PRESSURE DETECTION: IMPROVING PROSTHETICS EFFICACY

By

Nathan Beauchamp

Sihao Chen

Mickey Zhang

Final Report for ECE 445, Senior Design, Fall 2017

TA: Yuchen He

13 December 2017

Project No. 21

Abstract

Current commercially available upper limb prosthetic devices use surface electromyography (sEMG) sensors to enable the user to operate such device. Although sEMG system provides an acceptable level of control accuracy, one of its shortcomings is that it is not very robust -- EMG signal incurs a lot of noise from many sources, such as high impedance of the skin, external shock, shifting of the arm, and more. This results in unintended movement of the prosthetic fingers, leading to increased device abandonment [1]. Our solution is to propose an alternative control system, based on TMG pressure sensors, which overcomes the aforementioned shortfalls.

Contents

1. Introduction

1.1 Objective

As able-bodied humans, we often take for granted our ability to perform basic physical actions such as walking and picking up objects with our hands. Unfortunately, for many people around the world, freeform body movement is something that exists only in dreams and the imagination. For example, in the United States, 16.3% of adults have some kind of physical functioning difficulty [2]. In many cases, engineering principles- including those of electrical and computer engineering- can be used to improve the lives of the disabled. One particularly important disability to address is the loss of arm function due to amputation. In the United States, 41,000 people have suffered the loss of a hand or a complete arm [3]. Engineering has attempted to address this problem with a prosthetic upper limb- however, there are issues with current implementations, namely efficacy and high cost.

One of the major challenges in designing a prosthetic upper limb is the effectiveness of the sensing method that collects biological signals from the patient's upper arm or shoulder and maps them to various hand and wrist movements. The traditional sensing method for this purpose is surface electromyography (sEMG) [4]. However, this method yields inconsistent results in practical use due to the fact that the sEMG electrodes oftentimes capture a noisy mixture consisting of several arm muscles [5]. Several researchers have demonstrated that, when attached to the arm, an array of tactile pressure sensors can be used to capture muscle bulges and infer the patient's desired wrist and hand movements [5]. Therefore, a pressure sensor array should be considered as a viable input mechanism for a prosthetic arm.

Our objective is to design and build a system to perform intent classification in a prosthetic arm. Our system will use a pressure sensor array to replace the existing sEMG-based sensing solution. The system should integrate cleanly into the prosthetic arm developed by Psyonic, as described below.

1.2 Background

Psyonic is a local startup that is developing an affordable prosthetic hand for people with upper-limb amputations. Currently, they have a completed and working product that uses electromyography (EMG) to enable the patient to operate the hand. However, they face several obstacles in the prosthetic-arm interface. One of their main challenges is that the EMG signal incurs a lot of noise from many sources, such as high impedance of the skin, external shock, shifting of the arm, sweat, and more. This results in unintended movement of the prosthetic fingers. After communicating with the Psyonic team, we believe that we can overcome many existing obstacles by replacing the existing EMG model with a model based on pressure sensing.

Psyonic aims to provide an affordable prosthetic hand that will entail no out-of-pocket cost to patients with health insurance [6]. Given this, the components we choose will need to be as inexpensive as

possible while still fulfilling their intended purpose. For example, the microcontroller that runs the classification algorithms will need to be small-scale and low-power, yet still powerful enough to run the algorithms.

1.3 Performance Requirements

- The hardware component (i.e. the circuitry to collect, process, and store the pressure sensor inputs) should deliver the inputs with as little noise as possible and store the collected data, allowing the microcontroller to easily sample the values.
- The microcontroller should execute code to interface with the hardware component and run classification algorithms, mapping the patient's input to one of several hand gestures in real-time 50 ms windows.
- The machine learning model must classify the intensity map patterns read from the pressure sensors, outputting a set of hand/wrist movements. Training should be done on a per-person basis.
- Our design needs to integrate with Psyonic's existing prosthetic arm design; therefore, the final product will need to be small enough to fit comfortably into the bionic arm enclosure.

1.4 Block Diagram and Description

As shown in Figure 1 on the next page, the hardware component of our design can be divided into three primary blocks: the sensor module, the sampling/modulation module, and the control module. First, the sensor module reads the pressure applied by the patient's upper-arm muscles at various points along the human-arm interface. Since this module collects input pressure data, it must therefore also serve as the physical interface for the patient. This module gathers the input data needed for patient intent classification; as a result, it is imperative that it reads inputs accurately and with little noise. Second, the modulation module samples and quantizes the analog input data generated by the sensor module. It also stores the sampled results so they can be sent to the control module running the classification algorithm. Each modulation unit handles a single array of pressure sensors, adding a layer of indirection between the sensor and control units. This makes it easier for the control module to get the data it needs in a timely manner. Third, the control module executes the classification algorithm to map input pressure maps to output movement classes in real-time. Finally, the power supply and physical arm modules are provided by Psyonic and are as such outside the scope of our project. In the final version of our product, the control module would also communicate with actuators to drive the Psyonic mechanical hand (shown in Figure 2 on the next page). However, as we are developing a proof-of-concept prototype for Psyonic, we will instead simply output classification results to the terminal of a connected PC.

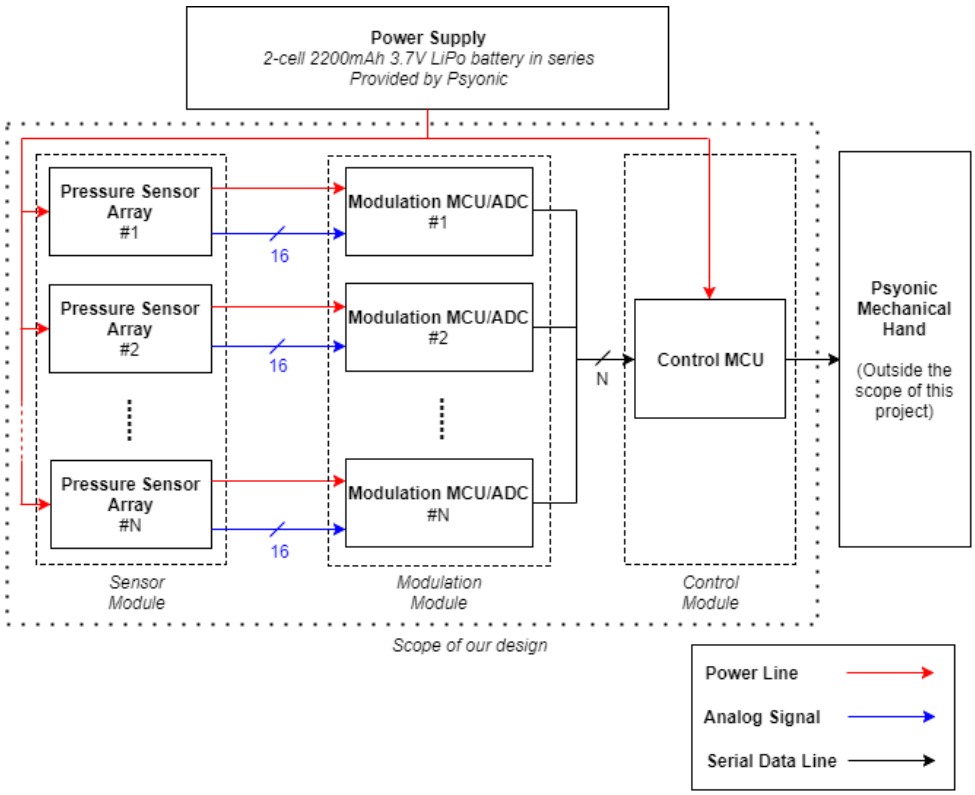

Figure 1: Block Diagram for the Hardware Component

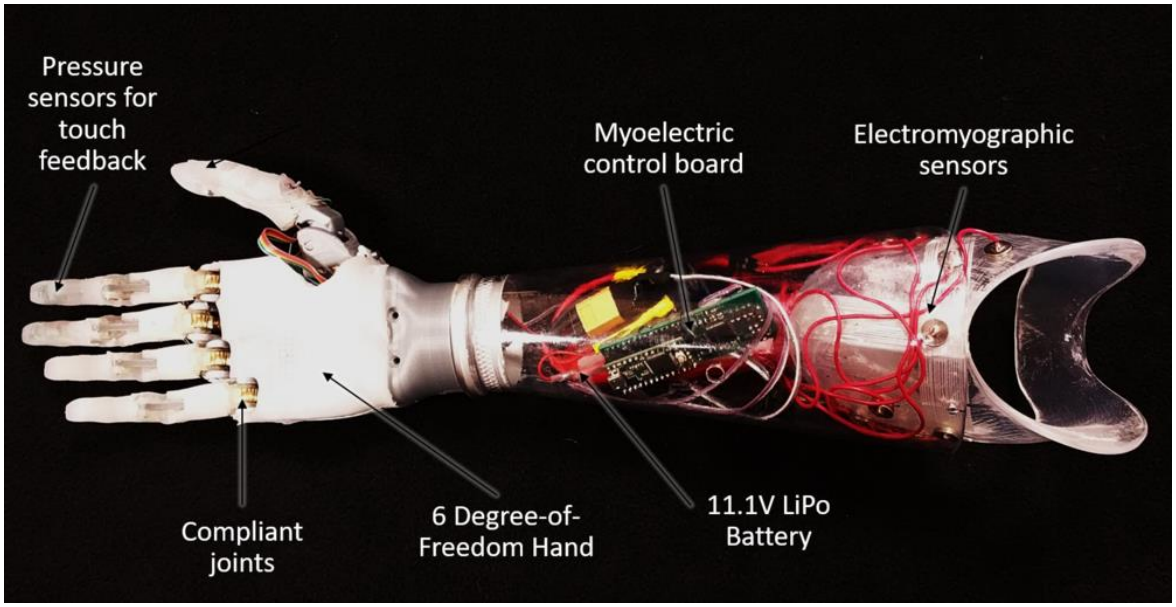

Figure 2: Current iteration of the Psyonic prosthetic arm prototype [6]. Our design replaces the EMG sensors with pressure sensor boards and the myoelectric control board with a new control board.

