we added some redundancies for the board such as the power. To help ensure that the reliability to power our device remains consistent, we included two ways of receiving power to our wireless sensor system. We decided

to include an RF energy harvesting port to utilize the ambient sources in order to convert the 915 MHz frequency to DC power and a Micro USB input in case the RF to DC converter is not properly creating enough power. The RF to DC converter we are using will help us get a constant output voltage at 3.8V and supply current of 30mA at a frequency of 915 MHz. The power sources are all interconnected on our PCB board as well as multiple other new features we have added to help with the overall project requirements. Some of these new features include an ON/OFF switch that easily be activated through a tactile button switch in order to minimize power from being wasted when our device is not being used. On a similar note, we also included a rechargeable battery that will help store the power collected from the previously mentioned power sources. Finally, our PCB board features pin connectors in order to easily interchange the battery and vital sign sensors if they are broken or not being used. These are some of the new components that we have added to our new wireless sensor system in order to help create a more reliable personal health monitoring device.

2.4 Safety

Safety is one of the biggest concerns that usually arise when creating a device that will be used by the public especially with products that will be implemented as wearable technology. With that in mind, when designing our whole personal health monitoring device, we took into account both long term and short term problems that may arise when a user is using our device. For instance, when dealing with radio frequencies that help power our whole wireless sensor system, we decided to choose a low frequency at 915 MHz because it is within the legal regulations/restrictions and is safe enough where there are no harmful effects to the user's health in the long run [4]. As for our wireless sensor system itself, we designed the PCB board so that everyone can easily use the system without needing any kind of technical background. The pin connectors are already included on the board to prevent the user from needing to solder components onto the board where they could possibly hurt themselves and instead can effortlessly plug in the sensors and rechargeable battery to the right port when they are going to use the device.

2.5 Art

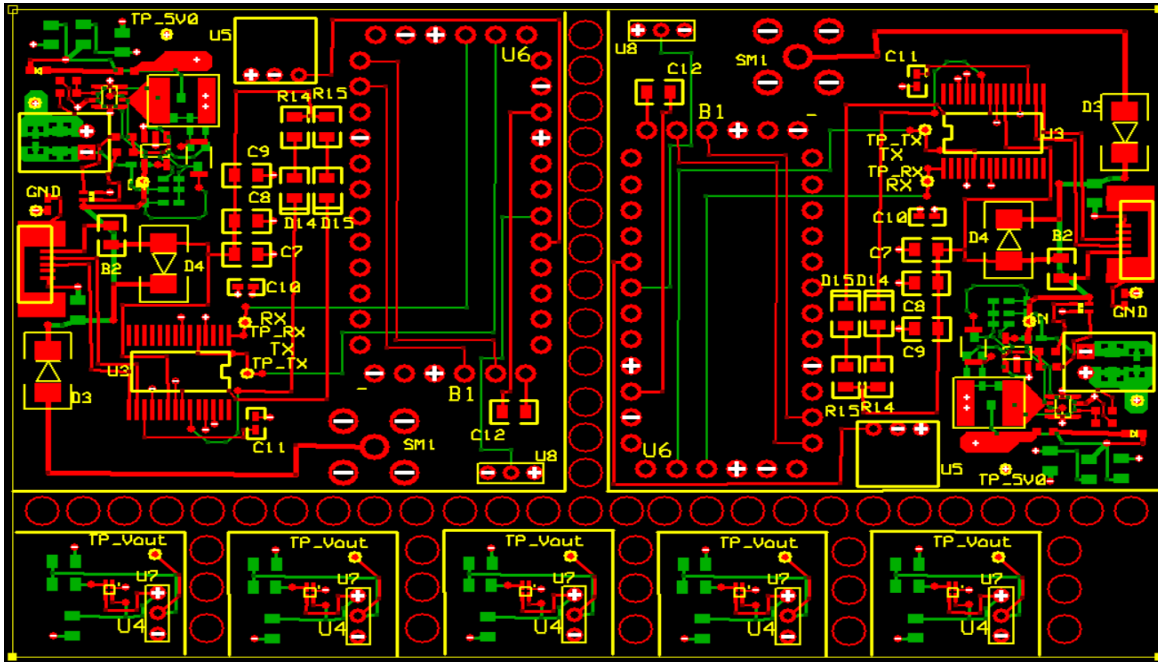


Figure 2.1: PCB Board Layout Design

Figure 2.1 shows the complete layout of our PCB board that was manufactured. The complete PCB includes two main boards that house the wireless sensor system, and five temperature sensor boards. The large circles between the boards are there so that we can easily cut the boards apart.

2.6 Timeline

Ambient Life	2017																													
	Jan	1	2	3	4	Feb	1	2	3	4	Mar	1	2	3	4	Apr	1	2	3	4	May	1	2	3	4	Jun	1	2	3	4
Implementation phase																														
Test and Calibrate Sensors and Power Modules	n	n	n	n	n	n	n	n	n	n	n	n	n	n	D	f														
PCB Design of Wireless Sensor System	n	n	n	n	n	n	n	n	n	n	n	n	n	n	D	f														
Solder and Troubleshoot PCB Board	n	n	n	n	n	n	n	n	n	n	n	n	n	n	D	f														
Implement Bluetooth Communication for Data Transfer	n	n	n	n	n	n	n	n	n	n	n	n	n	n	D	f														
Create and Design a RF Energy Harvesting Board	n	n	n	n	n	n	n	n	n	n	n	n	n	n	D	f														
Closure phase																														
Final Report																n	n	n	n	n	n	n	n	n	n	n	n	D	f	
Presentation Prepared																n	n	n	n	n	n	n	D	f						
Annotations																														
W Duration (Weeks)																														
n	normal																													
D	Deadline																													
f	Finished																													

Figure 2.2: Schedule

In Figure 2.2, we show the timeline of our yearly schedule for our Senior Design Project. As you can see, we prioritized majority of our project work during the months of January and

February for Winter Quarter in order to get a head start and hopefully finish some of the implementation phases of our project. For the Spring Quarter, we wanted to focus on finishing up the remainder of the project as well as begin creating and practicing our PowerPoint presentation for the Senior Design Conference in the month of May. After we finish with the implementation phase and presentation, we are now solely focusing on finishing the Senior Design Thesis that we will turn in at the end of the year.

Chapter 3 Hardware Design

3.1 Wireless Sensor System Design

We designed our own wireless sensor system from the one that we inherited from the previous project. There were many improvements that we wanted to make and many features that we wanted to include in our sensor system compared to the original device. We were fortunate that the device that we inherited functioned well, so it was a very good place to start.

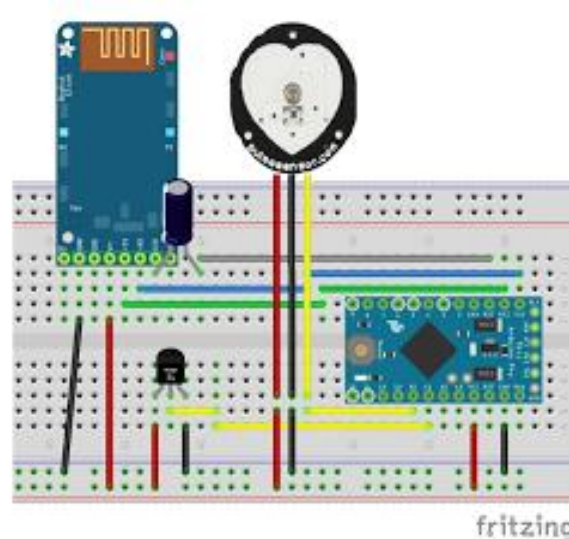


Figure 3.1: Original Device Design

In Fig. 3.1 is our original system that was inherited from a previous group. This design includes the basic features that are required for the device to work as designed. The first feature is an Arduino Pro Mini microcontroller to process and send data that we receive. Next, we have an Adafruit Bluefruit EZ-Link Bluetooth module to send and receive data from the microcontroller. Finally, we have our pulse sensor and our TMP35 temperature sensor. This design was very useful in testing different components as they can be easily interchanged without any major modifications or soldering work. The overall size of this device is around 45.94 cm^2 . This size is much too large to be used in any kind of wearable application. One of our main project objective was to miniaturize this device so that it could be easily worn by a person

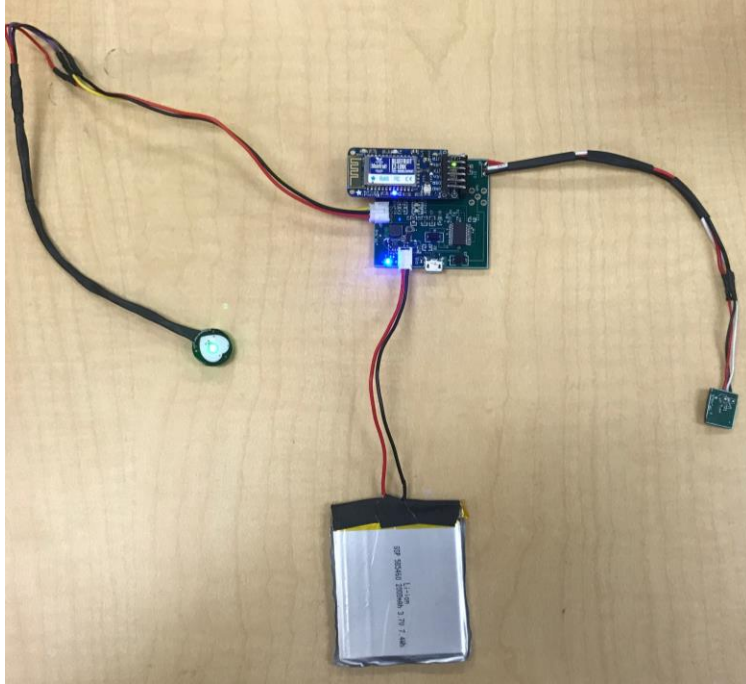


Figure 3.2: Final Device Design

In Fig. 3.2 is the final design of our wireless sensor system. We designed and assembled our own PCB to replace the breadboard from the original design. We used a lot of the same major components as the original design, but also used a couple of new and replacement parts. For example, we included a rechargeable battery in our new design that allows the device to be used for hours without having to be plugged in. We also replaced our temperature sensor with a new module that is much smaller and much more accurate. We did however, keep the pulse sensor, Arduino, and Bluetooth module the same. We were able to reduce the area to 22.84 cm², a reduction of around 48%. This device is much more adaptable for wearable applications and met our project objective.

3.2 PCB Design

3.2.1 Design Iterations

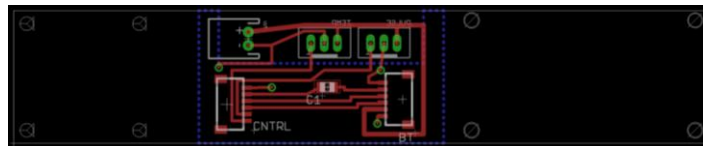


Figure 3.3: First Iteration Board Layout



Figure 3.4: First Iteration Physical Build

In both Fig. 3.3 and Fig. 3.4, we show the first iteration of our PCB design and physical build respectively. The top figure shows the layout in EAGLE, and the bottom figure shows the manufactured and assembled PCB. In this iteration, we attempted to keep all of the components on the top of the board to reduce danger associated with components touching skin of the wearer. We also attempted to use the TinyDuino stackable Arduino platform. This Arduino features several 2 cm² boards that stack on top of each other using header pins. We used the main board, the micro USB connector board, and the prototyping board so that we could connect all of our sensors and modules. This board has the same overall features as the original design, utilizing both pulse and temperature measurement, as well as Bluetooth transmission. The first problem that we saw with this design is that it was not a good shape. Although the area is only 21.42 cm², the long, narrow shape of the board is not becoming of a wearable design, which should be more compact. Another problem with this design is in the wires that must attach from the top of the TinyDuino to the board itself. These wires are not only not aesthetically pleasing, but they pose a safety risk if they come loose and cause a short circuit in the device, or come into contact with the wearer's skin. We kept these issues in mind while designing our second PCB.

