PROSTHETIC CONTROL BOARD

By

Caleb Albers

Daniel Lee

Final Report for ECE 445, Senior Design, Fall 2017

TA: Yamuna Phal

13 December 2017

Project No. 46

Abstract

This project involves creating a circuit board and accompanying software framework to control a robotic prosthetic hand. The goal was to provide a solution in a small form-factor, in an affordable manner, and adhere to applicable regulatory compliance guidelines to fast-track product approval for the Food and Drug Administration (FDA). Our project achieved these objectives and can control a six degree-of-freedom (DOF) prosthetic hand with brushed direct current (DC) motors, quadrature encoders, pressure sensors, an onboard temperature sensor, a status light emitting diode (LED), and input commands given via an Inter-IC (I^2C) bus. The finished product can perform multiple grasp actions, has significant safety features built in, and is ready for integration with a large-scale prosthetic arm system.

Contents

1. Introduction

Over eleven million people worldwide have hand amputations. Eighty percent live in developing nations, and less than three percent have access to affordable prosthetic devices [1,2]. We have partnered up with PSYONIC, a local startup, who aims to tackle this problem through the development of a low-cost robotic prosthetic hand. The focus of this project was to deliver a feature-rich, affordable, and standards-compliant robotic hand control system for integration into PSYONIC's upper-limb prosthesis. Specifically, we developed a circuit board that fits inside the size constraints of a normal human female's hand, can control six brushed DC motors that drive each digit, and has a variety of input and output methods for interacting and understanding the environment.

Our solution utilizes quadrature encoders for feedback on finger position, and supports up to six pressure sensors that can be embedded inside the silicone fingers and used for grasp support. The system developed has a status LED to give visual indication of error conditions to the user, utilizes an on-board temperature sensor to detect potential thermal issues, has a Universal Serial Bus (USB) interface for debugging, and features an I^2C communication bus for input commands.

All project goals have been realized. Both the hardware and software components were tested and are fully operational. A detailed description of our engineering approach and results are given in this paper.

2 Design

Due to our project being a part of a separate entities product, we first had to consider that any design choices we made should work along with the company product. PSYONIC was generous enough to provide a dimensioned model, as shown in Figure 1. All components and package sizes were chosen to ensure no interference with the mechanical design of the prosthetic hand would take place.

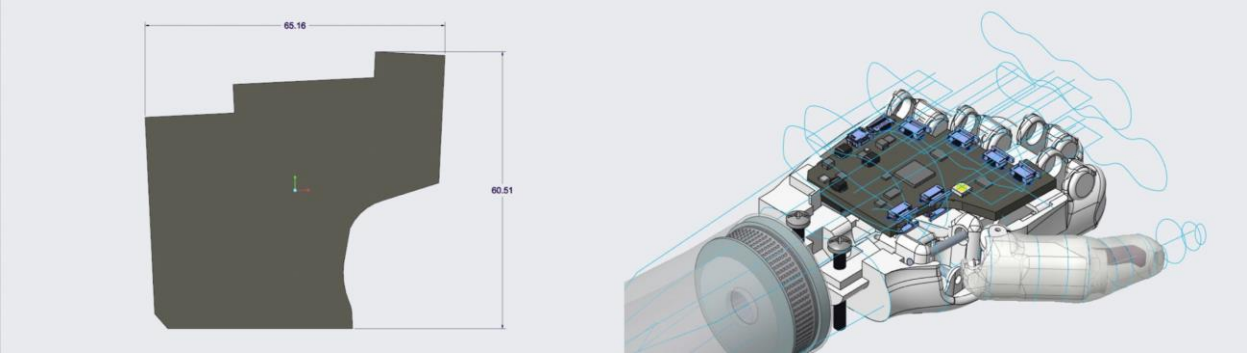

Figure 1: 3d model and basic dimensions

2.1 Block Diagram

Our project is a subsystem in the PSYONIC prosthesis. Figure 2 shows the various sub-components of our design, including how they integrate into the other PSYONIC subsystems (represented by the “External (Out of Scope)” segment of the block diagram. The onboard components include the microcontroller, LED, temperature sensor, and motor controllers. The three dual H-bridge motor controllers drive six brushed DC motors, which are located on independent circuit boards, along with quadrature encoders. Finally, external pressure sensors are supported, with the intention of embedding the pressure sensors in the silicone finger tips of the prosthetic hand in the future. Together, these various subsystems form the basis for our work.

