Prosthetic Control Board

ECE 445: Design Document — Fall 2017

Team 46: Caleb Albers & Daniel Lee

TA: Yamuna Phal

Introduction

Objective

As a company whose mission is to provide a feature rich prosthetic device, but at an affordable cost, PSYONIC has undoubtedly come to a crossroad as to which platform to continue development on. From arduino to teensyduino, PSYONIC has gone through several iterations of design and have decided upon a move to a more production-ready architecture with impressive toolchain support. This decision lead to discussion with the development team to move to an ARM based microcontroller.

Our solution to PSYONIC's need for a robust, standards-adherent, and mass-producible control board is outlined in this project proposal. We intend to design a printed circuit board that fits into the palm of PSYONIC's hand that will serve as a central controller to interface with motors, sensors, and other prosthetic hand subsystems. This board will include a revamped microcontroller architecture, multiple I^2C interfaces, a temperature sensor, a status LED, and conformal coating. Together, this hardware platform will provide a drop-in replacement for PSYONIC to use in their current prosthetic prototypes.

Background

PYSONIC aims to provide inexpensive and feature-filled prosthetic devices to the masses, with the intention to have full costs covered by insurance. In pursuant of this goal, we propose an inexpensive, water-resistant controller board and associated codebase such that PSYONIC can easily interface with finger actuators, sensors, and patient input. This board will be designed for mass-manufacturing and provide built-in safety features. Additionally, our design will be geared towards helping PSYONIC adhere to regulations put forth by the US Food and Drug Administration (FDA) such that the entire prosthetic device can be more quickly approved for resale.

High-level Requirements

- A printed circuit board must be designed to support the power and signal requirements that facilitate microcontroller operation and sufficient I^2C ports for communication with external devices.
- The microcontroller must write instructions to peripherals (led, motor driver) over I^2C.
- The microcontroller must read data from temperature and pressure sensors over I^2C.

Design

Block Diagram

Our project is broken down into three main sections. The first of which is an external section made up of the battery, EMG sensor board, and external IO board, all of which are subsystems inherent in PSYONIC's design which sit outside of the scope of our project. The second section consists of the off-board peripherals (motors, encoders, and pressure sensors), which our high-level design requirements dictate our control board must interface with. The third major section consists of the on-board components (microcontroller, status LED, temperature sensor, and motor controllers) that act together to ingest sensor information, control state, and output status to the patient. These are shown in Figure 1.