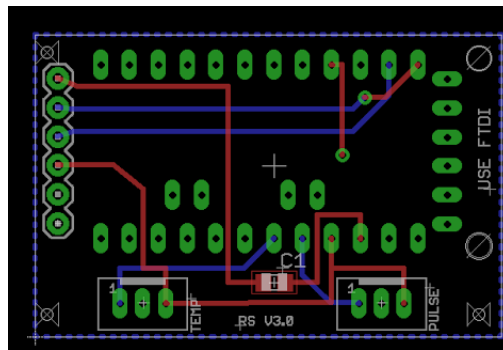


Figure 3.5: Second Iteration Board Layout

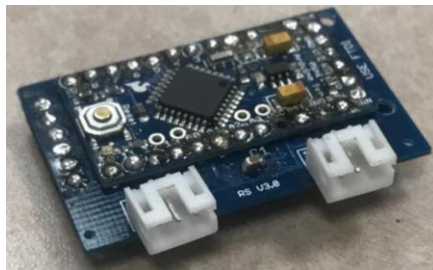


Figure 3.6: Second Iteration Physical Build

In Fig. 3.5 and Fig. 3.6, we show our second board iteration layout and assembled PCB, respectively. This board is functionally the same as the previous iteration, with the exception of the return of Arduino Pro Mini, in place of the TinyDuino. This board is arranged slightly differently in that we attempted to conserve space by placing the Arduino on the top side of the board and the Bluetooth module on the bottom of the board. This board did an excellent job of saving space, as it came in at 11.13 cm², and did so in a much more compact design than the previous board. While the dimensions of this board met our expectations and requirements, we wanted our final project to have more functionality than the barebones capabilities of our original design, most importantly a rechargeable battery. Furthermore, we realized that it was not a good idea to have the Bluetooth module mounted on the bottom of the board as the monopole antenna

would not be able to get as good of a signal. It is vital that the antenna have the best signal possible while being used in an application such as monitoring an infant to help prevent SIDS. These issues were the main inspiration for us as we designed our final PCB layout.

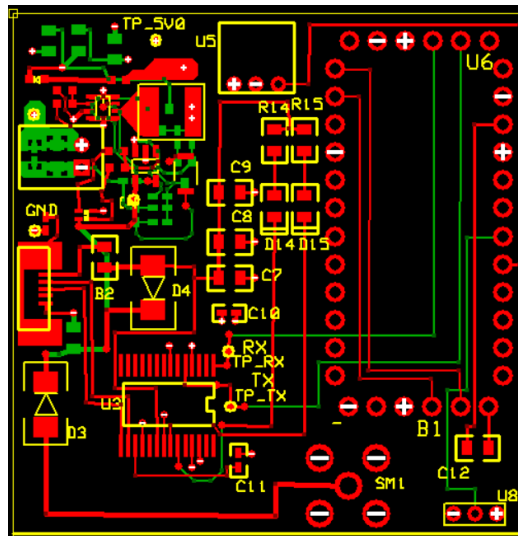


Figure 3.7: Final Iteration Layout



Figure 3.8: Final Iteration Physical Build

After a longer design process, we successfully manufactured our final board layout (Fig. 3.7), and manufactured it (Fig. 3.8). Coming from our second iteration, we had to make this board slightly larger in order to make room for the additional functions that we wanted to include. The area of this board measures around 22.84 cm^2 , which is still an area reduction of approximately 48% compared to our original breadboard layout. We will go into more detail about the functions in the next section. One of the major layout improvements that we made on this board concerning the placement of the Bluetooth module and Arduino board is that we used header pins to situate the Bluetooth module above the Arduino. However, the Bluetooth chip, while securely in place, can be easily removed if the user needs to access the Arduino underneath. We are very happy with this design and we believe that it meets all of our project objectives concerning the PCB layout.

3.2.2 Overview of Features



Figure 3.9: Final PCB Design

While designing our circuit board, there are several features that we included to give our device more functionality, while also completing the same functions as the original design. The first thing as shown in Fig. 3.9 is that we added was a USB to serial converter to complement the Bluetooth module. The USB to serial device can upload code to the board, and download data from it when the Bluetooth connection is not working. The converter is more of a failsafe, as we want the Bluetooth module to be the primary method of transmission for our device. The next feature that we added was a physical micro USB connector. We chose this connector over others, such as mini USB because micro USB is the connection used by most android phones. Because of this, we figured that most people using the product would have a cable of their own to use with the device. Users can use this port to upload or download from the board, or to charge the rechargeable battery that we have also included. The battery plugs in next to the USB port on the device and is capable of powering the device for several hours on a single charge. The next function that we added to the board is LED indicators. Two LEDs closest to the Bluetooth module indicate data being uploaded or downloaded from the board through the USB to serial converter. Another set of LEDs closest to the battery port indicate whether the device is charging. The orange LED indicates that the device is charging and should be left plugged in while the green LED indicates that the device is fully charged and ready to be used. The final LED indicator is a blue LED that indicates whether the device is powered on. This LED illuminates when the tactile push button on the board is pressed. This button is used to power down the device when it is not in use in order to save battery power for when the device is needed. The next feature that we added were connectors for the sensors as well as the battery. These connectors were used in case a sensor breaks and needs to be replaced, or if the battery needs to be replaced. With connectors, changes can be made without having to do any soldering, a skill that not everyone using our device will have. Finally, since we are using two different power source circuits, we wanted to make sure that no power was able to travel backward in the system and damage another part of the circuit. To accomplish this, we included two Schottky diodes in series with both of our sources. This way, when one power source is in use, the other will be safe from damage.

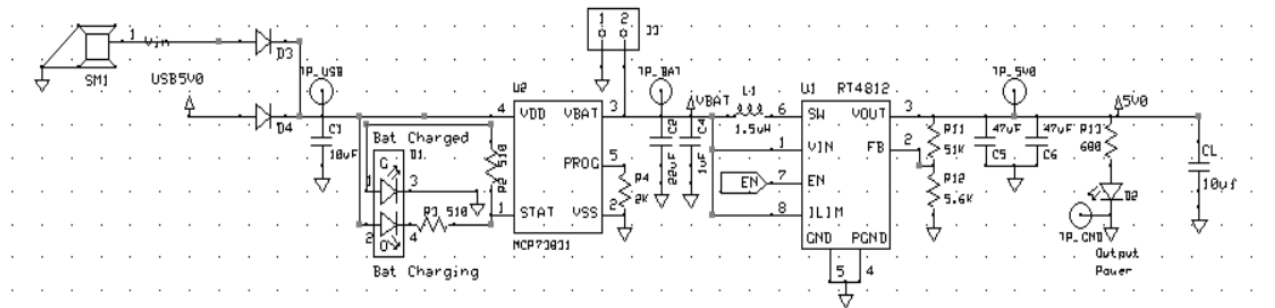


Figure 3.10: Charging Circuit

This circuit in Fig. 3.10 is used to power the device. It consists of a boost converter, a battery charge controller, and a battery connection. The dual LED in the circuit is used to show the status of the battery. An orange light signals battery charging, and a green light signals a charged battery. The capacitors C1, C2, C5, and C6 are used to help measure the voltage at the specified test points to verify voltages in order to debug the circuit. The feedback resistors are used to reduce noise from the voltage output. The blue LED light indicates that the whole circuit is ON and functioning. The enable pin is connected to another circuit with a switch that turns the whole system ON and OFF. Finally, the Schottky diodes are used to prevent power from going backward in the system.

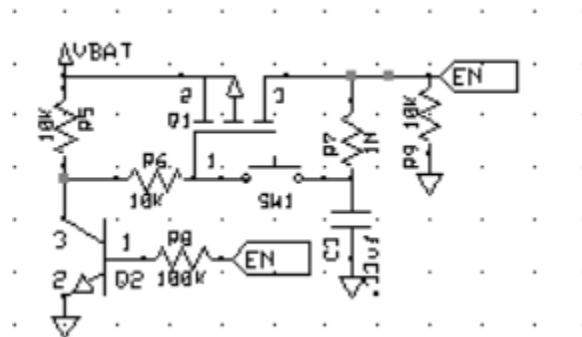


Figure 3.11: Switch Circuit

As for Fig. 3.11, we are showing our switch circuit that turns our whole wireless sensor system ON/OFF. The Vbat is the battery voltage that is fed from the battery of the first circuit. Once the switch is ON (closed) the BJT becomes saturated and the current flows through the Drain and Source of the MOSFET due to the negative voltage to the gate. The voltage at 3.3V is then transferred through the boost converter in the first circuit to boost the voltage to 5V.

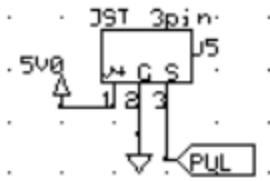


Figure 3.12: Pulse Rate Sensor

For Fig. 3.12, this Pulse Rate Sensor has 5V connected from the output of the boost converter and the other pin connected to the Arduino that will output the results.

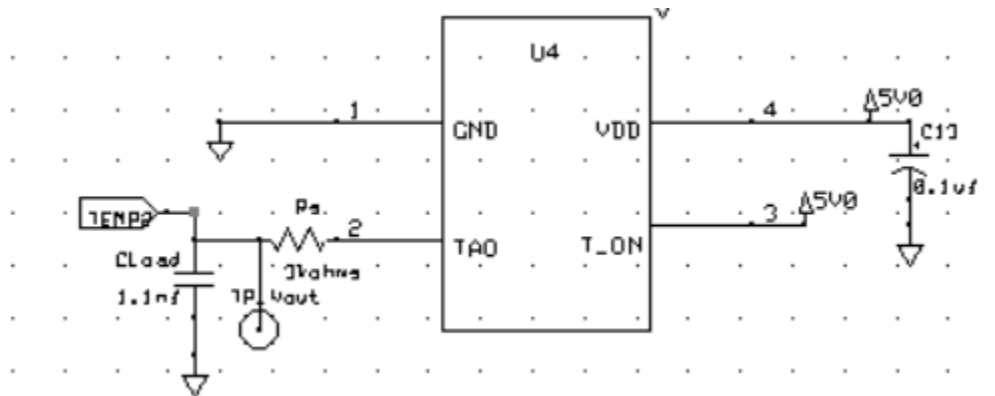


Figure 3.13: LMT70 Temperature Sensor

In Fig. 3.13, this is the temperature sensor that has 5V connected to the VDD and T_ON. The filter capacitor CLoad at the VDD helps minimize noise coupling and to maintain stable conditions for the temperature sensor. The Bypass capacitor C13 and resistor Rs help conduct an alternating current and are also connected to the TEMP pin that allows the temperature sensor to connect with the Arduino device. The GND pin is connected to ground.

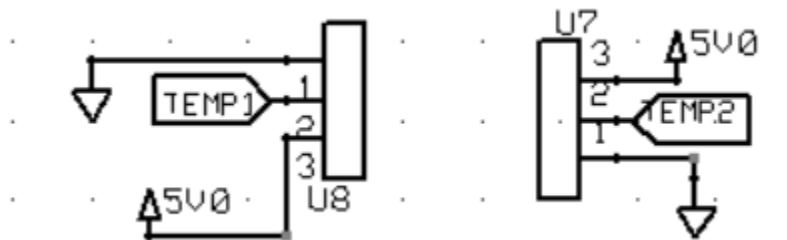


Figure 3.14: Temperature Sensor Circuit Ports

The circuit in Fig. 3.14 allows us to connect our temperature sensor to our Wireless Sensor System using wires to be able to touch certain parts of the skin and is separate from the whole sensor hub.