2 Design

2.1 Hardware Component

2.1.1 Pressure Sensors

The pressure sensor array block contributes to the overall design by providing input pressure data from the user, which will later be classified and mapped to an intent by the machine learning algorithm. Given only the idea that a bionic arm's EMG input system should be replaced with an input system based on pressure sensing, there are multiple ways by which the sensor module subsystem could be designed. The most important design decision for this module is the choice of pressure sensor. There are many different types of pressure sensors, including capacitive and piezoelectric varieties. However, we decided to use resistive tactile pressure sensors as they provide simplicity, safety, and consistent contact with the patient's skin, enabling easy integration into the bionic arm. They also have a linear output characteristic within a reasonable force input range [5], which facilitates the operation of the classification algorithm.

The tactile pressure sensors, as seen in Figure 3 below, consist of two electrodes and conductive sensory material placed on top. A variety of materials can fit this role- including human skin, as it turns out. However, as shown by Weiss and Worn, the material that yields the most accurate, classifiable output is conductive elastomer foam [7]. Meanwhile, the electrodes will simply be printed on a PCB- since we're aiming for 12 cells per array, 24 electrodes will need to be printed on each PCB. The output of a single tactile cell is the resistance between the two electrodes, which depends on the volumetric resistance of the sensor material as well as on the pressure applied to the material [5]. The resistance of the contact between the electrode and the material will change depending on the pressure applied- therefore, if R_v is the material resistance and R_{s1} and R_{s2} are the variable contact resistances, the overall resistance across the sensor is $R_t = R_v + R_{s1} + R_{s2}$ [5]. Then, a voltage divider circuit (shown in Figure 4 on the next page) can be used in order to more easily measure the pressure sensor readings. The output voltage can be calculated using the voltage divider rule:

$$V_{out} = 3.3\,V * \frac{R_t}{10\text{K}\Omega + R_t}$$

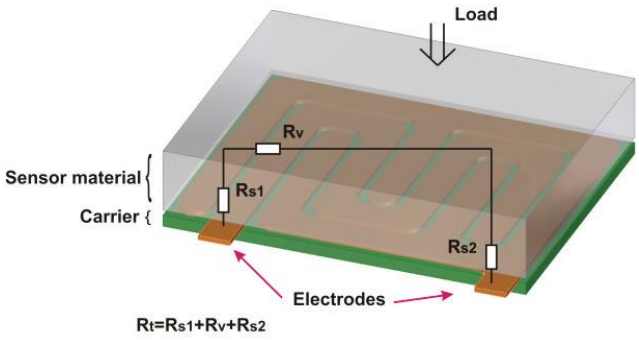

Figure 3: Schematic for a single tactile cell [5]

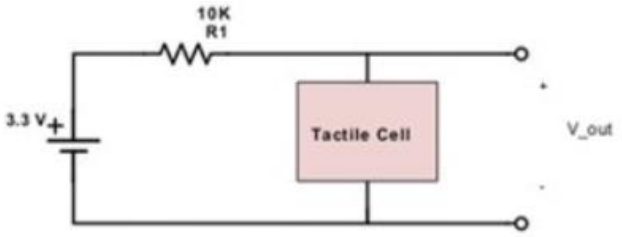

Figure 4: Voltage divider circuit to collect data from a single tactile cell.

During normal system operation, each pressure sensor in a given array will output the signal V_{out}, which will pass through the ADC and be sampled by the sampling microcontroller. The $10\text{K}\Omega$ resistor value was chosen so that the sensor would be able to produce a reasonable range of outputs and operate in a linear regime. This desired operational regime is shown in Figure 5 below.

Another relevant aspect of the pressure sensor design is that of consumed power. Since we are designing a system that is housed in a wearable device, we need to ensure that our system is not consuming enough power to harm or discomfort the user. With the stated design values, each tactile cell will consume the following amount of power:

$$P_{cell} = \frac{(3.3\ V)^2}{10\text{K}\Omega + R_t}$$

Here, R_t is the resistance of the tactile cell. Since there are 12 pressure sensors per board and five boards in total, the total power consumed by all of these is 60*P_{cell}. At its maximum, P_{cell} will be 0.001089 W, implying that the pressure sensor module will consume a maximum of 0.06534 W of power. This is a fairly insignificant quantity, and is definitely fine for a wearable device.

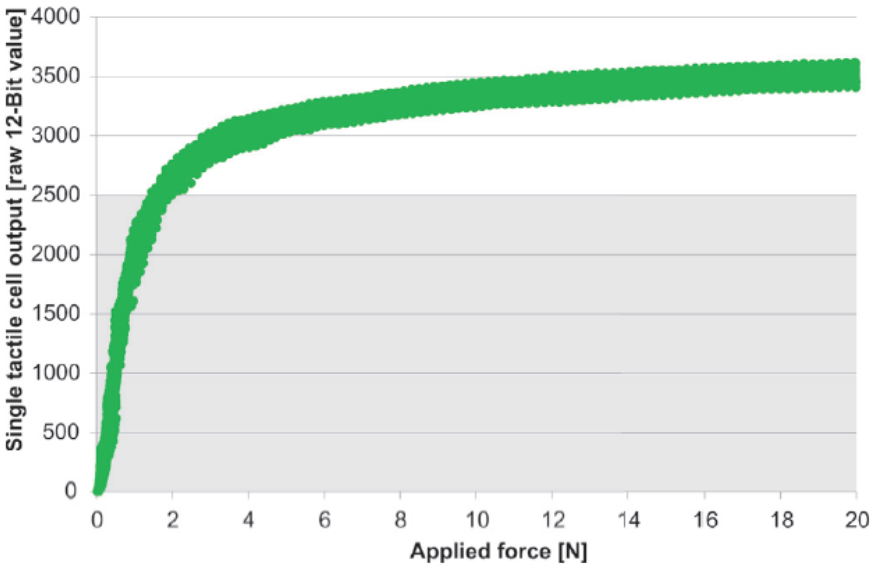

Figure 5: Sampled and quantized output of a single tactile cell for various applied forces [5]. The shaded portion of the plot indicates the normal (linear) mode of operation of the sensor [5].

A final interesting aspect of the pressure sensor design is their role in the design of our printed circuit boards (PCBs). Unlike the rest of our parts, our pressure sensors are not an off-the-shelf part; rather, their electrodes are etched on our PCBs directly as SMD pads. This is shown in Figure 6 on the next page. This fact results in two important design considerations for the board. First, no top-layer traces can be routed near the area of the board that houses the pressure sensor array. Traces placed too close to their sensors would affect their resistance curve, negatively affecting the data produced and the performance of the classification algorithm as a result. Furthermore, all surface-mount components, including the sampling microcontroller, needed to be placed on the bottom layer of the board. The second design consideration is the necessity of a T-stop layer surrounding the grid of pressure sensors. This ensures that solder mask will be poured on the area surrounding the sensors, isolating the sensors from the copper pour. Once again, this serves to safeguard the resistance curve and output voltage characteristics of the sensors.

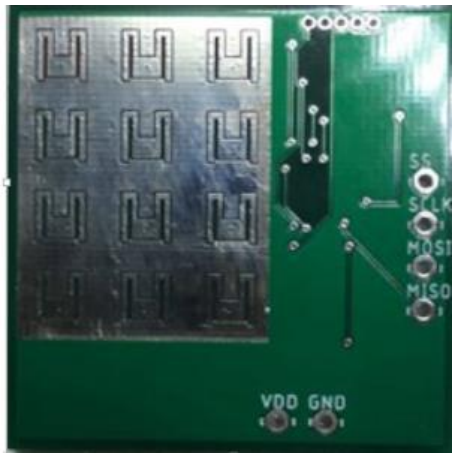

Figure 6: Our fabricated pressure board. The pressure sensor array is located in the upper-left corner.

2.1.2 Sampling Microcontroller

The sampling/modulation block consists of an analog-to-digital converter to render the pressure sensor signals usable for processing, and a small-scale microcontroller to sample and store the data for the pressure sensor array to which it is assigned. The two are in the same block because they will be present on the same chip- the microcontroller that we will use, PIC18LF2XK22, has 17 input ADC channels with a 10-bit resolution ADC [7]. Each array will contain 12 pressure sensors, which is why we chose a microcontroller with at least 12 input ADC pins; otherwise, we would need to utilize analog multiplexing in addition to that already used by the microcontroller (it would be unrealistic to expect a microcontroller to have 12 separate ADC modules). This would be needlessly complex and would likely cause some performance decrease, which is why it makes sense to use a microcontroller with at least 16 ADC pins. This microcontroller also has two SPI digital communication peripherals with which it can communicate with the main microcontroller [7]. Furthermore, PIC18LF2XK22 has an internal oscillator that is configurable to speeds of up to 16 MHz [7], which is sufficient to sample and store the data in line with the necessary ADC acquisition time. The analog signals generated by the pressure sensors are aperiodic, and the classification algorithm requires the average value to be sampled in order to work properly. Therefore, there is no particular requirement on the sampling rate of the ADC unit; only the

acquisition time is relevant, to ensure that the microcontroller can sample 12 analog inputs within each time window. PIC18LF2XK22 allows the programmer to select and configure the ADC acquisition time, so we will be able to fine-tune the device for our particular system. Given the simplicity of the operation it's performing, the sampling microcontroller does not need to run an operating system, nor does it have any particular memory requirements.

