FOAM PRESSURE-SENSOR BASED CONTROL METHOD FOR CONTROLLING PROSTHETIC HANDS

By

ENLIANG LI

YANGGE LI

ZHOUSHI ZHU

Final Report for ECE 445, Senior Design, Spring 2019

Supervisor: Professor Michael L. Oelze

TA: Amr M. Martini

30 APR 2019

Project No. 19

Abstract

Nowadays, prosthetic hands are commonly controlled by Electromyographic (EMG) method which evaluates the electrical activity produced by skeletal muscles. However, the traditional EMG method has several drawbacks that limits its performance: the measurements of electrical signal suffer from high level noises induced by human skin and the number of sensors is sometimes insufficient to acquire enough data points to precisely track the muscle movements due to overcomplicated physical layout and high cost of EMG sensors.

In this senior design project, our goal is to develop a foam pressure-sensor based method as an alternative of the EMG method for controlling prosthetic hands provided by PSYONIC Inc., which includes a design of PCB to carry the electrodes array with its corresponding communication peripherals and programing of the communication protocol. On top of the PCB, a layer of electrically conductive foam will be placed on the electrodes to function as the pressure sensor which records the compression of the foam. The pressure sensor method is expected to be less noisy and less expensive while being as accurate as the traditional EMG method. Preliminary research shows promising result to this pressure sensor method.

Contents

1. Introduction

1.1 Statement of Purpose

Nowadays, prosthetic hands are commonly controlled by Electromyographic (EMG) method which evaluates the electrical activity produced by skeletal muscles. However, the traditional EMG method is not accurate enough, because the measurements of the electrical signal suffer from high level noises come from the users' skin. In addition, due to the physical layout and high cost of EMG sensors, the number of sensors is insufficient to acquire enough data to track the muscle movements precisely.

In this senior design project, we are collaborating with the PYONICS Inc., a customized prostheses manufacturing company, to design and prototype an interface platform to control a prosthetic hand based on foam pressure-sensor as an alternative of the controlling system based on EMG currently used by the PYONICS Inc. The project includes:

a. Design of the senor module which carries the electrodes array with its corresponding communication peripherals and the communication master device which processes data from sensor modules.

b. Programing of the communication protocol.

c. Design and programming of hand gesture classification algorithm

d. The soldering and assembly of all the components.

This project is sponsored by the PSYONIC Inc. The pressure sensor method is more accurate, less noisy and cheaper, and preliminary research[1] shows promising results regarding this pressure-sensing method.

1.2 Existing Work & Objective

The aforementioned paper introduces a methodology of controlling prosthetic hand based on pressure sensors located around testers arm. The researchers designed a tactile bracelet composed of 10 sensor boards which was deployed around the test subject's residual limb or forearm, in the case of able-bodied or disabled subjects. The pressure sensors are built with electric conducting foam and electrodes, utilizing the foam's property of changing resistance while compressed by pressure.

The experiment consisted of two phases: training phase and testing phase. During the training phase, the test subjects were asked to try to make the intended hand movement for a period of time while wearing the bracelet. Meanwhile, the sensors on the bracelet collected the pressure distributions caused by muscle movement on subjects' arm to train the classification algorithm which classify the pressure distribution into categories according to different hand movement. During the testing phase, the test subjects were asked to repeat the movements in training phase with the bracelet on and the pressure distribution collected were classified by the pre-trained classification algorithm and transferred. The classification results were compared with the ground truth hand movements used in the training phase and an average inference accuracy was calculate for both disabled subjects and able-bodied subjects. The researchers claimed that the average accuracy is 89.15% for able-bodied and 93.07% for disabled subjects.

The PSYONIC Inc. has a finished product of prosthetic hand based on the EMG method and we integrate the prosthetic hand provided by the company with our prototype. By deploying similar methodology introduced in the paper, we build an interface platform which interacts with the electric conducting foam and prosthetic developed by PSYONIC Inc. The previous research only focused on classification results but we are moving one step forward to utilize the classification results to generate according command to control the prosthetic hand.

The main goal for designing and prototyping such a platform is to prove the efficacy of a new methodology and by reconducting similar experiment mention above, we are expecting to reproduce similar inference accuracy. Therefore, the classification algorithm that we are planning to develop is only expected to classify the pressure distribution into limited groups to achieve predefined simple hand motion control. PSYONIC Inc will potentially apply this new methodology in their products in the future depending the performance of the prototype and they will potentially implement more complex classifiers such neural networks to allow more sophisticated hand motion control.

1.3 High-level requirements

There are two major requirements for our design:

- Our system should be able to sense and process one specific user’s muscle movement and convert it to the command for prosthetic.
- The user should be able to control the prosthetic hand to complete motions including turning hand in four directions and clenching and releasing of the fist.

2. Design

2.1 Block Diagram

The system we designed will consist of a master device and some slave devices. The slave devices will hold the pressure sensors, collect pressure data from user and transmit them to the master device. The master device will collect data from slave devices and communicate with PSYONIC's prosthetic hand.

