ECE 388: Embedded System Design Project

Final Report Functioning Pedometer

I certify that this work is original and not a product of anyone's work but my own.

Isabella Di Bona

Submitted: December 8^{th}, 2020

Due by: December 8^{th}, 2020

Graded by: _____ Date: _____

Contents

List of Figures

List of Tables

List of Files

Abstract

The goal of this laboratory experiment was to construct a functioning pedometer utilizing an accelerometer to capture data and an OLED display to give feedback to the user. This design implemented hardware for capturing information based on user interface, and firmware to generate feedback regarding the number of steps and distance travelled as well as the battery level being supplied to the device interfacing with an organic light emitting diode. EAGLE AutoDesk was used to source appropriate component symbols in which would match that of the actual design components in order to ensure the board footprint contained accurate sizing, spacing and connection pads after manufacturing for hand soldering and the pick and place machine. Final hardware components were then placed on the final design using a pick and place machine and firmware was loaded onto the ATmega328PB using an AVR ISP.

1 Introduction

The purpose of this project was to create a pedometer in which a 3-axis accelerometer took in data based on the gravitational positioning and velocity of the motion and movement of a “person”. This data was then translated into voltage readings where the ATmega328PB analog to digital converters were used to convert these voltage readings into 10-bit digital values. The data was then analyzed under different test conditions in order to determine whether the data was valid. Valid data consisted only of actions and movement (or velocity) in which could realistically be performed by a human wearing this device on their body when either walking or running. Misuse of this device would be a result of "user-error" rather than a testing condition that should indicate invalid data. Misuse would consist of an instance such as manipulating the device to somehow generate a step by moving the device in a motion in which would typically consist of valid data. Data that was deemed valid within firmware was transmitted to a 128x32 Pixel OLED display using I^2C where the number of steps, distance in miles and battery life were reflected on the screen when powered.

The pedometer power supply consisted of two +3V coin cell batteries which were be placed in series to reflect +6V. There were variations in voltage readings from the accelerometer when the battery supply started to decrease, therefore in firmware there were not hard-coded digital data references. Instead, the accelerometer was calibrated to check for the sensor position and thresholds were determined for realistic step measurement based on these thresholds. A +5V to +3.3V linear voltage regulator was used to restrict the voltage supplied to the accelerometer as the voltage required should never be greater than +3.7V. The ATmega328PB and the OLED display were supplied power right from the supply voltage which is roughly between +5.5V and +6V considering the OLED contained a backpack with a linear regulator already implemented. Once

the accelerometer data for the step count was determined to be valid, the number of steps was be divided by an average number of steps per mile to determine the distance in miles walked. The supply voltage was measured using an ADC and converted into a 10-bit digital number. The largest 10-bit digital value is 1023 therefore to determine how far from full battery the supply was consisted of simple math to compute the difference.

Eagle AutoDesk software was used to generate a schematic and synchronized board layout in which once approved was sent to a board house for manufacturing. This process required great attention to detail specifically in the layout of components. Where the pedometer ended up being such a small design with very minimum components, the board size resultantly was about 27x35 mm after laying all parts out. This was accomplished by making some design sacrifices which could be adjusted in firmware instead of hardware. Sacrifices such as removing the external 16 MHz oscillator to function on the ATmega328PB internal 8 MHz clock were made as well as removing data measurements from the z axis and accounting for different thresholds within the x and y planes. More details regarding the design will be discussed latter in the report.

2 Methods and Procedures

2.1 System Under Test

The system under unit testing consists of a pedometer in which hardware and firmware revisions have been made to demonstrate accuracy and precision in displaying the users step count and distance travelled in miles. The display also contains the battery level represented as a battery icon in the upper right corner of the OLED screen.

A blue LED should light when power is supplied to the circuitry, however to run firmware on the prototype, the Xplained Mini will be powered by USB and supplies the circuit +5V by default. Therefore, when the battery pack is turned on, the LED should light. In the working PCB design where the firmware was loaded onto the ATmega using an AVR ISP, the blue LED should light based solely on power supplied from the battery pack and will represent the implication of the LED lighting when power is supplied to the circuitry.

2.2 Test Setup

Equipment Needed:

- Chocolate Board for final prototype and manufactured PCB for final design.
- +6V battery supply with OLED interfacing for observing resulting data.

- Atmel Studio and a device to run Atmel Studio for final prototype, ISP was then used to load firmware onto PCB therefore only power is required for the final PCB demonstration.

Final SMT Components:

- (1) ADXL327 3-Axis $\pm$ 2g Accelerometer
- (1) 128x32 Pixel OLED Display with I^2C Interface
- (1) 5V to 3.3V TLV1117 Linear Voltage Regulator
- (2) Coin Cell +3V Batteries
- +3V Coin Cell Battery Holder with Switch
- Male and Female pin headers
- (5) 0.15 μF Capacitor
- (3) 4.7 nF Capacitor
- (2) 10 μF Capacitor
- (2) 330 Ω Resistor
- (1) Blue LED
- ATmega328PB Microcontroller
- AVR ISP for programming final PCB design

2.3 Test Procedure Overview

1. Develop a problem statement and flowchart containing the features required for the design and any additional implementations that are desired.
2. Ensure the flowchart is sound and develop an initial plan for hardware components believed necessary for a pedometer based on the flowchart and research.
3. Each component from the schematic shown in figure [4] was utilized in the prototype and placed onto the final design.

4. Develop an initial plan for firmware and recall that if firmware is necessary (it likely will be), that an ISP programming port is implemented as a part in EAGLE and connected to the appropriate ATmega328PB pins on the EAGLE schematic and board layout. Ensure MISO and MOSI from the ISP are connected to MISO0 and MOSI0 on the ATmega.

5. If an analog read out accelerometer is used or any analog devices/components, the ATmega ADC's will be necessary in which the AREF and AVCC pins require power as well as VCC and ground on the ATmega chip.

6. All power supplies require a 10 μF decoupling capacitor. For this design the accelerometer required 4.7 nF decoupling capacitors at the axis outputs and a 0.1 μF for supply voltage decoupling.

7. Place an LED where power should be supplied to an important component, displaying that once power is supplied the LED should light will be used as a form of debugging. Ensure the LED is in series with a resistor of at least 330 Ω's and is connected to the anode, where the cathode should go to ground.

8. Place jumpers to connect the battery pack for power and ground.

9. Place four jumper pins connected in the order the OLED pins headers are labeled according to the desired placement of the OLED.

10. Place additional jumpers to measure from the accelerometer axis outputs for debugging purposes. There are no required ground connections for the ADXL327 accelerometer chip, however pin 1 was an optional ground pin and was used for this purpose. Grounding the IC ensures a reference point for the output axis reads and allows for more accurate results.

11. Ensure power is supplied to each component where power is required and incorporate a voltage regulator where necessary, i.e. for the accelerometer and an OLED without a backpack (the OLED in this design has a backpack in which utilizes a voltage regulator).

12. For this design, power was supplied via a battery pack with a switch consisting of two +3V coin cell batteries in series supplying roughly +6V.

13. Solder components either by hand or by pick and place once manufactured board is returned.

14. Use an AVR ISP to load program from Atmel onto board with 8 MHz core processor speed and minimal startup delay.

15. Power circuit and ensure results are as expected and correct errors in which need alterations.

16. Document results and possible corrections in either hardware or firmware or both that would allow for a more accurate and precise pedometer design.

The test procedure is now complete.

2.4 Test Matrix

The testing matrix consists of the engineering requirements and engineering design specifications in which were tested to ensure the pedometer portrayed accurate data to the user. The test's performed were based on analyzing the behavior of final pedometer prototype and altering the firmware to allow for greater accuracy in the counting and displaying of steps. Hardware components were no longer necessary for voltage measurements from the accelerometer as firmware was instead utilized to watch for accuracy in variable changes.

This matrix was tested and checked for each of the conditions in figure [1], and the expected versus actual results were recorded. If the actual results were in unison with what was expected of the test, a "P" for $pass$ was granted for this performance check. On the other hand, if the data from the experiment and the expected data were not comparable or had grave difference, this will be something that needs to be adjusted in either hardware, firmware, or both. Such unexpected tasks were momentarily determined to have $failed$ the performance check and were deemed an "F" until the design passed this performance check in the future.