The Sampling MCU/ADC block interfaces with two other blocks in our design: the Pressure Sensor Array block, and the Control MCU block. First, analog voltage inputs from the Pressure Sensor Array block will connect to the ADC pins on the microcontroller so that they can be sampled. Furthermore, each sampling MCU will communicate with the main control MCU via the SPI communication protocol. We decided to the use the SPI protocol over the I2C protocol- the other option available on most microcontrollers- due to the difference in attainable throughput. The I2C protocol supports a maximum throughput of 3.2 Mbps in its "high speed" mode [8]; in contrast, the throughput supported by SPI depends on the clock speed used. Since the clock speed of PIC18LF2XK22 can be set as high as 64 MHz [7], implying that the SPI bus speed can be as high as 32 MHz, we will be able to achieve much greater throughputs with SPI. A master-slave protocol will be used- at the beginning of each time window, the control MCU will poll each of the "slave" sampling MCUs, requesting the most recent sensor data. Assuming time windows of 50 ms, 5 arrays of 12 pressure sensors each, and 10 bit ADC resolution (packaged as 16 bits), the amount of data that will need to be transferred to the control MCU per second is 20*5*12*16=19200 bits. This required data rate can be handled by the SPI protocol, thus providing quantitative justification for choosing SPI.

The schematic of the sampling microcontroller breakout is shown below in Figure 7.

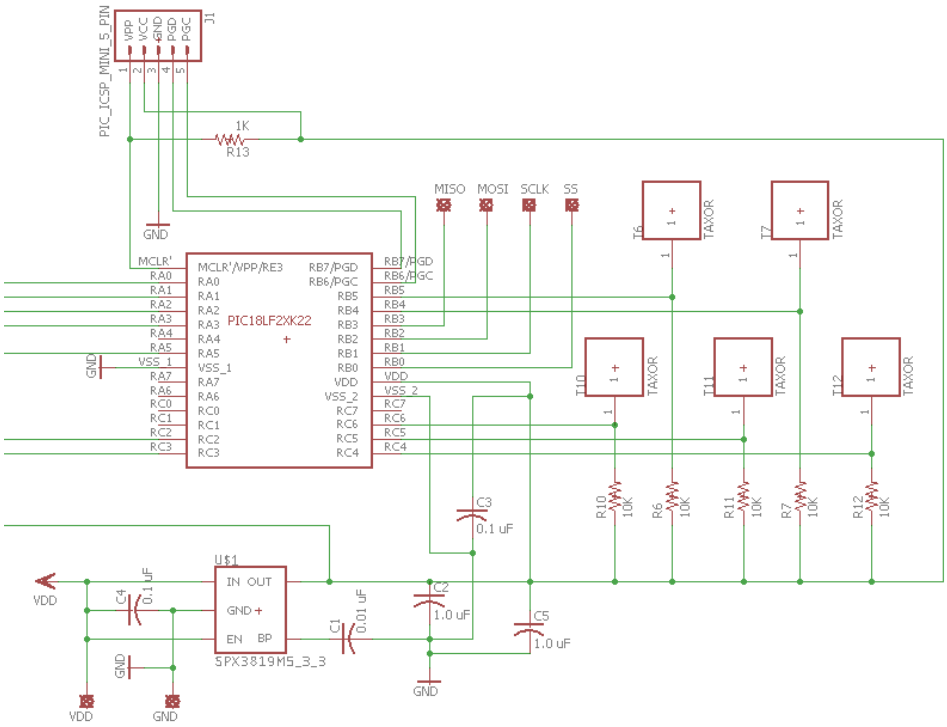

Figure 7: Circuit schematic for the pressure sensors and sampling MCU.

Overall, the sampling MCU circuit design is fairly standard. Tactile cells T6-T7 and T10-T12 are located on the right side of the schematic. The other seven cells are connected via the trailing wires on the left; they were omitted from this schematic in order to preserve readability. Each of the pins connected to a sensor is configured as an input ADC pin, while the pins that are broken out into an SPI bus are configured accordingly for SPI slave operation. Power is delivered through Psyonic's power supply and stepped down to 3.3 V using a linear regulator. Filtering capacitors are placed between each V_{dd}-ground pair. These capacitors exist to filter out high-frequency noise from the power signal; if left unchecked, such noise could damage the IC and impact its operation. Finally, pins on the PIC microcontroller are broken out to a five-pin connector to enable writing to the microcontroller's main flash (i.e. programming it).

As we did for the pressure sensors, we will analyze the sampling MCU circuit in terms of power consumption. First, the PIC microcontroller is rated at a maximum power dissipation of 1 W [7]. Next, resistor R13 will dissipate $\frac{(3.3\ V)^2}{1\text{K}\Omega} = 0.01089\ W$ of power, and the capacitors will dissipate no power. Finally, the SPX3819 linear regulator is rated at up to 500 mA current load [9]. Therefore, assuming that it receives a 5 V input and steps it down to 3.3 V, it will dissipate (5 V-3.3 V)*500 mA=0.85 W of power. Overall, under the described assumptions, the combined power dissipated by all five pressure boards (not counting the sensors themselves, which were handled in the previous section) can be written as follows:

$$P_{consumed} = 5 * (0.01089 + 1 + 0.85) = 9.30445\ W$$

While this isn't an insignificant amount of power, it should be suitable for a wearable device given that the enclosure successfully meets requirements.

2.1.3 Control Board

The Control MCU acts as the central control unit for the system. This board reads in voltage data from pressure sensor boards, normalizes the data to {0, 1}, and runs the classification algorithm. The algorithm has two phases, training phase and prediction phase. In training phase, the user is directed to do a series of pre-defined actions that populates the machine learning model. In prediction phase, the prediction goes to the movement class with the highest prediction confidence with respect to each binary classification boundary. Additionally, the classification result must be translated and transmitted via serial data line so that it meets the requirement of the existing interface of the Psyonic robotic hand.

The unit that we have chosen, STM32F401RBT6, satisfies the clock speed requirement, data transmission requirement, and memory requirements [10]. The Control MCU will first read pre-processed pressure sensor data via SPI interface from the registers of each modulation MCU. The MCU has 64KB SRAM, which is more than enough to store the intermediate values required by the ML algorithm. The schematic of the microcontroller breakout is shown in Figure 8. We based the breakout on the functionality we needed to perform. The microcontroller communicates to five (5) pressure boards over SPI, which requires five dedicated general purpose I/O pins. We also have a USB-serial port that we can use to communicate with the chip, which enable us to have a terminal based user interface. We program the chip using a dedicated programmer, ST-Link V2, on SWCLK and SWDIO pins.

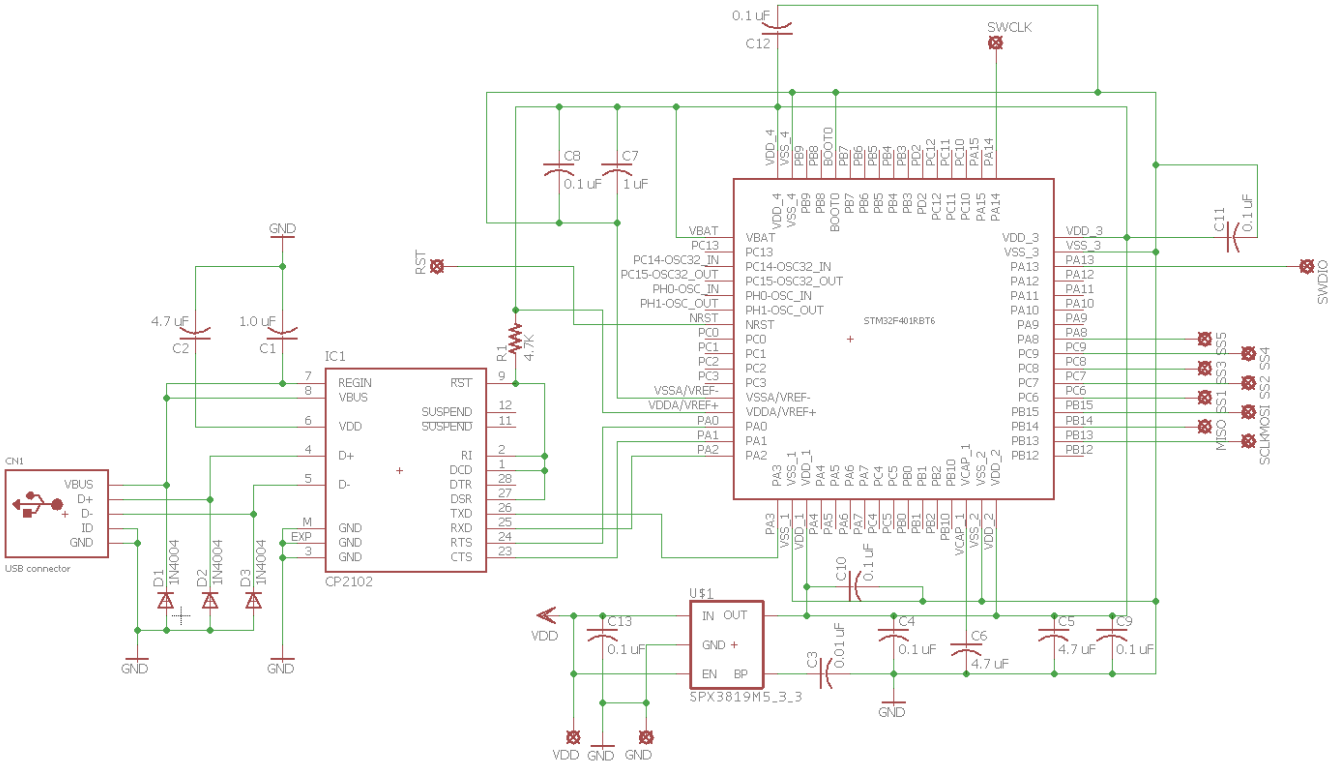

Figure 8: Control board schematic

2.2 Software Component

The primary role of the software component is polling the sampled sensor data from modulation module via SPI communication, and predict the user's movement intention by a machine learning classification model with per-person based training. Figure 9 shows the flow of the software module. The user inputs a mode selection byte at system start to choose to use the pre-trained model for prediction or enter training mode using real-time data collection. The input byte was sent to the microcontroller via USB-serial interface. The prediction result will be sent via the same USB-serial interface to and displayed on the terminal of the connected computer.