2.1.1 System Overview

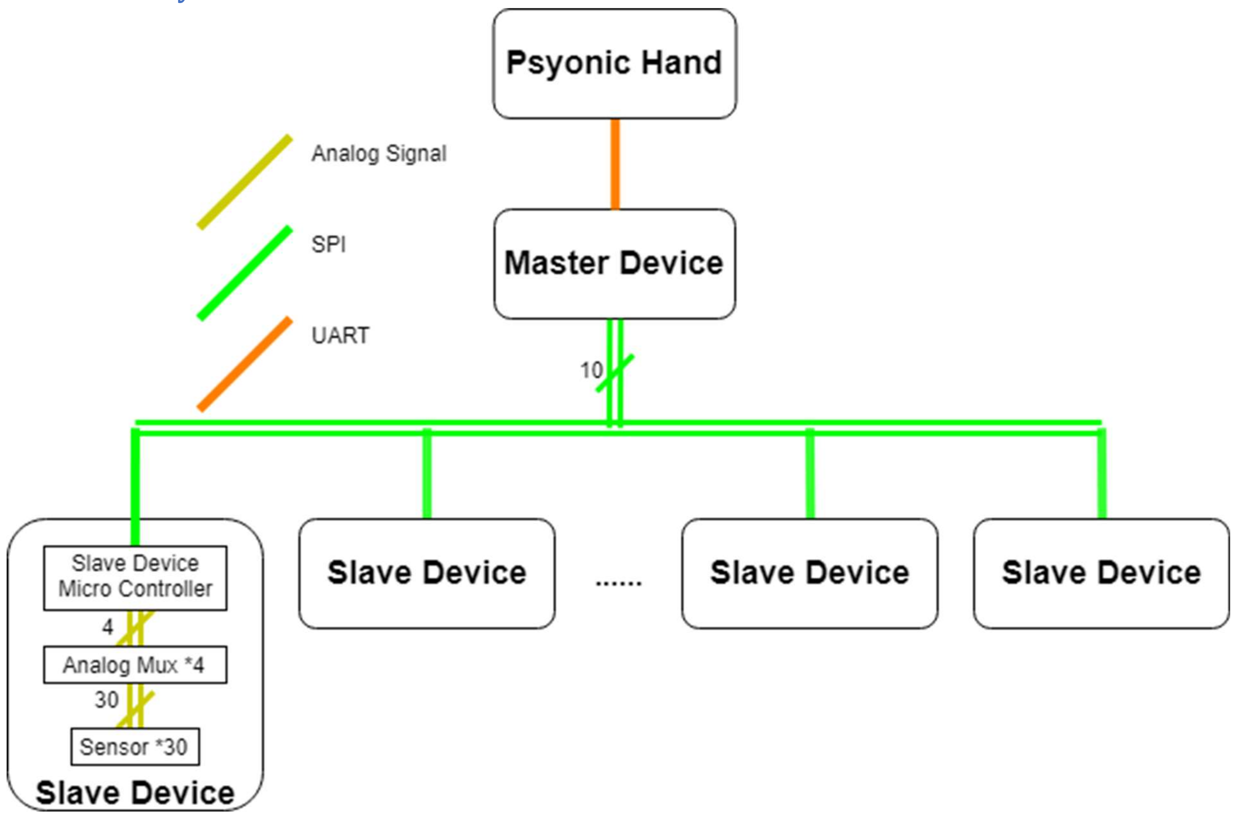


Figure 1: System Overview Block Diagram

2.1.2 Master Device

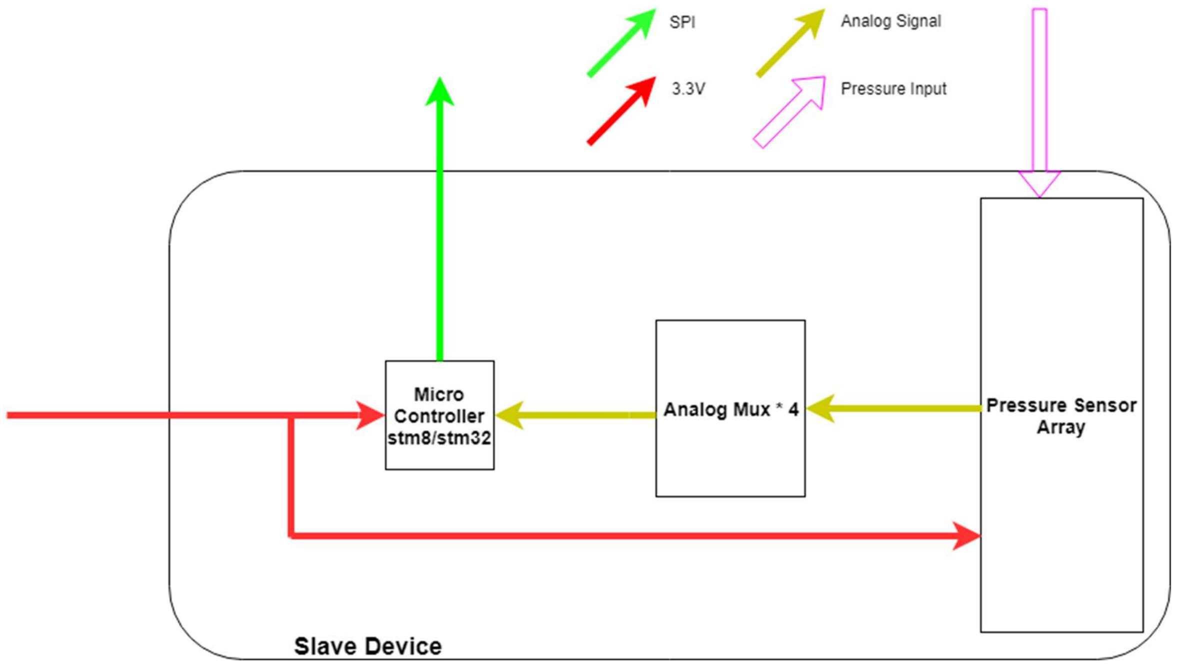


Figure 2: Master Device Block Diagram

2.1.3 Slave Device

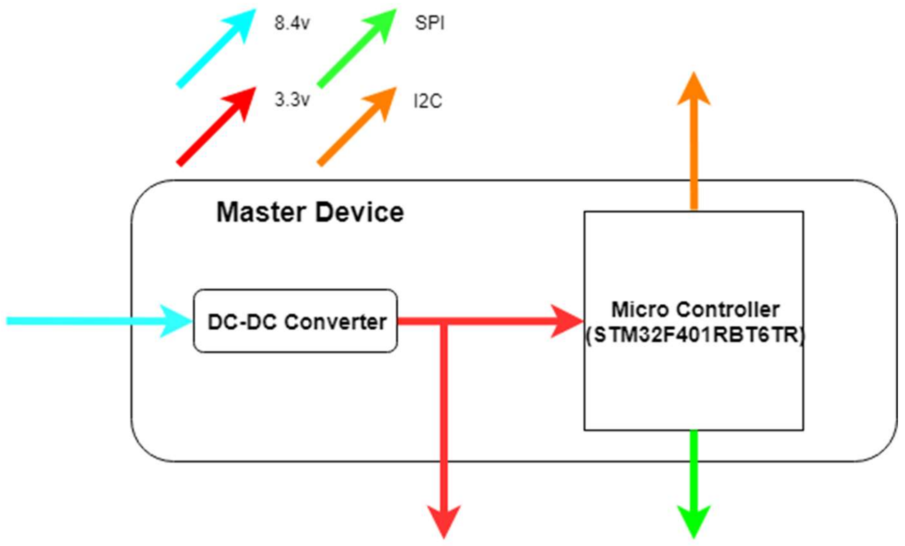


Figure 3: Slave Device Block Diagram

2.2 Physical Design

The final prototype consists of a bracelet holding 4 slave devices and the master device as shown in figure 4. The bracelet size is adjustable to fit testers' arm thickness and provide proper tension.

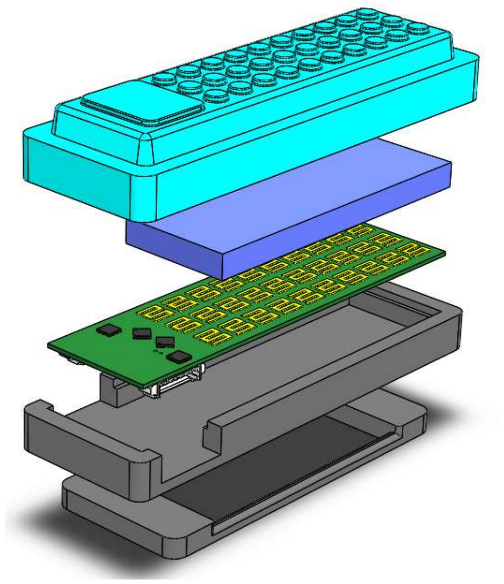


Figure 4: Physical Design of the Slave device

Each slave device PCB is contained in a 3D printed rectangular enclosure. One side of the case has a slot for the bracelet to pass through and the other side of the case is left open. The electrical conducting foam resides on the sensors on the PCB from the open side of the enclosure with a silicone cover around the enclosure to hold all parts together. The master device is attached to the outer surface the bracelet, locating at the closest slave device. The following 3D models portrays the overall physical structure.

2.3 Block Description

2.3.1 Master Device Microcontroller (STM32F401RB)

Input: 1.7V-3.6V power input.
Output: Command for PSYONIC's hand.
Communication: I2C communication with PSYONIC's hand; SPI communication with slave devices.
Description:

The STM32F401RB is using an ARM 32-bit Cortex-M4 CPU with floating-point unit (FPU). It has frequency up to 84 MHz, and contains up to 256 Kbytes of Flash memory, 512 bytes of OTP memory and up to 64 Kbytes of SRAM. [2]