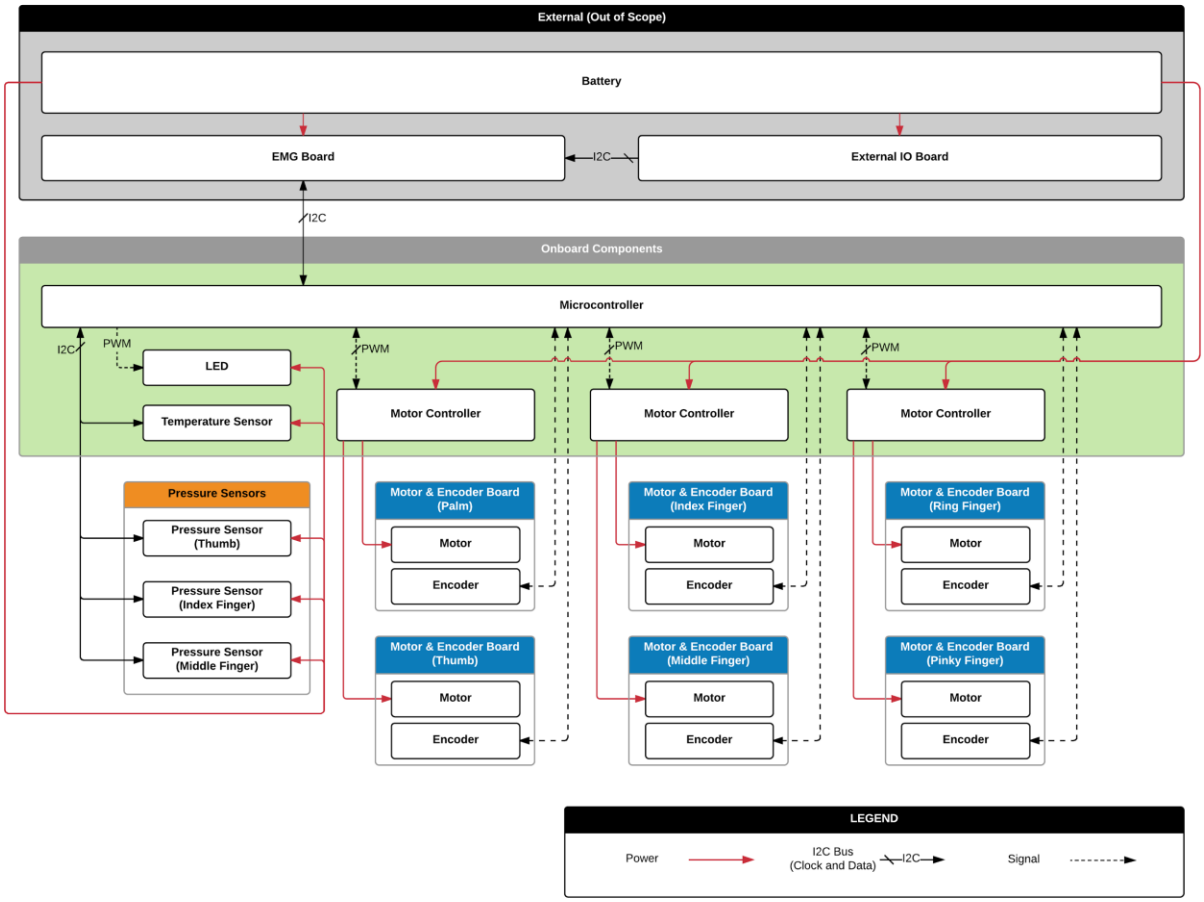

Figure 2: Block Diagram

Each subsystem on the block diagram is also present in the schematic, shown in Figure 3. These blocks are examined in detail as independent parts sections, each with their own requirements, verification procedures, and engineering design choices.

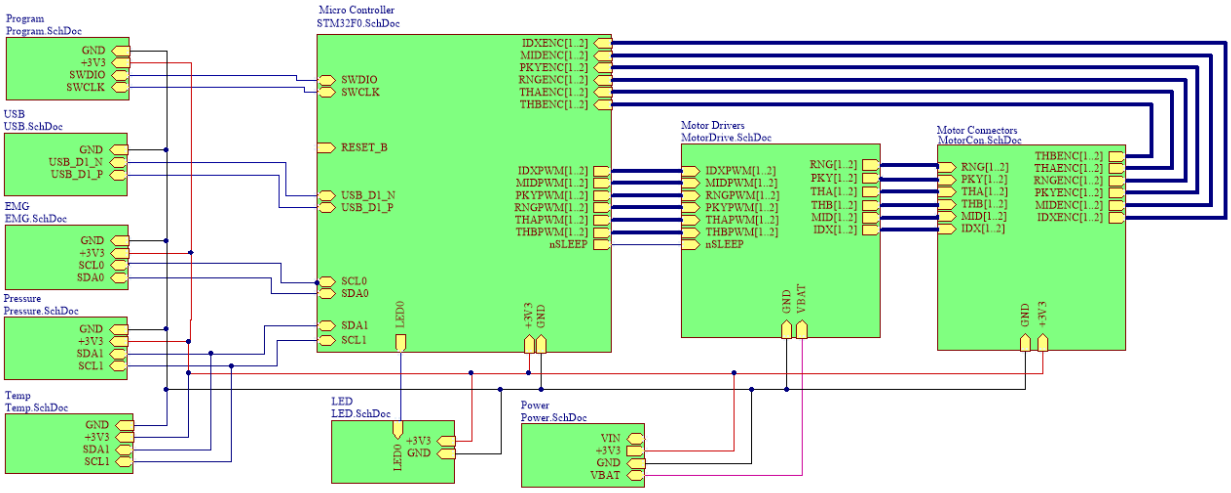

Figure 3: Overview schematic

2.2 Microcontroller

The microcontroller of choice was the STM32F072RBT6 [3], a low cost 32-bit microcontroller with ARM architecture. The microcontroller has many GPIO pins and support serial protocols such as I^2C, SPI, and CAN. The most important protocol was I^2C, as it is how we communicated with all the sensors in our design. The microcontroller also allowed us to setup a USB serial connection over two pins for debugging purposes. Another major factor was the availability of a development board that rapidly sped up progress on the project by allowing us to work on known working hardware. Additionally, the board and in turn the microcontroller is supported by the Mbed framework; allowing for an easier, more readable, and well documented implementation. As shown in Figure 4, our project uses almost every pin available on the microcontroller.