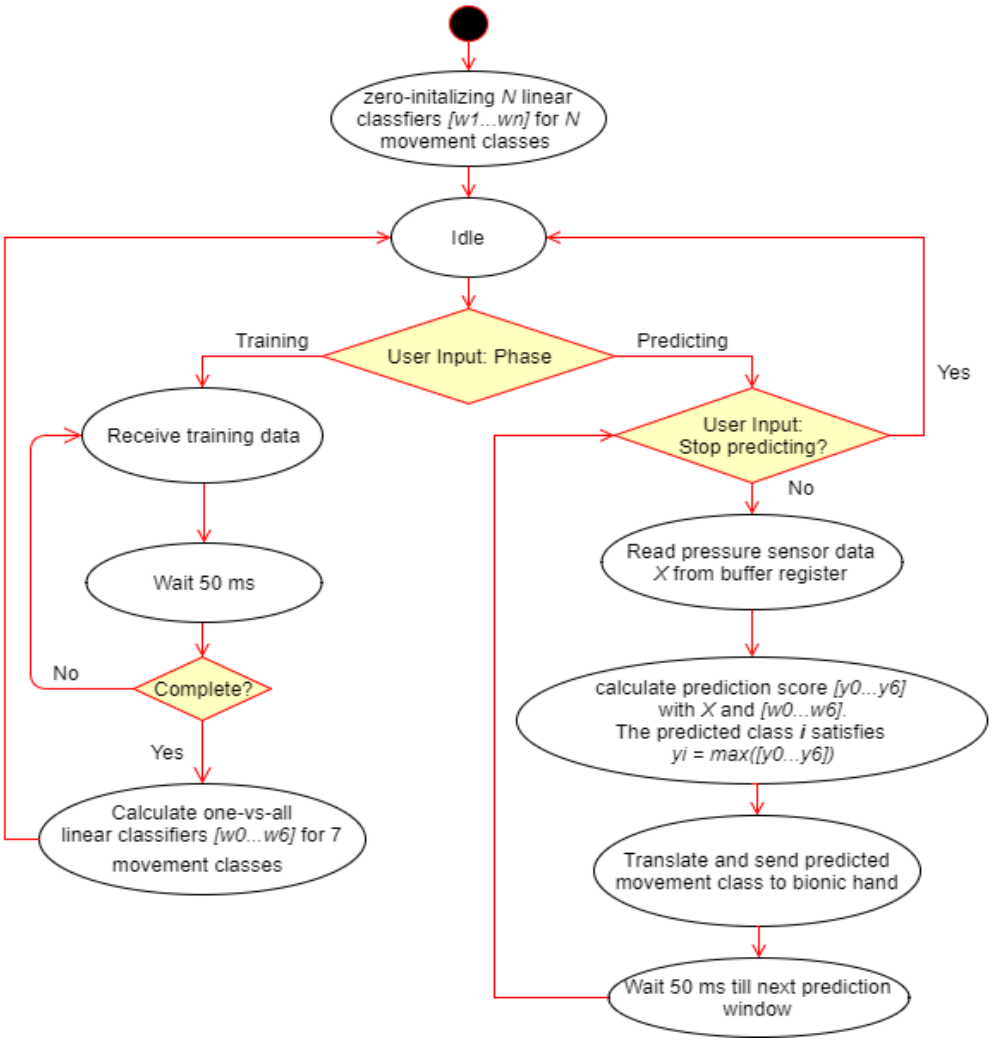

Figure 9: Software flowchart

2.2.1 Classification Model

For this project, we follow the definition of 6 movement classes described in Koiva et al 2015 [5], including three finger movement classes – index finger flexion, little finger flexion and thumb rotation; three wrist movement classes – wrist supination, extension and flexion. We also introduce a non-activation movement class where no intended movements are performed. Figure 10 shows the target gesture for each of the seven movement classes.

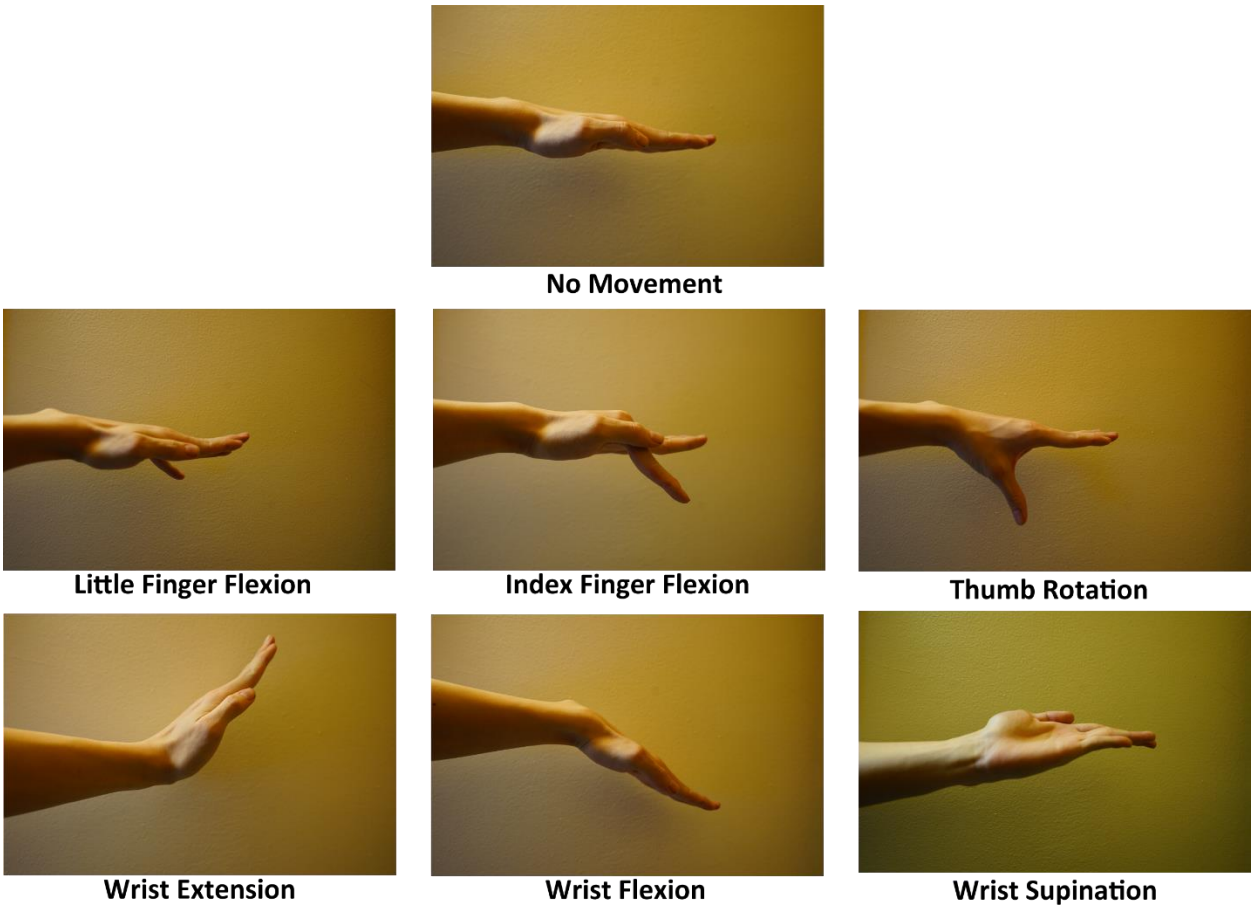

Figure 10: Demonstration of Seven Movement Classes

We implemented our classification model with seven one-vs-all L1-Loss soft margin support vector machine as. The objective function is shown below,

$$argmin_w \frac{1}{2}||w||^2 + C\sum_{k=1}^{m} \max\{0, 1 - w \cdot x_k \cdot y_k\}$$

where $\{x_k, y_k\}$ $\forall k \in 1..m$ is the training set. Each x_k has 60 features -- each feature corresponds to one of tactile cell on our pressure sensor board. $y_k \in \{0 \dots 6\}$ is the movement class label. C is the penalty parameter, allowing outliers to be at the opposite side of support vector margin at the cost of increasing hinge loss error scaled by C.

We train each individual SVM using a variant of stochastic gradient descent described in Panagiotakopoulos et al. 2013 [11], which enforces the bound on L2-norm of w by relaxing decision margin at each training time step. At each time step, one training sample is randomly selected from our training set. A binary label $y_{k,c}{}'$ is produced for each of the seven classifiers SV_c of movement class $c \in \{0 \dots 6\}$, such that $y'_{k,c} = 1$ only when $y_k = c$, and $y'_{k,c} = 0$ otherwise. At each time step, the weight vector of each of the seven classifiers was updated exactly once.