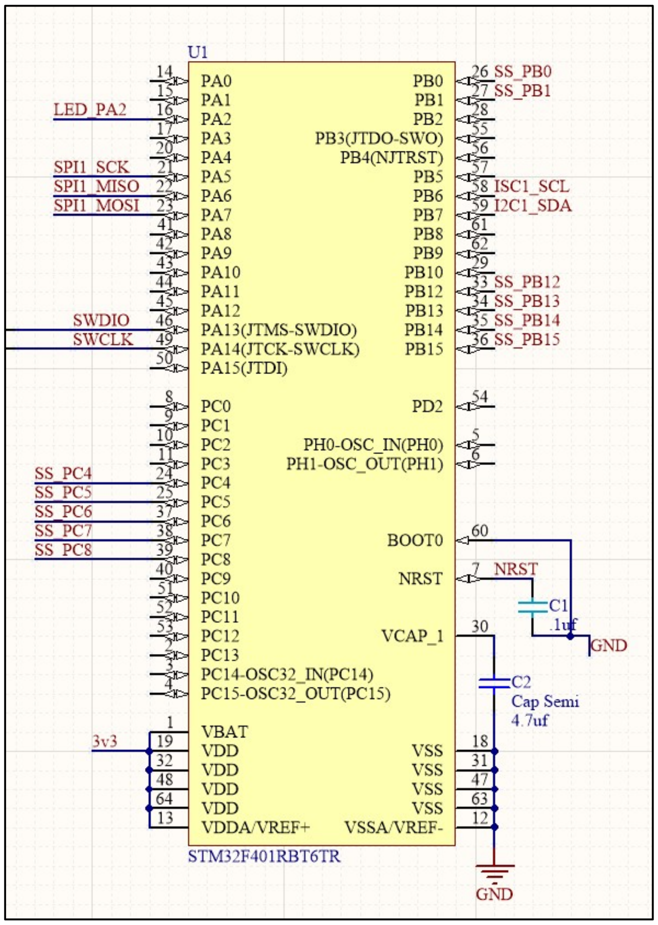


Figure 5: STM32F401RBT6 Schematic

This unit is supposed to collect data from all slave devices through SPI interface (MISO, MOSI, SCK, SS). It will then process the collected data and convert the data to command towards the PSYONIC's hand. The command will be transmitted through I2C interface (SDA and SCL) to the PSYONIC's hand.

This unit will be powered by a 3.3V power input regulated from an 8.4V voltage supplied from the PSYONIC's hand. A pin layout summary of these interfaces is shown in figure 5.

Considering integrating with the system developed by PSYONIC Inc., we choose this specific part depending on the stock of PSYONIC Inc. and the chip is also powerful mathematically.

2.3.2 Slave Device Microcontroller (STM32F030K6)

Input: 2.4V-3.6V power input; Analog signals from MUXes.
Output: Digitized pressure sensor reading.
Communication: SPI communication with master device.
Description:
The STM32F401RB is using an ARM 32-bit Cortex-M0 CPU. It has frequency up to 48 MHz. It contains up to 256 Kbytes of Flash memory and up to 32 Kbytes of SRAM. In addition, the microcontroller also has one 12-bit ADC with up to 16 channels and conversion range 0 to 3.6 V. [3]

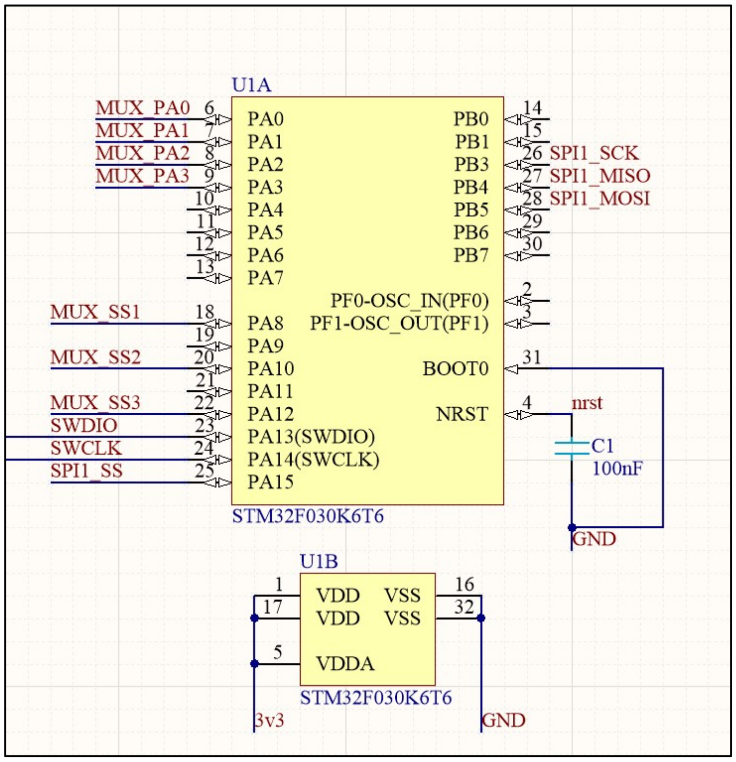


Figure 6: STM32F030K6 Schematic

This unit is supposed to read analog voltage input from the MUXes. It will digitize the input using the built in 12-bit ADC of the microcontroller. The result data will be transmitted to the master device using SPI interface (MISO, MOSI, SCK, SS). In addition, four 8-to-1 analog MUXes are used to choose between different sensors' signal supporting up to 32 pressure sensors on each slave device. The microcontroller will also have three pins connected to all the MUXes to select between signals.

This unit will be powered by the 3.3 V input from the DC-DC Converter on the master device. A pin layout summary of these interfaces is shown in figure 6.

This part is also chosen from the stock of PSYONIC Inc. Since it is not responsible for any mathematical operations, we choose it for its relatively low cost and still moderate processing power.

2.3.3 DC-DC Converter (TPS82140SILR)

Input: 3V-17V power supply, typically 8.4V (from PSYONIC's hand in our design)
Output: 0.9V to 6V Adjustable Output Voltage, in our case configured as 3.3V

Description:
The TPS82140 is a step-down converter MicroSiP™ power module optimized for small solution size and high efficiency. It supports input voltage from 3V to 17V and an adjustable voltage output from 0.9V to 6V. The converter support 2A continuous output current. [4]

This unit will take the 8.4V voltage from the PSYONIC's hand convert it to the 3.3V voltage that power the system we designed. It also supports other input power range from 3V to 17V. The 2A output current is enough to power all the rest of the system and a detailed calculation to justify this statement can be found in part 2.6.2 below. A pin layout summary of these interfaces is shown in figure 6.

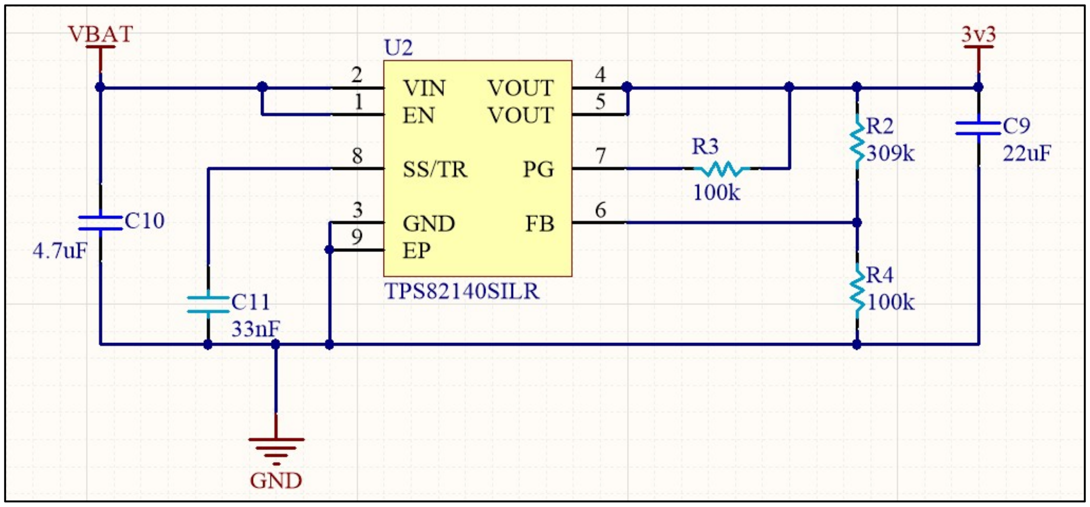


Figure 7: TPS82140SILR Schematic

This particular part is chosen for its large output current because we expect to power up to 10 slave devices and a master device using one regulator.