Test Number	Test Performed	Expected Result	Actual Result	Pass/ Fail
(1)	When the power is turned on, i.e. the battery switch is switched to the "on" position.	LCD display should turn on with no messages, then remain on with varying messages. Green LED should light.	LCD displays with no messages and remains on. Green LED lights.	P, P
(2)	Compile C code on ATmega328PB.	LCD should display: "Steps: 0" on line one, "Distance: 0.00 mi" on line two and "Battery: *battery percentage may vary* %" on the third line. With a battery indication in the upper right hand corner.	Result identically as expected.	P
(3)	Walk with accelerometer, or gently rock accelerometer back and forth in a walking motion.	LCD should display a step value that represents the actual number of steps taken.	LCD counts every back and forth motion as an individual step.	P
(4)	Shake accelerometer aggressively.	LCD should display the number of steps prior to shaking aggressively, i.e. these steps should not be counted as they could never represent an actual human walking or running.	LCD counts steps in this state.	F
(5)	Hold accelerometer at a strange angle with no movement.	LCD should not increment in steps or distance.	LCD displays unincremented steps and distance.	P
(6)	Hold accelerometer upside down and walk with accelerometer or gently rock back and forth in a walking motion.	LCD should perform as in step (3).	LCD performs as in step (3).	P
(7)	Walk ten steps.	Distance should change from "0.00 mi" to "0.01 mi".	Distance display changes from "0 mi" to "0.01 mi"	P
(8)	Check battery level over time.	The battery level should decrease over time.	I have not experienced a decrease in overall battery life yet, but with a potentiometer I was able to vary the battery percentage to have the adjusted battery level display appropriately.	P
(9)	Leave the accelerometer untouched.	The number of steps displayed as well as the distance should not change at all.	The number of steps displayed as well as the distance traveled does not change at all.	P
(10)	Check to ensure that refresh of values only occurs when new values are obtained.	Refresh of row name, i.e. "Steps, Distance, Battery" should never happen, only when values change from previous should there be a refresh.	Row names never change, refresh only occurs upon change of value.	P
(11)	Check to ensure battery stays in one position once level if obtained, ensure battery level not fluctuating with ADC too much for user interface	Battery percentage acquires new battery level if battery status has changed.	Battery percentage fluctuates between current and previous reading due to change in ADC converter.	F

Figure 1: Up-to-Date Test Matrix

3 Laboratory Experimental Results

The schematic of the pedometer shown in figure [4] was greatly optimized from the previous experiment. The 16 MHz frequency oscillator crystal was removed in which requires that in firmware the internal 8 MHz clock is enabled which allowed for a gain of about 4 mm of surface area on the board layout. The z component on the accelerometer was also removed as the changes in the wrist movement correlating to the z direction were mainly impacted by bending the wrist in and out, which is not a common way of walking. This data was not being utilized in code as it was causing a much higher level of sensitivity in the counting of the steps. Therefore the elimination was not detrimental to the experiment.

In lieu of the voltage divider circuit that was supplying the accelerometer on the previous schematic submission, a 5V to 3.3V linear voltage regular was incorporated which although is a larger component, saved the space of three larger resistors and was more efficient. The AREF and AVCC on the ATmega were also incorporated and supplied voltage to ensure that the analog features of the device had a reference point and were properly supplied, this was necessary as several ADC converters were used in this design. A pull-up resistor was added to the schematic reset pin as well for the programming AVR ISP firmware circuitry and several decoupling components were also added to the schematic for noise cancellation.

Figure [3] represents the final working design of the pedometer project. The decision to remove the external 16 MHz frequency oscillator was detrimental to the circuit. The function of the manufactured design operates at half the speed the prototype in figure [13] does and is therefore extremely inaccurate. Steps are not displayed as quick as they are counted where measurements were checked using a mutlimeter and measurements are changing accordingly. Within firmware the frequency of the core processor unit was updated to reflect the 8 MHz rather than the 16 MHz CPU.

When programming the ATmega328PB on the final board layout, the 8 MHz clock was selected to change the fuse bits as explained in the ATmega datasheet and therefore the device may have slower processing due to the CPU speed. An AVR ISP was purchased from Amazon without updated software which took a great deal of time to reprogram for the ATmega328PB and for updating the software. Altering the code on the prototype to function with an 8 MHz clock displays fine. Rather than the 4.7 nF decoupling capacitors, 10 nF capacitors were used for the final design on the accelerometer which may be a factor for why the display is inaccurate, but that is not a great variation in capacitance therefore is likely not the cause.

The difficulty with displaying the battery percentage where the value would fluctuate even after calculating and accounting for the variation appropriately was not an issue in the final design with fast fluctuation. This was another indicator that the processor speed being cut in half was the

issue in terms of why the final design was not displaying accurate information. Another indicator as to why it is believed to be the CPU speed is due to the fact that the LED's indicating which parts of the board are receiving power is very well lit and the measured display of the battery is accurate according to calculations.

Figure [2] represents the manufactured board with all of the soldered components. The OLED is not currently connected in this picture considering it overlaps all connections and would interfere with the displaying of the soldered components. All components were placed by the pick and place machine with the 4.7 nF decoupling capacitors for the accelerometer outputs having been replaced with 10 nF decoupling capacitors. The pick and place machine also mentioned about was extremely temperamental therefore C3, C5, C7 and C9 were adjusted by hand and then the board was placed through the reflow oven for final soldering. The jumpers and LED were hand soldered and the final working design is shown in figure [3]. As mentioned above, the design works at half the clock speed as the prototype and this was directly reflected in the observation of the final design. However, in figure [3] the pedometer did exactly as expected based on firmware. The initial display was that the step count and distance traveled were at zero, and the measured battery level was displayed and accurate in terms of the percentage and battery indicator in the upper corner of the OLED display in increments of 10, where 90 to 100 percent should be displayed as a full battery level and at 99% this shows to be accurate in the figure.