3. Design Verification

3.1 Hardware Component

3.1.1 Pressure Board

The pressure board verification was centered on ensuring three things: that the pressure sensors were able to output deterministic voltages based on the force applied, that the sampling microcontroller was able to correctly read the pressure sensor outputs, and that the sampling microcontroller was able to correctly send the read data to the control microcontroller. These ideas are greatly expanded upon in the first six requirements of the R&V table given in Appendix A, which describe the main goals for this part of the project. Overall, we were able to successfully verify five of the six requirements. Three of the successful requirements (pressure sensor spatial resolution, pressure sensor accuracy, ADC accuracy) were verified by collecting data, while the remaining two (ADC acquisition time, PIC clock frequency) were verified by timing program executions. The data collected during the verification process are presented in graph form in Appendix B. Unfortunately, we were never able to successfully verify the SPI requirement- this is because, due to synchronization issues between master and slave, we were unable to establish correct SPI communication. After considerable debugging, we hypothesize that the large difference between the PIC's clock speed and the SPI bus's clock speed led to the slave writing to and reading from the bus out of turn. Since the PIC utilizes a hardware SPI module, there is likely no way to fix this issue in code. Instead, in the future, we will develop synchronization logic to smooth the edges between the two clock domains, thus enabling synchronous communication.

3.1.2 Control Board

The control board verification is relatively straightforward. We need to ensure that it can function as expected in our design. The detailed requirements are referenced in Appendix A, and we successfully verified all the requirements. Although the control board does not have many quantitative verifications, it is a solder intensive task to build the board. One component, in particular, was very challenging to solder on. The component in question, CP2012 USB com chip, was a highly integrated and small package that had its connection pins on the underside of the chip instead of having extended legs. Thus, it is not possible to use the conventional soldering technique. The assembly of the chip required the "hot air soldering" technique, in which solder is applied on all the soldering pads before blowing hot air in the area and achieve all the connections at once under a microscope. Although a complicated build process, it was easy to verify its functionality – the computer recognized the device when we connected an USB cable.

3.2 Software Component

3.2.1 Classification Model

We first justify our choice of classification algorithm by verifying the linear separability on a simulated dataset. We provide a brief definition of linear separability here. Two sets of points $A = \{a_i \in R^n\}, B = \{b_i \in R^n\}$ in Euclidean space R^n is linear separable if there exists a Euclidean surface W, such that all points in A are in one side of the plane while all points in B are on the other side. Linear separability is a

concept in Euclidean geometry, which is widely used in machine learning as an empirical way to decide whether linear classifier (in Euclidean space) would suffice to classify the data.

We follow the description of collected data in Koiva et al, 2015 [5] to generate the mock data. Although the paper didn't provide statistics on the data distribution, we follow the description of characteristic muscle groups involved in each movement class, and generate set of training & testing data using method described below.

Each training/testing sample is a $4 * 15$ grayscale image, each pixel having value in $[0, 1]$, between the lowest (black) force detected and highest (white) force detected. For each movement class, we first generate $Uniform(0.1,0.2)$ background noise at each pixel. For the three finger movement classes, Koiva et al. mentioned that because of the similar flexor muscle usage in performing the three gestures, similar patterns are observed in the force image [x], but the force intensity levels are different. Similar observations were made for the three wrist movement classes. For each class, we uniformly sample a random point within radius of 5 pixels of a fixed muscle location for three finger movement classes and another location for three wrist movement classes. We then apply a gaussian filter of pixel radius 3, standard deviation of 1 pixel, and peak intensity sampled from $Uniform(0.25,0.65)$. Next, we random sample 10% of the pixels, and assign $Uniform(0,1)$ noise to them. For each class, we generated 10 training samples and 100 testing samples. Figure 11 shows an example pattern for each of the seven classes.

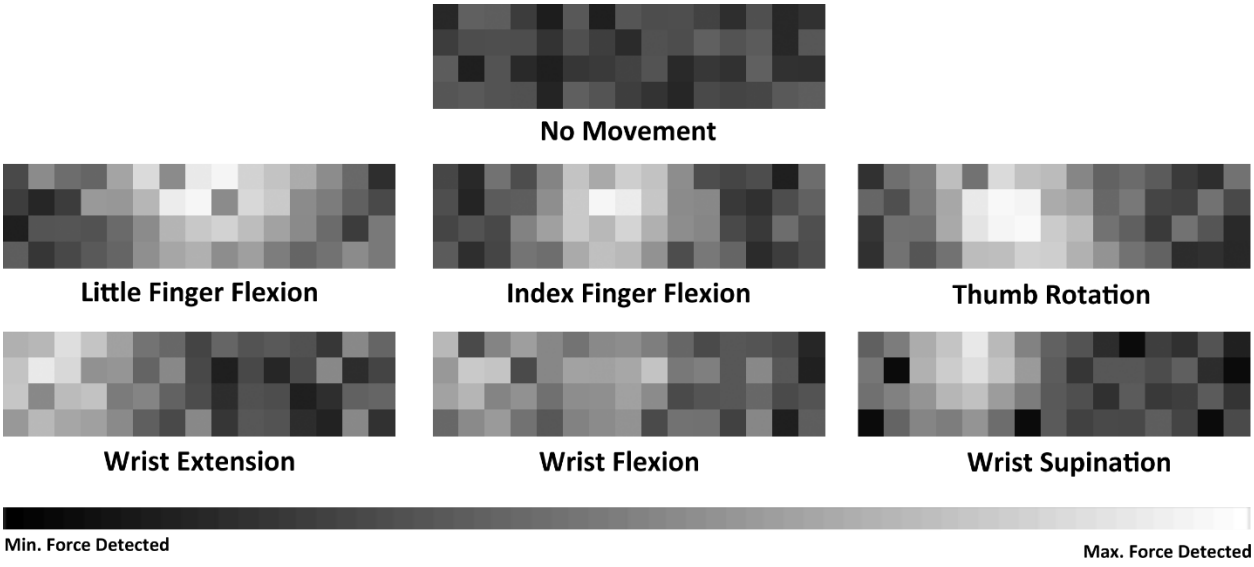

Figure 11: Example Simulated Sensor Data of Seven Movement Classes

We use the $tSNE$ technique [12] to reduce the dimensionality original data to two dimensions for visualization. The Euclidean distance between two data points in the projected space has a positive correlation to the Euclidean distance in the original high dimension space. Figure 12 and Figure 13 below show the transformed result of our simulated training and testing dataset.

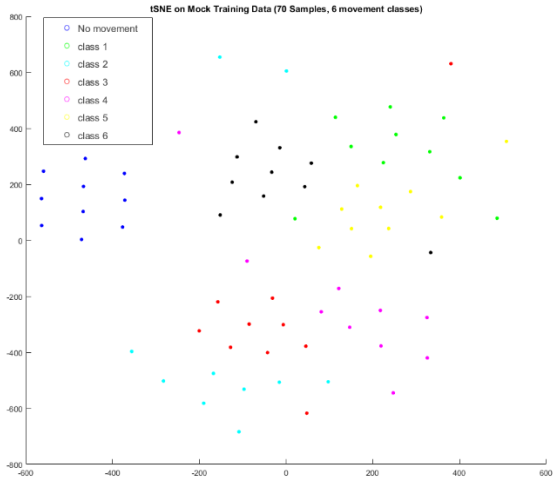

Figure 12: Visualization of Simulated Training Set

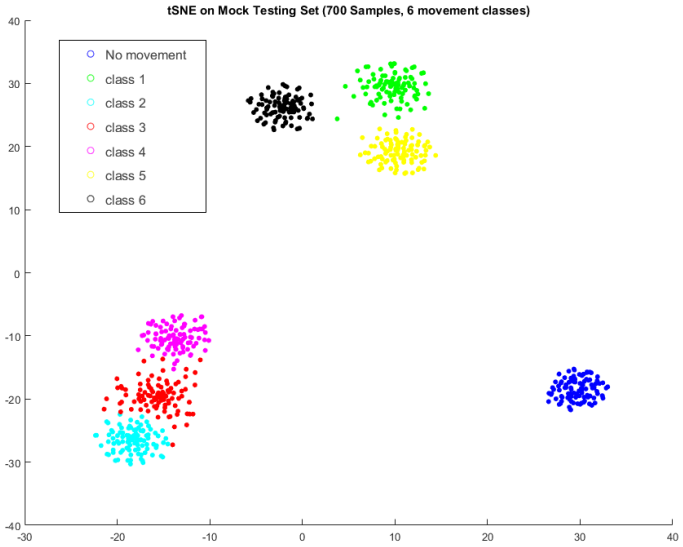

Figure 13: Visualization of Simulated Testing Set

Although we only observe a weak clustering pattern in Figure 12 due to limited number of samples present in the training set, Figure 13 shows a strong clustering pattern for each movement class. Furthermore, three finger movement classes (lower-left) showed strong distinction from three wrist movement classes (top). And the "no movement" class is further away from both groups, as expected. We can visually confirm that there exists a line that could roughly divide every two classes. This tells us that in the original high dimensional space, there is at least a decision surface of equivalent order that could divide our data. Given a decision surface $wx^T + b = 0$, and a transformation matrix H, such that $y = Hx^T$ is the $tSNE$ projection to 2-D space of x.

$$yH^{-1} = x,$$

$$yH^{-1}w^T + b = 0$$

Let $w' = H^{-1}w^T$. Then w' is the decision surface of y. Although the $tSNE$ transformation is non-linear, but since the data is linear separable in 2D space, $w = Hw'^T$ has could at least express a linear surface in original dimension, thus verifying our claim that the data is linear separable.

For completeness, we verify the linear separability quantitatively by training a hard margin SVM on our training set of 70 samples. The result is presented in Table 1 below.