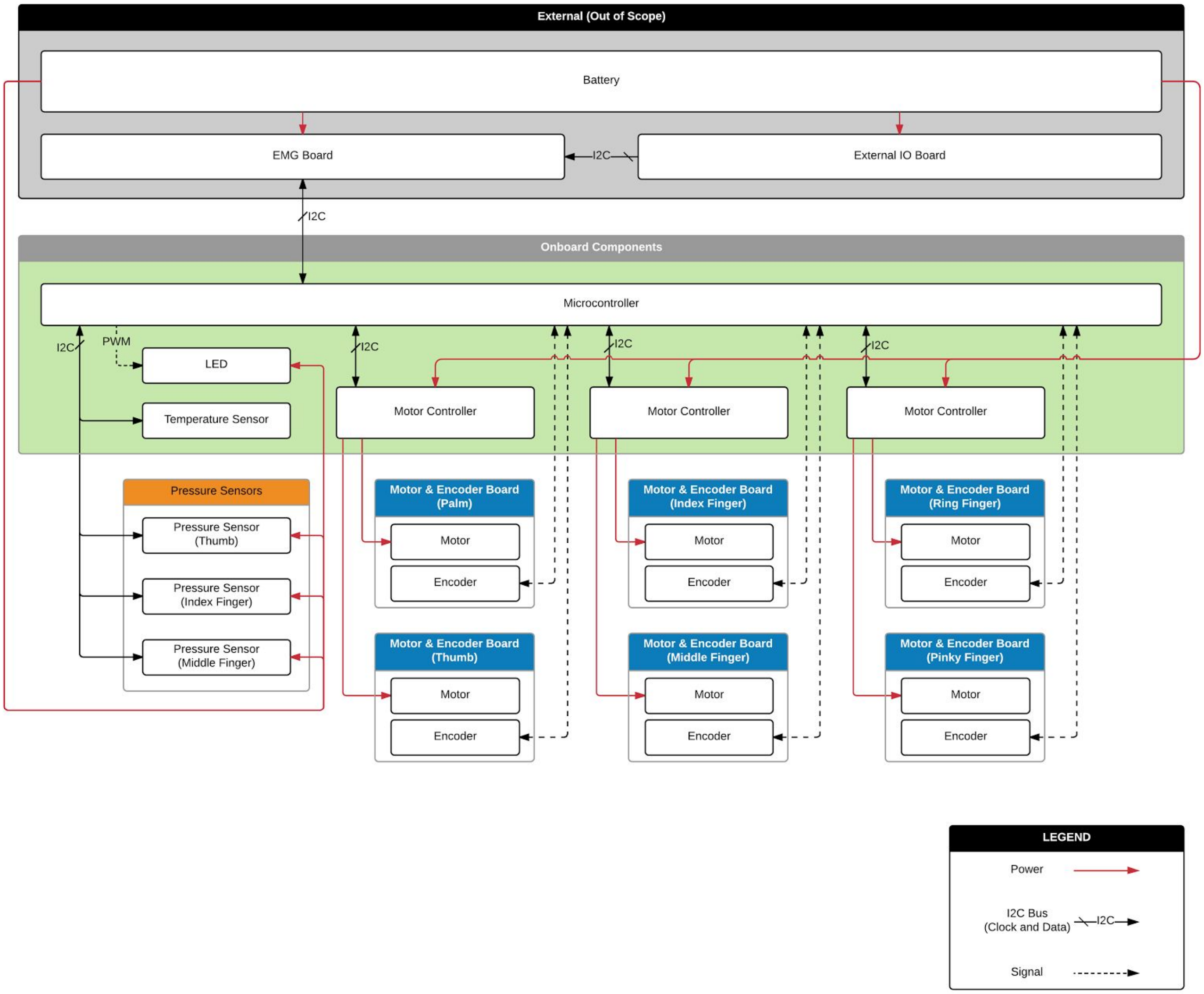


Figure 1: Block Diagram

Physical Design

The physical dimensions of our control board are restricted by the mechanical design of the prosthetic hand. As such, PSYONIC is working to finalize and send us the latest revision of dimension restrictions for the control board. Our board will be be shaped to roughly approximate the dimensions of the average female palm and fit within the current PSYONIC hand design. Figures 2 through 6 show CAD renderings of how the Control Board (artist's concept shown in Figure 5 and 6) will be mechanically placed inside the prosthetic device.