2.3.4 Analog MUXs (NX3L4051HR,115)

Input: 1.4V-4.3V input power, typically 3.3V; Analog pressure sensor reading, range between 0-3.3V; Three selection signal (S1, S2, S3).
Output: Selected analog pressure sensor reading to the microcontroller.
Description:
The NX3L4051 is a low-ohmic 8-channel analog switch. It supports a 1.4V to 4.3V supply voltage and has a maximum of 900$m\Omega$ on resistor. [5]

Since as we mentioned above, each slave device is going to hold up to 32 pressure sensors, which exceeds the 16 ADC channels we have on the microcontroller. Therefore, four 8-to-1 analog MUX(s) are used to select between different inputs. In case of including 30 pressure sensors on a slave module, three MUX(s) will each handle 8 analog signals and one MUX will handle 6 analog signals. All 4 MUX(s) will share the same selection signal from the microcontroller and connected to different ADC channel on the microcontroller. An example pin layout summary of these interfaces is shown in figure 8.

This particular part is chosen for its QFN package and, therefore, its small size because of

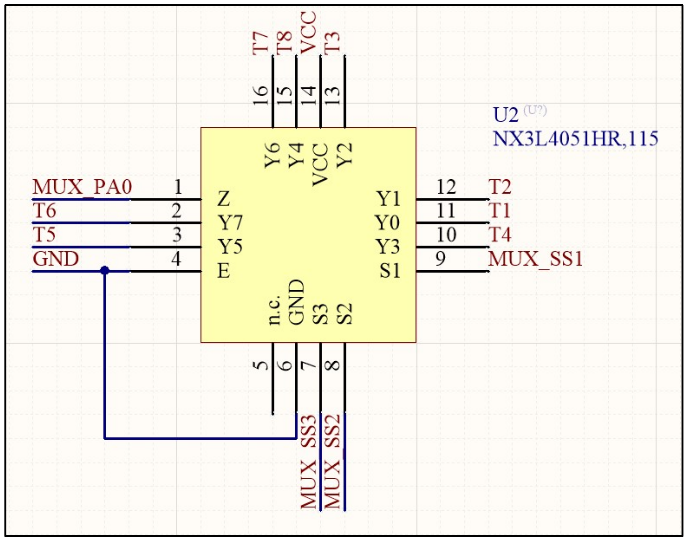


Figure 8: NX3L4051HR Schematic

the limited space on slave device PCB. It's also a good choice due to its low inner resistance.

2.3.5 Pressure Sensor

Input: 3.3V input power; Pressure input from human muscle.
Output: Analog voltage change corresponding to the change of pressure.
Description:
Each pressure sensor is based on a resistive working principle in which the interface resistivity between two surfaces changes according to the applied load. We will use metal trail on PCB as electrodes and use conductive foam as the sensor material. When load is applied the resistance between the electrodes will be changed and we can use the resistance change to sense pressure change. Therefore, we are going to apply the voltage division principle to convert the resistance change to voltage change, which is demonstrated in figure 8. A detailed explanation about the choice of foam and the resistance of R can be found in section 2.6.1 below.
Figure 9 from [1] illustrates the physical design of the pressure sensor.

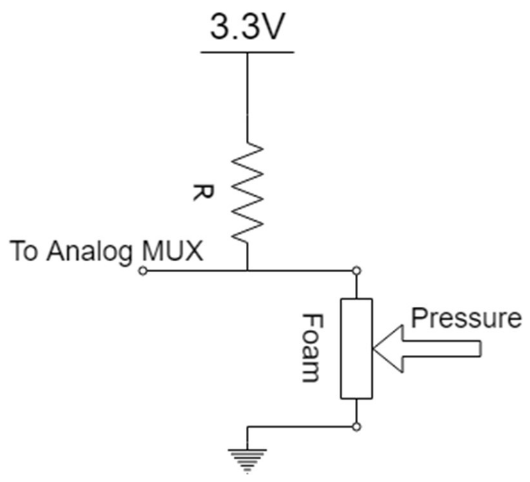


Figure 9: Pressure Sensor Model

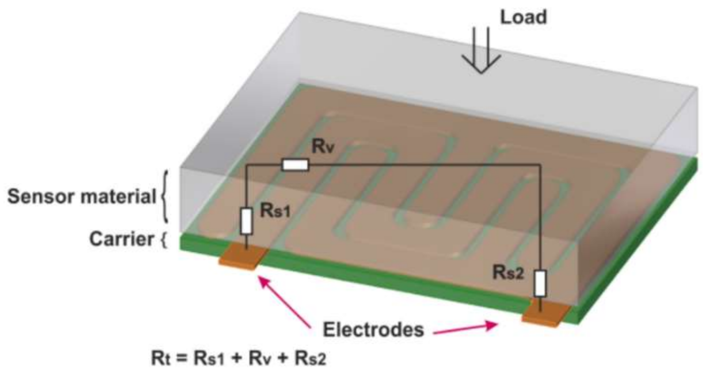
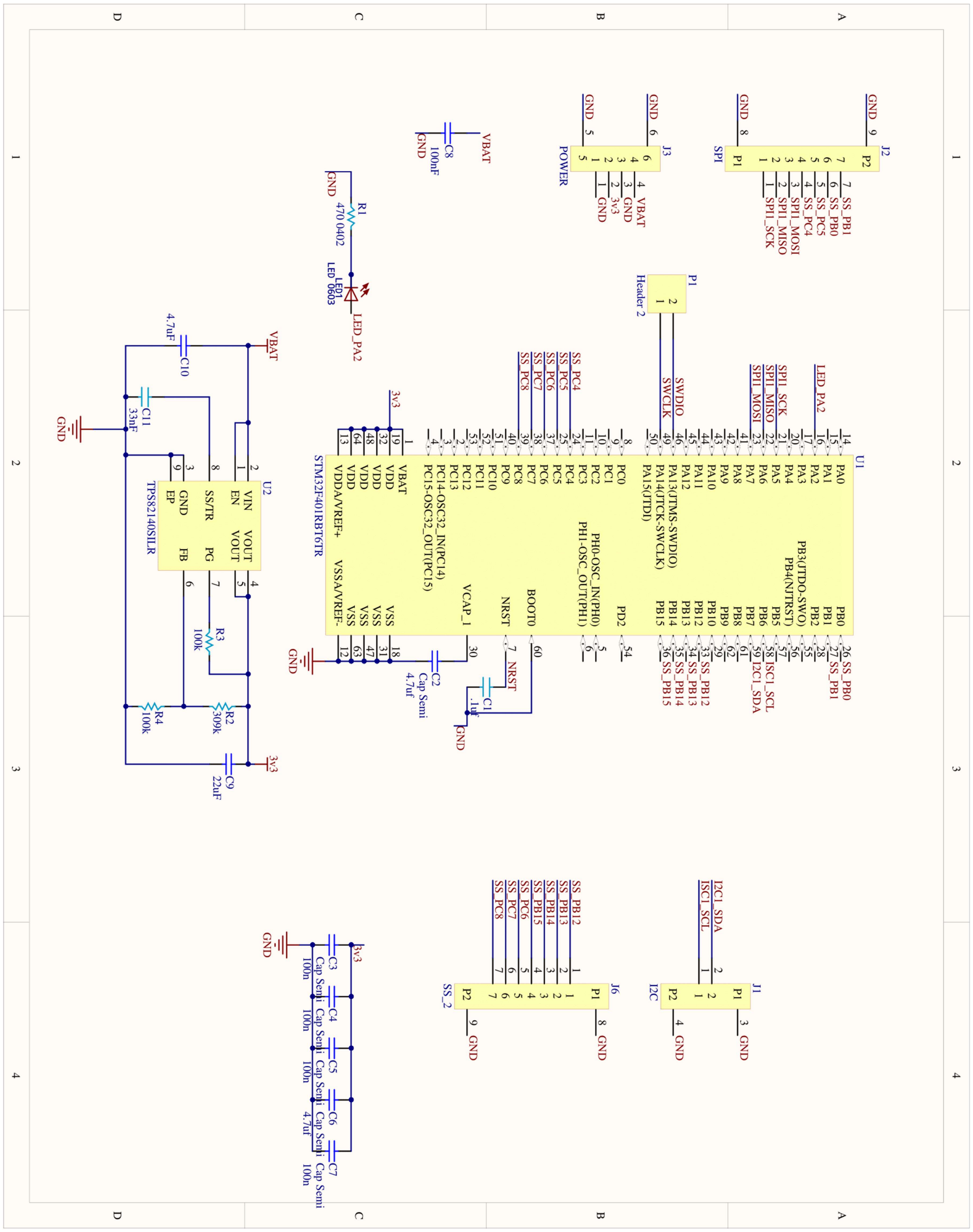