Table 1: Hard SVM training result on training set

Precision	*1.0*
Recall	*1.0*
F1	*1.0*

For user-friendliness, we require the running time for classification model training to be less or equal to 30 seconds. Figure 14 below shows a screenshot from one of our training session on the control MCU.

```
--              Current Parameters              --
Objective:                                      L-1 Loss C-SVC
C:                                              10.000000
Iterations:                                     7000
Training Elasped Time:                          6.162500 s
Training Sample Size:                           70
Training Accuracy:                              1.000000
Test Sample Size:                               700
Test Accuracy:                                  0.855714
```

Figure 14: Screenshot of software terminal output

Next, we present the evaluation result on testing dataset of 700 samples.

Table 2: Evaluation Result on Testing Set

Class	No Move	Wrist Flex.	Little Flex.	Index Flex.	Thumb Rot.	Wrist Ext.	Wrist Sup.
No Move	100	0	0	0	0	3	0
Wrist Flex.	0	94	0	0	0	6	0
Little Flex.	0	0	94	7	0	2	0
Index Flex.	0	0	5	80	15	1	0
Thumb Rot.	0	0	0	11	85	2	3
Wrist Ext.	0	3	1	2	0	85	0
Wrist Sup.	0	3	0	0	0	1	97

Table 3: Evaluation Result on Testing Set

Precision	*0.9944*
Recall	*0.8961*
F1	*0.9427*

4. Costs

As is the case for any significant undertaking, our project has costs associated with its development. Our expenses can be divided into two categories: the cost of obtaining parts and the cost of labor.

4.1 Parts

Our parts costs are shown in Table 4 below. All cost columns with the exception of "bulk purchase cost" already account for the quantity ordered, meaning that the cost values shouldn't be multiplied by quantity. In contrast, the "bulk purchase cost" refers to price per unit when ordered in bulk. The "retail cost" is the listed price of the given quantity of items, while the "actual cost" is what we actually paid for the items, accounting for shipping costs. Since we generally purchased multiple items per order, we were able to divide the shipping costs between them. Overall, the cost was significantly higher than originally anticipated in the design document; this is mostly due to the fact that we had to place multiple PCBway board orders due to bugs in the initial board. Also, it should be noted that this does not include the cost of parts that were available in the senior design laboratory for no charge, such as soldering irons, power supplies, and the PICKit3 debugger.

Table 4 : Parts Costs

Part	Quantity	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
Pressure Board PCB, original	10	PCBway	5.00	0.2445	16.50
Control Board PCB, original	5	PCBway	5.00	0.3436	16.50
Pressure Board PCB, rev. 1	10	PCBway	5.00	0.2445	28.00
Pressure Board PCB, rev. 2	10	PCBway	39.00	0.2445	61.00
Modulation MCU/ADC (PIC18LF2XK22)	5	Microchip	8.10	0.89	8.10
Control MCU (STM32F401RBT6)	1	Mouser	5.33	2.68	5.33
10KΩ 0402 SMD Resistor, 5% Tolerance	120	Bourns Inc.	1.25	0.00334	3.08
1KΩ 0402 SMD Resistor, 5% Tolerance	10	Bourns Inc.	0.26	0.00341	2.09
0.1 μF 0402 ceramic capacitor	34	KEMET	1.05	0.00640	2.88
10 nF 0402 ceramic capacitor	10	Murata Electronics	0.11	0.00233	1.94
1 μF 0603 ceramic capacitor	22	Samsung Electro-Mechanics	0.90	0.01029	2.73
3.3 V, 500 mA linear regulator (SPX3819M5)	10	Exar Corporation	6.31	0.28832	8.14

4.7KΩ resistor (through-hole)	10	Stackpole Electronics	0.40	0.00729	2.23
4.7 μF 0603 ceramic capacitor	10	Yageo	0.90	0.02261	2.73
USB-UART bridge (CP2102)	3	Silicon Labs	4.20	1.22501	6.03
Micro USB connector	3	Amphenol	1.38	0.26196	3.21
1N4004 diode	6	Micro Commercial Co.	0.66	0.01930	2.49
Total					**172.98**

4.2 Labor

Throughout the 16-week semester, each of us worked an average of approximately 12 hours per week, with more downtime in some weeks than others. Based on this and assuming a pay rate of $30 per hour, the labor costs for each teammate can be calculated using the following formula:

$$C_{labor} = \frac{\$30}{hour} * \frac{12\ hours}{week} * 16\ weeks * 2.5 = \$14400$$

Thus, since there are three teammates, the total labor cost for the project amounts to $43,200.

5. Conclusion

Our project was overall successful. We verified all of our individual requirements, and parts are integrated except a timing issue in the SPI communication between the control board and the pressure board. If given more time to implement the project, this bug can be easily fixed. Our user interface is easy to use and intuitive. Furthermore, our machine learning algorithm performed better than expected in both higher-than-expectation testing accuracy and significant reduced training time.

5.1 Ethical considerations

5.1.1 Wearable Medical Device Concerns

In designing and building our system, we must remain cognizant of potential ethical issues surrounding the process. Since our system is meant to be used in a wearable medical device, we must be especially attuned to the first tenet of the IEEE Code of Ethics: "to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment" [13]. Due to the importance of the arms in daily life, we are ethically obligated to do everything in our power to ensure that our design is effective, safe, and comfortable for the user. We are also ethically motivated to keep parts costs as low as possible, so that the design will be accessible to those in need regardless of economic status. Thankfully, since we're integrating with Psyonic's existing design, they will be able to help verify that our design meets these requirements. Finally, it should be noted that, unlike costlier bionic limbs such as i-limb [14], our design will not feature bluetooth connectivity or an associated mobile app. This means that we will not have to contend with computer security concerns and related ethical dilemmas, as hackers will be unable to modify the operation of the arm without direct access to the hardware.

5.1.2 Battery Concerns

Lithium-ion based batteries are volatile by nature, and they can catch on fire or even explode if not charged properly or exposed to extreme temperatures. Overheating of the battery can be caused by an internal short circuit due to contaminants introduced in manufacturing [15]. However, these events are fortunately rare- the worst failure rates experienced (i.e. the ones that trigger device recalls) are generally around one in 200,000 [15]. Since we are not creating the power circuits as part of our project, we will work closely with Psyonic to make sure that the existing power circuit conforms to safety standards. Our team and Psyonic will also ensure that the specific batteries used in our prototype have no defects and work properly.

5.2 Future work

Our proof of concept project is successful. However, for the production design, we can replace our lead-oxide t-stop layer on the pressure sensor with a gold etched layer. This will further improve the precision of the pressure sensors, which will in turn increase the classification accuracy. Given the time constraints, we only collected test data from one user; we need to collect more data from amputees to analyze the confidence interval of our classification result and evaluate its performance. Finally, we can develop synchronization logic to fix our SPI bugs and enable real-time data collection.

References

[1] A.l Lauer, K. Longenecker Rust, R. O. Smith. Factors in Assistive Technology Device Abandonment: Replacing “Abandonment” with “Discontinuance”. (2015) [Online]. Available: http://www.r2d2.uwm.edu/atoms/archive/technicalreports/tr-discontinuance.html.

[2] Centers for Disease Control and Prevention. (2017, May). *National Center for Health Statistics: Disability and functioning*. Retrieved from https://www.cdc.gov/nchs/fastats/disability.htm.

[3] Industrial Safety and Hygiene News. (2014, February). *Statistics on hand and arm loss*. Retrieved from http://www.ishn.com/articles/97844-statistics-on-hand-and-arm-loss.

[4] S. Micera, J. Carpaneto, and S. Raspopovic, “Control of hand prostheses using peripheral information,” IEEE Reviews in Biomedical Engineering (R-BME), vol. 3, pp. 48–68, 2010.

[5] Koiva, Risto, et al. “Shape Conformable High Spatial Resolution Tactile Bracelet for Detecting Hand and Wrist Activity.” 2015 IEEE International Conference on Rehabilitation Robotics (ICORR), 2015, doi:10.1109/icorr.2015.7281192.

[6] Psyonic. (n.d.). Technology. Retrieved October 10, 2017, from http://www.psyonic.co/technology/.

[7] *28/40/44-Pin, Low-Power, High-Performance Microcontrollers With XLP Technology*. DS40001412G. Revision G. Microchip Technology Inc. 2016. Retrieved October 4, 2017, from http://ww1.microchip.com/downloads/en/DeviceDoc/40001412G.pdf.

[5] Radmand, Ashkan, et al. “High-Density Force Myography: A Possible Alternative for Upper-Limb Prosthetic Control.” Journal of Rehabilitation Research and Development, vol. 53, no. 4, 2016, pp. 443–456., doi:10.1682/jrrd.2015.03.0041.

[6] Ravindra, Vikram, and Claudio Castellini. “A Comparative Analysis of Three Non-Invasive Human-Machine Interfaces for the Disabled.” Frontiers in Neurorobotics, vol. 8, 2014, doi:10.3389/fnbot.2014.00024.

[7] Weiss, Karsten, and Worn, Heinz. “The working principle of resistive tactile sensor cells.” IEEE International Conference Mechatronics and Automation (ICMA), vol. 1, 2005, pp. 471–476.

[8] *I2C-Bus Specification*. UM10204. Version 6.0. NXP Semiconductors. 2014. Retrieved October 4, 2017, from http://cache.nxp.com/documents/user_manual/UM10204.pdf.

[9] *500 mA Low-Noise LDO Voltage Regulator*. SPX3819. Revision 2.0.0. Mouser Electronics. 2012. Retrieved November 4, 2017, from http://www.mouser.com/ds/2/146/SPX3819_DS_R200_082312-17072.pdf.