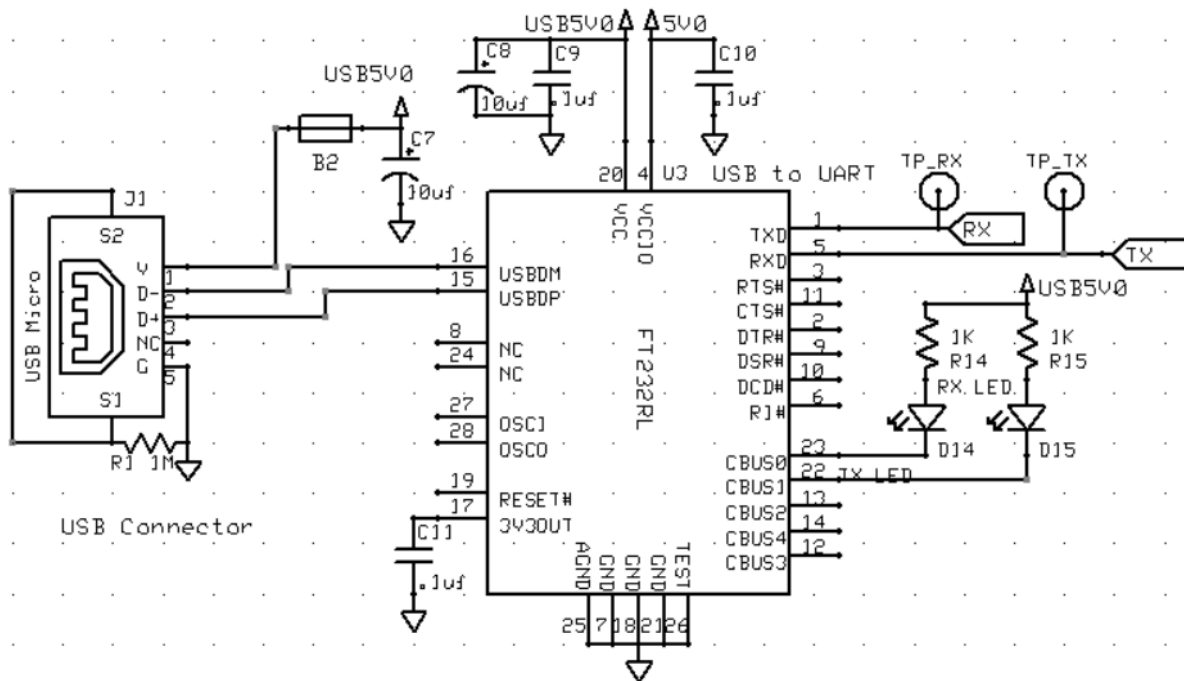


Figure 3.16: FTDI Chip and Micro USB input

This circuit in Fig. 3.16 is our USB and USB to serial device. The larger device is the FTDI chip. It reads data coming in from the USB port (on the left side) and converts it into data that the Arduino can read and process. It also takes data from the Arduino and converts it into data that a computer can process. There are capacitors on the voltage pins to minimize noise. One LED is used to signal data being transmitted, and another is used to signal data being received.

3.3 Calibration of Sensors

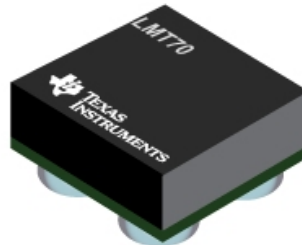


Figure 3.17: LMT70 Temperature Sensor

The TMP35 temperature sensor as shown in Fig. 3.17 is the original temperature sensor that we used to measure the body temperature vital sign. Unfortunately, with this particular sensor, the margin of error at $\pm 5^{\circ}\text{C}$ is too large to differentiate at temperature ranges such as high fevers

where it can be the difference between life and death for a user and a body size of 17.54 mm x 5.21 mm. With our new LMT70 temperature sensor, the margin of error is now at $\pm 0.1^{\circ}\text{C}$ and with a body size of 0.88mm x 0.88 mm which is an area reduction of around 98% from the original TMP35 temperature sensor. The parameters of this new temperature sensor consist of an operating temperature range from -20°C to 90°C , a supply voltage range from 2V to 5.5V, and a minimum supply current at 12uA. The specs of our LMT70 temperature sensor will have a temperature range from 29°C to 40°C in order to be able to monitor changes in a user's temperature to detect symptoms for colds that occur below 29°C and fevers that are dangerous above 40°C . Some other specs of our sensor include a supply voltage at 5V, which is the voltage being sent all throughout our wireless sensor system, a supply current at 30mA and an operating voltage output at 940mV.

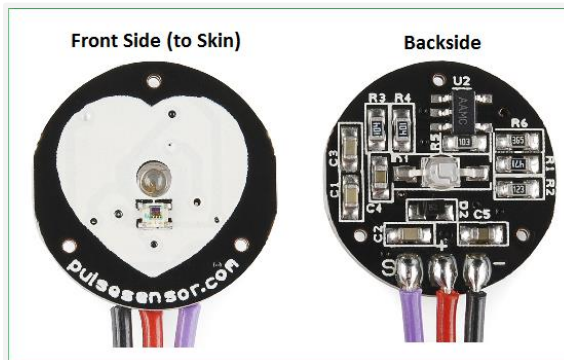


Figure 3.18: Pulse Sensor Front and Back

When calibrating our Pulse sensor in Fig. 3.18, we faced some initial challenges that prevented us from getting good, accurate readings of a person's pulse and BPM. These challenges consisted of ambient light, skin oil interference and ideal placement within the sensor. In order to compensate for the ambient light and skin oil interference, we decided to place a thin vinyl over the pulse sensor that minimized the noise and helped improve the accuracy of the readings. On a similar note, another problem we originally faced with sensor was with the placement of the user's finger because the sensor moves around a lot. To help solve this problem, we decided to attach a Velcro strap onto our sensor so that the user can easily place his/her finger to record their pulse and BPM measurements.

According to the American Heart Association, the average adult can range from 60 to 100 BPM [6]. While for newborn infants, the average can range from 120 to 160 BPM. These BPM reading ranges are very important to our project because it can help detect any kind of changes in a user's health. For instance, if a newborn infant is having a BPM reading that is less than 100, this is an indication that the infant is dealing with low blood oxide within their body. The control BPM reading ranges that we will specifically focus on for our senior design project are in the age groups from 18-25 years old since we will be measuring our own BPMs. These readings can range from 40 to 62 BPM for more athletic users and range from 74 to 81 BPM for the average person within this age group [5] [12].



Figure 3.19: LMT70 Evaluation Board

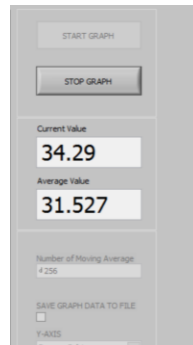


Figure 3.20: Data Display of LMT70 Evaluation Board

This is the LMT70 temperature sensor evaluation board shown in Fig. 3.19 that we were able to acquire from Texas Instruments. We used this evaluation board as a control temperature value in order to help validate the results we measured through the data display we created ourselves. In Fig. 3.20, this is the data display of the evaluation board using a Texas Instrument program that allowed us to measure a skin temperature value at 34.29°C. This is very close to the typical skin temperature value at 34°C according to the mayo clinic.

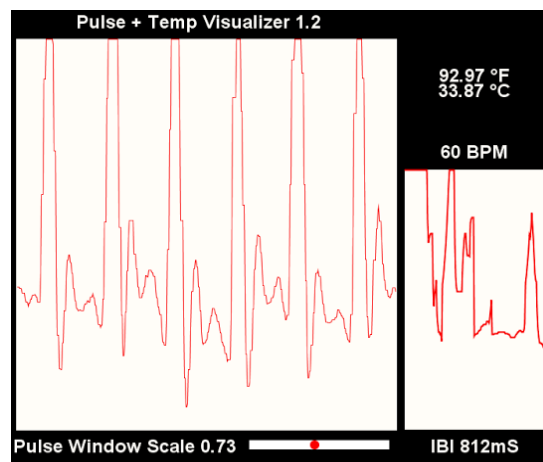


Figure 3.21: Data Display of Wireless Sensor System

To be able to read the measurements recorded from our wireless sensor system, we created our very own data display that shows the user's skin temperature at Celsius and Fahrenheit, pulse rate and BPM in Fig. 3.21. For our measured skin temperature, we were getting a value at 33.87°C that is converted to 92.97°F. As a result, there is a good variation of $\pm 0.42^{\circ}\text{C}$ between our measured skin temperature value and the typical skin temperature for validation. On a similar note, the pulse rate value we measured through our data display was at 60 BPM which is still between our control pulse rate ranges showing the validation of our pulse sensor.

Chapter 4 Energy Harvesting

4.1 RF Energy Harvesting

RF energy harvesting is an energy harvesting technique that involves the conversion of radio frequency, in our case 915 MHz, waves. RF waves are constantly around us in our everyday environment. We can capture and transmit these waves using antennas. Depending on how we configure our antennas, we can make them radiate in a certain way and operate at a certain frequency. We can evaluate these antennas operating frequency based on their S11 reflection coefficient. This parameter shows how well and how much power is received and reflected by an antenna at a particular frequency because we chose the operating range of 915 MHz, we had to find antennas that were able to transmit and receive signal effectively at this frequency.

We chose the frequency range 902-928 MHz because it is designated by the FCC for amateur radio use as well as radiolocation, and RF Identification. Although there are several frequency bands like this, such as 2.4 GHz, or 5.8 GHz, we decided to stay at a lower frequency because it is safer to use a lower frequency while transmitting higher power waves needed for RF-DC conversion.

4.2 RF to DC Converter

To power our device, along with USB power, we are also implementing an RF to DC converter to harvest RF energy from the air. The device that we used to accomplish this is the Powercast P2110B Powerharvester. We purchased the module on the P2110B Evaluation board. The evaluation board includes several charging capacitors to store and deliver the output power, as well as places to measure the output current.



Figure 4.1: P2110B Evaluation Board

Pictured above in Fig. 4.1 is the P2110B evaluation board. On the board, the small chip next to the brass connector is the actual P2110 module. This board is capable of outputting 50 mA of current at 4.8 volts. Out of the box, the board is set to output 3.3 volts, but can be modified to deliver more voltage using a resistor that is added to the board. The data sheet includes an equation to calculate the resistance needed to boost the voltage. We wanted to get around 3.8 volts, as this is the voltage required by our boost converter to charge the battery. Using the equation in the datasheet, we obtained a resistance of 3.16 MOhms. The red square in the figure above shows the location of this resistor.

4.3 Antennas



Figure 4.2: 915 MHz Patch Antenna



Figure 4.3: 915 MHz Dipole Antenna

Our power harvesting module came with two antennas, a patch antenna (Fig. 4.2), and a dipole antenna (Fig. 4.3). The patch antenna is a directional antenna, meaning that it only receives and transmits signal in one direction. Because of this, we decided to use this antenna as our transmitter. This antenna also has a gain of 6 dBi. The dipole antenna is an omni-directional antenna that can radiate in all directions. We use this antenna as the receiving antenna on the converter so that the converter can be placed anywhere as long as it is near the transmitter. This antenna has a gain of 1 dBi according to the manufacturer's data sheet.

It was important to test the operating frequency of these antennas to make sure that they function effectively at 915 MHz and will work with our converter.