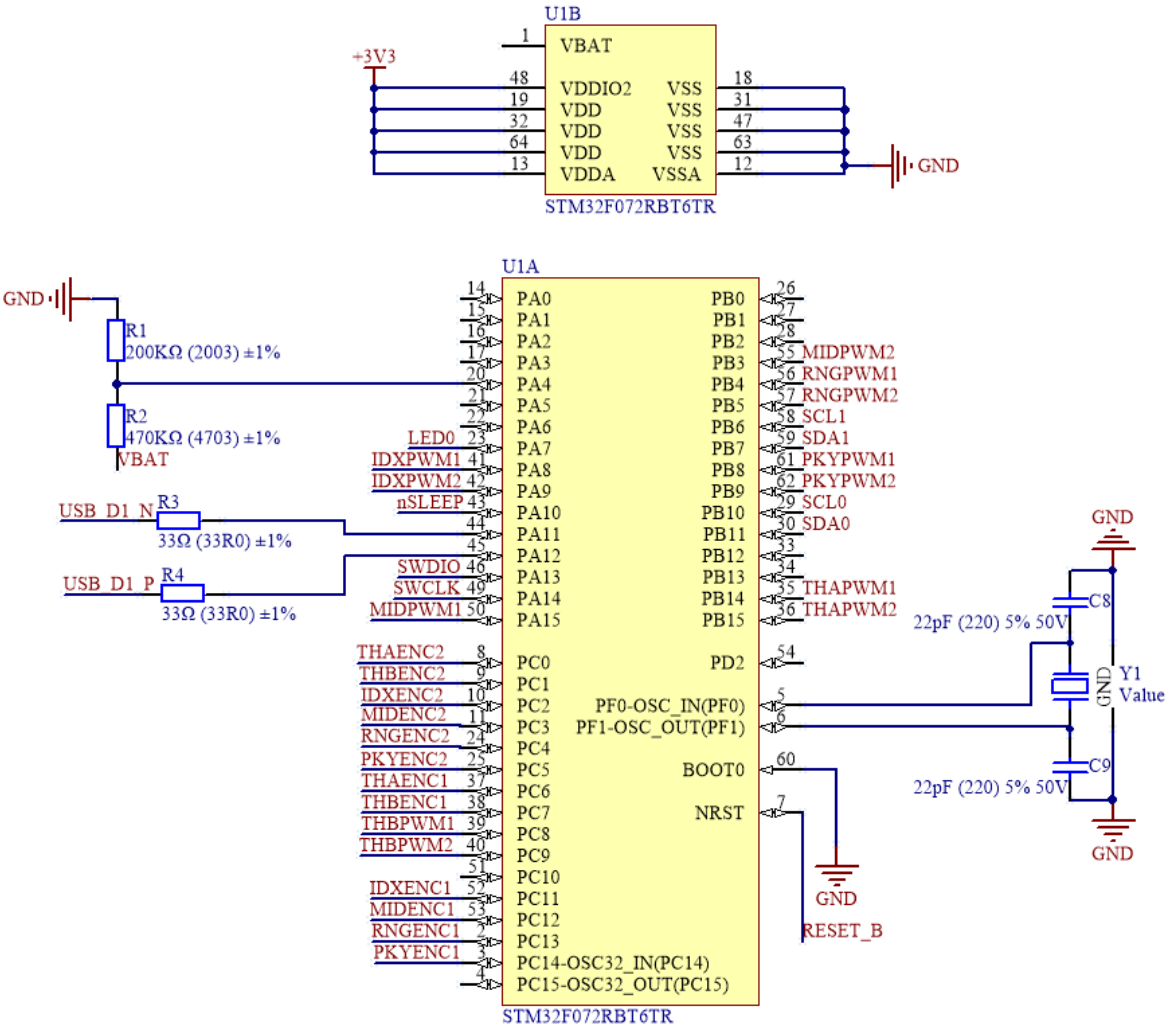

Figure 4: Schematic for the microcontroller

2.3 Motor Controllers

The motor controllers are a key component of the project as they drive the actuation and flexion of each finger. Since the motors are brushed DC, we had to use a controller that would allow us to drive the motors in both the forward and reverse direction. PSYONIC has tested various offerings in the past and from those we selected the DRV8881P [4] for its size and availability at PSYONIC. The Texas Instruments DRV8881P has a simple Pulse-Width Modulation (PWM) interface requiring two inputs to control a motor, and includes the ability to put the driver into a sleep mode where very little power is used. Figure 5 shows the schematic pertaining to three motor controllers, along with the other passive components for configuration and usage. Each controller is a dual H-bridge design, which means that two motors can

be independently driven. This allows for a total capability of controlling six motors for our implementation.

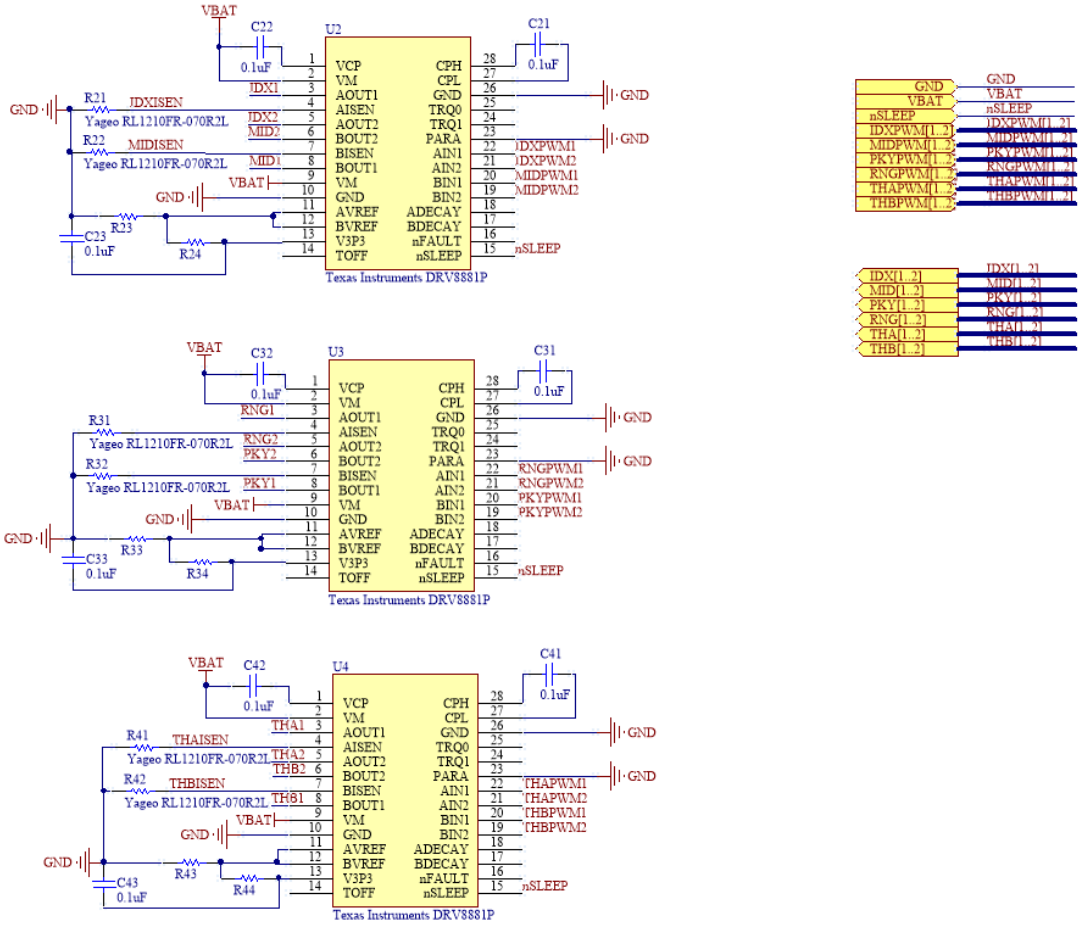

Figure 5: Schematic showing the motor controllers and additional components

2.4 Pressure Sensors

The pressure sensors shown on Figure 6 are key feature to the prosthetic hand as they can be used to provide feedback to the user. The pressure sensors are a product from STMicroelectronics called the LPS22HB [5], which feature a pressure sensor and a temperature sensor. The sensors were chosen by PSYONIC and integration of the sensors into the tips of the fingers was out of the scope of our project. Our project instead worked to implement a standard connection interface as well as software integration. To create the physical connection, we pioneered a connector from JST with an extremely small profile. Though more difficult than most common connectors, we gain board area and thin flexible wires. We then worked to communicate with the sensor over the I^2C protocol, which outputs three bytes of pressure data that can be converted to Hectopascal (hPa). This sensor is PSYONIC's choice for the foreseeable future, but an implementation with any sensor that can communicate over I^2C should also work if they ever decide to change.

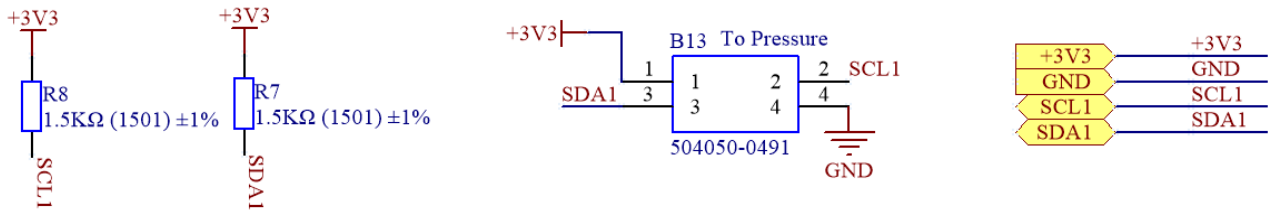

Figure 6: Schematic for one of the pressure sensor connectors and pull-up resistors for I^2C

2.5 Temperature Sensor

An onboard temperature sensor shown in Figure 7 was selected to keep track of any thermal errors that may present themselves on the hand control board. Specifically, this was done to specify a threshold temperature that, if surpassed, would indicate an overheating condition. Given that the acrylonitrile butadiene styrene (ABS) plastic that the hand is constructed of starts to exhibit signs of deformation at around 60 °C, we choose 55 °C as a cutoff threshold. When breached, the microcontroller will immediately stop all motors, the status LED will turn red, and an error will be printed via the USB serial console.