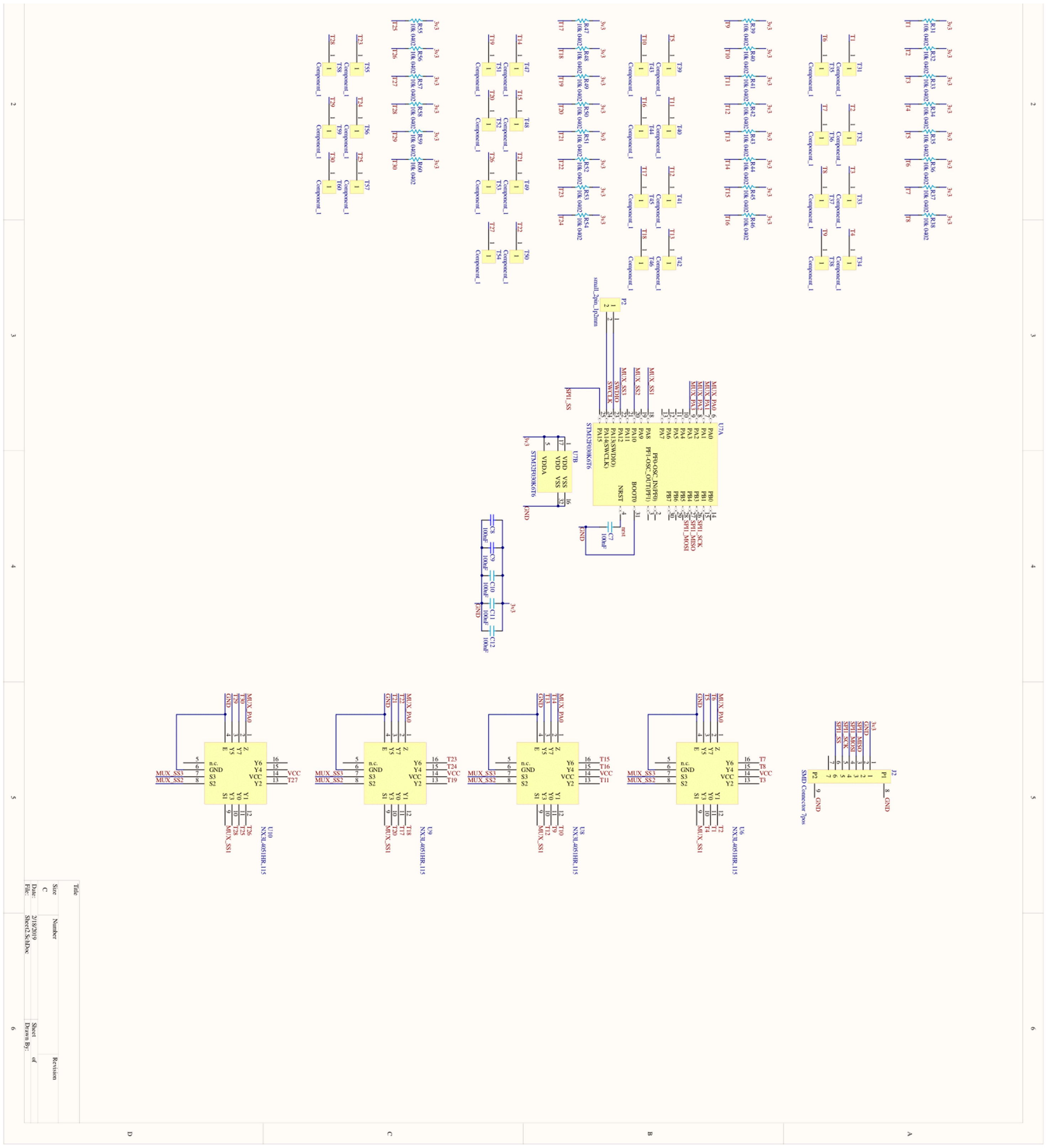
Figure 10: Pressure Sensor Physical Design

2.4 Schematics

2.4.1 Master Device Schematics



2.4.2 Slave Device Schematics



2.5 Software Description

2.5.1 Master Device Software

The microcontroller we are going to use as master in our system is STM32F401RBT6, which is a HAL C platform supporting the strict C89/C90 standard. Thus, we are going to use the "-std=c99 -pedantic-errors" compiling flag in this implementation.

Our system could be viewed as a common multiple Slaves single Master model. And three required control / data flow are attached as below, which will be controlled by our Master Device Software.

In a typical communication cycle, the Master Device Software will do the following,

a. Set the SS signal for the slave currently talking with to low (active low) using GPIO pin of the microcontroller, meanwhile, all other SS signals should remain at high to prevent MISO conflict.
b. As the Slave Clock (SCK) ticking, master chip will wait for the message from the selected slave by listening to the MISO.
c. SS for the current slave will be now set to high in order to end its permission to write to the MISO.
d. Now, the microcontroller on master device is ready for next cycle.

2.5.2 Slave Device Software

The microcontroller we are going to use as slave in our system is STM32F030K6T6, which is a HAL C platform supporting the strict C89/C90 standard. Thus, we are going to use the "-std=c99 -pedantic-errors" compiling flag in this implementation.

The microcontroller on the Sensors Module Chip is expected to complete the pressure sensors signals' Analog-to-Digital Conversion (ADC) as well as the communication with master device.

For the ADC part, our Slave Device Software will do the following,

a. Scan 4 ADC pins at once and register their binary values.
b. Increment the counter to select next group of 4 sensors by giving correct combination of Chip-Select signal to the 8-to-1 MUX(s) we have.
c. Keep doing this until data from all 30 sensors are registered.

The task for communication with the master device is relatively simpler,

a. Waiting for the MOSI signal with the corresponding user-defined message indicating the start of transmission process.
b. Execute the scanning process and preserve only the latest data awaiting to be sent.
c. Write data to the MISO if its own SS is in low (active low).

The timing will be the most significant consideration when the time turns to the integration of the Master Device Software and Slave Device Software.

2.5.3 MATLAB Script on PC

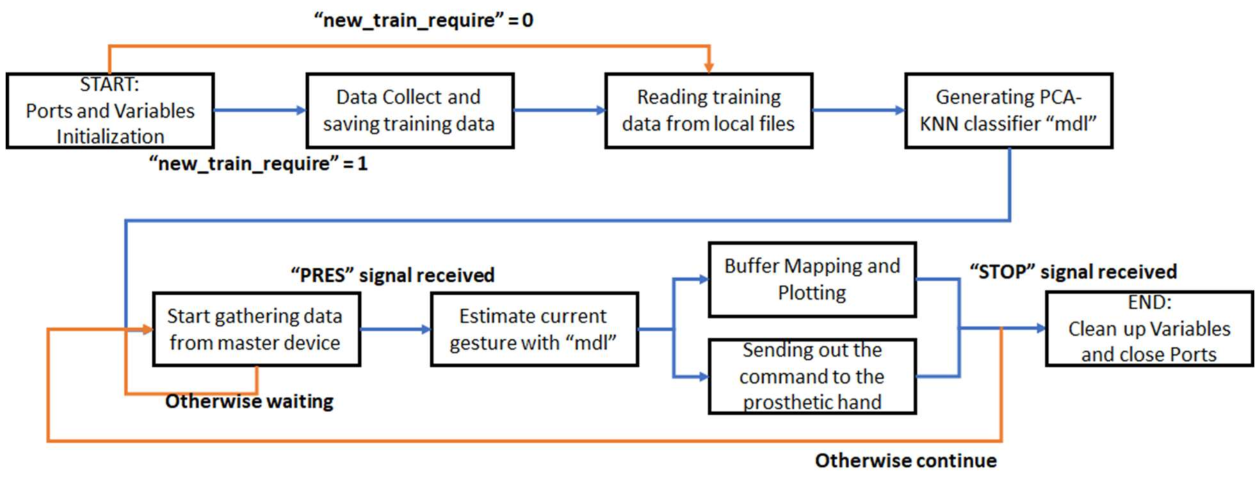

"new_train_require" = 0
START: Ports and Variables Initialization
Data Collect and saving training data
Reading training data from local files
Generating PCA-KNN classifier "mdl"
"new_train_require" = 1
"PRES" signal received
Start gathering data from master device
Estimate current gesture with "mdl"
Buffer Mapping and Plotting
Sending out the command to the prosthetic hand
"STOP" signal received
END: Clean up Variables and close Ports
Otherwise waiting
Otherwise continue