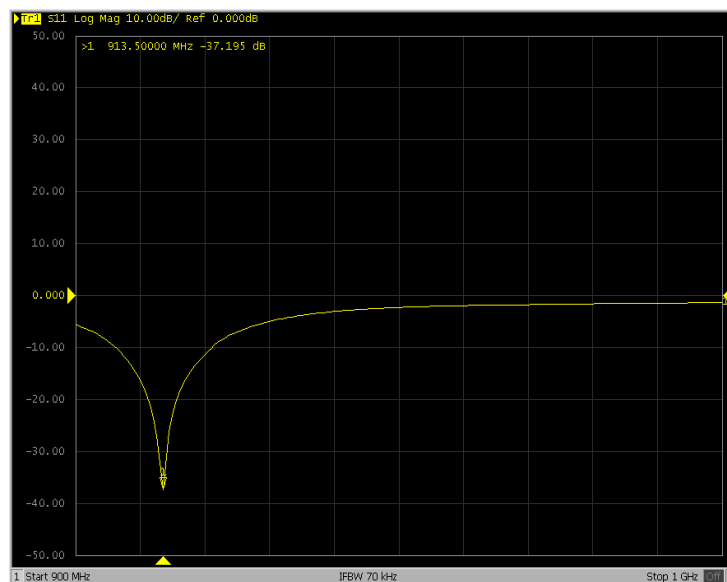


Figure 4.4: Patch Antenna S11 Parameter

In Fig. 4.4 is the S11 parameter for the patch antenna that came with our RF-DC converter [10]. To measure this parameter, we used a Vector Network Analyzer (VNA). From the display, we can see that the antenna has an effective operating frequency of 913.5 MHz, which is close to 915 MHz and in our converter's operating frequency range.

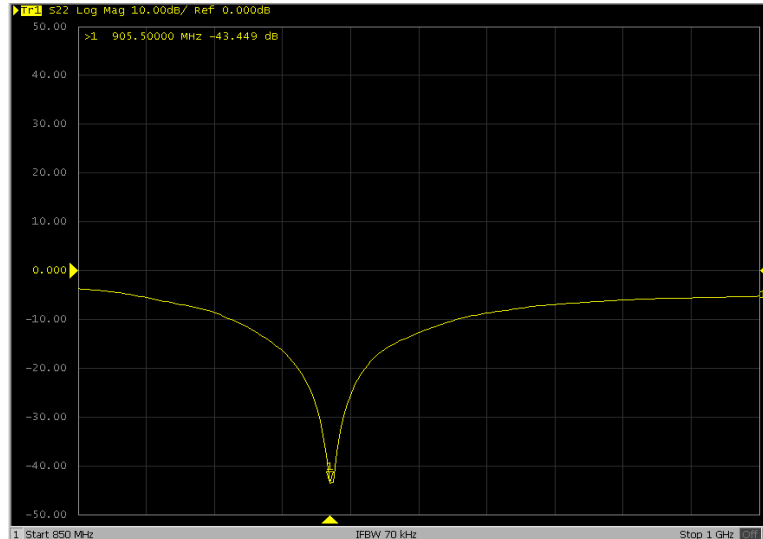


Figure 4.5: Dipole Antenna S11 Parameter

While Fig. 4.5 shows the S11 parameter of the dipole antenna. The VNA shows that the antenna is radiating electromagnetic power effectively at 905.5 MHz and exhibits minimum reflection. While this is not as close to 915 MHz as the patch, it is still well within the operating range of the converter of 902-928 MHz. Before we purchased these antennas, we attempted to use some of the antennas that are available to us in the lab. When we measured the S11 parameters of these antennas, we found that they operated at frequencies closer to 1.1 GHz. These antennas do not operate in the effective range of our converter and therefore cannot be used.

4.4 Performance of Converter

The RF-DC converter works by attaching the patch antenna to a signal generator machine set to 915 MHz and any power level above 0 dBm, as the converter does not power up at power levels below 0 dBm as shown in Fig. 4.6. We chose 3 dBm because it is a high enough power level to keep the converter powered at a reasonable distance.

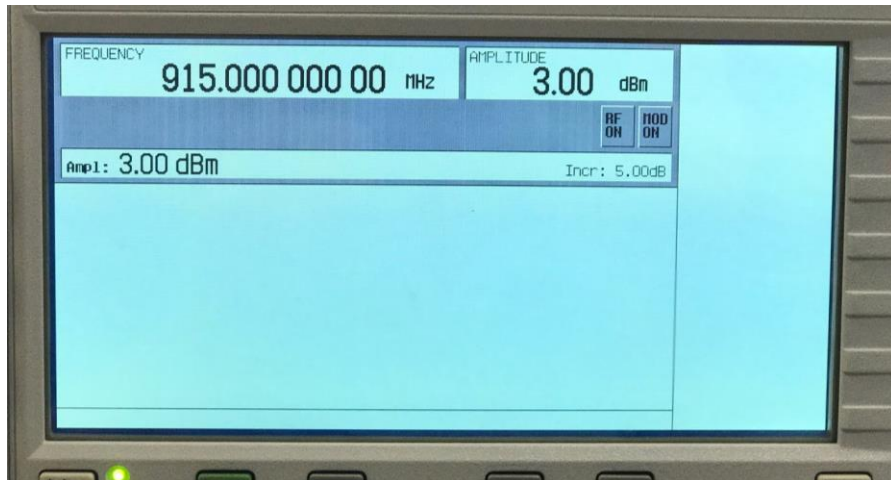


Figure 4.6: Signal Generator Settings

Once the Signal Generator is running, we place the antennas in front of each other in Fig. 4.7. When we do this, the converter powers on and begins storing charge. We see that the converter is turned on by a green LED indicator illuminating in Fig. 4.8. The figure below shows the converter powered on. The red arrow points to the indicator LED.

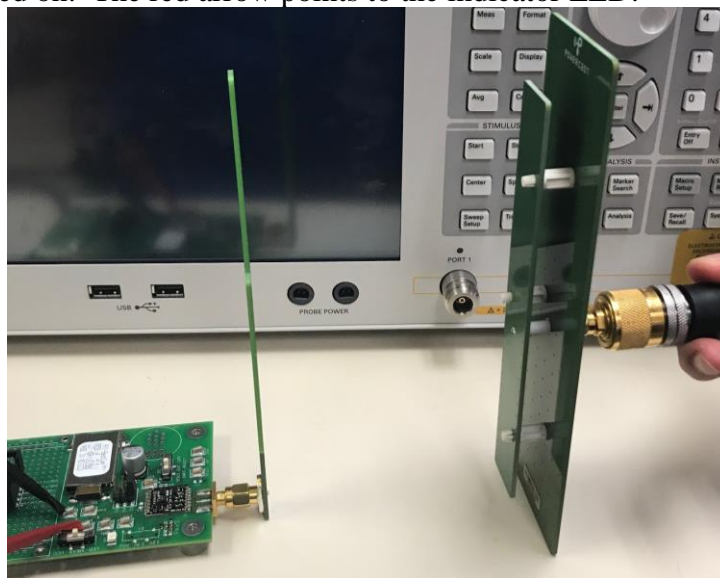


Figure 4.7: Antenna Placement

Chapter 5 Constraints

5.1 Manufacturability

Manufacturability is one of the first constraints we faced when dealing with our senior design project. The original plan involving our wireless sensor system was to design a PCB board that was small enough for wearability on the wrist. However, we needed to include more features onto the PCB board in order to improve the basic functionality of our wireless sensor system and design the sensor hub in time to fabricate and finish building the prototype before our designated deadlines. To accommodate for the constraints, we decided to give up a bit of space on the final iteration of our PCB board so that we could include the extra features on top of the original components that were necessary to complete the wireless sensor system. This final design is able to still accomplish our system to be a wearable device specifically onto the wrist as well as expand on the functionality to improve the overall device.

For our RF energy harvesting setup, we decided to create our very own RF to DC converter in order to minimize cost and have a working prototype to help show new ways of powering wearable technology. The original converter design consisted of a charge pump, schottky diodes and capacitors that would help convert radio frequencies at 915 MHz into DC power for our device. Unfortunately, a constraint we came across with our design was that the board was not getting the right voltage output and supply current to power our device. We also were not able to properly interconnect our original converter with our PCB board. As a result, the backup we decided to choose was the P2110B RF energy harvesting chip for our wireless sensor system. This new evaluation board is able to create a voltage output of 3.8V and supply current of 30 mA that will help power our personal health monitoring device.

5.2 Environmental Impact

For our Senior Design Project, we are creating a Wireless Sensor System that is using energy harvesting methods to power our whole system in order take the first step to ending the dependency on nonrenewable energy sources. With that in mind, our group has designed a PCB layout of our Wireless Sensor System that has a hybrid capability of being powered through RF energy harvesting and a basic Micro USB. Our design also includes a temperature sensor and heartbeat sensor that can be easily used by the user and interchanged if the sensor breaks or malfunctions allowing for the users to continually operate the same Wireless Sensor System and replace only the small aspects of the system to minimize cost and resources needed for our product.

The impact our Senior Design Project can have on the environment when disposing materials at the end of the product's life is very substantial because majority of the materials can be recycled to be used for some other purpose. However, if the materials or product are not disposed properly, it can have some impact on the environment because some of the materials can be harmful to animals and the planet. As a result, we have designed our Senior Design Project to have an expected duration of around 5 years.

5.3 Social Sustainability

Our product is designed to benefit users that want to take a more active role in their health and well-being. The device will measure their body temperature and their pulse rate to give them

a glimpse into their overall health. We are making the device to be as easy to use as possible and as maintenance free as possible. We want the user to be able to have the user turn on the device, connect it to their phone, and not have to worry about it again.

Furthermore, we want the application to be equally as easy to use as the device itself. The user will be able to record their readings at the touch of a button and view their results immediately. We hope to make the system inexpensive and easily obtainable for all users. Since our system will be used as a medical device to help people monitor their health, we do not foresee any negative impacts on health or welfare. However, we do not want people to think that our device is a replacement for regular check-ups with a doctor. People should continue to see their doctor and use our device supplementary. One area in which we must exercise caution is in our use of a high-power RF transmitter as the source of our RF power. High power RF waves can be harmful to humans at certain wavelengths, so we will make sure to use a setting that is not harmful.

5.4 Ethics/Social Context

Ethical Justification:

In our healthcare system today, many medical and technological advances have simplified and increased the productivity of professional medical doctors and surgeons in hospitals that would not have been possible in the past. However, an increase in the world population has made it much more difficult to constantly monitor patients in hospitals and at home due to time constraints with doctors and the expensive cost of healthcare. With these parameters, we have set out to accomplish, our personal health-monitoring device will in turn help monitor basic vital signs such as body temperature and pulse rate in order to detect any changes in their health. As a result, health monitoring applications such as wearable technology is the next step in technological advances for healthcare. This type of application allows for constant checkups on a patient's health resulting in fewer visits to the doctor only when needed which ultimately cheapens the cost of healthcare and an improves the overall medical system.

What it means to be a good engineer:

Our goal of any engineering project should be to help the end user. Our project is no different. We want to make sure that our product works when the patient receives it and that any repairs that need to be done to the device are simple and easy for any person to perform. We don't want the patient or their health care provider to have to spend valuable time and energy trying to fix a poorly made device that is difficult to work on. We want our product to be safe to use for both the patient and anyone around them, we will make sure to keep all voltage sources protected, and all connections are secure. As problems arise, we plan to make engineering choices that make our device safer and easier to use above all else.

Social Context:

Safety is one of our main concerns when designing our Senior Design project because we want to make sure that the patients are not at any risk of danger when using our product. The challenges we face when dealing with specific engineering ethics for our project is building a wireless sensor system that can properly record the necessary measurements accurately and precisely for the patients since the sensors from our system such as the temperature sensor cannot have big margins of error due to certain temperatures being at the threshold of life and death. Another challenge that is raised through our project is the implementation of energy harvesting

power sources. In our project, we are attempting to be more eco-friendly with our product in order to stop the dependency of using power outlets that use non-renewable resources to charge our technology such as fossil fuels and oil.

5.5 Economics

Making this product financially sustainable comes from the cost of the individual components. Currently the device is affordable since almost all of the hardware components are open sourced from well-known companies that provide reliable parts at low costs.