3.1 Manufactured and Soldered Board

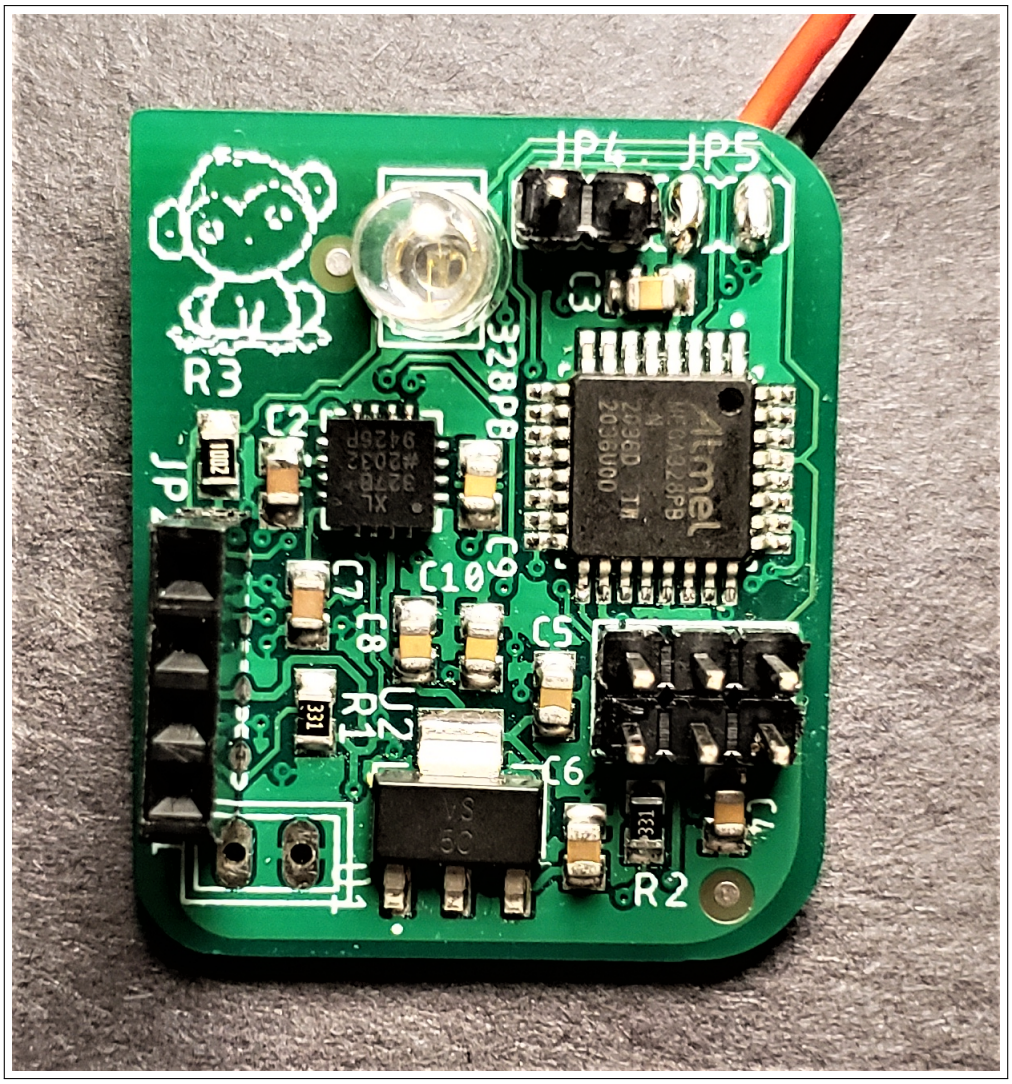

Figure 2: Manufactured Board with Components

3.2 Working Manufactured Design

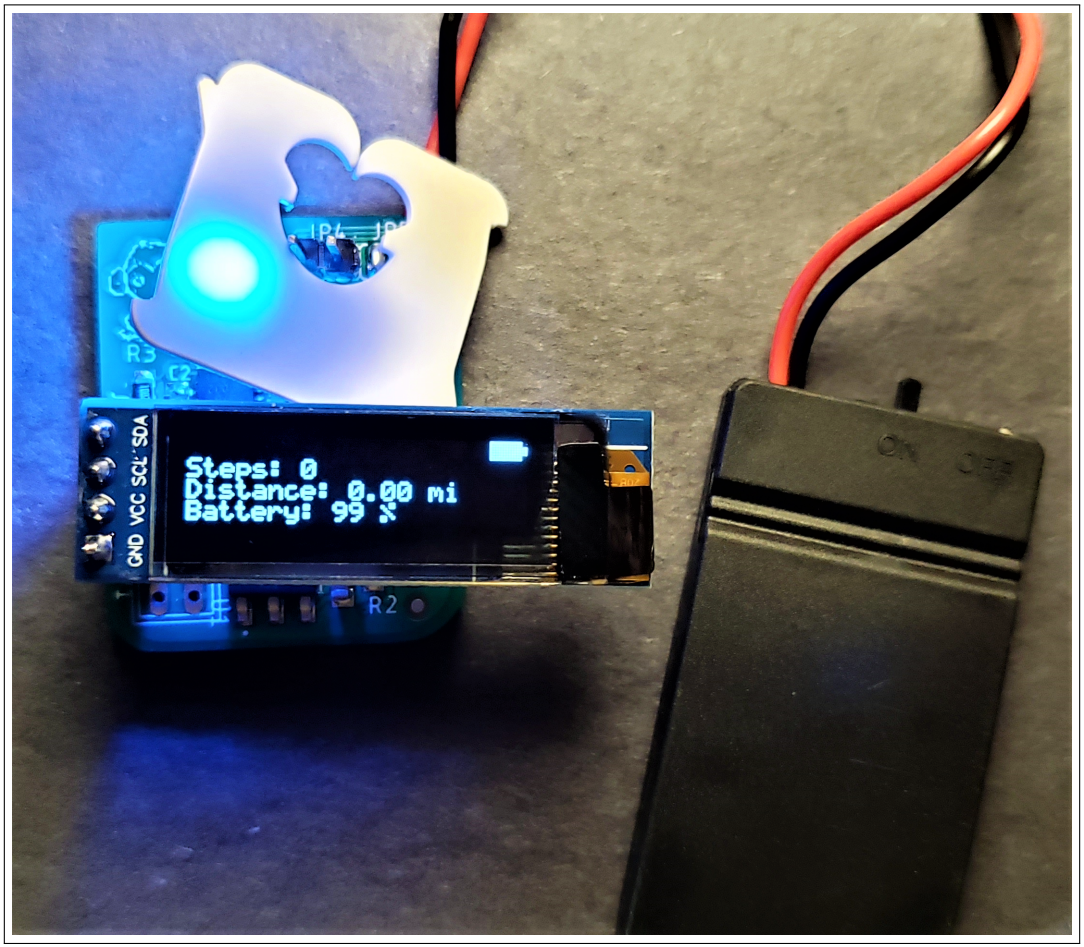

Figure 3: Final Design with Power Supplied

4 Discussion

The prototype for the pedometer design were finalized in Eagle software and manufactured by LCS board house. The pedometer schematic was greatly minimized and improved to generate a lightweight and realistically small sized PCB in which may comfortably be mounted to a human wrist for user interfacing. Revisions to the schematic and areas where improvement was required and recognized will be discussed in further detail below.

4.1 Revisions in Eagle Schematic and ERC's for Final Design

Many changes were made to the schematic which were crucial to the functionality of the manufactured PCB. Had these changes not been made, the design would have failed despite all of the prototyped tests and test matrices. Such revisions consisted of decoupling all supply pins to emulate lower input voltage frequency signals and clear out the high frequency noise components. They also ensured there were no voltage spikes in which may result in a mV supply difference which is enough to fry components such as operational amplifiers and in this case the accelerometer and OLED screen.

The decoupling capacitors were added to the voltage regulator which was implemented from the previous reports design recommendations. The linear regulator TLV1117 was used to convert a +5V supply into a +3.3V supply without causing one load to draw more current and change the voltage drop in which was initially happening with the voltage divider circuitry. This was supplied to the accelerometer because the required supply is between 1.8V and 3.7V where the supply should never exceed 3.7V, this may easily happen with a spike therefore this was necessary. The OLED requires no more than +3.3V supplied to the power pin, however the regulator was not necessary for this supply pin as the OLED backpack already contains a voltage regulator, therefore +5V was supplied to the OLED VCC pin.

The main ERC that was corrected was a no connection pin on the accelerometer being tied to ground. This was fixed by editing the library for the accelerometer device symbol and removing the no connection pin and replacing it with a passive input/output pin instead. The capacitors on the output pins for the x and y components were also placed in parallel to properly decouple the output signals. This was properly incorporated in the prototype, however the schematic placement was incorrect. The z component on the accelerometer was removed to save the space of a pin header connection for debugging, a capacitor and a wire connection to the ATmega. The z component was not in use in code as it was causing the step counter sensitivity to count steps unrealistically fast. The z plane correlated to the wrist bending in and out from normal stance which would not impact the counting of each step and would rather only be noise compared to the important data.

Another major change which was done to save space rather than due to error was that of the oscillator crystal. The crystal itself took up about 25 mm of the board width and about 4 mm of the board length. This space was instead used to place the voltage regulator on the board considering the pedometer would function similarly with the internal 8 MHz frequency oscillator built into the ATmega once firmware was implemented accordingly. This internal clock was enabled when the AVR ISP was used to load the firmware onto the final design by adjusting the fuse bits and the F_CPU was adjusted to 8000000 rather than 16000000.

The final schematic change that was made was supplying the AREF and AVCC pins on the

ATmega with the device supply voltage. This was necessary, and something that was not realized, in order to give the analog devices within the micro controller a proper analog signal and a proper analog reference point. The pedometer design utilizes roughly four of the seven ADC's located on PORTC of the ATmega, therefore the readings from the board ADC's would have been extremely inaccurate had this not been addressed by the TA.

Figure [6] represents the electronic rule check for the schematic layout in which displayed that the board and schematic were consistent and also displayed several warning messages as well as approved errors/warnings. The consistency was important to ensure the schematic was directly translating to the board layout which would check that all connections and components on the schematic were also on the board layout. There would be major manufacturing errors had they not been consistent. The warnings displayed are due to the pin headers or jumpers not being given a value, it was not necessary to give values to these components as they have no value and were properly connected to their designated connections once the board was populated. The approved warnings and errors consisted of the frame having no value which was a similar case as the jumpers, where the outline of the board did not have value. The output and supply pins being mixed was a result of mislabeling on the linear voltage regulator device symbol which would not have caused any issues on the board itself therefore they were approved (this was also advised per Tim that they may be approved).

4.2 Revisions in Eagle Board Layout and DRC's

The revisions made on the board layout were directly correspondent to the changes made within the schematic. Components such as the oscillator were removed to save space and utilize the internal clock of the ATmega which allowed for the additional surface area needed for the voltage regulator to regulate the input to the accelerometer supply pin. Once all of the ERC's were fixed, the components were then placed onto the board itself.

Placing components onto the board was the most challenging part of the layout as there was much consideration needed for where larger components would fit while still being able to place the decoupling capacitor close enough to make a difference in the signal, as well as whether there would be enough space between components to make the proper connections. The ATmega328PB was placed in the upper left corner of the board where the controller was flipped to ensure PORTC was on the left hand side of the board. This made it easier to connect the jumpers to PORTC PIN0 and PIN1 without needing to place wires all the way across the board. The AVR ISP was placed close to the MISO0, MOSI0, RESET and SCK pins on the ATmega for the same reason, this also ensured there was still plenty of space left for the other components. The wires were placed with the knowledge that the top layer wires could not go under pads, however the bottom layer could.

Therefore the top layer was traced to a point where a via could be placed through to the bottom layer and the trace could then follow on the bottom layer this way no pads were interrupted or overlapped.

The accelerometer, voltage regulator and two LED's were then placed onto the board as well as the four jumpers on the bottom of the board. The jumpers were placed in a way that the OLED would face in the direction that the pedometer would rest on the wrist of the user. This meant that the breakout board pins in the order SDA1, SCL1, VCC and GND needed to be in the same order on the jumpers where the OLED would resultantly lie across the board in parallel direction to the monkey friend. The LED to the right of these jumpers was placed right below the OLED which would indicate whether the ATmega was receiving power by turning the LED on when power is supplied and turning the LED off when power is not supplied. The other LED would indicate whether the entire circuit was powered on or whether the device was turned off.