We intended the maximum temperature range for our device to be between -25 °C and 55 °C. As such, a temperature sensor with a greater range was required. Additionally, at least an accuracy of ± 1 °C needed to be maintained to ensure that any error was minimized. A Microchip-brand MCP9800A5T [6] temperature sensor was selected, as it utilizes an I2C communication bus (which was already present on the PCB for other components) and has a small form-factor SOT-23A-5 package. The specific sensor used has an operating range of -55 °C to +125 °C and an accuracy of ± 0.5 °C, both of which are well within tolerance.

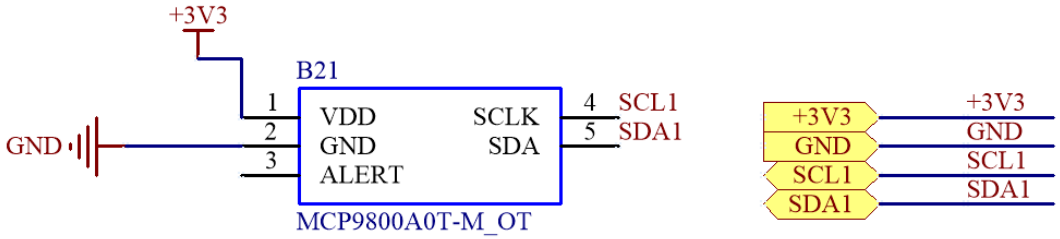

Figure 7: Temperature sensor schematic

2.6 Status LED

The status LED shown in Figure 8 is digitally controllable and can produce a red, green, and blue (RGB) output. The WS2812 [7] LED was chosen, as it only requires power, ground, and a singular data connection. The protocol for the LED is a variation of PWM that carries similarities to the Serial Peripheral Interface (SPI) protocol. We found during development for it to be a nuisance as it requires precise timing. We have considered alternatives and a suitable replacement would appear to be the APA102C, but it would unfortunately be slightly more expensive and require another digital pin.

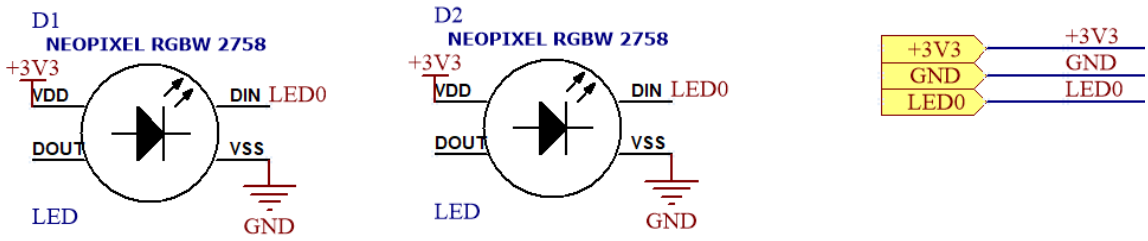

Figure 8: LED schematic

We found that the WS2812 LED does not appear to produce an accurate representation of true color for a given RGB color value. Research into the topic revealed that the human eyes perceive brightness in a logarithmic manner whilst the LED is linear. To mitigate this issue, we implemented a gamma correction using a lookup table. The values were provided through some experimentation by Ryan O'Hara [8] which provide a reasonable approximation. The result was a more accurate color spectrum that converts a given RGB input to the corresponding lookup equivalent as shown in Figure 9.

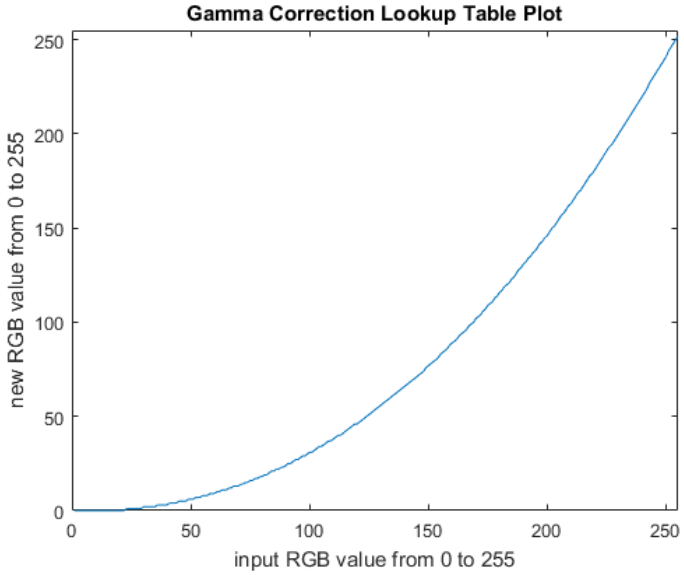

Figure 9: Lookup Table Plot

Although we had the ability to choose any color within the 24-bit RGB color spectrum, we decided it was best to choose a base palette of colors. After some research, we chose the Natural Color System (NCS) which has the six elementary colors shown in Figure 10. The premise of NCS is to produce a set of colors based on how humans naturally perceive color.

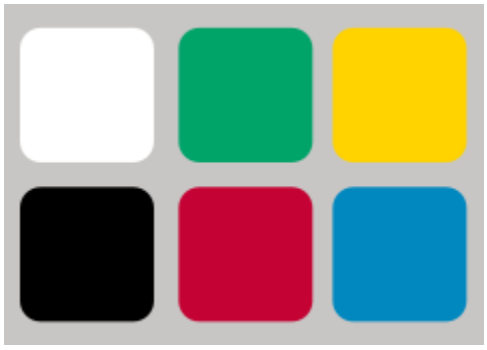

Figure 10: Natural Color System elementary colors. [9]

3. Design Verification

3.1 Microcontroller

To confirm I^2C communication we attached a digital logic analyzer to the I^2C data lines, and converted the signals to ASCII and hex values using a built-in decoder. Looking at Figure 11 we can see the decoded signals as blue message boxes above the wave forms. The top waveform represents SDA (Serial Data), whilst the bottom waveform represents SCL (Serial Clock). The I^2C protocol calls for start and end conditions, which the microcontroller is correctly outputting as the digital logic analyzer is marking them as green and red circles accordingly.

The two commands being sent are to first set which register in the temperature sensor we desire to access, followed by reading the data from this register as two bytes of data. To confirm we had a proper transmission of data we calculated the temperature given back from the sensor. Bit shifting the first received value of 0x1A eight positions to the left and then logic OR the shifted number by the second value (0x80), gives 0x1A80. Converting 0x1A80 to decimal is 6784, and then dividing by the correction factor 256 gives a temperature of 26.5 °C. This value was very close to the room temperature we expected within given tolerances, thus confirming we had working I^2C communication.