Very little is made “in house,” and this outsourcing of manufacturing basic parts allows our team to focus on pushing the technologies not readily available on the market. If we were a startup company just beginning to build our first prototype, we would be able to allocate more of our financial resources to building the RF energy harvesting aspect of the design. As the company grows, it could hire more staff to make previous parts more accurate (the temperature and pulse sensors). If we decided that we wanted to stick with buying basic components externally, the market ready components would improve in their reliability, functionality, and accuracy over time. In addition, these parts would cost less over time as well (looking at computer processing units as an example).

As the technology of the device evolves, so will the laws that require certain restrictions and usage conditions. When RF energy harvesting panels are reduced enough in size and are able to provide enough power to make the system self-sustaining, there will be other outcomes and effects when the wearer passes by various systems that emit signals within the range the device is able to absorb and convert to power. If it turns out that there are inconveniences and unexpected effects from various wearables taking in radio frequencies from others’ Wi-Fi routers, for example, each consumer could also pay a very small monthly RF usage fee that would compensate for the inconvenience on men and women who are unintentionally providing RF sources to power their device.

5.6 Health/Safety

Our product is designed to benefit users that want to take a more active role in their health and well-being. The device will measure their body temperature and their pulse rate to give them a glimpse into their overall health. We are making the device to be as easy to use as possible and as maintenance free as possible. We want the user to be able to have the user turn on the device, connect it to their phone, and not have to worry about it again.

Furthermore, we want the application to be equally as easy to use as the device itself. The user will be able to record their readings at the touch of a button and view their results immediately. We hope to make the system inexpensive and easily obtainable for all users. Since our system will be used as a medical device to help people monitor their health, we do not foresee any negative impacts on health or welfare. However, we do not want people to think that our device is a replacement for regular check-ups with a doctor. People should continue to see their doctor and use our device supplementary. One area in which we must exercise caution is in our use of a high-power RF transmitter as the source of our RF power. High power RF waves can be harmful to humans at certain wavelengths, so we will make sure to use a setting that is not harmful.

5.7 Civic Engagement/Regulatory Standards

For the RF harvesting portion of the project, we had to select a frequency to use for harvesting energy. We decided to use the frequency band of 902 - 928 MHz. We used this frequency because it is designated by the FCC for amateur use. This means that there are no major restrictions on it. Furthermore, we would not be interfering with any major transmissions. There are other frequency bands designated for this use, including 2.4 and 5.8 GHz, but we determined that since we would be using high power RF waves, it would be safer for us to use a lower frequency.

For our sensors, we wanted to make sure that our sensors had a margin of error that was within an acceptable range of safety for humans. For example, our previous temperature sensor had a margin of error of around 5 degrees. This was unacceptable because the normal range of a person's temperature usually fluctuates around 95-100 degrees. The sensor could be reading normal values, while the person may have a deadly fever. We made sure that our new sensor had a much smaller degree of error.

Chapter 6 Conclusion

6.1 Accomplishments

One of the first accomplishments that we were able to achieve through our senior design project was in miniaturizing the personal health-monitoring device for wearability. Through our different iterations in creating the best possible PCB board design, we were ultimately able to decrease the foot print by 48% from the original design to our final iteration. As a result, we also chose components that are included on the wireless sensor system that are small enough to fit onto the PCB board and still easily usable by the user.

The rechargeability aspect of our senior design project consisted of using RF energy harvesting and a Micro USB input as two ways of powering our device and rechargeable battery. In order to implement this, we included two terminals that connected to the rechargeable battery with Schottky diodes on our PCB board to prevent power from traveling backwards where it could possibly damage our power sources. Our main power source will be the RF to DC converter and the Micro USB input will act as the failsafe in case the RF energy harvesting port cannot properly generate enough power.

Bluetooth Communication was a key objective that we needed to execute for our project because wirelessly transferring data through our personal health monitoring device would help prevent users from being tethered to a computer and would make our system much more user friendly. As a result, we implemented a Bluetooth board onto our wireless sensor system that is compatible with our Arduino mini and vital sign sensors. Once we established connection between the necessary components, we then used the Bluetooth to send the data collected through the sensors to a data display we created onto a laptop.

Originally, our vital sign sensors were much bulkier and inaccurate due to challenges we faced with integrating them onto our wireless sensor system. By finding new pulse and temperature sensors that improve the accuracy of the measurements and decrease the body size of the overall sensors, this will help with the overall personal health monitoring system. The margin of error of the new sensors are much more accurate compared to old sensors and with a smaller body size, we can easily implement it onto our smaller PCB board design.

Finally, the last accomplishment to our senior design project was involving RF energy harvesting. With our original RF to DC converter, we were not able to properly create enough voltage output and supply current. We decided to choose another RF to DC converter that was able to get us a base voltage output at 3.3V and a supply current at 30 mA. Fortunately, with some modifications to the converter we were able to boost the voltage output to 3.8V and keep the supply current at 30mA which are the parameters needed to help power our rechargeable battery and entire wireless sensor system.

6.2 Future Work

Throughout the progress of our senior design project, we were eventually able to accomplish the project objective we set out to achieve. However, there are still improvements to our personal health-monitoring device that can be implemented in order to help improve the overall system. The first improvement would be to include an LED light to indicate the temperature sensor is on or off. Currently, the only way to tell if the sensor is working is to look at the changes in output voltage using a multimeter.

Another improvement that we took into account for our senior design project was to include a mobile app. By implementing a mobile app with our personal health monitoring device, this will be much more user friendly since users can control the device all through a single app. It would also help interconnect our system under one app where the data can easily be collected through our device and sent to an app on a user's phone allowing for easy accessibility to a user's health information.

Our original intent with our personal health monitoring device was to push towards implementing in a manner that will allow our device to become wearable for a user. With this in mind, we are hoping that a future group can create a case and a strap that a patient can wear and easily record their vital signs without it being too complicated for the user.

With the power aspect to our project, some improvements we see for the future is to possibly implement the RF energy harvesting chip onto the PCB design of our wireless sensor system. By applying the chip on our personal health monitoring device, this will minimize the size of the actual converter since it would now be directly interconnected with the entire wireless sensor system. On top of this, another improvement that would help with the RF energy harvesting is by creating or buying a cheap RF transmitter to help send the RF frequency waves to our personal health monitoring device to power the system. In doing so, it would take out the necessity of an expensive signal generator to send RF waves and would be much more simplified for a user to function since these RF transmitters can easily be turned ON/OFF.

Finally, including additional vital sign sensors on our wireless sensor system will help expand the functionality of the personal health monitoring device towards other applications such as helping patients recover from addictions through the rehabilitation process or even monitoring newborn infants and preventing Sudden Infant Death Syndrome by detecting any changes in their breathing pattern via monitoring pulse rate. Electrocardiogram Sensors help detect electronic impulses from the heart which can be used to observe possible irregularities within the user. On the other hand, Cortisol sensors can detect stress within a user through measurements of a person's saliva. Although this sensor is geared more towards patients dealing with addictions, it is very applicable for this purpose since addictions such as smoking tend to be triggered through stress [1] [9] [2].

6.3 Relevant Coursework

Several courses that we have taken at Santa Clara University were particularly helpful in the completion of our senior design project. ELEN 164 and the knowledge that we gained about DC-DC converters was helpful for implementing a boost converter to charge the battery of our device. ELEN 192 was very important for our project as it showed us how to properly design a circuit board, as well as how to solder and assemble the board. ELEN 105 was also very helpful as we learned about antennas, as well as how to use a Vector Network Analyzer, which were very important for the energy harvesting side of the project. COEN 44 and 12 were also helpful in writing some of the code that was required on the software side of the project.

References

- [1] <http://www.medicalnewstoday.com/info/addiction>
- [2] https://www.cdc.gov/tobacco/data_statistics/fact_sheets/adult_data/cig_smoking/
- [3] <http://cs.emis.de/LNI/Proceedings/Proceedings15/GI-Proceedings.15-14.pdf>
- [4] <http://sandersrfconsulting.com/rf-engineering-resources/fcc-part-15-intentional-radiators/>
- [5] <http://www.livestrong.com/article/152337-what-is-a-normal-heart-rate-for-men/>
- [6] <https://www.americannursetoday.com/identifying-complete-heart-block-in-elderly-patients/>
- [7] <http://www.telegraph.co.uk/technology/0/bluetooth-5-need-know-new-wireless-technology/>
- [8] <https://www.cdc.gov/sids/data.htm>
- [9] <https://my.clevelandclinic.org/health/articles/stress-stress-management-and-smoking>
- [10] Sadiku, Matthew N. O. *Elements of Electromagnetics*. New York: Oxford UP, 2015. Print.
- [11] <http://www.mayoclinic.org/first-aid/first-aid-fever/basics/art-20056685>
- [12] http://www.heart.org/HEARTORG/HealthyLiving/PhysicalActivity/FitnessBasics/Target-Heart-Rates_UCM_434341_Article.jsp#.WTW7eGjvtPY

Appendix

Code

Arduino Code:

arduinoMain:

```
Sketch: main.ino
* ----- Notes -----
* This code:
* 1) Blinks an LED to User's Live Heartbeat PIN 13
* 2) Fades an LED to User's Live HeartBeat
* 3) Determines BPM
* 4) Prints All of the Above to Serial
*
* Read Me:
*
https://github.com/WorldFamousElectronics/PulseSensor\_Amped\_Arduino/blob/master/README.md
*
* created by Joel Murphy and Yury Gitman
* modified 04/21/2016
* by Chan Hee Lee
* -----
*/

#define ENABLE_DEBUGTEMP 0
#define ENABLE_DEBUG_PULSE 0 // debugging messages enabled
// enable the following separatley
#define ENABLE_DEBUG_PULSEINFO 0
#define ENABLE_DEBUG_IBI 0
#define ENABLE_DEBUG_BPM 0 // see how bpm is calculated

// Variables (pins)
int pulsePin = 14; // Pulse Sensor purple wire connected to
analog pin 0
int blinkPin = 13; // pin to blink led at each beat
int fadePin = 5; // pin to do fancy classy fading blink at
each beat
int fadeRate = 0; // used to fade LED on with PWM on fadePin
int tempPin = 15; // Temperature Sensor wire connected to
analog pin 1

// Volatile Variables, used in the interrupt service routine!
volatile int BPM; // int that holds raw Analog in 0. updated
every 2mS
volatile int Signal; // holds the incoming raw data
volatile int IBI = 600; // int that holds the time interval between
beats! Must be seeded!
volatile boolean Pulse = false; // "True" when User's live heartbeat is
```

```

detected. "False" when not a "live beat".
volatile boolean QS = false;      // becomes true when Arduino finds a beat.

// Regards Serial OutPut -- Set This Up to your needs
static boolean serialVisual = false; // Set to 'false' by Default. Re-setz
to 'true' to see Arduino Serial Monitor ASCII Visual Pulse

void setup(){
  pinMode(blinkPin, OUTPUT);      // pin that will blink to your
  heartbeat!
  pinMode(fadePin, OUTPUT);       // pin that will fade to your heartbeat!
  Serial.begin(115200);           // we agree to talk fast!
  interruptSetup();              // sets up to read Pulse Sensor signal
  every 2mS

  // IF YOU ARE POWERING The Pulse Sensor AT VOLTAGE LESS THAN THE BOARD
  VOLTAGE,
  // UN-COMMENT THE NEXT LINE AND APPLY THAT VOLTAGE TO THE A-REF PIN
  // analogReference(EXTERNAL);
}

// Where the Magic Happens
void loop(){
  serialOutput();                // 1) Uncomment so the "Processing" can access the
  data
                                  // 2) Disable 'serialVisual' also.
                                  //
  if (QS == true){              // A Heartbeat Was Found
                                  // BPM and IBI have been Determined
                                  // Quantified Self "QS" true when Arduino finds a
  heartbeat
    fadeRate = 255;              // Makes the LED Fade Effect Happen
                                  // Set 'fadeRate' Variable to 255 to fade LED with
  pulse
    serialOutputTemperature();
    serialOutputWhenBeatHappens(); // A Beat Happened, Output that to
  serial.
    QS = false;                  // reset the Quantified Self flag for next
  time
  }

  ledFadeToBeat();              // Makes the LED Fade Effect Happen
  delay(20);                    // take a break
}

void ledFadeToBeat() {
  fadeRate -= 15;                // set LED fade value
  fadeRate = constrain(fadeRate, 0, 255); // keep LED fade value from
going into negative numbers!
  analogWrite(fadePin, fadeRate); // fade LED
}