The accelerometer was placed more in the center of the board and was placed so that when the device is placed on the users wrist, the accelerometer would be facing right side up. This did not make a difference in the actual measurement of the data, that is the direction of the accelerometer, however this simplified wiring to decoupling capacitors and the direction of the device would be clear for the pick and place machine. Other components such as the remaining capacitors, resistors and the regulator were placed in convenience according to the previously placed components. There was intentional space for the devices that requires decoupling capacitors, however resistors were placed where they fit.

The DRC was run by clicking the DRC caution symbol on the board side panel, loading the text file in and running the check. This ensured there were no overlaps or errors which would result in manufacturing issues. The DRC was consistently checked to ensure issues were resolved along the way rather than needing to do a complete remake of the board due to an overload of manufacturing errors. Once the board was complete, a ground pour of copper was placed on the top layer by using the polygon feature and doing the same on the bottom layer. The miter tool was used to round the corners of the board and the dimension feature was used to check the sizing of the board. Images were placed on the bottom layer using the run ULP, import-bmp, importing the bmp picture file, resizing the pixels to be greater than or equal to 0.155 mm per manufacturing specification, and then placed in a location where the pixels did not overlap device pads. This was done on the top and bottom layers, layer 21 tPlace and 22 bPlace respectively. The text feature was then used to put the name of the student, the course and project as well as the github link for the project. The copper pour and images may be seen in figures [9] and [10].

4.3 Gerber and Assembly Files

The CAM files were processed with ECE388.cam and the Gerber files were generated for all silkscreens, drill holes, pad placement, sizing, etc. Assembly files were generated for the pick and place machine which was extremely temperamental the day boards were being assembled. The files were sent to the LCS board house and properly manufactured and returned with no errors. The Gerber and assembly files may be observed in figures [7] and [8].

4.4 SMT Components and Debugging Manufactured Board

Once the manufactured board was returned from the board house, components were placed using the Gerber and Assembly files together to help the pick and place populate the components. The pick and place very carefully determined the orientation of the board by referencing the fiducials. The machine then determined where each component would be placed based on the starting component on the final bill of materials submission. The machine was fairly inaccurate and placed capacitors sideways, upside downs and even occasionally threw them on the ground. Therefore, these components were placed by hand on the solder masked board and placed in the reflow oven once connections were repaired. After the reflow oven, the pin headers were placed onto the board in their respective locations, then the battery pack was soldered onto the back of the board to allow for the pack to be "out of site," which is in quotes because it is a large battery pack.

Since the OLED display was too large and overlapped the ISP pins, additional female pins were soldered to the board to give the OLED extra height in which the OLED fit snug onto the board thereafter. Figure [2] shows a fully populated board with the exception of one LED missing. The reason for the LED missing was due to the fact that the LED surface mounts were not placed onto the board for a reason unknown to the student, and the through hole LED's were much too bright to have two of them mounted on such a small surface.

Figures [11] and [12] shown the front and back of the unpopulated board where the pixels of the monkey and yin yang cats were clearly within the board house sizing specification. A casing for the pedometer was not made considering SENG 210 was being taken apart and relocated, meaning the 3D printers were doing the same and were not assembled prior to not returning to campus for Thanksgiving. Therefore the final design may be viewed in figure [3] fully functioning in the original state, however not properly functioning in the counting stage due to a CPU frequency that is too slow and has not yet been adjusted. As mentioned in section [3], the measurements from the pin headers for the accelerometer x and y axis were measured using a multimeter and appear to measure and vary as they did in the working prototype, this was why the reasoning for dysfunctionallity was deemed to be clock related.

5 Conclusion

It was essential that all deadlines for this experiment were met in accordance to their scheduled date. The pedometer schematic and board layout were submitted several times where changes were repetitively made prior to the board being given the pass for manufacturing. Overall, Eagle was accessible for the production of a pedometer schematic which synchronously generated a board layout with the same components and connections in the form of a PCB footprint. It was also essential to ensure the footprint placed on the PCB was the same as that mentioned in the data sheet for the device used within the pedometer project, such as the accelerometer.

There were several recommendations that were implemented in this final design that have been brought about by self reflection, peer reflection and TA guidance and this design would not be manufacturable had exterior influence not been available. The layout of a printed circuit board requires much thought in terms of the placement of each component and the efficient wiring of connections. The extensive prototyping that was done to ensure a working PCB allowed for better formatted firmware and more precise data portrayal for user interfacing.

After receiving the final manufactured board components were placed based on the Gerber and assembly files, which it is now understood why it was crucial these were correctly generated. After the board was populated, the AVR ISP was used to load the most current firmware from Atmel onto the ATmega328PB final design. The final design does not work exactly as expected, but revisions to firmware may account for some sacrifices made in hardware. Once these revisions due to clock speed are made, the pedometer should function similarly to the prototype. Although the final design is not entirely accurate, the final prototype reflects the proper functionality of a working pedometer with the exception of the battery percentage display. After rigorous changes to firmware, the battery percentage still fluctuates greater than expected and it would be beneficial to solely use the battery indicator icon instead. If the prototype were a reasonable size, it would make a great pedometer.

6 Recommendations

It is highly recommended that if this project was to be recreated or reconstructed that the design implement the 16 MHz external frequency oscillator. By utilizing the same decoupling capacitors when possible and utilizing only one resistor for both LED's, the space saved could have made room for the frequency oscillator. Also, the battery percentage may be entirely replaced by the battery icon allowing for less firmware as well as an extra output line for potential desired interfacing. The OLED display itself, although the smallest available for the pricing, was still rather large for interfacing, potentially orienting the OLED to generate characters as a 32x128 Pixel OLED may

be extremely beneficial.

7 Laboratory/Course Reflection

This design has so many different areas where improvement may be implemented, specifically in firmware. With several libraries and over 300 lines of code, this code could surely be optimized. If the project were group oriented, the hardware part of the project would have been more interesting and indigestible. As an electrical engineering student with less knowledge on overall optimization, it would have been beneficial to work with a computer engineer on a slightly larger design scale. This being said, the same capacitor may have been used for the decoupling purposes as well as the resistors for the LED's which would have saved even more space on this tiny design. Also allowing for a daughter board with a 16 MHz clock in case the 8 MHz was not enough would have been a better route to take, or at least an easier one. The datasheet will once again be analyzed to incorporate fixes to the final design allowing for proper functionality. The errors in the final design are believed to be directly due to a slower processing speed and instruction readability by the processor therefore allowing for the quick fix of adding the external oscillator would have been the best approach. Once this issue is fixed, it is believed that the final design will work as the final prototype does.

Overall, this course has been extremely painful, but it can be said students have absolutely grown so much because of it. The restrictions of COVID such as less time on campus, less limitless access to labs, less teamwork and less direct communication were extreme challenges this semester. However, having the ability to complete the project, although it is far from fully functioning, has been such an experience. This pedometer design is no FitBit, nor is it even a WalMart step counter, there are many areas in firmware that could be improved, however as an electrical engineering student the parts of hardware intended to be completed and accomplished were just that. Having resources such as Atmel Studios, EAGLE, Amazon, the internet, Tim Chase and other students were pertinent to student's success in 388 this semester.

If it were feasible to take less classes and work less, having time to truly focus on this course would have been amazing as it is imagined this is what it would be like in the field except with more help and much more experience. Having said this, there are many students who had an underwhelming experience in ECE 388 this semester. Whether it was due to expectations or a lack of finances and external resources normally supplied by the school, this class was extremely unfair. Not only have ECE students always had lab partners up until this point, but they have never had to do anything like this with full interfacing in hardware and firmware with little help. Many students did not even get the chance to develop a working prototype. It is not understand how senior design was allotted teams, however this course, a "mini senior design", was not granted this

privilege. In this case the student was fortunate enough to have supplies at home, able to order an OLED in which was the last part to arrive on campus, have a multimeter and whatnot, however many students did not have these advantages. Some students needed power supplies and function generators to test their equipment which was not feasible outside of lab time.