Figure 11: MATLAB Script Flow chart

In order to pre-process and classify the raw data we collect from the M-shape pressure sensors, we prefer the data from master device transmitted on to the PC through SPI protocol.

The whole MATLAB Script could be divided into three parts, data collecting, PCA-KNN classification and real-time estimation.

For the data collecting part, we record 40 datapoints for each gesture thus, however, we will only pick up the first 20 datapoints for our training process as we pass these datasets into the PCA-KNN model.

Every time as you move your arm position or the first time you put the bracelet on, a "new_train" is required to make sure the whole system is working in its best situation. And the flag "new_train_require" will need to be set to 1 at the very beginning of the whole program.

The second part involves the training of Principal component analysis (PCA) – K nearest neighbors (KNN) network. We need to first calculate the PCA of the heatmap and then paired them with the given ground truths, [1, 2, 3, 4] represents [Sign of the horns, Open, Gripping, Holding the fist] respectively.
After we put the PCA of the heatmap and the ground truth into the MATLAB function "fitcknn", it will return a classifier named "mdl".

Everything in the third part, namely, the second row of the flow chart, are placed in a while loop to make the MATLAB runs as a server to keep receiving the data streams from our master device and estimate the current gestures and send out proper command to the prosthetic hand until a "STOP" signal is received from the master device or we kill the MATLAB program manually.

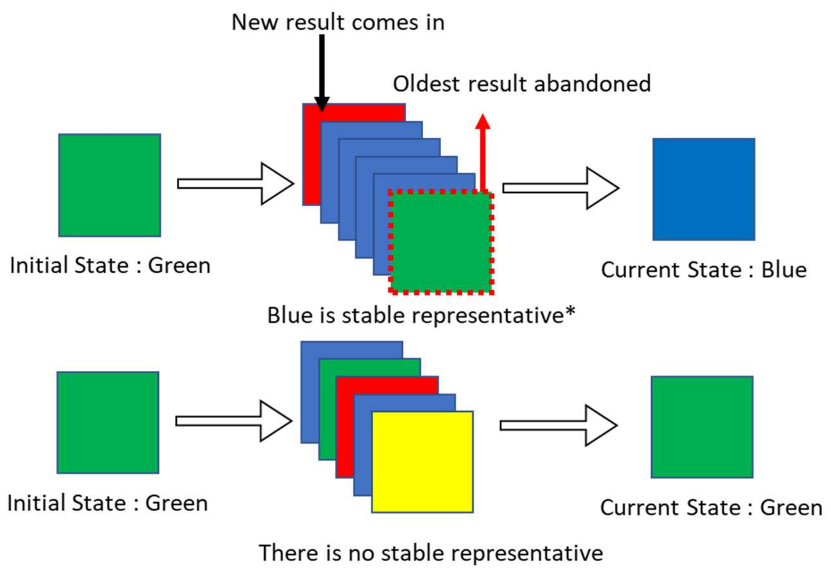


Figure 12: Voting Scheme

Since the PCA-KNN estimation is not perfect, jitters should be expected in the decisions made by the estimator. To prevent the prosthetic hand from reacting too fast or even receiving conflict command, we design a voting scheme that considers the most recent five results as a whole, and they are held in a container named “decision buffer”.

The logic itself is simple, as in the examples we list above: we will change a state, for example, from open to hold a fist gesture when and only when a certain state has a frequency of not less than four in the “decision buffer”, we call it “stable representative”. Otherwise, as in the second example, if there is no stable representative, we will leave the old state unchanged.

3. Design Verification

3.1 Functionality test – Microcontroller

After every PCB is soldered, following tests are conducted to make sure the microcontroller on the PCB is functional. The test is performed to both Master Device and Slave Device.

1. Power the microcontroller with 3.3v input from the DC power supply. Observe the current drawing from the DC power supply. When the microcontroller is in idle, a reasonable current drawn should not be too low (less than 0.001A which means the microcontroller may not be powered at all) or too high (greater than 0.1 A which means there may be a short or the microcontroller may be broken).
2. Write a simple program to blink the LED connected to the microcontroller. Download the program to the microcontroller check and check if there is any error during the downloading process. If the program failed to download, it may because the following reason: bad soldering, broken microcontroller, or broken debugger/programmer.
3. Run the program mentioned above on the microcontroller. See if the LED blinks. If the LED is not blinking properly, it means: bad soldering on some of the pins on microcontroller or wrong polarity of the LED.
4. Set a break point in the programming environment. Check the debugging features of the programming environment including setting breakpoints and single stepping through code.

It is verified that all 5 PCBs that we use in the final design have their microcontrollers properly soldered and ready for further development.

3.2 Functionality test – UART

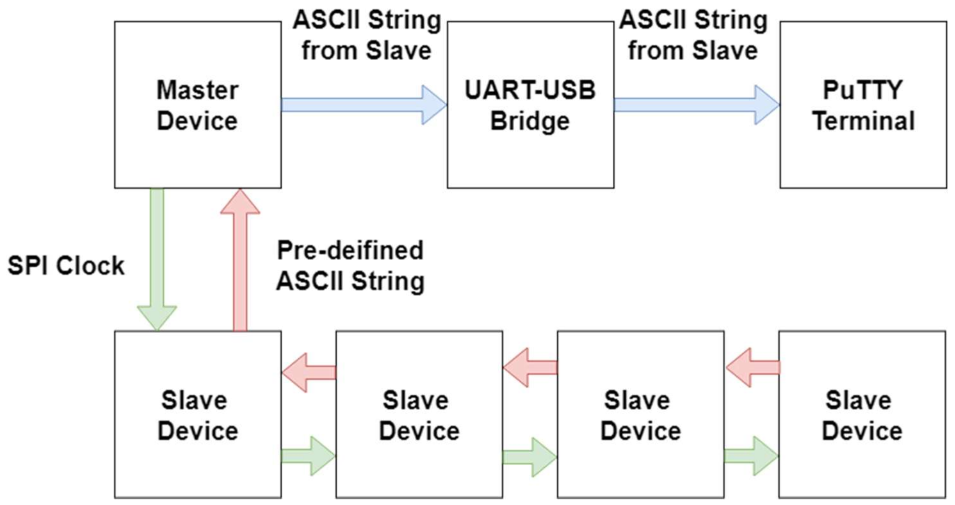
After testing the Microcontroller, the UART test is performed on the Master Device to make sure the communication between master and PC is correct.

1. Program the microcontroller to send pre-defined ascii string periodically via UART.
2. Connect the Master Device's UART port to an UART to USB bridge and connect the UART to USB bridge to a PC.
3. Display the received data on a PC in PuTTY terminal. Check whether the received data is consistent with the data programed in Master Device.

It is verified that the master device is able communicate with a PC over UART protocol using baud rate of 115200.