```

AllSerialHandling.ino

```
/*
 * Sketch: AllSerialHandling.ino
 *
 * This is a serial handling Code for pulse sensor.
 * It's Changeable with the 'serialVisual' variable
 * Set it to 'true' or 'false' when it's declared at start of code.
 *
 */

// Decide How To Output Serial.
void serialOutput(){
  if (serialVisual == true) {
    arduinoSerialMonitorVisual('-', Signal); // goes to function that makes
Serial Monitor Visualizer
  } else {
    sendDataToSerial('S', Signal); // goes to sendDataToSerial function
  }
}

// Decides How To OutPut BPM and IBI Data
void serialOutputWhenBeatHappens() {
  if (serialVisual == true){ // Code to Make the Serial Monitor
Visualizer Work
    Serial.print("*** Heart-Beat Happened *** "); //ASCII Art Madness
    Serial.print("BPM: ");
    Serial.print(BPM);
    Serial.println(" ");
  } else {
    sendDataToSerial('B',BPM); // send heart rate with a 'B' prefix
    sendDataToSerial('Q',IBI); // send time between beats with a 'Q'
prefix
  }
}

// boolean to integer
int bool2int(boolean stat) {
  if (stat) {
    return 1;
  } else {
    return 0;
  }
}

// Sends Data to Pulse Sensor Processing App, Native Mac App, or Third-party
Serial Readers.
void sendDataToSerial(char symbol, int data ) {
  Serial.print(symbol);
  Serial.println(data);
}

void sendDataToSerialTemp(char symbol, float data) {
  Serial.print(symbol);
```



```

    Serial.println(data);
}

// Code to Make the Serial Monitor Visualizer Work
void arduinoSerialMonitorVisual(char symbol, int data ){
    const int sensorMin = 0;        // sensor minimum, discovered through
    experiment
    const int sensorMax = 1024;     // sensor maximum, discovered through
    experiment

    int sensorReading = data;
    // map the sensor range to a range of 12 options:
    int range = map(sensorReading, sensorMin, sensorMax, 0, 11);

    // do something different depending on the
    // range value:
    switch (range) {
    case 0:
        Serial.println("");;        //ASCII Art Madness
        break;
    case 1:
        Serial.println("----");
        break;
    case 2:
        Serial.println("-----");
        break;
    case 3:
        Serial.println("-----");
        break;
    case 4:
        Serial.println("-----");
        break;
    case 5:
        Serial.println("-----|-");
        break;
    case 6:
        Serial.println("-----|---");
        break;
    case 7:
        Serial.println("-----|-----");
        break;
    case 8:
        Serial.println("-----|-----");
        break;
    case 9:
        Serial.println("-----|-----");
        break;
    case 10:
        Serial.println("-----|-----");
        break;
    case 11:
        Serial.println("-----|-----");
        break;
    }
}
}

```

Interrupt.ino

```
/*
 * Sketch: Interrupt.ino
 *
 */

volatile int rate[10]; // array to hold last ten IBI
values
volatile unsigned long sampleCounter = 0; // used to determine pulse timing
volatile unsigned long lastBeatTime = 0; // used to find IBI
volatile int P = 512; // used to find peak in pulse
wave, seeded
volatile int T = 512; // used to find trough in pulse
wave, seeded
volatile int thresh = 525; // used to find instant moment of
heart beat, seeded
volatile int amp = 100; // used to hold amplitude of pulse
waveform, seeded
volatile boolean firstBeat = true; // used to seed rate array so we
startup with reasonable BPM
volatile boolean secondBeat = false; // used to seed rate array so we
startup with reasonable BPM

void interruptSetup(){
  // Initializes Timer2 to throw an interrupt every 2mS.
  TCCR2A = 0x02; // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC
MODE
  TCCR2B = 0x06; // DON'T FORCE COMPARE, 256 PRESCALER
  OCR2A = 0x7C; // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE RATE
  TIMSK2 = 0x02; // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND OCR2A
  sei(); // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}

// THIS IS THE TIMER 2 INTERRUPT SERVICE ROUTINE.
// Timer 2 makes sure that we take a reading every 2 milliseconds
ISR(TIMER2_COMPA_vect){ // triggered when Timer2 counts
to 124
  cli(); // disable interrupts while we
do this
  Signal = analogRead(pulsePin);
// read the Pulse Sensor

  sampleCounter += 2; // keep track of the time in mS
with this variable
  int N = sampleCounter - lastBeatTime; // monitor the time since the
last beat to avoid noise

#ifdef ENABLE_DEBUG_PULSE
#ifdef ENABLE_DEBUG_PULSEINFO
  Serial.print("Counter reading (N): ");
  Serial.println(N);
  Serial.print("Pulse signal reading: ");
  Serial.println(Signal);
#endif
#endif
}
```

```

#endif

    // find the peak and trough of the pulse wave
    if (Signal < thresh && N > (IBI/5)*3) { // avoid dichrotic noise by
waiting 3/5 of last IBI
        if (Signal < T){ // T is the trough
            T = Signal; // keep track of lowest point
in pulse wave
        }
    }

    if (Signal > thresh && Signal > P) { // thresh condition helps
avoid noise
        P = Signal; // P is the peak
    } // keep track of highest point in
pulse wave

    // NOW IT'S TIME TO LOOK FOR THE HEART BEAT
    // signal surges up in value every time there is a pulse
    if (N > 250) { // avoid high frequency
noise
        if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) ){
            Pulse = true; // set the Pulse flag when
we think there is a pulse
            sendDataToSerial('X',bool2int(Pulse)); // send the status for second
beat with a 'Z' prefix
            digitalWrite(blinkPin, HIGH); // turn on pin 13 LED
            IBI = sampleCounter - lastBeatTime; // measure time between
beats in mS
            lastBeatTime = sampleCounter; // keep track of time for
next pulse

            #if ENABLE_DEBUG_PULSE
            #if ENABLE_DEBUG_IBI
            Serial.print("IBI: ");
            Serial.println(IBI);
            Serial.print("last beat time: ");
            Serial.println(lastBeatTime);
            #endif
            #endif
        }

        if( secondBeat ) { // if this is the second
beat, if secondBeat == TRUE
            // sendDataToSerial('Z',bool2int(secondBeat)); // send the status
for second beat with a 'Z' prefix
            secondBeat = false; // clear secondBeat flag
            for(int i=0; i<=9; i++){ // seed the running total to get
a realisitc BPM at startup
                rate[i] = IBI;
            }
        }

        if( firstBeat ) { // if it's the first time we
found a beat, if firstBeat == TRUE
            firstBeat = false; // clear firstBeat flag
            secondBeat = true; // set the second beat flag
            sei(); // enable interrupts again

```

```

        return; // IBI value is unreliable so
discard it
    }

    // keep a running total of the last 10 IBI values
    word runningTotal = 0; // clear the runningTotal
variable
    for(int i=0; i<=8; i++){ // shift data in the rate array
        rate[i] = rate[i+1]; // and drop the oldest IBI
value
        runningTotal += rate[i]; // add up the 9 oldest IBI
values
    }

    rate[9] = IBI; // add the latest IBI to the
rate array
    runningTotal += rate[9]; // add the latest IBI to
runningTotal

    #if ENABLE_DEBUG_PULSE
    #if ENABLE_DEBUG_BPM
        Serial.print("runningTotal: ");
        Serial.println(runningTotal);
    #endif
    #endif

    runningTotal /= 10; // average the last 10 IBI
values
    BPM = 60000/runningTotal; // how many beats can fit into
a minute? that's BPM!
    QS = true; // set Quantified Self flag
    // QS FLAG IS NOT CLEARED INSIDE THIS ISR
    }
}

    if (Signal < thresh && Pulse == true){ // when the values are going down,
the beat is over
        digitalWrite(blinkPin, LOW); // turn off pin 13 LED
        Pulse = false; // reset the Pulse flag so we can
do it again
        amp = P - T; // get amplitude of the pulse wave
        thresh = amp/2 + T; // set thresh at 50% of the
amplitude
        P = thresh; // reset these for next time
        T = thresh;
    }

    if (N > 2500) { // if 2.5 seconds go by without a
beat
    #if ENABLE_DEBUG_PULSE
        Serial.println("Beat is not detected");
        Serial.println("Resetting by default");
    #endif
        thresh = 512; // set thresh default

```

```

    P = 512;           // set P default
    T = 512;           // set T default
    lastBeatTime = sampleCounter; // bring the lastBeatTime up to
date
    firstBeat = true; // set these to avoid noise
    secondBeat = false; // when we get the heartbeat back
}

sei(); // enable interrupts when youre
done!
} // end isr

```

Timer_Interrupt_Notes.ino

/*

These notes put together by Joel Murphy for Pulse Sensor Amped, 2015

The code that this section is attached to uses a timer interrupt to sample the Pulse Sensor with consistent and regular timing. The code is setup to read Pulse Sensor signal at 500Hz (every 2mS). The reasoning for this can be found here:
<http://pulsesensor.com/pages/pulse-sensor-amped-arduino-v1dot1>

There are issues with using different timers to control the Pulse Sensor sample rate.

Sometimes, user will need to switch timers for access to other code libraries.

Also, some other hardware may have different timer setup requirements. This page

will cover those different needs and reveal the necessary settings. There are two

part of the code that will be discussed. The interruptSetup() routine, and the interrupt function call. Depending on your needs, or the Arduino variant that you use,

check below for the correct settings.

ARDUINO UNO, Pro 328-5V/16MHZ, Pro-Mini 328-5V/16MHZ (or any board with ATmega328P running at 16MHZ)

>> Timer2

Pulse Sensor Arduino UNO uses Timer2 by default.

Use of Timer2 interferes with PWM on pins 3 and 11.

There is also a conflict with the Tone library, so if you want tones, use Timer1 below.

```

void interruptSetup(){
    // Initializes Timer2 to throw an interrupt every 2mS.
    TCCR2A = 0x02; // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO
INTO CTC MODE
    TCCR2B = 0x06; // DON'T FORCE COMPARE, 256 PRESCALER
    OCR2A = 0X7C; // SET THE TOP OF THE COUNT TO 124 FOR 500Hz

```

```

SAMPLE RATE
    TIMSK2 = 0x02;      // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND
OCR2A
    sei();              // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
    }

```

use the following interrupt vector with Timer2

```
ISR(TIMER2_COMPA_vect)
```

>> Timer1

Use of Timer1 interferes with PWM on pins 9 and 10.

The Servo library also uses Timer1, so if you want servos, use Timer2 above.

```

void interruptSetup(){
    // Initializes Timer1 to throw an interrupt every 2mS.
    TCCR1A = 0x00; // DISABLE OUTPUTS AND PWM ON DIGITAL PINS 9 & 10
    TCCR1B = 0x11; // GO INTO 'PHASE AND FREQUENCY CORRECT' MODE, NO
PRESCALER
    TCCR1C = 0x00; // DON'T FORCE COMPARE
    TIMSK1 = 0x01; // ENABLE OVERFLOW INTERRUPT (TOIE1)
    ICR1 = 16000; // TRIGGER TIMER INTERRUPT EVERY 2mS
    sei();        // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}

```

Use the following ISR vector for the Timer1 setup above

```
ISR(TIMER1_OVF_vect)
```

>> Timer0

DON'T USE TIMER0! Timer0 is used for counting delay(), millis(), and micros().