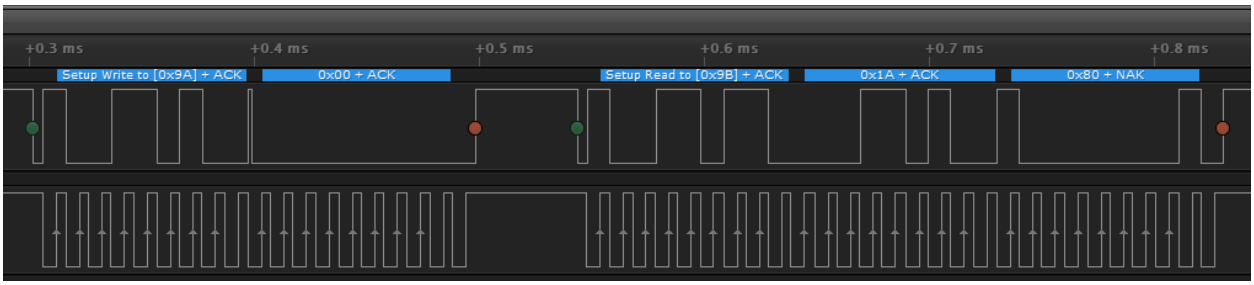

Figure 11: Screen capture of digital logic analyzer output for I^2C with temperature sensor

The next item of importance was being able to process the sensor data, which we verified by connecting the microcontroller over USB to a computer. This computer would have a serial monitor to display the output of the microcontroller with formatting for easier readability. This is shown in Figure 12 where we see the temperature in Celsius, raw pressure data, pressure in Hectopascals (hPa) and an encoder counter, from left to right respectively.

```
Temperature: 25    Pressure(raw): 4045240.000000    Pressure(hPa): 987.607422    Encoder Count: 1
Temperature: 25    Pressure(raw): 4045216.000000    Pressure(hPa): 987.601562    Encoder Count: 1
Temperature: 25    Pressure(raw): 4045188.000000    Pressure(hPa): 987.594727    Encoder Count: 1
Temperature: 25    Pressure(raw): 4045185.000000    Pressure(hPa): 987.545654    Encoder Count: 1
Temperature: 25    Pressure(raw): 4045169.000000    Pressure(hPa): 987.590088    Encoder Count: 1
Temperature: 25    Pressure(raw): 4045182.000000    Pressure(hPa): 987.593262    Encoder Count: 1
```

Figure 12: Screen capture of the serial monitor

3.2 Motor Controllers

We verified the operation of the motor controllers by programming the microcontroller to cycle motors forwards and backwards. We first tested one at a time to confirm each controller was working properly,

and then all at the same time. This concluded with our demonstration where the hand cycled from an open to a fully closed grasp.

3.2.1 New connectors

We verified an improvement in connectors by testing the common issues PSYONIC had encountered with previous revisions. The older connectors could be frequently mishandled during insertion, resulting in accidental shorts and upside-down connections. Another issue was the connectors would disconnect during finger movements, especially if the connector was not fully seated. The new connectors have vertical pins for connections so inserting upside-down is virtually impossible without a decent amount of force and damage to the connector. The connector also features sliding rail-like structures with clip points that will guide the connector into place and then clipping together once fully inserted. During construction and testing with PSYONIC's hand we encountered no issues with the new connectors even during finger movement.

We verified the connectors could handle at least 1.5 amps of current by soldering a connector onto an extra board and crimping a new header for a test setup. This setup was then attached to a DC power supply where a current limit was set to 1.5 amps. Since the setup is effectively a low ohmage resistor, the power supply switched to constant current mode. In constant current mode the power supply drops the voltage of the output and keeps the current within the set max. During multiple trials the connector appeared to be visually unaffected and electrically sound.

3.2.2 Encoder Feedback

We verified the encoder feedback by first probing the respective pins that exist as a mirror on the backside of the PCB. When powered up and rotated a pulse that would switch between low and high states would be observed on the waveform. To confirm the encoders were being properly read by the microcontroller we performed a series of test where a serial monitor would display the position of the encoder as we moved the magnetic rotor back and forth.

3.2.2 Conformal coating

We visually verified the application of conformal coating by the semi-clear amber appearance of the PCB after application. Further test confirmed its properties as probing two electrically connected points with a multimeter in continuity mode produced no shorts or low resistance values.

3.3 Status LED

We verified the operation of the status LED by loading some code onto the microcontroller cycling through various colors in the RGB spectrum. To conform with our choice to use the natural color system as a base palette of colors we later loaded a script to cycle through the NCS color space, with black shutting the LED off altogether. We were surprised to see how bright the LED could become and were able to see the LED from 45° off normal. In fact, we could move to nearly perpendicular to LED before losing sight of the LED.

3.4 Pressure Sensor

We verified communication with the pressure sensor over I^2C by programming a microcontroller to rapidly read data from the sensor and display to a serial monitor. We then logged data from a test where we put varying amounts of force on the sensor and plotted it in MATLAB. The applied force ranged from barely taps to as much we could physically exert, and the results can be seen in Figure 13.

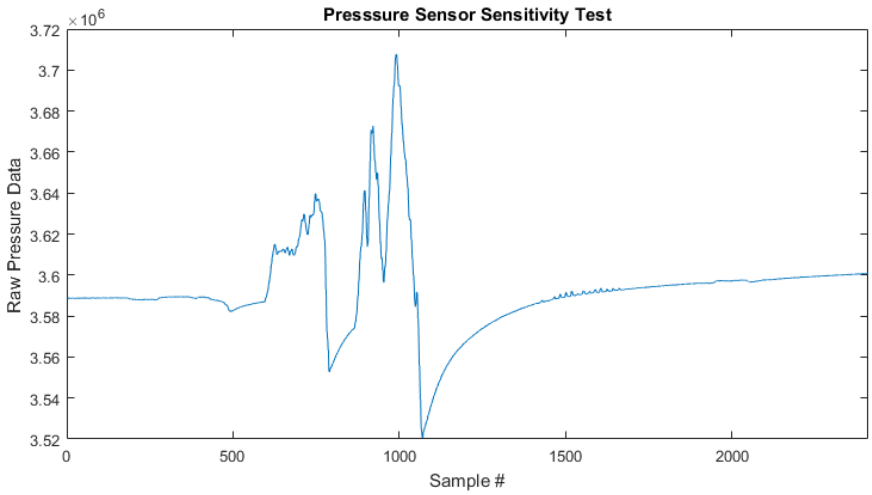

Figure 13: Plot of raw pressure values from pressure sensor

3.5 Temperature Sensor

We verified communication with the temperature sensor by programming the microcontroller to frequently read data and display on a serial monitor. We first noted the temperature the sensor was reporting and by placing a finger on the sensor seeing if the values would increase. We then removed the finger and observed the reported temperature gradually drop back to the base temp. This test had a range from around room temperature at 26 °C, increasing to approximately 30 °C, then dropping back down to 26 °C.

We verified a safety shutdown by using a hot-air gun and blowing towards the region where the temperature sensor was located on the PCB. The microcontroller was programmed to go into a critical error state if the sensor reported a temperature above 55°C. In the critical error state, the motors would shut down, the LED would turn red, and the serial monitor would display an error message such as the one shown in Figure 14.