[10] *ARM Cortex-M4 32b MCU+FPU, 105 DMIPS, 256KB Flash/64KB RAM, 11 TIMs, 1 ADC, 11 comm. Interfaces*. STM32F401xB. Mouser Electronics. Retrieved October 5th, from http://www.mouser.com/ds/2/389/stm32f401cb-956305.pdf.

[11] Panagiotakopoulos, Constantinos, and Petroula Tsampouka. "The stochastic gradient descent for the primal l1-svm optimization revisited." Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Berlin, Heidelberg, 2013.

[12] Maaten, Laurens van der, and Geoffrey Hinton. "Visualizing data using t-SNE." Journal of Machine Learning Research 9.Nov (2008): 2579-2605.

[13] Institute of Electrical and Electronics Engineers. (n.d.). IEEE code of ethics. Retrieved September 21, 2017, from http://www.ieee.org/about/corporate/governance/p7-8.html.

[14] Touch Bionics. (n.d.). I-limb ultra. Retrieved September 21, 2017, from http://www.touchbionics.com/products/active-prostheses/i-limb-ultra.

[15] Cadex Electronics. (n.d.). Lithium-ion safety concerns. Retrieved September 21, 2017, from http://batteryuniversity.com/learn/archive/lithium_ion_safety_concerns.

Appendix A Requirement and Verification Table

Table 5 : System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
Pressure sensors must provide approximately 5 mm spatial resolution.	A. Set up the voltage divider circuit in Figure 7 for a single tactile cell. B. Apply a constant load to the center of the electrode for this cell. C. Note the voltage output for this case. D. Apply the same constant load to a points 5 mm and 10 mm away from that central point.	Y

	E. Verify that the output voltage is significantly different (i.e., not within the sensor tolerance) from the original. F. Repeat for the other cells to verify that all are working properly.	
Pressure sensors should provide sufficient accuracy such that level changes (0, 0.25F, 0.5F, 0.75F, F where F is full force) can be easily distinguished (i.e. not within 5% of each other).	A. Set up the voltage divider circuit in Figure 7 for a single tactile cell. B. Apply the maximum expected load ("full force) to the center of the electrode for this cell. C. Note the voltage output for this case. D. Repeat steps B and C for the other force levels. E. Verify that the voltages measured for the five different force levels are not within 5% of one another. F. Repeat for the other cells to verify that all are working properly.	Y
The ADC conversion time needs to be less than 4.167 ms (this comes from dividing the time per sampling interval (50 ms) by the number of pressure sensors per board (12)). Note: all five pressure boards sample their inputs in parallel.	According to its datasheet, PIC18LF2XK22 allows the user to set the ADC sampling rate and acquisition time appropriately. A. Set the ADCS bits of the microcontroller's ADCON2 register to 111 to configure the ADC's conversion clock as a dedicated internal oscillator (600 KHz). B. Set the ACQT bits of the microcontroller's ADCON2 register to configure the acquisition time. Try values of 101, 110, and 111 (12TAD, 16TAD, and 20TAD respectively; at least 10TAD is required for 10-bit conversion). C. Select an input channel by setting the CHS bits, and set the GO/DONE' bit. D. The ADC will sample the input for the selected time and automatically begin the conversion. E. Read the result (placed in the ADRES register). F. Repeat the above process for the remaining 11 ADC input channels.	Y

	G. Repeat steps C-F in a loop of 100 iterations. H. Measure the time it took to do this (should take less than 5 seconds).	
The sampling microcontroller needs to have at least 12 functioning input ADC pins.	PIC18LF2XK22 has 17. A. Setup the ADC module as described in the verification procedure above. B. Attach an oscilloscope probe to an ADC input channel and set the oscilloscope to measure average voltage. C. Apply force to the corresponding pressure sensor, noting the converted ADC result. D. Verify that this result is within ADC tolerance (3.3 V/1023) of the average voltage value measured by the oscilloscope. E. Repeat the above process for all 12 in-use input channels. This will verify that they are all functioning as expected.	Y
The sampling microcontroller needs to run at least 1 MHz.	According to its datasheet, PIC18LF2XK22 has a CPU that is rated at up to 16 MIPS, depending on the speed the onboard configurable clock is set to. A. Write a small test program for the microcontroller, noting the total number of instructions that should be executed. B. Load the program into program memory and run it, recording the execution time for each one (for programs with many instructions, it is reasonable to do this without an RTC). C. Verify that the execution time is reasonably consistent with the onboard clock speed.	Y
The sampling microcontroller needs to be able to communicate with the control MCU with a data rate of at least 19.2 Kbps (this is because each of five slave microcontrollers need to send 24 bytes of data to the master in each 50 ms window).	This can be done via the SPI communication protocol- according to its datasheet, PIC18LF2XK22 has two SPI peripherals.	N

	A. Form a data packet of size 24 bytes (12 pressure sensors, each of which provide a 2-byte value). B. Send it to the control MCU over the SPI bus. C. Repeat the above process 1200 times (i.e. use a loop). D. Time the process above and verify that it took less than one minute to complete (one minute indicates a data rate of 19.2 Kbps).	
The control microcontroller must be clocked at at least 20MHz	With a loop in assembly, we can count the cycles it takes to execute the simple program. Using this information, we can calculate how many cycles per second the microcontroller operates	Y
The control microcontroller must be able to handle 32-bit float point calculations	A. Write a test program featuring 32-bit floating point addition, subtraction, multiplication, and division. B. Load the program into the microcontroller's program memory and run it. C. Verify that the output produced is the same as the output of an equivalent program running on a desktop PC.	Y
The controller can support the required data transmission speed (19.2 Kbps) over SPI based on our data structure size calculation.	A. Verify the max frequency is greater than the required speed in the PC based config tool. B. Transmit data over the SPI interface, measure the clock and data line frequency to verify that the bit rate is higher than that of the requirement.	Y
Linear separability: Training Accuracy must be at least 90% using a linear SVM classifier on 10 samples for each movement class.	A. Read 100 pressure sensor array samples for each movement class, randomly take 10% of sample as training set. B. Train one-vs-all hard Support Vector Machine on each movement class C. Evaluate on the same data set obtained in Step A. The accuracy measured must be at least 90%.	Y

The elapsed time between the beginning and the ending of training execution of 10000 iterations on the Control MCU must be under 30 seconds.	A. Generate random mock training set with 10 example patterns for each of the 6 class independently B. For 10000 iterations, randomly choose a sample from training set mentioned in A, and run Stochastic Gradient Descent on the example with our classification model with the classification algorithm we decided to use C. Verify that time used for the training process over 10000 iterations is finished in under 30 seconds	
The algorithm for classification model must be able to achieve at least 70% accuracy on a testing set at least 5 times larger in size than the training set	A. Read 100 pressure sensor array patterns for each movement class on a user B. Randomly sample 10% of the collected patterns from each class as training, the rest 90% of the patterns as evaluation set. C. Use each of the trained classifier to predict on evaluation set in Step B. Verify that the prediction accuracy on the evaluation is over 70%	Y

Appendix B Pressure board verification data

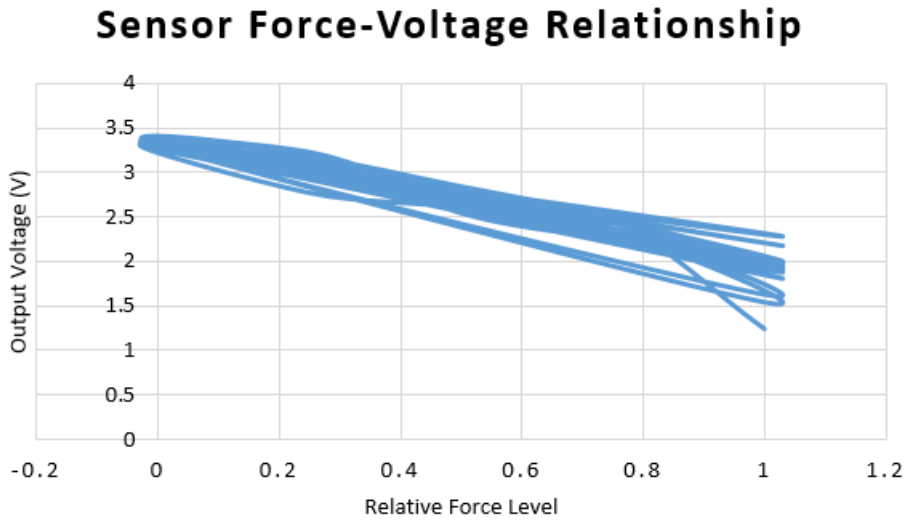

Figure 15: Results verifying the pressure sensor accuracy. We were able to achieve a linear response from the sensor

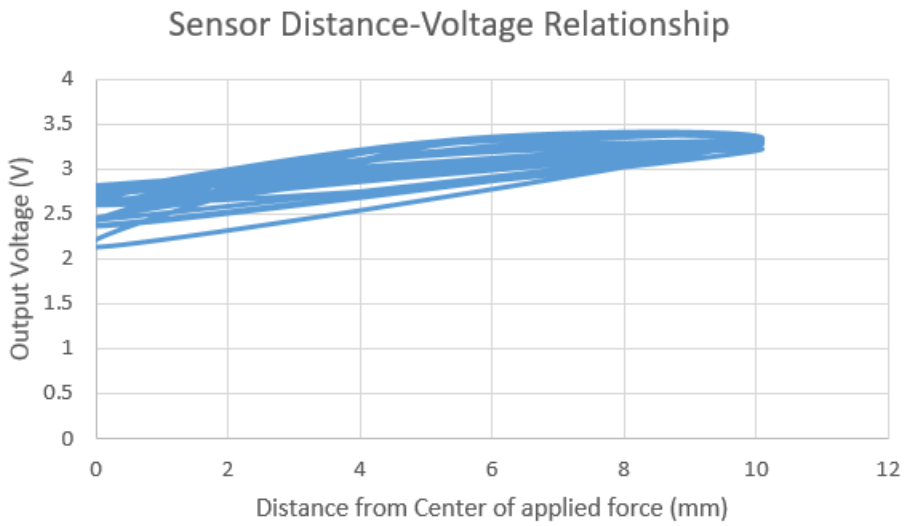

Figure 16: Results verifying the pressure sensor spatial resolution. 5 mm is on the sensor's edge. Resolution is adequate but could be improved with gold etched t-stop and better foam.