Messing with Timer0 is highly unadvised!

```

*****
*****

```

ARDUINO Fio, Lilypad, ProMini328-3V/8MHz (or any board with ATmega328P running at 8MHz)

>> Timer2

Pulse Sensor Arduino UNO uses Timer2 by default.

Use of Timer2 interferes with PWM on pins 3 and 11.

There is also a conflict with the Tone library, so if you want tones, use Timer1 below.

```

void interruptSetup(){
    // Initializes Timer2 to throw an interrupt every 2mS.
    TCCR2A = 0x02;      // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO
INTO CTC MODE
    TCCR2B = 0x05;      // DON'T FORCE COMPARE, 128 PRESCALER
    OCR2A = 0X7C;      // SET THE TOP OF THE COUNT TO 124 FOR 500Hz
SAMPLE RATE

```

```

OCR2A    TIMSK2 = 0x02;      // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND
        sei();             // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
    }

```

use the following interrupt vector with Timer2

```
ISR(TIMER2_COMPA_vect)
```

>> Timer1

Use of Timer1 interferes with PWM on pins 9 and 10.

The Servo library also uses Timer1, so if you want servos, use Timer2 above.

```

void interruptSetup(){
    // Initializes Timer1 to throw an interrupt every 2mS.
    TCCR1A = 0x00; // DISABLE OUTPUTS AND PWM ON DIGITAL PINS 9 & 10
    TCCR1B = 0x11; // GO INTO 'PHASE AND FREQUENCY CORRECT' MODE, NO
PRESCALER
    TCCR1C = 0x00; // DON'T FORCE COMPARE
    TIMSK1 = 0x01; // ENABLE OVERFLOW INTERRUPT (TOIE1)
    ICR1 = 8000;  // TRIGGER TIMER INTERRUPT EVERY 2mS
    sei();       // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}

```

Use the following ISR vector for the Timer1 setup above

```
ISR(TIMER1_OVF_vect)
```

>> Timer0

DON'T USE TIMER0! Timer0 is used for counting delay(), millis(), and micros().

Messing with Timer0 is highly unadvised!

```

*****
*****

```

ARDUINO Leonardo (or any board with ATmega32u4 running at 16MHz)

>> Timer1

Use of Timer1 interferes with PWM on pins 9 and 10.

```

void interruptSetup(){
    TCCR1A = 0x00;
    TCCR1B = 0x0C; // prescaler = 256
    OCR1A = 0x7C; // count to 124
    TIMSK1 = 0x02;
    sei();
}

```

The only other thing you will need is the correct ISR vector in the next step.

```
ISR(TIMER1_COMPA_vect)
```

```
*****  
*****
```

```
ADAFRUIT Flora, ARDUINO Fio v3 (or any other board with ATmega32u4 running  
at 8MHz)
```

```
>> Timer1
```

Use of Timer1 interferes with PWM on pins 9 and 10.

```
void interruptSetup(){  
  TCCR1A = 0x00;  
  TCCR1B = 0x0C; // prescaler = 256  
  OCR1A = 0x3E; // count to 62  
  TIMSK1 = 0x02;  
  sei();  
}
```

The only other thing you will need is the correct ISR vector in the next step.

```
ISR(TIMER1_COMPA_vect)
```

```
*****  
*****
```

```
ADAFRUIT Gemma (or any other board with ATtiny85 running at 8MHz)
```

NOTE: Gemma does not do serial communication!
Comment out or remove the Serial code in the Arduino sketch!

```
Timer1
```

Use of Timer1 breaks PWM output on pin D1

```
void interruptSetup(){  
  TCCR1 = 0x88; // Clear Timer on Compare, Set Prescaler to 128  
TEST VALUE  
  GTCCR &= 0x81; // Disable PWM, don't connect pins to events  
  OCR1C = 0x7C; // Set the top of the count to 124 TEST VALUE  
  OCR1A = 0x7C; // Set the timer to interrupt after counting to  
TEST VALUE  
  bitSet(TIMSK,6); // Enable interrupt on match between TCNT1 and  
OCR1A  
  sei(); // Enable global interrupts  
}
```

The only other thing you will need is the correct ISR vector in the next step.

```
ISR(TIMER1_COMPA_vect)
```

Temperature


```

/*
 * runs once when you turn your Arduino on. We initialize the serial
connection with the computer
 */

void serialOutputTemperature() {
  //getting the voltage reading from the temperature sensor
  // int reading = analogRead(tempPin);

  int average = 0 ;
  for (int i=0; i < 100; i++) {
    average = average + analogRead(tempPin);
  }
  average = average/100;

  // converting that reading to voltage, for 3.3v Arduino use 3.3
float voltage = average * 5000.0 / 1024.0;
//float temp_c = (-0.193) * voltage + 212.0;
float temp_c = (1098.99- voltage) / (5.2);
//(slope * voltage) + 1097.36;

  // raw data
#ifdef ENABLE_DEBUGTEMP
  //Serial.println(ENABLE_DEBUGTEMP);
  //Serial.print("raw signal reading: ");
  //Serial.println(reading);
  //Serial.print("evaluated voltage: ");
  //Serial.println(voltage);
#endif

  if (serialVisual == true) {
    printTemp(temp_c);
  } else {
    sendDataToSerialTemp('T', temp_c);
    sendDataToSerialTemp('F', (temp_c * 9 / 5) + 32 );
  }
  //printTemperature(voltage);
  //delay(250);
}

void printTemp(float temp_c) {
  //to Celsius
  Serial.println("==== temperature =====");
  Serial.print(temp_c);
  Serial.println(" degrees C");

  //to Fahrenheit
  float temp_f = (temp_c * 9 / 5) + 32;
  Serial.print(temp_f);
  Serial.println(" degrees F");
}

```

```
}
```

Processing Code

processingMain

```
/*
 * Sketch: processingMain.pde
 *
 * THIS PROGRAM WORKS WITH PulseSensorAmped_Arduino-xx ARDUINO CODE
 * THE PULSE DATA WINDOW IS SCALEABLE WITH SCROLLBAR AT BOTTOM OF
SCREEN
 * PRESS 'S' OR 's' KEY TO SAVE A PICTURE OF THE SCREEN IN SKETCH
FOLDER (.jpg)
 *
 * CREATED BY Joel Murphy
 * MODIFIED BY Chan Hee Lee
 * @ APRIL, 2016
 * see original code here
 *
https://github.com/WorldFamousElectronics/PulseSensor\_Amped\_Processing\_Visualizer/
 */

import ddf.minim.*;
import processing.serial.*;
import processing.sound.*;
PFont font;
Scrollbar scaleBar;
Serial port;
SoundFile file;
AudioSnippet alert;
Minim minim;

int Sensor; // HOLDS PULSE SENSOR DATA FROM ARDUINO
int IBI; // HOLDS TIME BETWEEN HEARTBEATS FROM ARDUINO
int BPM; // HOLDS HEART RATE VALUE FROM ARDUINO
float temp_c; // HOLDS TEMPERATURE VALUE IN CELSIUS FROM ARDUINO
float temp_f;
int[] RawY; // HOLDS HEARTBEAT WAVEFORM DATA BEFORE SCALING
int[] ScaledY; // USED TO POSITION SCALED HEARTBEAT WAVEFORM
int[] rate; // USED TO POSITION BPM DATA WAVEFORM
float zoom; // USED WHEN SCALING PULSE WAVEFORM TO PULSE WINDOW
float offset; // USED WHEN SCALING PULSE WAVEFORM TO PULSE WINDOW
color eggshell = color(255, 253, 248);
int heart = 0; // This variable times the heart image 'pulse' on
screen
```

```

// THESE VARIABLES DETERMINE THE SIZE OF THE DATA WINDOWS
int PulseWindowWidth = 490;
int PulseWindowHeight = 512;
int BPMWindowWidth = 180;
int BPMWindowHeight = 340;
boolean beat = false; // set when a heart beat is detected, then
cleared when the BPM graph is advanced
boolean alertNow = false;
boolean firstBeat = false;
boolean secondBeat = false;

void setup() {
  size(700, 600); // Stage size
  frameRate(100);
  font = loadFont("Arial-BoldMT-24.vlw");
  textFont(font);
  textAlign(CENTER);
  rectMode(CENTER);
  ellipseMode(CENTER);

  // Scrollbar constructor inputs: x,y,width,height,minVal,maxVal
  scaleBar = new Scrollbar(400, 575, 180, 12, 0.5, 1.0); // set
parameters for the scale bar
  RawY = new int[PulseWindowWidth]; // initialize raw pulse
waveform array
  ScaledY = new int[PulseWindowWidth]; // initialize scaled pulse
waveform array
  rate = new int[BPMWindowWidth]; // initialize BPM waveform
array
  zoom = 0.75; // initialize scale of
heartbeat window

  // set the visualizer lines to 0
  for (int i=0; i<rate.length; i++) {
    rate[i] = 555; // Place BPM graph line at bottom of BPM Window
  }

  for (int i=0; i<RawY.length; i++){
    RawY[i] = height/2; // initialize the pulse window data line to V/2
  }

  // GO FIND THE ARDUINO
  println(Serial.list()); // print a list of available serial ports

  // choose the number between the [] that is connected to the Arduino
  port = new Serial(this, Serial.list()[2], 115200); // make sure
Arduino is talking serial at this baud rate
  port.clear(); // flush buffer
  port.bufferUntil('\n'); // set buffer full flag on receipt of
carriage return

  minim = new Minim(this);
  alert = minim.loadSnippet("alert.wav");
}

void draw() {
  background(0);

```

```

noStroke();

// DRAW OUT THE PULSE WINDOW AND BPM WINDOW RECTANGLES
fill(eggshell); // color for the window background
rect(255, height/2, PulseWindowWidth, PulseWindowHeight);
rect(600, 385, BPMWindowWidth, BPMWindowHeight);

println(port.readStringUntil('\n'));

// DRAW THE PULSE WAVEFORM
drawPulse();
/*
 * INPUT:
 * - Sensor, RawY, ScaledY, zoom
 * OUTPUT:
 * - Draw waveform according to the value of 'ScaledY'
 */

// GRAPH BPM WAVEFORM
drawBPM();
/*
 * INPUT:
 * - beat, rate, BPM,
 * OUTPUT:
 * - Draw waveform for BPM
 */

// DRAW HEART BEATING
// drawHeart();

// PRINT THE DATA AND VARIABLE VALUES
fill(eggshell); // get ready to
print text // tell them
  text("Pulse + Temp Visualizer 1.2", 245, 30); // tell them
what you are
  text("IBI " + IBI + "mS", 600, 585); // print the
time between heartbeats in mS
  text(BPM + " BPM", 600, 200); // print the
Beats Per Minute
  text("Pulse Window Scale " + nf(zoom,1,2), 150, 585); // show the
current scale of Pulse Window
  text(temp_c + " °C", 600, 120);
  text(temp_f + " °F", 600, 100);

// DO THE SCROLLBAR THINGS
scaleBar.update (mouseX, mouseY);
scaleBar.display();

// DRAW TEMPERATURE VALUE
// drawTemp();

  //playAlert();
} //end of draw loop