8 References

[1] "ATmega328PB Xplained Mini," 2017. [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/50002660a.pdf

[2] "AVR® Microcontroller with Core Independent Peripherals and PicoPower® Technology," 2018. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/40001906C.pdf

[3] "How to Make a Practical Pedometer," Apr 2018. [Online]. Available: https://blog.adafruit.com/2018/04/11/how-to-make-a-practical-pedometer/

[4] C. Woodford, "How Do Pedometers Work," Jun 2020. [Online]. Available: https://www.explainthatstuff.com/how-pedometers-work.html

9 Appendices

9.1 Schematic Layout

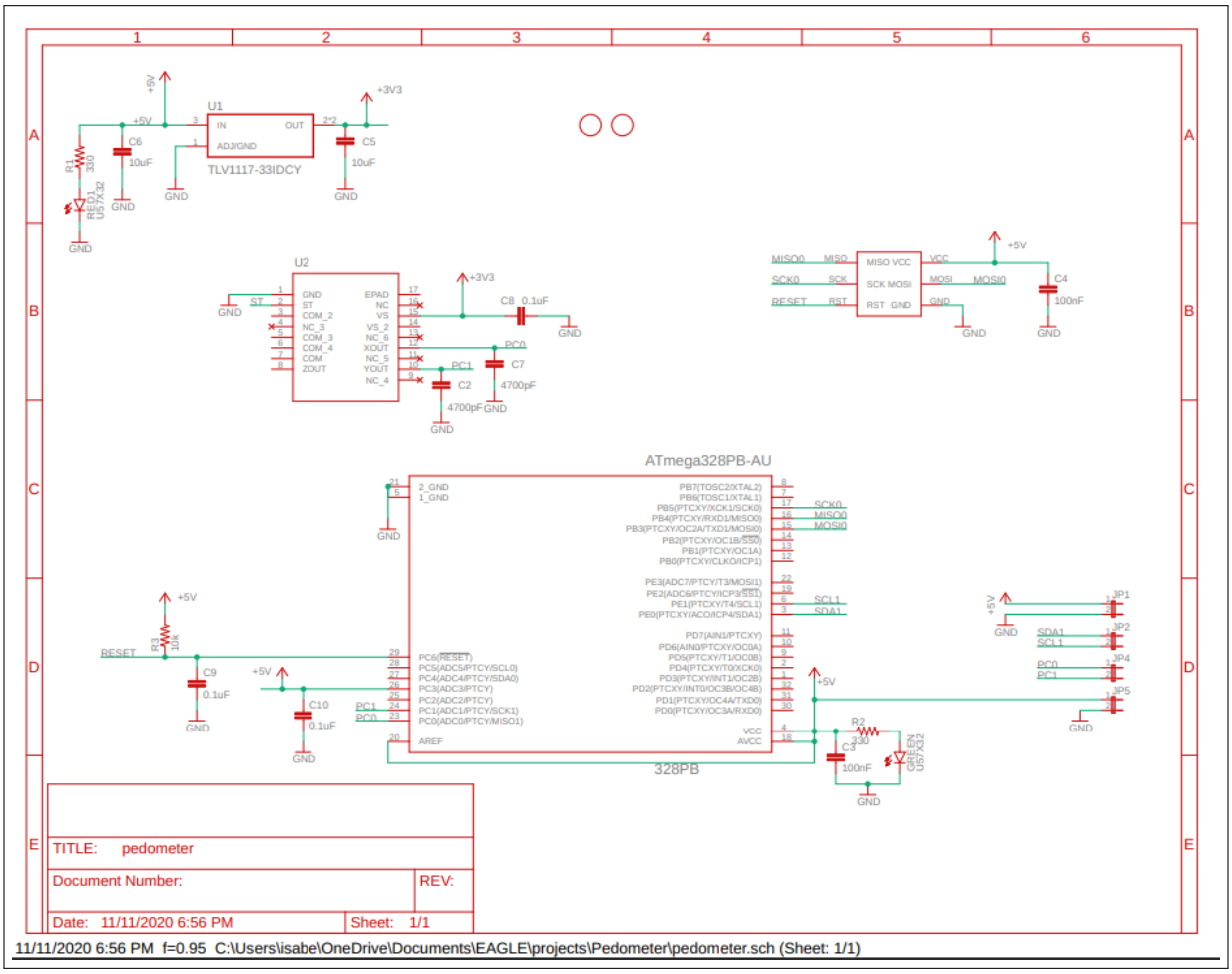

Figure 4: Pedometer Schematic Design

9.2 Board Layout