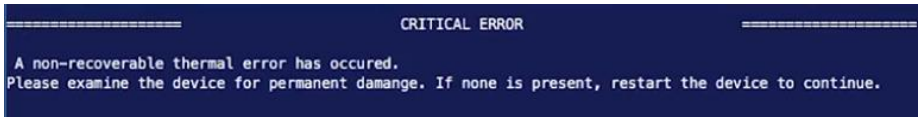

Figure 14:Screen capture of the error message given during thermal shutdown test

4. Software Architecture

In addition to the hardware implementation of our project, an extensive software architecture was developed to tie all the individual components together and control the hand.

4.1 Design Approach

All software aspects of this project were designed to be as general as possible, such that much of the code developed could be transferred to future iterations of PSYONIC's product line without major modifications. Given that we utilized an ARM microcontroller, we decided to approach the software architecture by creating C++ libraries based off the ARM Mbed operating system and associated development toolchain.

Figure 15 shows a top-down view of our software stack. Several open source libraries were used to drive the motor controller, interpret the quadrature encoders, calculate control loops, interface via USB, and drive the LED. Since most of the aforementioned objects are implementation-dependent, we created code wrappers on top of the libraries to offer a consistent application programming interface (API) that the hand control framework and software used. This allows for implementation-specific details, such as what motor driver model is used, to be switched out without having a large impact on the rest of the code-base.

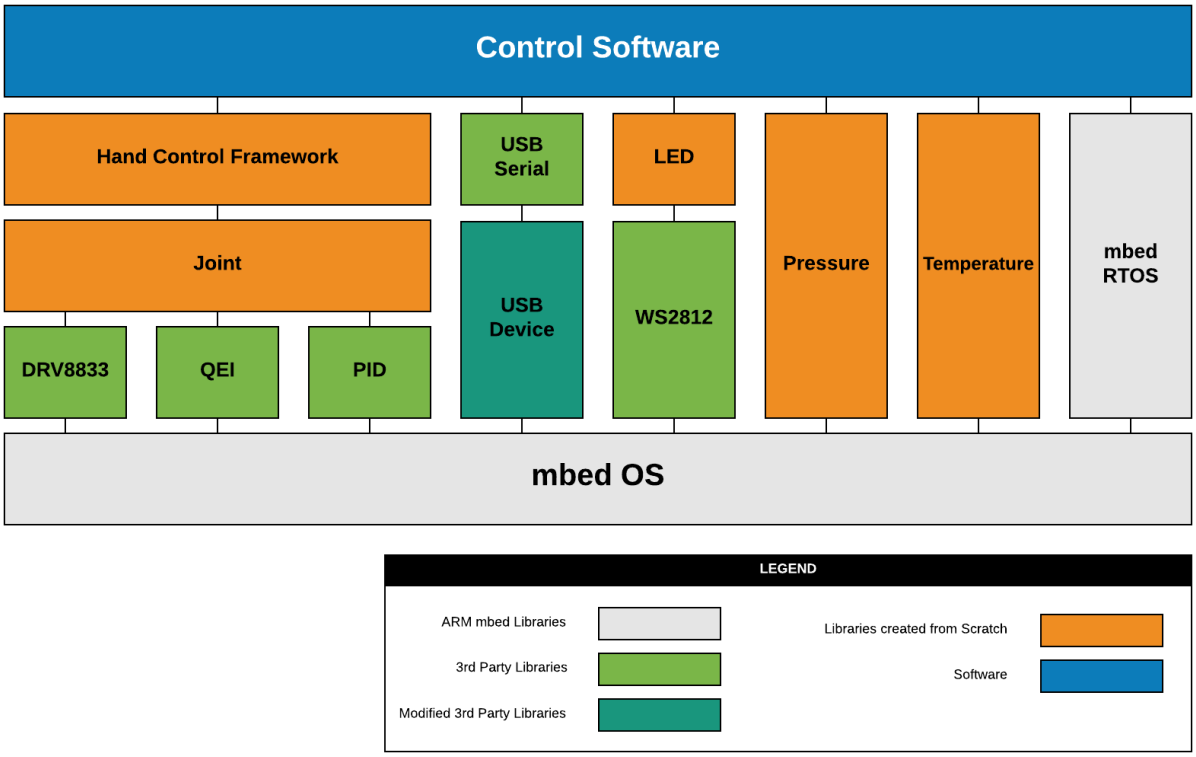

Figure 15

4.2 Hand Control Framework

The Hand Control Framework was developed to interpret instructions over the I2C bus from the electromyography solution and then move the fingers into their appropriate positions for each grasp action (i.e. an open palm or a closed fist). This was built by modeling each finger as a series of *joints* which further encapsulate the motor, position feedback, and control system details in a common interface.

A table of grasp actions was created to be specify the desired position of each finger by declaring one or more *waypoints*, which have a specific location a finger should move to, along with a speed multiplier to determine how fast the motor should move. This allows for complex movements and basic path planning. This grasp table can be extended to an arbitrary number of grasps, waypoints, and joints.

4.3 Real-Time Operating System

The ARM Mbed Real-Time Operating System (RTOS) was utilized to allow for multiple software threads. This simplified the timing issues present in ensuring that safety-critical error checking (such as for overheating) could be given priority, however additional design time was required to deal with the heavy preemption that occurred due to the encoders, as each encoder was made up of two Hall-effect sensors that interrupted code flow on the microcontroller whenever a finger would move.

The implementation of an RTOS allows for future advancement and flexibility in task scheduling, deterministic behavior, and predictable memory utilization. These are useful in safety-critical systems. This will allow PSYONIC to easily extend the control software as needed to add more features.

5. Costs

5.1 Parts

The hand control circuit board alone consists of 98 total components, 22 of which are unique (a single model number used multiple times). The total cost of our project, including components, PCB manufacturing, and development boards, came out to approximately $59.42. Buying in bulk for manufacturing can have a significant impact on total cost by taking advantage of quantity-driven price breaks. Figure 16 below shows the component cost (not including PCB manufacturing or assembly) for the hand control board based on the quantity of boards produced. PSYONIC intends to sell 40 devices within the first year on the market. This would bring the component cost down to approximately $21.20 per board.