```

Alert

```
/*
 * Sketch: serialEvent.pde
 *
 * This sketch manages the serial event.
 */

void playAlert() {
  // Pulse Alert condition
  if (IBI > 1300 || IBI < 260) {
    alertNow = true;
  }

  if (BPM > 140 || BPM < 30 ) {
    alertNow = true;
  }

  // Temperature Alert Condition
  if (temp_c > 40 || temp_c < 29 ) {
    alertNow = true;
  }

  playNow(alertNow);
  alertNow = false;
}

void playNow(boolean trigger) {
  if (trigger) {
    alert.rewind();
    alert.play(200);
  }
}
```

drawPulse

```
/*
 * Sketch: drawPulse.pde
 *
 * CREATED BY Joel Murphy
 * MODIFIED @ APRIL, 2016
 */

// DRAW THE PULSE WAVEFORM
void drawPulse() { //<> //<> //
  // prepare pulse data points
  RawY[RawY.length-1] = (1023 - Sensor) - 212; // place the new raw
datapoint at the end of the array
  zoom = scaleBar.getPos(); // get current
waveform scale value
  offset = map(zoom,0.5,1,150,0); // calculate the
```

```

offset needed at this scale

    for (int i = 0; i < RawY.length-1; i++) {           // move the pulse
waveform by
        RawY[i] = RawY[i+1];                           // shifting all raw
datapoints one pixel left
        float dummy = RawY[i] * zoom + offset;         // adjust the raw data
to the selected scale
        ScaledY[i] = constrain(int(dummy), 44, 556); // transfer the raw
data array to the scaled array
        // println(dummy + "\n");
        // println(ScaledY[i] + "\n");
    }

    // draw it according to the 'ScaledY' value.
stroke(250,0,0); // red
noFill();
beginShape(); // using beginShape() renders fast

    for (int x=1; x < ScaledY.length-1; x++) {
        vertex(x+10, ScaledY[x]);                       //draw a line
connecting the data points
    }
    endShape();
}

// GRAPH BPM WAVEFORM
void drawBPM() {
    if (beat == true) // move the heart rate line over one pixel every
time the heart beats
    {
        beat = false; // clear beat flag (beat flag was set in
serialEvent tab)
        for (int i=0; i<rate.length-1; i++){
            rate[i] = rate[i+1];                         // shift the bpm Y
coordinates over one pixel to the left
        }

        // then limit and scale the BPM value
        BPM = min(BPM, 200);                             // limit the highest
BPM value to 200
        float dummy = map(BPM, 0, 200, 555, 215);       // map it to the heart
rate window Y
        rate[rate.length-1] = int(dummy);               // set the rightmost
pixel to the new data point value
    }

    // GRAPH THE HEART RATE WAVEFORM
stroke(250,0,0); // color of heart rate
graph
strokeWeight(2); // thicker line is easier
to read
noFill();
beginShape();
    for (int i=0; i < rate.length-1; i++){ // variable 'i' will take
the place of pixel x position

```

```

        vertex(i+510, rate[i]);                // display history of heart
rate datapoints
    }
    endShape();
}

// DRAW HEART BEATING
void drawHeart() {
    fill(250,0,0);
    stroke(250,0,0);
    // the 'heart' variable is set in serialEvent when Arduino sees a
beat happen
    heart--;                                // heart is used to time how long the
heart graphic swells when your heart beats
    heart=max(heart,0);                    // don't let the heart variable go into
negative numbers

    if (heart > 0)                        // if a beat happened recently,
    {
        strokeWeight(8);                  // make the heart big
    }

    smooth();                             // draw the heart with two bezier curves
    bezier(width-100,50, width-20,-20, width,140, width-100,150);
    bezier(width-100,50, width-190,-20, width-200,140, width-100,150);
    strokeWeight(1);                       // reset the strokeWeight for next time
}

/*void drawTemp() {
    fill(250,0,0);
    stroke(250,0,0);
    /
}*/

```

keyboard_mouse

```

/*
 * Sketch: drawPulse.pde
 *
 *
 *   CREATED BY Joel Murphy
 *   MODIFIED @ APRIL, 2016
 */

// DRAW THE PULSE WAVEFORM
void drawPulse() { //<> //<> //
    // prepare pulse data points
    RawY[RawY.length-1] = (1023 - Sensor) - 212; // place the new raw
datapoint at the end of the array
    zoom = scaleBar.getPos(); // get current
waveform scale value
    offset = map(zoom,0.5,1,150,0); // calculate the

```

```

offset needed at this scale

    for (int i = 0; i < RawY.length-1; i++) {           // move the pulse
waveform by
        RawY[i] = RawY[i+1];                           // shifting all raw
datapoints one pixel left
        float dummy = RawY[i] * zoom + offset;         // adjust the raw data
to the selected scale
        ScaledY[i] = constrain(int(dummy), 44, 556); // transfer the raw
data array to the scaled array
        // println(dummy + "\n");
        // println(ScaledY[i] + "\n");
    }

    // draw it according to the 'ScaledY' value.
stroke(250,0,0); // red
noFill();
beginShape(); // using beginShape() renders fast

    for (int x=1; x < ScaledY.length-1; x++) {
        vertex(x+10, ScaledY[x]); //draw a line
connecting the data points
    }
    endShape();
}

// GRAPH BPM WAVEFORM
void drawBPM() {
    if (beat == true) // move the heart rate line over one pixel every
time the heart beats
    {
        beat = false; // clear beat flag (beat flag was set in
serialEvent tab)
        for (int i=0; i<rate.length-1; i++){
            rate[i] = rate[i+1]; // shift the bpm Y
coordinates over one pixel to the left
        }

        // then limit and scale the BPM value
        BPM = min(BPM, 200); // limit the highest
BPM value to 200
        float dummy = map(BPM, 0, 200, 555, 215); // map it to the heart
rate window Y
        rate[rate.length-1] = int(dummy); // set the rightmost
pixel to the new data point value
    }

    // GRAPH THE HEART RATE WAVEFORM
stroke(250,0,0); // color of heart rate
graph
strokeWeight(2); // thicker line is easier
to read
noFill();
beginShape();
    for (int i=0; i < rate.length-1; i++){ // variable 'i' will take
the place of pixel x position

```



```

        vertex(i+510, rate[i]);           // display history of heart
rate datapoints
    }
    endShape();
}

// DRAW HEART BEATING
void drawHeart() {
    fill(250,0,0);
    stroke(250,0,0);
    // the 'heart' variable is set in serialEvent when Arduino sees a
beat happen
    heart--;                               // heart is used to time how long the
heart graphic swells when your heart beats
    heart=max(heart,0);                     // don't let the heart variable go into
negative numbers

    if (heart > 0)                         // if a beat happened recently,
    {
        strokeWeight(8);                   // make the heart big
    }

    smooth();                               // draw the heart with two bezier curves
    bezier(width-100,50, width-20,-20, width,140, width-100,150);
    bezier(width-100,50, width-190,-20, width-200,140, width-100,150);
    strokeWeight(1);                       // reset the strokeWeight for next time
}

/*void drawTemp() {
    fill(250,0,0);
    stroke(250,0,0);
    /
}*/

```

ScaleBar

```

/*
 * Sketch: scaleBar.pde
 *
 * THIS SCROLLBAR OBJECT IS BASED ON THE ONE FROM THE BOOK
"Processing" by Reas and Fry
 */

class Scrollbar {
    int x,y;                               // the x and y coordinates
    float sw, sh;                          // width and height of scrollbar
    float pos;                              // position of thumb
    float posMin, posMax;                   // max and min values of thumb
    boolean rollover;                       // true when the mouse is over
    boolean locked;                         // true when it's the active scrollbar
    float minVal, maxVal;                  // min and max values for the thumb

    Scrollbar (int xp, int yp, int w, int h, float miv, float mav) { //
values passed from the constructor
        x = xp;

```

```

    y = yp;
    sw = w;
    sh = h;
    minVal = miv;
    maxVal = mav;
    pos = x - sh/2;
    posMin = x - sw/2;
    posMax = x + sw/2; // - sh;
}

// updates the 'over' boolean and position of thumb
void update(int mx, int my) {
    if (over(mx, my) == true) {
        rollover = true; // when the mouse is over the
scrollbar, rollover is true
    } else {
        rollover = false;
    }
    if (locked == true) {
        pos = constrain (mx, posMin, posMax);
    }
}

// locks the thumb so the mouse can move off and still update
void press(int mx, int my){
    if (rollover == true){
        locked = true; // when rollover is true, pressing the
mouse button will lock the scrollbar on
    } else {
        locked = false;
    }
}

// resets the scrollbar to neutral
void release(){
    locked = false;
}

// returns true if the cursor is over the scrollbar
boolean over(int mx, int my) {
    if ((mx > x-sw/2) && (mx < x+sw/2) && (my > y-sh/2) && (my <
y+sh/2)){
        return true;
    } else {
        return false;
    }
}

// draws the scrollbar on the screen
void display () {
    noStroke();
    fill(255);
    rect(x, y, sw, sh); // create the scrollbar
    fill (250,0,0);
    if ((rollover == true) || (locked == true)){
        stroke(250,0,0);
        strokeWeight(8); // make the scale dot bigger if you're

```

on it

```
    }
    ellipse(pos, y, sh, sh);    // create the scaling dot
    strokeWeight(1);           // reset strokeWeight
  }

  // returns the current value of the thumb
  float getPos() {
    float scalar = sw / sh; // (sw - sh/2);
    float ratio = (pos-(x-sw/2)) * scalar;
    float p = minVal + (ratio/sw * (maxVal - minVal));
    return p;
  }
}
```

serialEvent

```
/*
 * Sketch: serialEvent.pde
 *
 * This sketch manages the serial event.
 */

void serialEvent(Serial port) {
  String inData = port.readStringUntil('\n');

  if (inData == null) { // bail if we didn't get
anything
    return;
  }
  if (inData.isEmpty()) { // bail if we got an empty
line
    return;
  }

  inData = trim(inData); // cut off white space
  (carriage return)
  if(inData.length() <= 0) { // bail if there's nothing
there
    return;
  }

  if (inData.charAt(0) == 'S'){ // leading 'S' for sensor data
    inData = inData.substring(1); // cut off the leading 'S'
    Sensor = int(inData); // convert the string to
usable int
  }

  if (inData.charAt(0) == 'B'){ // leading 'B' for BPM data
    inData = inData.substring(1); // cut off the leading 'B'
    BPM = int(inData); // convert the string to
usable int
  }

  beat = true; // set beat flag to advance
heart rate graph
  heart = 20; // begin heart image 'swell'
```

```

timer
    }

    if (inData.charAt(0) == 'Q'){           // leading 'Q' means IBI data
        inData = inData.substring(1);     // cut off the leading 'Q'
        IBI = int(inData);                 // convert the string to
usable int
    }



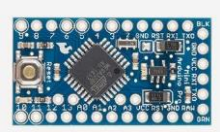

    if (inData.charAt(0) == 'T') {         // leading 'T' means
temperature in Celsius
        inData = inData.substring(1);     // cut off the leading 'T'
        temp_c = float(inData) ;          // convert the string to
usable int
    }

    if (inData.charAt(0) == 'F') {         // leading 'F' means
temperature in Fahrenheit
        inData = inData.substring(1);     // cut off the leading 'F'
        temp_f = float(inData);           // convert the string to
usable int
    }




    if (inData.charAt(0) == 'Y') {
        inData = inData.substring(1);
        firstBeat = boolean(inData);
    }
    if (inData.charAt(0) == 'Z') {
        inData = inData.substring(1);
        secondBeat = boolean(inData);
    }
}

```

Bill of Materials

Item Image	Product Name	Quantity	Price
	LMT70 Temperature Sensor	5	\$0.00
	<i>Bluefruit EZ-Link - Bluetooth Serial Link & Arduino Programmer - v1.3</i>	1	\$27.50
	Arduino Pro Mini 328 - 5V/16MHz	1	\$9.95
	<i>Asiawill® Pulse Sensor Module for Arduino - Red</i>	1	\$13.99

Item Image	Product Name	Quantity	Price
------------	--------------	----------	-------

	Power Management IC Development Tools Eval Board For P2110	1	\$172.0
	Lithium Ion Battery - 2000mAh	1	\$27.50
	PCB Board Fabrication	3	\$110.93