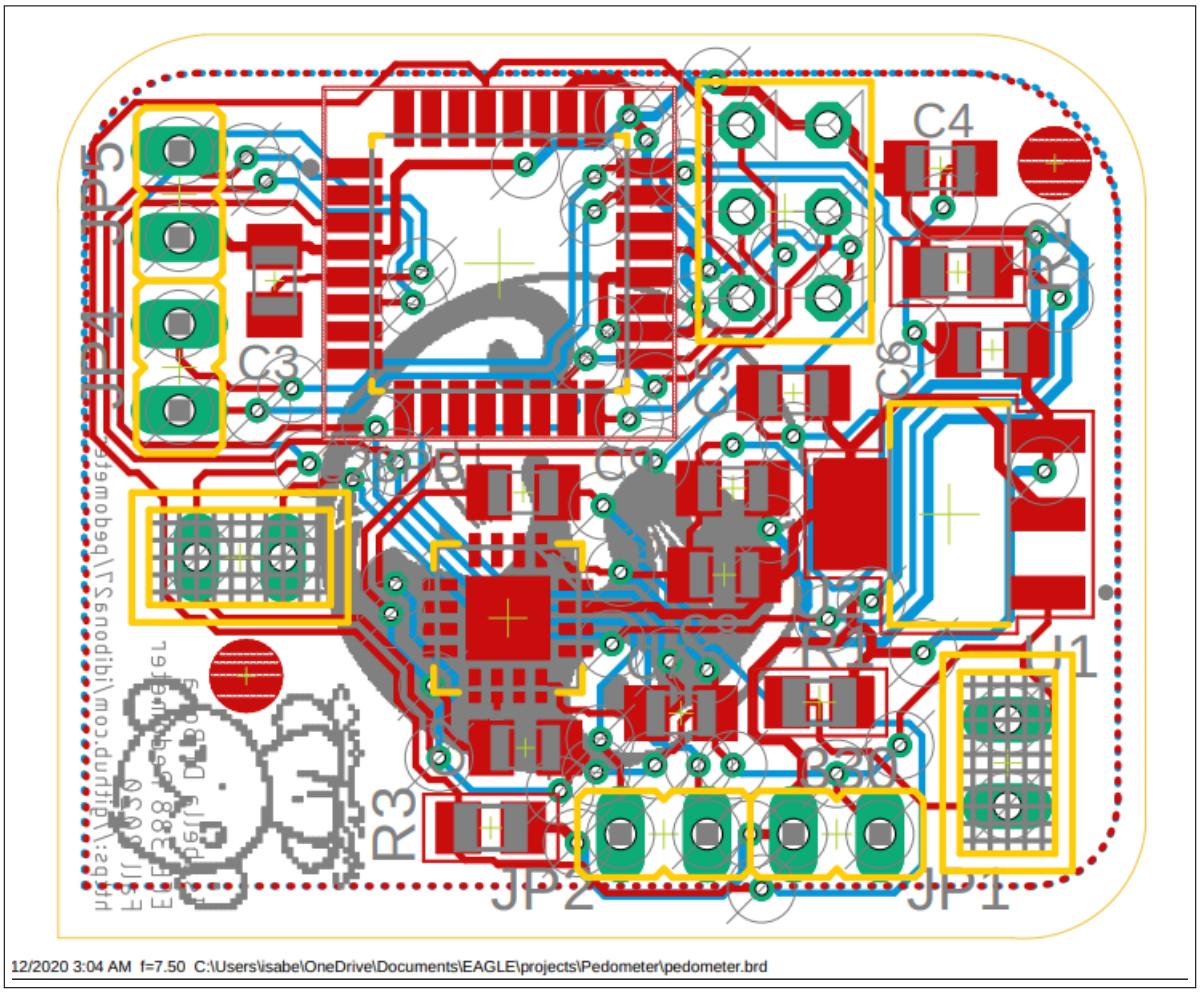

Figure 5: Pedometer Design PCB Board Layout

9.3 Schematic ERC

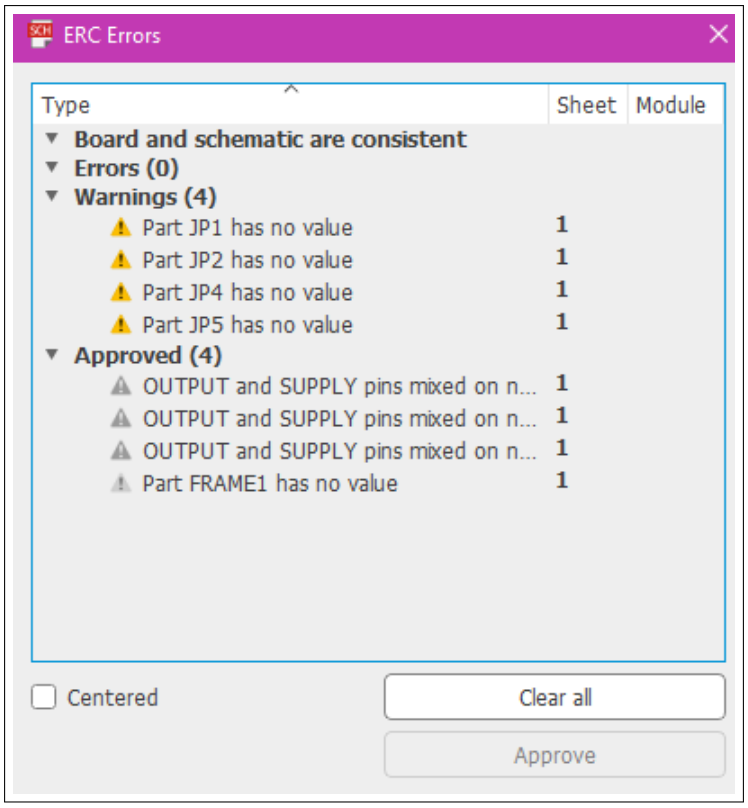

Figure 6: Pedometer ERC

9.4 Gerber Files

Figure 7: Gerber Files

9.5 Assembly Files

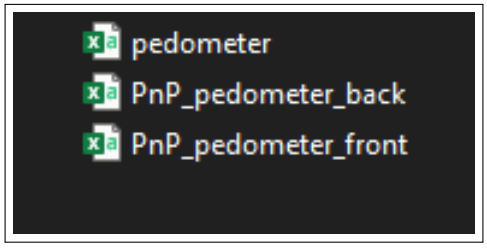

Figure 8: Assembly Files

9.6 Top Layer of PCB Design

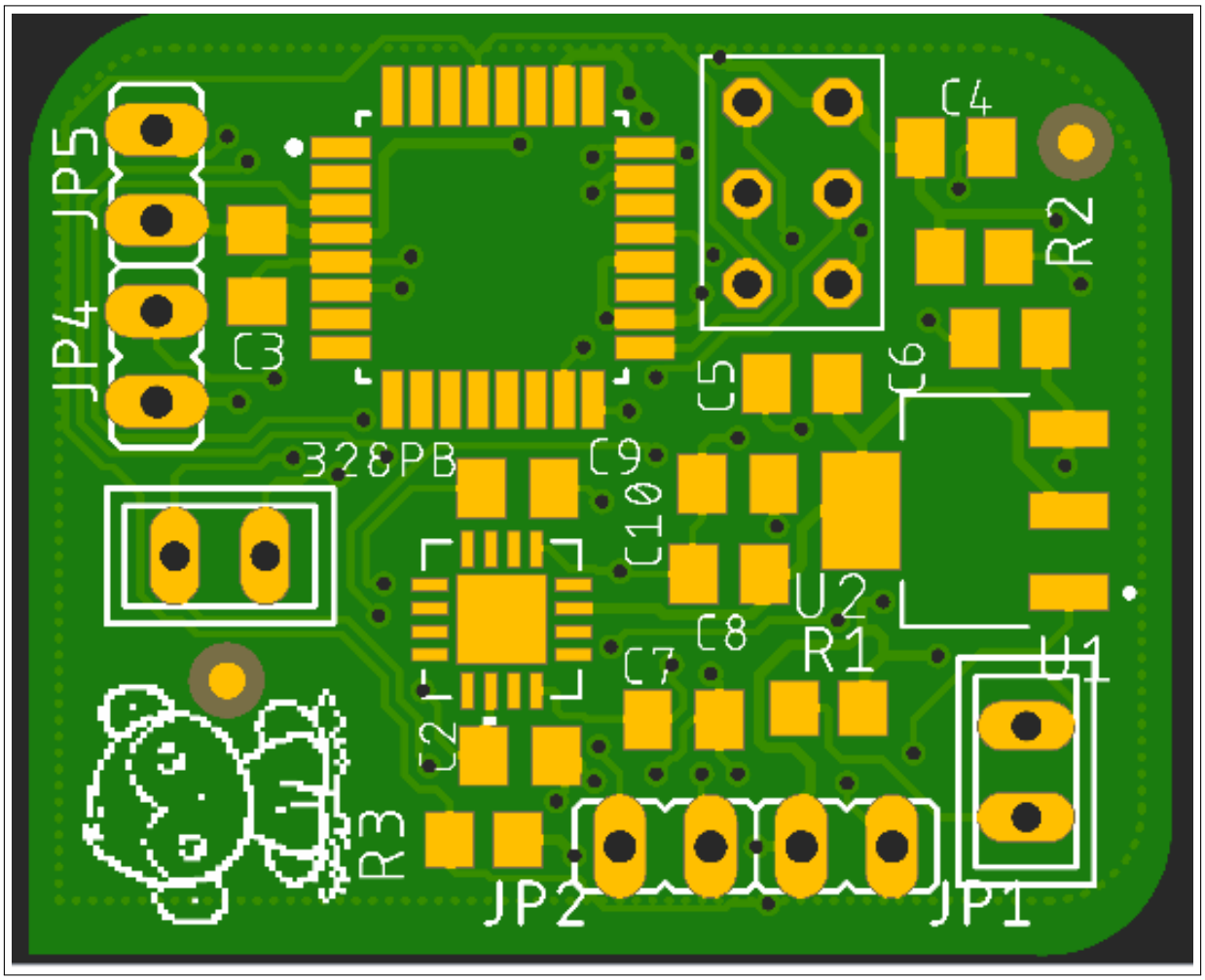

Figure 9: Top Layer

9.7 Bottom Layer of PCB Design

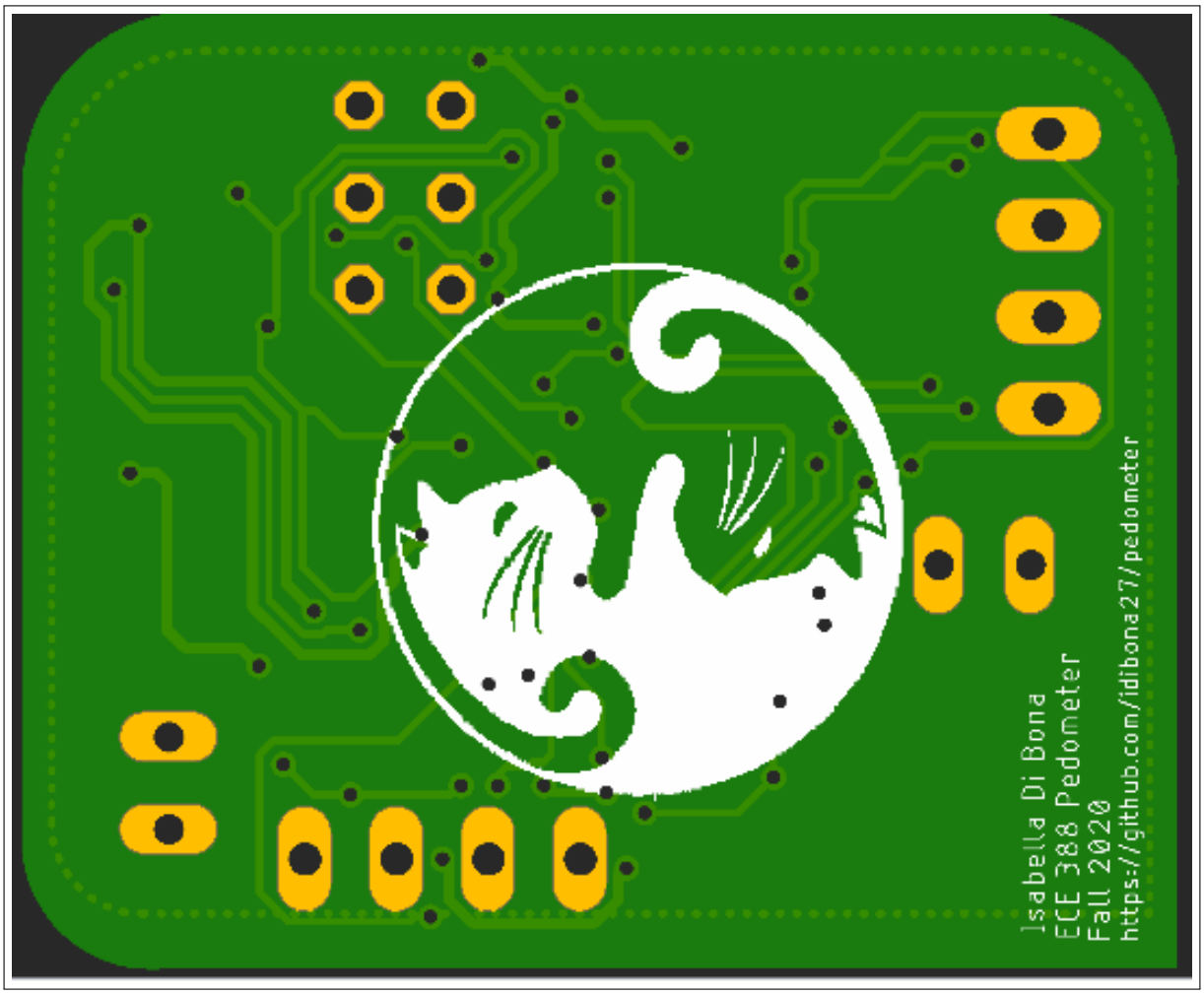

Figure 10: Bottom Layer

9.8 Manufactured Top & Bottom Layer of PCB Design

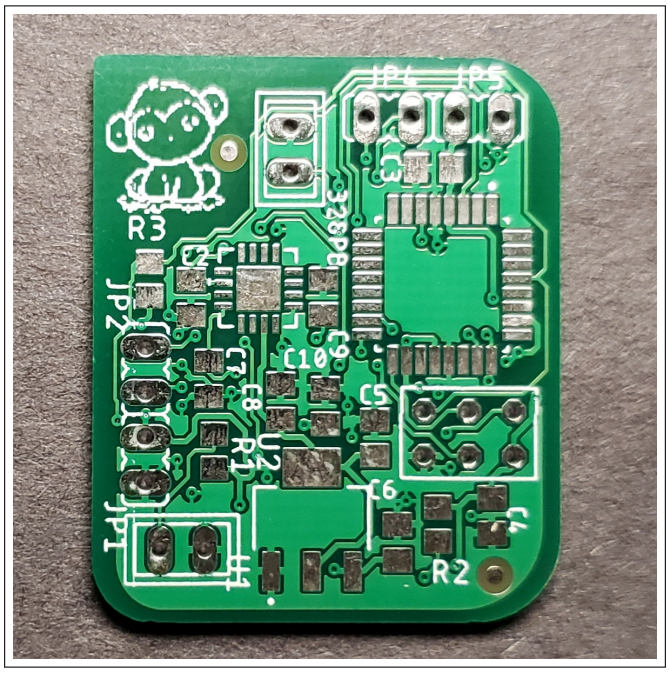

Figure 11: Top Layer

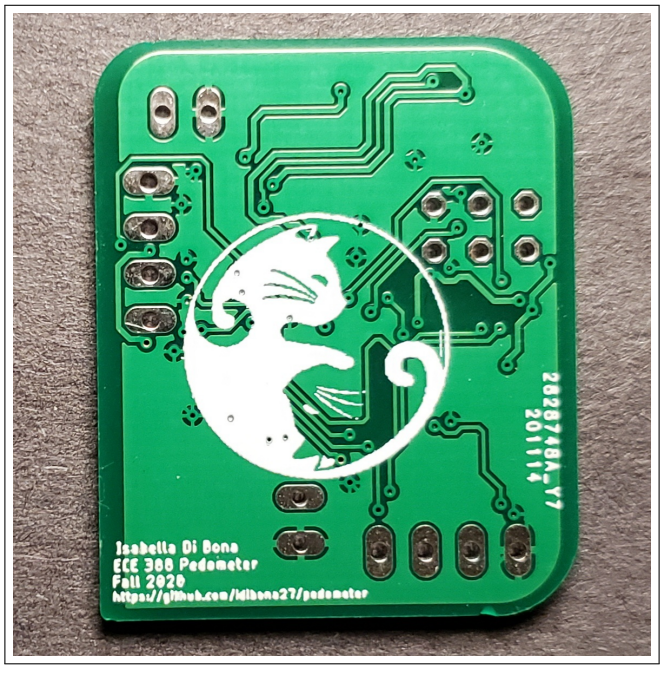

Figure 12: Bottom Layer

9.9 Working Prototype

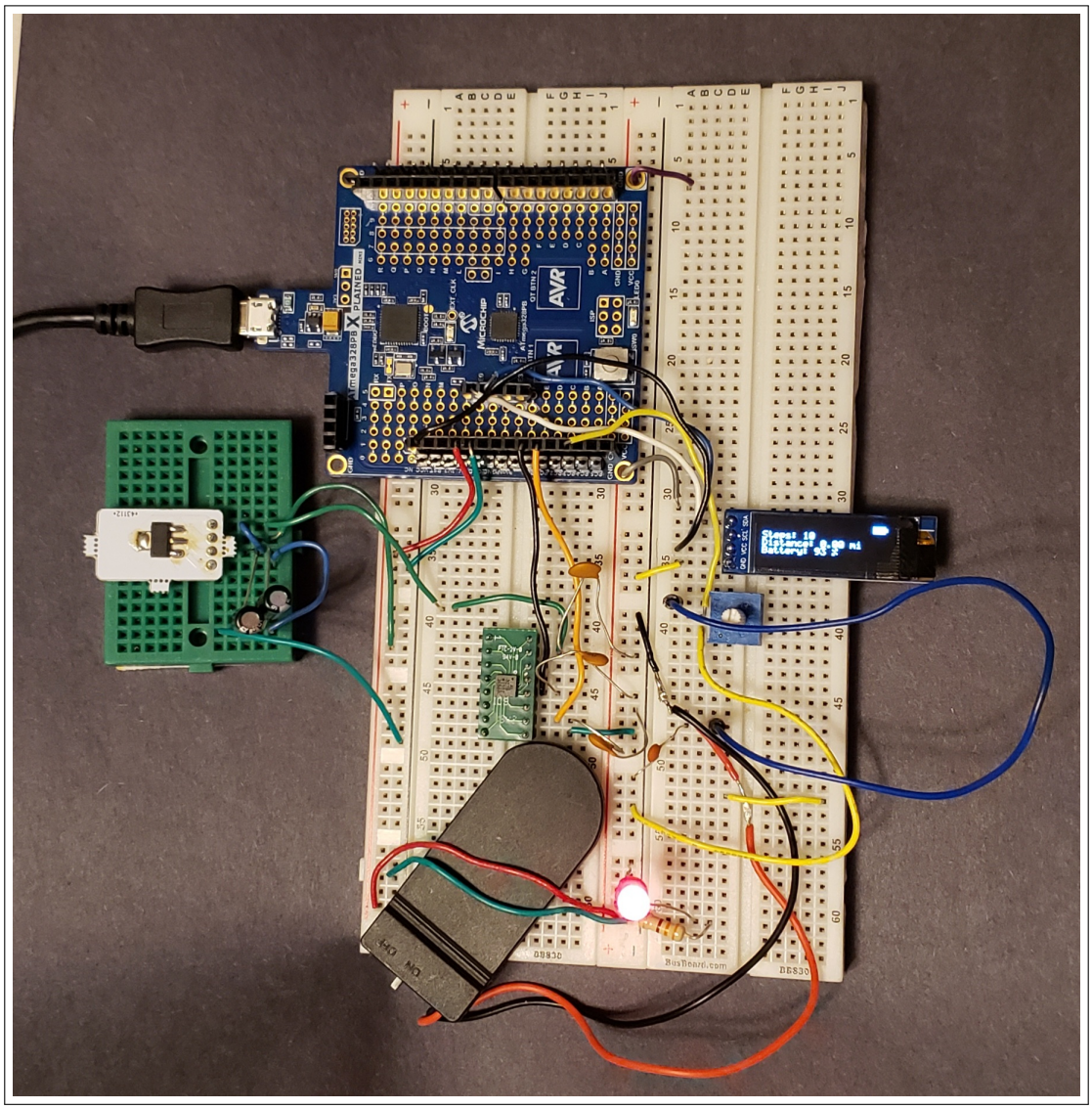

Figure 13: Prototype

9.10 Pedometer Firmware C Code

File 1: Pedometer Firmware

```
/*

 * Created: 10/3/2020 10:27:31 AM
 * Author : isabella dibona

*
*/

#include <avr/io.h>
#include "lcd.h"
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

//#define F_CPU 16000000
#define F_CPU 8000000
#include <util/delay.h>

//extern void lcd_backlight(char on);

//#define LOWER_threshold 400
//#define UPPER_threshold 600

/****prototype functions****/
void setUpperandLower();
void checkBattery_status();
void checkDistance();
void checkDeltaBattery();
void batterIconPercentage();
uint8_t accelInitialize();
uint8_t checkFor_newStep();
uint16_t AnalogRead16(uint8_t channel);
/***************************/