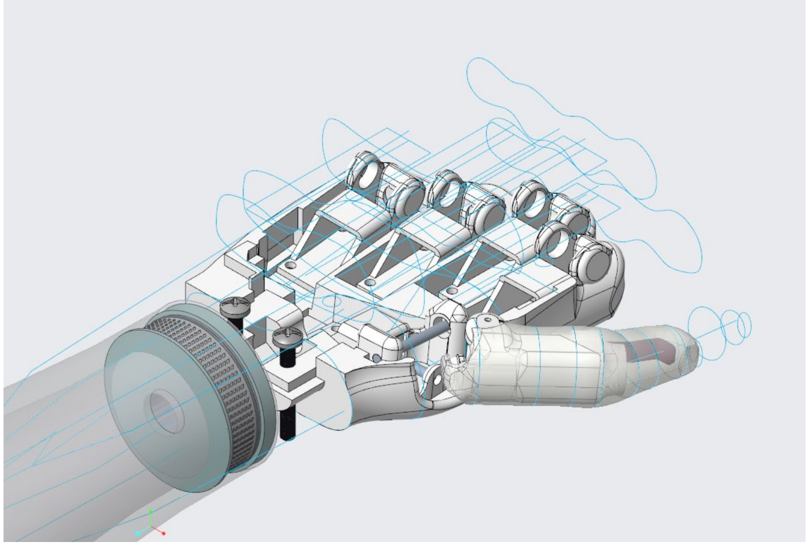


Figure 2: Upwards facing palm on which the Control Board will be placed

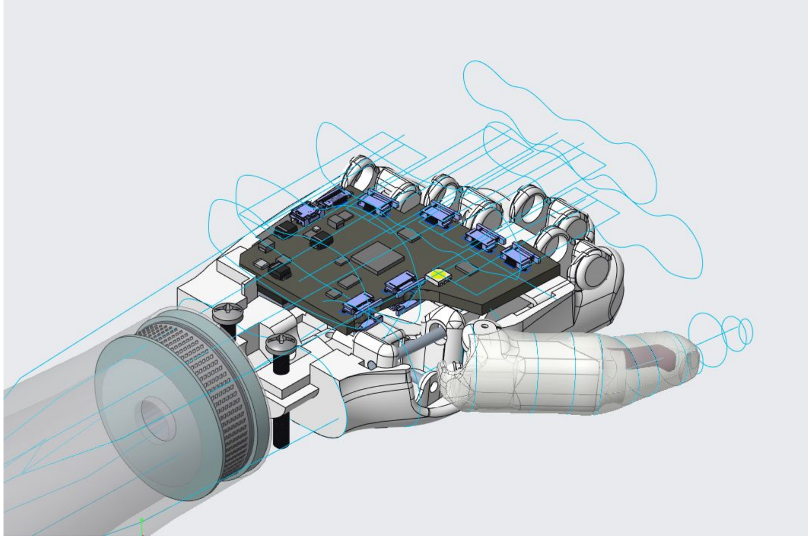


Figure 3: Mechanical mockup of Control Board on palm

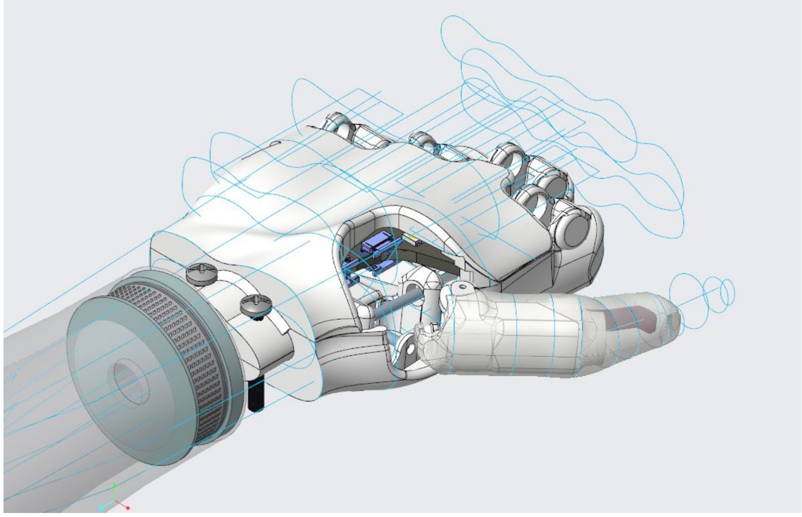


Figure 4: Rendering of closed palm on top of Control Board

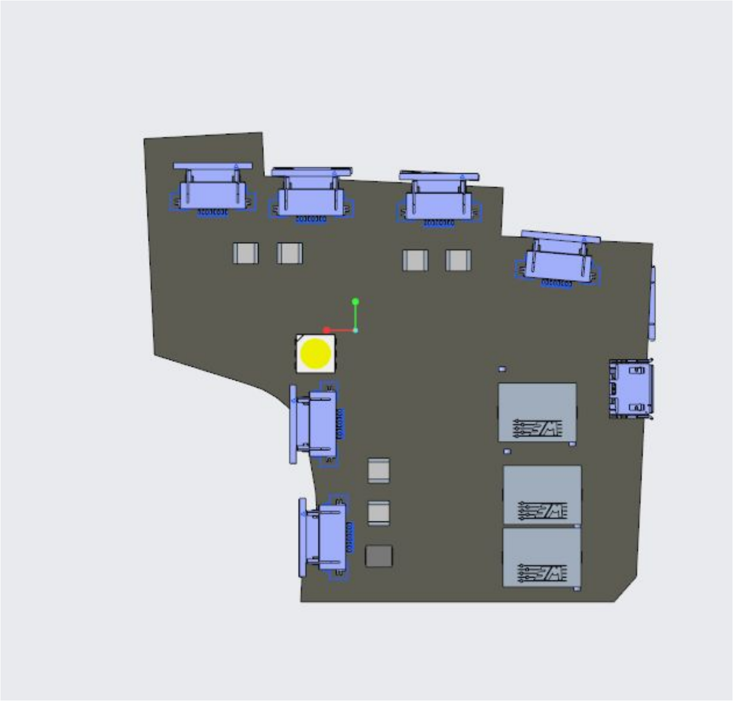


Figure 5: Render showing an artist's concept of the Control Board
(not a technical design)

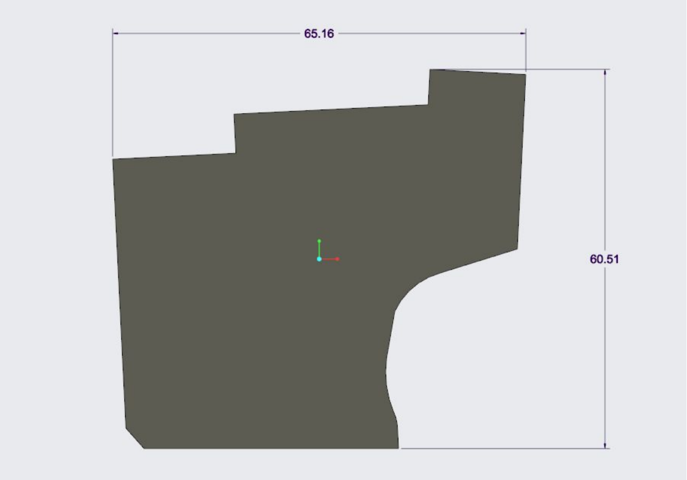


Figure 6: Artist's concept of the Control Board shape

Functional Overview

External

There are three major blocks of this diagram that, although not a part of our project, are major subsystems of the PSYONIC prosthetic arm. These subsystems include a battery to provide portable power, an EMG Sensor Board that reads input from the patient's arm, and an External IO Board that regulates USB-C charging and communicates with external devices via Bluetooth. All three of these subsystems have been encapsulated into a single block segment to represent other aspects of the PSYONIC product, which our project serves as a constituent part thereof.