3.3 Functionality test – SPI

With both the Master Device’s and Slave Device’s microcontroller functioning and UART communication between Master Device and PC tested, we are now able to conduct tests about the SPI communication between Slave Devices and Master Device. The idea of the test is shown in the figure below.



To simplify the problem, we will first test the communication between each Slave Device and Master Device.

1. Program the Slave Device to send pre-defined message through SPI to Master Device.
2. Program the Master Device to receive the message sent by Master Device and forward the received message to PC though UART.
3. Check the received message on PC in PuTTY terminal and check whether the received message is consistent with the programmed message on Slave Device.

After testing the communication between each Slave Device and Master Device, we can now set up and test the whole SPI bus.

1. Since currently we are only 4 Slave Devices, we connect four Slave Devices alone with the Master Device to form the complete SPI communication bus.
2. Program each slave device to send out different pre-defined messages to Master Device.
3. Program the Master Device to scan through all four Slave Devices to receive the messages from the Slave Devices and forward the received messages to a PC through UART.
4. Check the received message on PC in PuTTY terminal and check whether the received message is consistent with the programmed message on each Slave Device.

It is verified that all 4 slave devices are able to communicate with the master device over SPI protocol but during testing the overall communication stability using 4 slaves, we face problem that if the SPI data rate is too high, the slave microcontroller may not be able to catch up with it and the system desynchronizes. We reduce the SPI data rate and delay the master microcontroller after requesting data from slaves to solve the problem. The following table shows the results trying combinations of different SPI data rate and time delay on master device.

Delay / Data rate	No Delay	1 ms	10 ms	50 ms
128 kbps	Unstable	Stable	Stable	Stable
256 kbps	Unstable	Stable	Stable	Stable
512 kbps	Unstable	Unstable	Stable	Stable

As the results showing, 1 ms time delay on the master device allows the SPI protocol running with maximum data rate of 256 kbps. 1 millisecond time delay is the minimum time delay achievable on the master microcontroller and the data is enough for transfer over 10,000 sensor readings per seconds, which is sufficient for our system. Therefore, we determine to use this setting for the SPI communication.

3.4 Functionality test – MUX

The QFN package MUXes we are using proves to be really difficult to solder and test. However, with the UART communication between Master Device and PC and SPI communication between Master Device and Slave Device tested, we are now able to test the soldering of the MUXes.

1. Program the Slave Device to collect data from all pressure sensors and send the collected data to Master Device through UART.
2. Program the Master Device to receive data from Slave Device and forward the data to a PC through UART.
3. On PC, write a Matlab script to visualize the data as a heat map. When the sensor is open (nothing between the electrodes), the sensor reading should be approximately maximum which is about 4095. When we manually short the electrodes of sensor with wire, the sensor reading shoud be approximately 0. By checking this on each sensor and record the results, we are able to figure out which MUXes are not properly soldered.

It is verified that all MUXes are working properly after several testing and re-soldering. This test also proves the functionality of pressure sensor electrodes that every pair of sensors are sensing 4095 when they are not connected.

3.5 Functionality test – Overall Test

With all the blocks tested, we can now perform the overall functionality test of the whole system.

1. Put the bracelet with a Master Device and 4 Slave Devices on testee's arm. The testee will perform 4 predefined motions, holding the fist, releasing the fist, gripping with thumb and ink finger, and sign of horns and our system will record the user's muscle movement for the four gestures and compute the principle component and form a nearest neighbor classifier according to the calculated result.
2. The testee will then perform the 4 gestures again and record additional data as the test data. The principle component of the test data will be calculated and will be tested using the nearest neighbor classifier we get from the previous step. The accuracy of classification will be calculated.

The result of the overall test shown that our system is able to classify between different gestures. The holding the fist gestures have an accuracy of 85%, releasing the fist of an accuracy of 90%, gripping with thumb and ink finger have accuracy of 79.49% and sign of horns have accuracy of 100%. The overall accuracy of classification is 88.68%, which is pretty nice with such a naive classification algorithm.

4. Cost Analysis

4.1 Cost of All Parts (Currency in USD)

Part	Part Number	Cost/Unit	Quantity	Subtotal	Provider
		4x Slave Module			
4 layers PCB	with ENIG finish	3.5	4	**14**	JLCPCB
Microcontroller	STM32F030K6T6	1.26	4	**5.04**	STMicroelectronics
0.1 uF Capacitor	CC0402JRX5R6BB104	0.039	16	**0.624**	Yageo
4.7 uF Capacitor	EMK107ABJ475MA-T	0.124	8	**0.992**	Taiyo Yuden
7-pin Connector	53261-0771	1.26	8	**10.08**	Molex
2-pin SMD Header	N/A	0	4	**0**	N/A
10k Resistor	SFR01MZPJ103	0.023	120	**2.76**	ROHM Semiconductor
8-input MUX	NX3L4051HR	0.849	16	**13.584**	NXP Semiconductors
SMD LED	SML-LXFT0603UPGCTR	0.806	1	**0.806**	Lumex
		1 x Master Module			
PCB	with HASL finish	0.2	1	**0.2**	JLCPCB
Microcontroller	STM32F401RBT6	5.33	1	**5.33**	STMicroelectronics
SMD LED	SML-LXFT0603UPGCTR	0.806	1	**0.806**	Lumex
2-pin SMD Header	N/A	0	1	**0**	N/A
2-pin Connector	53261-0271	0.776	1	**0.776**	Molex
4-pin Connector	53261-0471	0.983	1	**0.983**	Molex
7-pin Connector	53261-0771	1.26	2	**2.52**	Molex
Linear Regulator	TPS82140SILR	3.38	1	**3.38**	Texas Instruments
0.1 uF Capacitor	CC0402JRX5R6BB104	0.039	5	**0.195**	Yageo
4.7 uF Capacitor	EMK107ABJ475MA-T	0.124	4	**0.496**	Taiyo Yuden
22 uF Capacitor	GRM188R61A226ME15J	0.35	1	**0.35**	Murata Electronics
33 nF Capacitor	AC0402KRX7R8BB333	0.053	1	**0.053**	Yageo
470 Resistor	RR0510P-471-D	0.076	1	**0.076**	Susumu
309k Resistor	RC0402FR-07309KL	0.012	1	**0.012**	Yageo
100k Resistor	RT0402FRE07100KL	0.056	2	**0.112**	Yageo
		MISC			
ST-LINK in-circuit debugger/programmer	N/A	8.6	1	**8.6**	N/A
FTDI FT232 USB to UART Converter	N/A	10.25	1	**10.25**	N/A
			Total:	**84.443**	

4.2 Cost of Labor (Currency in USD)

Worker	Pay Rate (Semi-Skill)	Weekly Hours	Total Pay
Yangge Li	25	12	300
Enliang Li	25	12	300
Zhoushi Zhu	25	12	300
		Total:	**900**

If we assume the actual working period takes about 12 weeks, the total cost of labor is about $\$900 * 12 * 2.5 = \$27000$

5. Conclusion

5.1 Accomplishment

By the end of the semester, we successfully design and prototype a functioning interface platform to control a prosthetic hand based on foam pressure-sensor including a classification algorithm that transfer pressure change into command to control a prosthetic hand. The system is able to fit on a specific user's arm, recognize four gestures and control the prosthetic hand to do the same motion. The gestures supported are holding the fist, releasing the fist, gripping with thumb and ink finger and make the sign of the horns.

The overall classification accuracy achieves the requirement set by PSYONIC Inc. and the total cost is less than 100 USD, which is much less than the cost for PSYONIC Inc. to develop an equivalent EMG system.

5.2 Safety & Ethics

Our major safety concern during the design process is the potential circuit short which may lead to extreme high temperature due to current surge if the PCB carries serious bugs or human error while testing. Circuit short hazard could burn down the PCB or scald our skins, and thus, it needs to be taken seriously. We won't allow any one in our group working alone with the PCB(s).