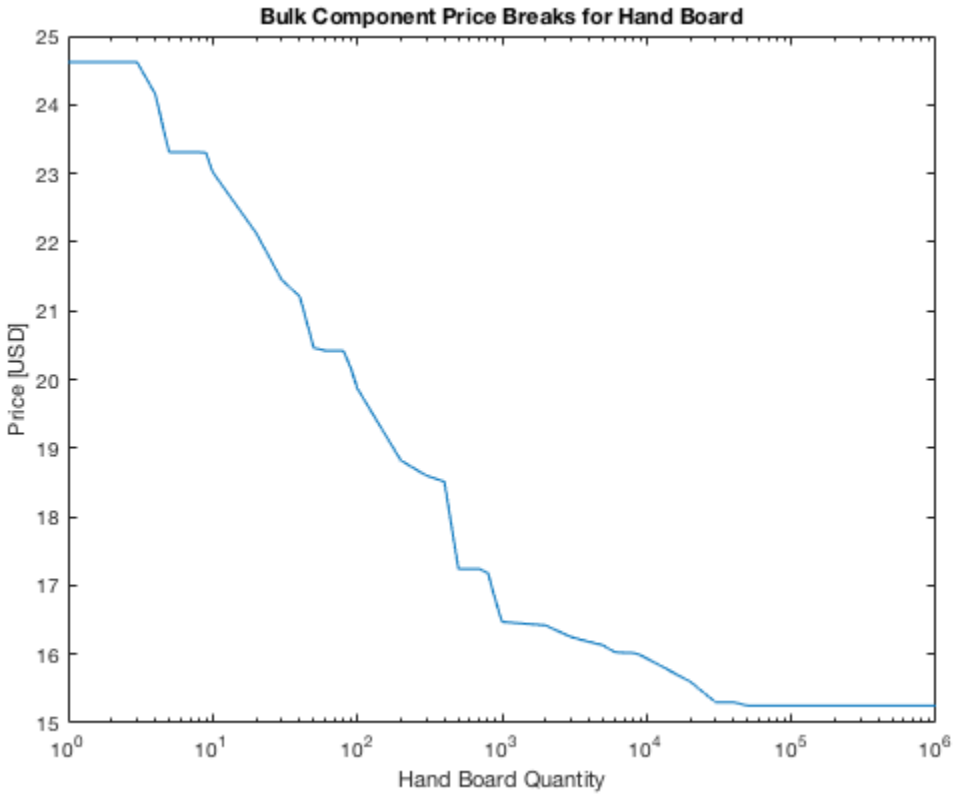

Figure 16

The estimated cost for PCB production, assembly, soldering, and testing came out to $9.85 per board (when quoting for a quantity of 40 boards). This brings the total estimated price of the finished and assembled hand control board to $31.05 each.

5.2 Labor

An estimation of labor was created by assuming a $40 hourly rate for 2 engineers. This was extrapolated for 12 hours per week across 16 weeks. To compensate for unforeseen issues that acted as time sinks, a 2.5x multiplier was attributed. This brings the total labor cost to $38,400.

$$2.5 \times 40 \times 2 \times 12 \times 16 = \$38{,}400 \quad (1)$$

Combining the labor rate with the estimated parts and PCB manufacture rate, the total project cost comes to:

$$\$38{,}400 + \$59.42 = \$38{,}459.42 \quad (2)$$

6. Conclusion

6.1 Accomplishments

Our project was a success. All requirements were fully met, and the final product can be produced for a reasonable price, something imperative for PSYONIC's goal of providing an affordable prosthesis. The hardware and software have both been handed over to our sponsor and are already integrated into a demonstration hand. PSYONIC has also already expressed interest in utilizing our software framework to replace their current code-base.

6.2 Uncertainties

Although our project did work, we did push the STM32F0 microcontroller to its limits as far as interrupt handling goes. The Nested Vectored Interrupt Controller (NVIC) that the ARM Cortex-M3 microcontroller framework utilizes can handle all the interrupts currently in place, however if a new joint was added (wrist rotation, for example), it would be difficult to implement more interrupt-driven encoders. Utilizing an alternative approach, such as a counter-based system, would relieve this. Additionally, we did have to make a couple of bodge wires across the microcontroller to be able to utilize all motor controllers effectively, as a mistake in pinouts was made due to limitations on Pulse-Width Modulation (PWM) capable pins. These issues could be rectified with minor changes in a new revision of the hardware, however.

6.3 Ethical Considerations & Regulatory Compliance

This project entailed creating a device intended to act as a medical replacement to alternative prosthetics. The finished product is intended to sit inside a patient's prosthesis and be used in real-world environments. As such, the patient health and safety are imperative. The IEEE Code of Ethics [10], subsections 1 and 5 speak directly to this by putting reiterating that health and safety are to be held paramount. Every design choice we made held these directives in mind, and it drove every action we took.

To help hold us and our project accountable to these ethical concerns, we employed a focus on regulatory compliance goals that help achieve Class 1 approval from the Food and Drug Administration (FDA). All software components were beholden to International Electrotechnical Commission (IEC) 62304 - "Medical Device Software", which specifies the need for unit-testable functions and extensive documentation [11]. Development of the hardware aspects of our project were driven by IEC 62353 - "Medical Electrical Equipment, Flammability Standard General Requirements for Safety." [12] This standard reinforced our extensive usage of ground planes and our application of a urethane conformal coating, among other things. Finally, UL94 V-1 "Flammability Standard" compliance was also achieved by

the conformal coating that was applied to the hand board. Specifically, this ensures that our PCB cannot combust after a standard specified flame test [13].

6.4 Future work

Taking this project from its current form to market necessitates a few future design changes. Most notably, switching from brushed DC motors to brushless direct current (BLDC) motors would allow for significant increase in torque and finger flexion speed. The use of counter-electromotive force (back EMF) instead of Hall effect-based quadrature encoders would allow for increased rotor velocity estimation, as well as reduced part cost. Integration with off-the-shelf electromyography solutions is intended to avoid the additional cost of gaining FDA approval for in-house EMG systems. Finally, all aspects of the design, both those within the scope of our project and those external as part of PSYONIC's larger solution, must go through third-party certification bodies such as UL or FM Global that can verify regulatory compliance.

References

[1] World Health Organization, "World Report on Disability," World Health Organization, New York, 2011. [Accessed: Dec - 2017].

[2] D. Cummings, "Prosthetics in the developing world: a review of the literature," Prosthetics and Orthotics International, no. 20, pp. 51-60, 1996. [Accessed: Dec - 2017].

[3] STM32F072RB Mainstream ARM Cortex-M0 USB line MCU with 128 Kbytes Flash, 48 MHz CPU, USB, CAN and CEC functions, datasheet, STMicroelectronics, 2017. Available at: http://www.st.com/content/ccc/resource/technical/document/datasheet/cd/46/43/83/22/d3/40/c8/DM00090510.pdf/files/DM00090510.pdf/jcr:content/translations/en.DM00090510.pdf

[4] DRV8881 2.5-A Dual H-Bridge Motor Driver, datasheet, Texas Instruments, 2015. Available at: http://www.ti.com/lit/ds/symlink/drv8881.pdf

[5] LPS22HB MEMS nano pressure sensor: 260-1260 hPa absolute digital output barometer, datasheet, STMicroelectronics, 2017. Available at: http://www.st.com/content/ccc/resource/technical/document/datasheet/bf/c1/4f/23/61/17/44/8a/DM00140895.pdf/files/DM00140895.pdf/jcr:content/translations/en.DM00140895.pdf

[6] MCP9800/1/2/3 2-Wire High-Accuracy Temperature Sensor, Microchip, 2010. Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/21909d.pdf

[7] WS2812 Intelligent Control LED Integrated Light Source, datasheet, Worldsemi, Available at: https://cdn-shop.adafruit.com/datasheets/WS2812.pdf

[8] R. O'Hara, Rgb-123.com. WS2812 Color Output | RGB-123. [online] Available at: http://rgb-123.com/ws2812-color-output/. [Accessed: Dec - 2017].