```

```
/*****initialize globals*****/
uint16_t step = 0;
uint16_t findInitialXcoord[5];
uint16_t findInitialYcoord[5];
uint16_t LOWER_threshold = 0;
uint16_t UPPER_threshold = 0;
uint8_t batteryLevel = 0;
uint8_t previousBatteryLevel = 0;
double thresholdMagnitude = 0;
double averageInitialX = 0;
double averageInitialY = 0;
double voltageLevel = 0;
double magAccel = 0;
double previous_Magnitude = 0;
double new_Magnitude[5];
char buffer[10], suffer[4], huffer[6];
/*************************/


 int main(void)
 {
  //OSCCAL =
  //CKSEL = (0<<CKSEL3) | (0<<CKSEL2) | (1<<CKSEL1) | (0<<CKSEL0);
  //CLKPR |= (1<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) |
     ↪ (0<<CLKPS0);
  //CLKPR |= (0<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (1<<CLKPS1) |
     ↪ (1<<CLKPS0);

  DDRC &= (0b11110000); //set pinc [0:3] to be inputs
  PORTC &= (0b11110000); //bit masking to preserve lower four bits
  DDRE &= 0xFC; //input is 0 for bits 0 and 1
  PORTE |= 0xFF; //enable pull up resistors to read in from pine 0 and
     ↪ 1

  lcd_init(LCD_DISP_ON);    // init lcd and turn on
  lcd_clrscr();
  setUpperandLower(); //initiates calibration position
  checkBattery_status();
```

```
  batterIconPercentage();

  lcd_gotoxy(0,1);
  lcd_puts("Steps: ");
  lcd_display();

  lcd_gotoxy(0,2);
  lcd_puts("Distance: ");
  lcd_display();

  lcd_gotoxy(0,3);
  lcd_puts("Battery: ");
  lcd_display();

    while(1)
    {
      //first line to display
      itoa(step, buffer, 10);
      lcd_gotoxy(7, 1);
      lcd_puts(buffer);
    lcd_display();

    checkDeltaBattery();
    batterIconPercentage();

      checkDistance();
      checkFor_newStep();
    }
 }

// set upper and lower limits --> type of calibration

 void setUpperandLower()
 {
  for (int i = 0; i < 5; i++)
  {
    findInitialXcoord[i] = AnalogRead16(0); //READ DELTA X
    findInitialYcoord[i] = AnalogRead16(1); //READ DELTA Y
```

```
    averageInitialX = averageInitialX + findInitialXcoord[i];
    averageInitialY = averageInitialY + findInitialYcoord[i];
  }

  averageInitialX = averageInitialX/5;
  averageInitialY = averageInitialY/5;

  thresholdMagnitude = sqrt(pow(averageInitialX,2) + pow(
     ↪ averageInitialY,2));

  LOWER_threshold = 0.95*thresholdMagnitude;
  UPPER_threshold = 1.13*thresholdMagnitude;
 }

//  convert voltage readings from accelerometer and battery into
   ↪ digital values

 uint16_t AnalogRead16(uint8_t channel)
 {
  ADMUX = (0b01<<REFS0) | (0<<ADLAR) | (channel<<MUX0);
  ADCSRA = (1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIE) | (0b111<<
     ↪ ADPS0);
  ADCSRB = (0b000<<ADTS0);
  ADCSRA |= (1<<ADSC);
  while((ADCSRA & (1<<ADIF)) == 0)
    ;
  return ADC;
 }

 // check for battery level/status

 void checkBattery_status()
 {
  voltageLevel = AnalogRead16(3);
  batteryLevel = voltageLevel/1023*100;

  itoa(batteryLevel, suffer, 10);
  lcd_gotoxy(9,3);
```

```
  lcd_puts(suffer);
  if(batteryLevel != 100)
    lcd_puts("␣%␣␣␣");
  else if(batteryLevel == 100)
    lcd_puts("␣%");
  lcd_display();
}

 void checkDeltaBattery()
 {
  uint16_t newVolt = 0;
  newVolt = AnalogRead16(3);
  uint8_t checkIfReadyforChange = 0;
  checkIfReadyforChange = abs(voltageLevel-newVolt);

  if (checkIfReadyforChange > 10)
  {
    lcd_gotoxy(9,3);
    checkBattery_status();
  }
 }

// get current state of accelerometer, poll for current magnitude

 uint8_t accelInitialize()
 {
  double x[5], y[5];
  double averageX = 0;
  double averageY = 0;
  uint8_t i;

  for (i = 0; i < 5; i++)
  {
    x[i] = AnalogRead16(0); //x coordinate = PC0
    _delay_ms(10);
    y[i] = AnalogRead16(1); //y coordinate = PC1
    _delay_ms(10);
  }
```

```
  for (i = 0; i < 5; i++)
  {
    averageX = averageX + x[i];
    averageY = averageY + y[i];
  }

  averageX = (averageX/5);
  averageY = (averageY/5);

  _delay_ms(10);

  magAccel = sqrt(pow(averageX,2) + pow(averageY,2));

  return magAccel;
 }

// check if magnitude increased after incrementing step by one, this
   ↪ will ensure
// the counter does NOT just continuously increase in one position

 uint8_t checkFor_newStep()
 {
  double averageNew = 0;

  accelInitialize();
  previous_Magnitude = magAccel;

  if((magAccel > LOWER_threshold) && (magAccel < UPPER_threshold)) //
     ↪ first resting condition check if movement from rest
  {
    for (int i = 0; i < 5; i++)
    {
      accelInitialize();
      new_Magnitude[i] = magAccel; //get average of new magnitude
      averageNew = averageNew + new_Magnitude[i];
    }
```

```
    _delay_ms(10);
    averageNew = averageNew/5; //new magnitude average over 5 samples

    if (((averageNew < 0.98*previous_Magnitude) || (averageNew > 1.03*
       ↪ previous_Magnitude)) && ((averageNew > LOWER_threshold) && (
       ↪ averageNew < UPPER_threshold)))
      step = step + 1;
    else
      step = step + 0;
  }

  else
    step = step + 0;

  return step;
}

void checkDistance()
{
  double howManyMiles = 0;
  howManyMiles = step/2000.0;

  dtostrf(howManyMiles, 4, 2, huffer);
  lcd_gotoxy(10,2);
  lcd_puts(huffer);
  lcd_puts("␣mi");
  lcd_display();

}

void batterIconPercentage()
{
  lcd_drawRect(114, 0, 125, 6, WHITE);
  lcd_drawRect(125, 2, 127, 4, WHITE);
  lcd_fillRect(125, 2, 127, 4, WHITE);

  if (batteryLevel <= 100 && batteryLevel >= 90){
   lcd_fillRect(114, 0, 124, 6, WHITE);
```

```
    lcd_gotoxy(4, 0);
    lcd_puts("␣␣␣␣␣␣␣␣␣␣␣");
  }

  else if (batteryLevel < 90 && batteryLevel >= 80){
    lcd_fillRect(114, 0, 123, 6, WHITE);
    lcd_fillRect(123, 1, 124, 5, BLACK);
    lcd_gotoxy(4, 0);
    lcd_puts("␣␣␣␣␣␣␣␣␣␣␣");
  }

  else if (batteryLevel < 80 && batteryLevel >= 70){
    lcd_fillRect(114, 0, 122, 6, WHITE);
    lcd_fillRect(122, 1, 124, 5, BLACK);
    lcd_gotoxy(4, 0);
    lcd_puts("␣␣␣␣␣␣␣␣␣␣␣");
  }

  else if (batteryLevel < 70 && batteryLevel >= 60){
    lcd_fillRect(114, 0, 121, 6, WHITE);
    lcd_fillRect(121, 1, 124, 5, BLACK);
    lcd_gotoxy(4, 0);
    lcd_puts("␣␣␣␣␣␣␣␣␣␣␣");
  }

  else if (batteryLevel < 60 && batteryLevel >= 50){
    lcd_fillRect(114, 0, 120, 6, WHITE);
    lcd_fillRect(120, 1, 124, 5, BLACK);
    lcd_gotoxy(4, 0);
    lcd_puts("␣␣␣␣␣␣␣␣␣␣␣");
  }

  else if (batteryLevel < 50 && batteryLevel >= 40){
    lcd_fillRect(114, 0, 119, 6, WHITE);
    lcd_fillRect(119, 1, 124, 5, BLACK);
    lcd_gotoxy(4, 0);
    lcd_puts("␣␣␣␣␣␣␣␣␣␣␣");
  }
```

```
  else if (batteryLevel < 40 && batteryLevel >= 30){
   lcd_fillRect(114, 0, 118, 6, WHITE);
   lcd_fillRect(118, 1, 124, 5, BLACK);
   lcd_gotoxy(4, 0);
   lcd_puts("           ");
  }

  else if (batteryLevel < 30 && batteryLevel >= 20){
   lcd_fillRect(114, 0, 117, 6, WHITE);
   lcd_fillRect(117, 1, 124, 5, BLACK);
   lcd_gotoxy(4, 0);
   lcd_puts("           ");
  }

  else if (batteryLevel < 20 && batteryLevel >= 10){
   lcd_fillRect(114, 0, 116, 6, WHITE);
   lcd_fillRect(116, 1, 124, 5, BLACK);
   lcd_gotoxy(5, 0);
   lcd_puts("LOW BATTERY");
  }

  else if (batteryLevel < 10 && batteryLevel >= 0){
    lcd_fillRect(115, 1, 124, 5, BLACK);
    lcd_gotoxy(5, 0);
    lcd_puts("LOW BATTERY");
  }

  lcd_display();
}
```