Since the prosthetic hand is comprised of many mechanical components, we also need to take precautions to avoid any possible cutting injuries caused by improper operations.

After reviewing the IEEE and ACM ethics, we agree on the following concerns to be presented in our project proposal,

5.2.1 IEEE Policies, Section 7, 7.8 IEEE Code of Ethics

8. to treat fairly all persons and to not engage in acts of discrimination based on race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression [6]

We need to be honest on the performance of our product. Due to current technical limitation, both EMG and pressure-sensing method could only work for the disabled who had amputation surgery below the elbow (and still have working residual limbs). We expect our product to work poorly for those who experienced amputation surgery above the elbow.

This may be a discrimination regarding the degree of disability.

9. to avoid injuring others, their property, reputation, or employment by false or malicious action [6]

&

ACM General Ethical Principles 1.2 Avoid harm. [7]

Our product may injure/harm (bring negative consequence) the user, or the objects hold by the prosthetic hand due to incorrect responds that against user's will. Furthermore, since user could not always control the force it applies to the project precisely, it's very possible that soft items are squeezed.

5.2.2 ACM General Ethical Principles

1.6 Respect privacy. [7]

As part of the training process, we need to collect sensitive data from user, such as the pressure patterns for different hand movements and store them into the master chip, which assumes the possibility of user's data leakage if the product hacked or missing.

5.3 Future works and Application

There exists several possible improvements to the system we designed:

a. The mechanical design of the enclosure can be improved to make our product more user-friendly and robust
b. A quantitative analysis on the characteristics of the foam can be developed to help find more suitable electrical conducting foam
c. A better classification algorithm can be developed so the classification and inference process can be more robust and accurate
d. Increase the number of pressure sensing units to provide more data points for muscle movement
e. Deploy the classification algorithm on the microcontroller so command could be sent directly from the master device to the prosthetic hand

For future application, the product we designed can cooperate with the existing EMG sensors on the prosthetic hand developed by PSYONIC Inc. to combine the advantages of both methods. By doing this, we expect to enhance the capability for the prosthetic hand to track the user's intended movement and benefit people with disability without increasing the cost remarkably.

References

[1] C. Castellini, R. Kõiva, C. Pasluosta, C. Viegas, and B. M. Eskofier, "Tactile Myography: An Off-Line Assessment of Able-Bodied Subjects and One Upper-Limb Amputee," *MDPI*, 23-Mar-2018. [Online]. Available: http://www.mdpi.com/2227-7080/6/2/38. [Accessed: 01-Feb-2019].

[2] STMicroelectronics, "STM32F401xB STM32F401xC Datasheet," *STMicroelectronics*, 06-Sep-2013. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f401cb.pdf. [Accessed: 01-Feb-2019].

[3] STMicroelectronics, "STM32F030x4 STM32F030x6 STM32F030x8 STM32F030xC Datasheet," *STMicroelectronics,* 04-Jul-2013. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f030k6.pdf. [Accessed: 01-Feb-2019].

[4] Texas Instruments, "TPS84210 2.95-V to 6-V Input, 2-A Synchronous Buck, Integrated Power Solution," *Texas Instruments*, 11-Apr-2018. [Online]. Available: http://www.ti.com/lit/ds/symlink/tps84210.pdf. [Accessed: 01-Feb-2019].

[5] NXP Semiconductors N.V., "NX3L4051 Product data sheet," NXP Semiconductors N.V., 03-Jul-2012. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/NX3L4051.pdf. [Accessed: 01-Feb-2019].

[6] Ieee.org, "IEEE IEEE Code of Ethics", 2016. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 1- Feb- 2019].

[7] "ACM Code of Ethics and Professional Conduct," *ACM Code of Ethics and Professional Conduct*. [Online]. Available: https://www.acm.org/binaries/content/assets/about/acm-code-of-ethics-and-professional-conduct.pdf. [Accessed: 01-Feb-2019].

[8] Zhang, C. (2019). How to run deep learning model on microcontroller with CMSIS-NN (Part 1). [Blog] DLology. Available at: https://www.dlology.com/blog/how-to-run-deep-learning-model-on-microcontroller-with-cmsis-nn/ [Accessed 19 Feb. 2019].

Appendix A

Master Device Microcontroller

Requirement	Verification
1. Able to both receive and transmit data over SPI at speeds greater than 12Mbit/s while running the proper master device software.	1. <u>SPI Verification</u> a. Connect the MISO, MOSI, SCK and SS signals to oscilloscope to verify there are signals. b. Using the signal from SCK to check if the speed of SPI is greater than 12Mbits/s. c. Use the debugger built inside the STM32 development environment to check the content of data from slave device is correct.
2. Able to both receive and transmit data over I2C while running the proper master device software.	2. <u>I2C Verification</u> a. Connect the SDA and SCL signals to oscilloscope to verify there are signals. b. Connect the master device to PSYONIC hand and send predetermined command to the hand and see if the hand is controlled.
3. The microcontroller (STM32F401RB) clock frequency should be higher than 64MHz.	3. <u>Computational power verification</u> Connect the system core clock output to the oscilloscope. The frequency of that signal should be higher than 64MHz.

Slave Device Microcontroller

Requirement	Verification
1. Able to both receive and transmit data over SPI at speeds greater than 12Mbits/s while running the proper slave device software.	1. <u>SPI verification</u> Please refer to the SPI verification of master device. If the Master Device SPI is proofed to be working, the Slave Device SPI have to work.
2. Able to convert analog signals ranging from 0-3.3V to corresponding 12-bit digital signals.	2. <u>ADC verification</u>

	a. Connect the ADC channel of the slave device microcontroller to the DC power supply. b. Adjust the output voltage of the DC power supply to be value between 0-3.3V. c. Use the debugger built inside the STM32 development environment to check the correctness of converted data.

DC-DC Converter

Requirement	Verification
1. Able to convert voltage ranging from 7.2-8.4V to 3.3±0.3V.	1. Provide 7.2-8.4V input voltage from DC power supply to the component. The output voltage should be 3.3±0.3V
2. Able to provide 500mA current to power the system.	2. Check if the system powered by the DC-DC Converter can function properly.

Analog MUXs

Requirement	Verification
1. The component is able to select between 8 analog signal ranging from 0-3.3V.	1. <u>MUX selection verification</u> a. Program the slave microcontroller to specify the Select Signal for the Analog MUX. b. Apply different pressures to eight sensors that are connected to the MUX being tested. c. Use the debugger built inside the STM32 development environment to check if the output of MUX reflects the change of pressure on the selected sensor.
2. The component should have a on resistance lower than 10.	2. Connect the output pin of the component to a 10 resistor. Provide 3.3V signal to input 0 and change selection signal to select input 0. The voltage between input 0 and output should be less than 3.3/2=1.65V.

Pressure Sensor

Requirement	Verification
1. The pressure sensor characteristic should report approximately same signal for same compression of a properly chosen form and different sensors should report approximately same signal for same compression. The difference between different test trial on the same sensor should be less than 1% and the difference between different sensors should be less than 10%.	1. Verification Process a. Connect the master device to a PC via a UART bridge. b. Program the master microcontroller to collect digitized data from slaves and transfer the data to PC using UART protocol. c. Compress the foam on top of the all the sensor to a specific distance for five times. Choose 10 different distance within the working range of the foam and repeat the test for 10 time respectively. d. Compare the value transferred from master microcontroller in the PC and check if the differences are larger than expected. e. Port the data to Matlab and plot a heatmap of the reported pressure distribution for more visualized test results.

Mechanical Constraint

Requirement	Verification
1. The X and Y dimension of the master device PCB should both be smaller than 3cm.	1. Measure the dimension of the manufactured master device PCB to check if it is oversized.
2. The X dimension of the slave device PCB should be smaller than 3cm, and the Y dimension should be smaller than 10cm.	2. Measure the dimension of the manufactured slave device PCB to check if it is oversized.