[9] En.wikipedia.org. Natural Color System. [online] Available at: https://en.wikipedia.org/wiki/Natural_Color_System. [Accessed: Dec - 2017].

[10] IEEE.org, "IEEE IEEE Code of Ethics", 2016. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: Dec - 2017].

[11] K. Hall, "Developing Medical Device Software to IEC 62304", MDDI Online, 2010. [Online]. Available: https://www.mddionline.com/developing-medical-device-software-iec-62304. [Accessed: Dec-2017].

[12] J. Backes, "Safety Testing of Medical Devices: IEC 62353 Explained", MDDI Online, 2007. [Online]. Available: https://www.mddionline.com/safety-testing-medical-devices-iec-62353-explained. [Accessed: Dec- 2017].

[13] Company, R. (2017). *UL94 V-0, V-1, V-2 Flammability Standard (UL Yellow Cards) - RTP Company*. [online] Web.rtpcompany.com. Available at: http://web.rtpcompany.com/info/ul/ul94v012.htm [Accessed Dec. 2017].

Appendix A Requirement and Verification Tables

Table 1 Microcontroller Requirements and Verifications

Requirement	Verification	Verification Status (Y or N)
1.Outputs an I^2C protocol signal 2.Communicates over I^2C	1. Connect a digital logic analyzer to the SDA, and SCL pins. Run sample script and verify output protocol. 2. Set up test with known working sensor or device with a given I^2C address. Request pertinent data from connected sensor and display on serial console. Alternatively set up another microcontroller in slave mode, send a command, interpret command on other microcontroller and output to serial console.	Y
Interpret sensor data	Connect sensor (temp/pressure) to I^2C bus of microcontroller. Request data from sensor and return the values via serial console.	Y
Detect rotor movement and give reference to what relative position the hand is in	● Assuming encoders are functioning. Connect encoders to PCB and manually move the encoder position. Run a script to detect encoder pulses, interpret the movement, and output to the serial console.	Y

Table 2 Motor Controller Requirements and Verifications

Requirement	Verification	Verification Status (Y or N)
Command the motor controller via I^2C	Connect a multimeter or oscilloscope to output pins of motor controller. Power board and send I^2C command to the motor controller. Measure signal outputted by the motor controller and verify correct operation. Alternatively connect motor and send command to move in one direction, then	Y

	verify, then send command to move in the opposite direction and verify.	
Improved connector resilience to improper use	Showcase an attempted connection in various orientations such as slightly diagonal and upside down.	Y
Connector can handle at least 1.5 amps of current	• Set up test circuit with resistor load bank to force ~1.5 amps through the connector and visually inspect connector condition. Check current throughput with power supply and multimeter.	Y
Conformal coating present	Visually inspect the surface of the PCB to verify the existence of the coating. Take a multimeter in Ohm measurement mode and attach one probe to an exposed wire (i.e. power input) and touch the surface of the coating to a part connected to exposed wire (i.e. LDO). Assuming coating is working as intended there should not be a low resistance, provided the probe did not pierce the coating.	Y

Table 3 Pressure Sensor Requirements and Verifications

Requirement	Verification	Verification Status (Y or N)
Communicate over I^2C	Connect pressure sensor to I^2C bus. Send command to pressure sensor and read data by displaying in a serial console. Continue to send requests while agitating sensor to induce pressure variance and therefore change in value for serial console.	Y

Table 4 Status LED Requirements and Verifications

Requirement	Verification	Verification Status (Y or N)
Communicate over digital input (PWM)	Turn LED on and off via microcontroller	Y
Cycle through multiple colors ● Display white light ● Display red light ● Display green light ● Display blue light	Begin a cycle sequence via serial console command to the Microcontroller where the LED is instructed to cycle between white, red, green, and blue. Observe colors outputted by LED.	Y
Light viewable form at least a 90° viewing area one foot away	● Turn the LED on red, green, and blue and verify that the color is clearly viewable one foot away from the board both straight on and 45° off normal in all directions.	Y

Table 5 Temperature Sensor Requirements and Verifications

Requirement	Verification	Verification Status (Y or N)
Communicate over I^2C	Connect temperature sensor to I^2C bus. Send a request for data from temp sensor and display read value in serial console	Y
Has accuracy of *±1°C*	Read data from temp sensor and compare value to either a temperature probe tool or (if unable to acquire tool) the temp of AC/Weather report allowing for more error.	Y
Simulate safety shutdown	● Set shutoff temperature to 37° C for testing purposes. ● Apply a hot air gun or hair dryer to achieve the 37° C. ● Verify that the LED displayed a solid red color indicating a critical error. ● Verify that no voltage is output from the motor controllers with a multimeter or oscilloscope.	Y

Appendix B Component Price Breakdown

Table 6 Parts Costs

Part	Manufacturer	Quantity	Retail Cost ($)	Bulk Purchase Cost ($)	Extended Bulk Cost ($)
504050-0691	Molex	6	1.0057	0.8839	5.3034
504051-0601	Molex	6	0.3065	0.1920	1.1520
504052-0098 (CUT STRIP)	Molex	36	0.1050	0.1050	3.7800
MCP9800A5T-M/OT	Microchip	1	1.2400	1.0000	1.0000
STM32F072RBT6	STMicroelectronics	1	2.2250	2.0024	2.0024
LP5907MFX-3.3/NOPB	Texas Instruments	1	0.5000	0.4000	0.4000
73L5R20J	CTS Components	6	0.0342	0.0257	0.1542
RC0603FR-074K7L	Yageo	6	0.0022	0.0017	0.0102
COM-11821	SparkFun	1	0.5000	0.5000	0.5000
SM04B-XSRS-ETB(LF)(SN)	JST	4	0.6465	0.5878	2.3510
2013499-1	TE Connectivity / AMP	1	0.5820	0.5820	0.5820
DRV8881PRHRT	Texas Instruments	1	1.9400	1.9400	1.9400
NX3225GA-16.000M-STD-CRG-1	NDK	1	0.6600	0.5840	0.5840
SISS23DN-T1-GE3	Vishay	1	0.4200	0.4000	0.4000
06034D106MAT2A	AVX	9	0.0300	0.0290	0.2610
02013A220GAT2A	AVX	2	0.0585	0.0585	0.1170
AC0603FR-07200KL	Yageo	1	0.0900	0.0160	0.0160
RC0603FR-07470KL	Yageo	1	0.0020	0.0020	0.0020
RC0603FR-0733RL	Yageo	2	0.0035	0.0035	0.0070
CRCW06031K50FKEA	Vishay	4	0.0064	0.0064	0.0256
GRM035R60J475ME15D	Murata	3	0.3945	0.1945	0.5835

GRM033R61A104K E84D	Murata	4	0.0166	0.0101	0.0404
Total:					**$21.20**