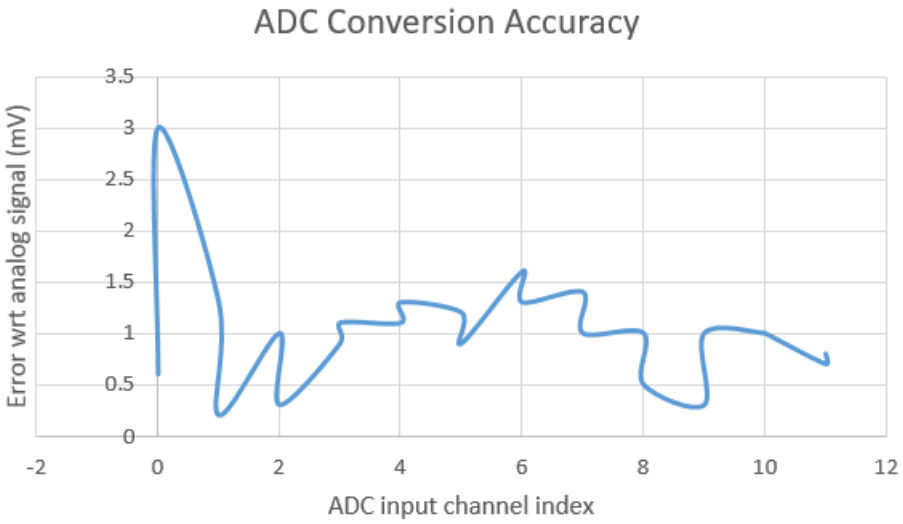

Figure 17: Results verifying the ADC conversion accuracy for all channels. The data was collected by measuring the analog sensor output and the ADC sensor output at the same time. "Error" refers to the absolute value difference between the two values.

Appendix C Board Layouts

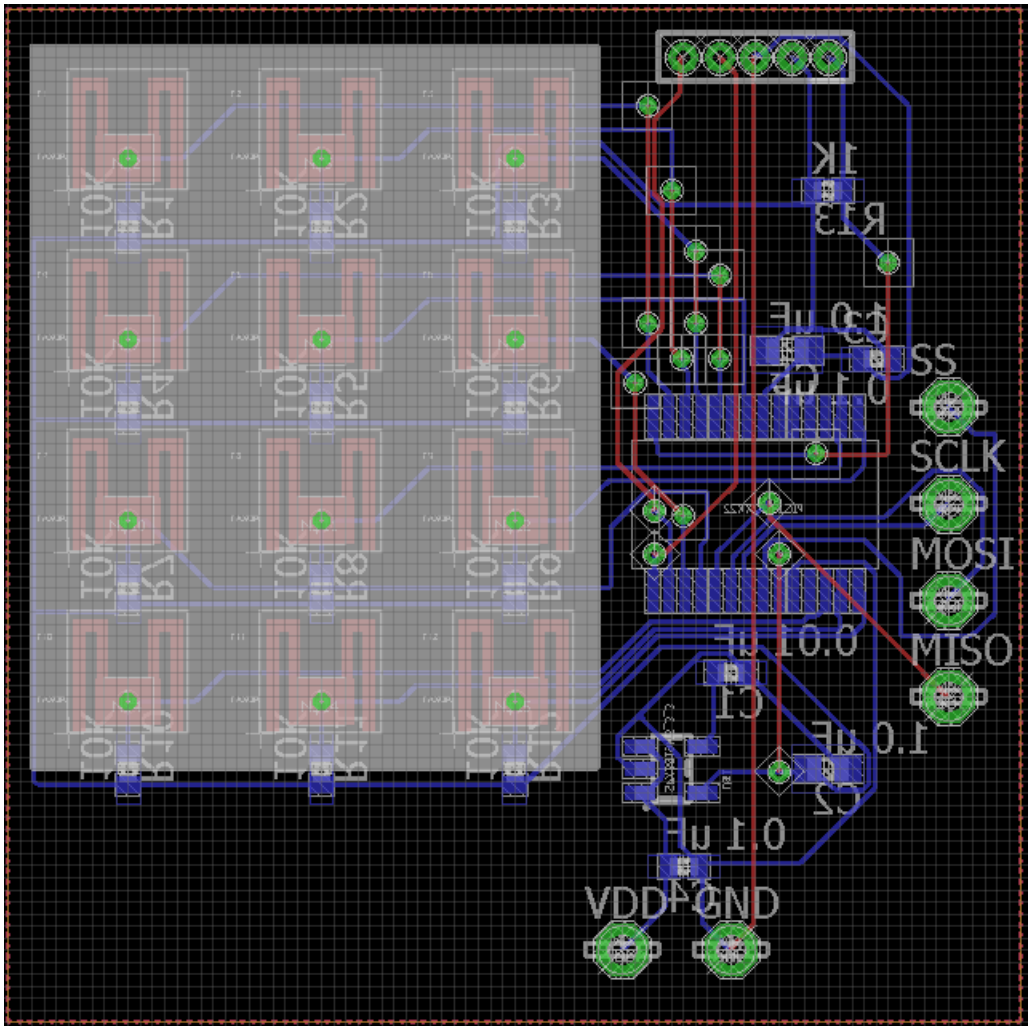

Figure 18: Layout of the pressure board PCB

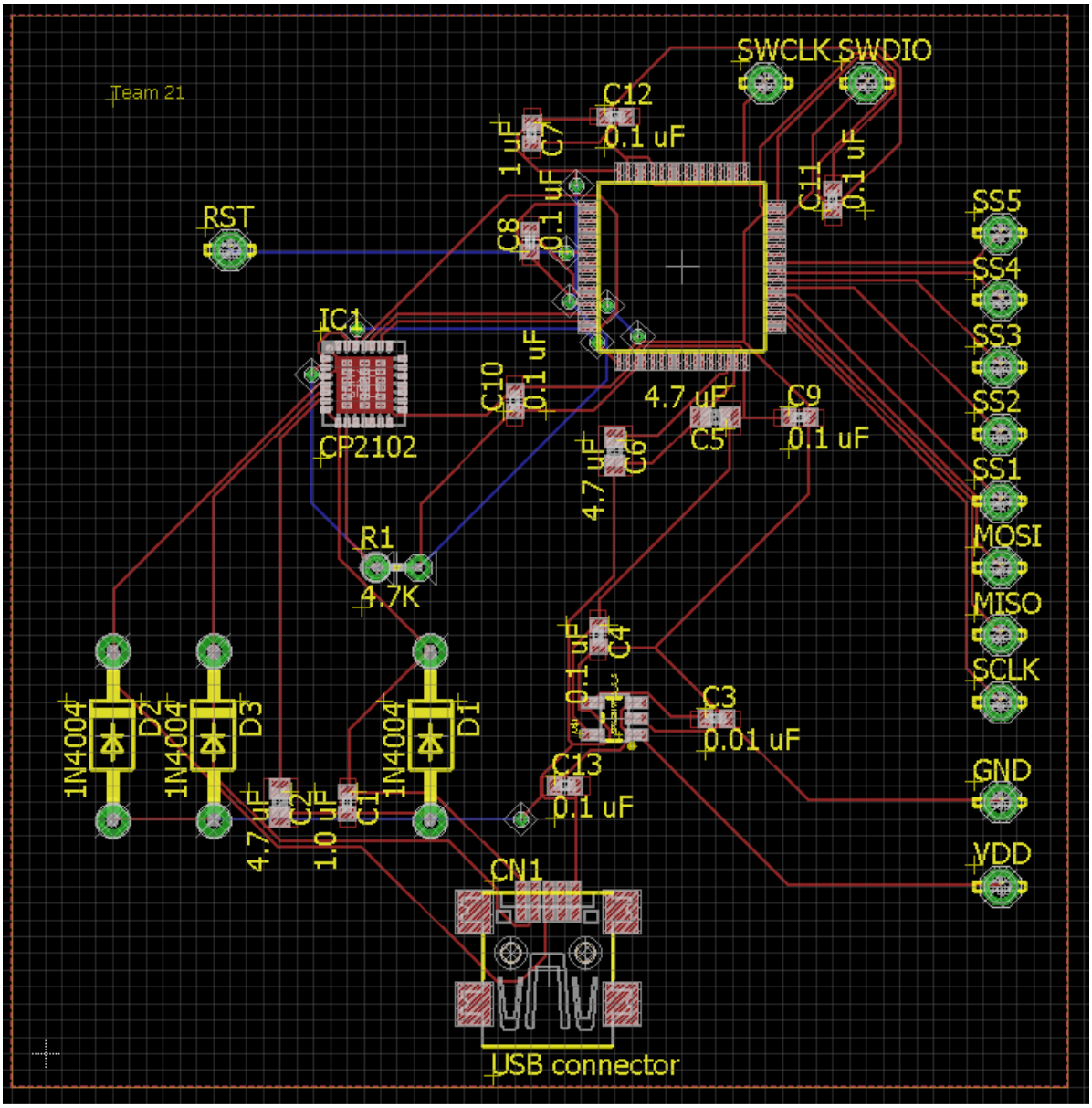

Figure 19: Layout of the control board PCB