Battery

A 7.4v 2200mah Lithium Polymer battery is currently used in the PSYONIC prosthetic arm to power the electronics and motors. Lithium Polymer batteries have a benefit for providing a large source of energy for a rather small and light form factor. They can also be re-charged if provided enough voltage though slow if the charging circuitry is lower than 1C (1*capacity of battery) of amperage. The downside to using lithium based batteries is the inherent danger that misuse can cause critical failure and potential external damage. Therefore protective circuitry is a must to limit the probability of such an event occurring.

EMG Sensor Board

The Electromyography (or EMG) Sensor Board is what actually allows a patient to control the PSYONIC prosthetic device. It works by sensing faint electrical signals from the skeletal muscles present in a patient's amputated arm, enhancing them, and decoding those signals into actionable grasp patterns (closed fist, only hand, et cetera). This subsystem interfaces with the External IO board for communicating with external devices. Although not directly an aspect that is under the realm of our project, the board we are producing *will* communicate with the EMG sensor board via I^2C. The EMG subsystem will send the decoded grasp patterns to our control board. The EMG sensor board also takes input from the pressure sensors our board streams to it and passes it on to other subsystems in order to provide haptic feedback to the user.

External IO Board

The External IO Board is a project a separate Senior Design group is completing. The subsystem works by utilizing the USB-C-PD standard to charge the battery and managing Bluetooth communications with external devices (such as a phone or computer). Communication between the External IO Board and our project is managed by the EMG Sensor Board and is generally out of scope of our project proposal.

Microcontroller [20 points]

STM32F072RB

We will utilize a low cost microcontroller from ST electronics that uses an ARM®Cortex®-M0 architecture on a 32 bit RISC core. The microcontroller will run up to 48 MHz with various standard communication interfaces: I^2C, SPI/I^2S, HDMI CEC and USARTs. This will serve as the brain of the hand board taking in various command, analyzing data from several sensors, and sending control signals to the motor drivers.

Requirement	Verification
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.
Interpret sensor data	Connect sensor (temp/pressure) to I^2C bus of microcontroller. Request data from sensor and return the values via serial console.
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.

Motor Controllers [15 points]

Each finger of the prosthetic hand is actuated by a single motor with a gear and spring system. The motor is mounted to a printed circuit board that contains a motor and an encoder (hall effect sensors) to measure relative motor position. These six motor and encoder combination PCBs are fed through a connector to the main PCB, where a microcontroller will read sensor data and communicate over I^2C to a motor driver that will power the motor. In an effort to save board space, each motor controller will be a dual H-bridge that is possible to drive two motors, therefore using only three motor controllers total. An external power connection will be provided by the battery, as higher voltage (7.4v) and current capacities are needed to drive the motors.

Our primary responsibility with the boards will be three-fold:

- We will replace the power and data connections between the finger/motor board and our control board by redesigning both schematics to utilize a new, more sturdy physical connector.
- We will explore ways to increase the robustness of the physical board by use of conformal coating, which adds protection from minor water damage.
- Low-level drivers will need to be created for the new microcontroller architecture we are utilizing on our control board, such that we can effectively communicate with and send commands to motor controllers, as well as read encoder values.

Requirements: command the motor controller via I^2C, read the encoder position via input pins, redesign board to make use of different physical connectors, apply a water protective coating

Requirement	Verification
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 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.
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.
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.

Pressure Sensors [7 points]

In addition to vast freedom of movement the prosthetic hand posses, each finger-tip is outfitted with a pressure sensor to act as a touch sensor. The data received from the pressure sensor can then be used to provide feedback to the user as a reference to what degree the fingers are imparting a force in its grasp. The feedback is fed as data to the EMG board which will perform haptic feedback based on the data fed from the microcontroller and in turn the sensors. These sensors are being implemented outside of our project, though we will be working towards interfacing with them.

Requirement	Verification
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.

Additional rated part requirements: operate between 1.95 to 3.6V, rated to work at -10℃ to 50℃, and accuracy of ±100pa.

Status LED [5 points]

Every product needs some method of communicating information to the user, and in the case of the prosthetic hand, it is two-fold as both status and error messages can be provided to a patient and also serve as a debugging tool during product development.

Requirements: communicate over I^2C, have multiple colors (red, green, and blue), and have a light viewable from at least a 90° viewing area.

Requirement	Verification
Communicate over digital input (PWM)	Turn LED on and off via microcontroller
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.
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.

The status LED will be used to give a visual indication as to the mode and state of operation, in accordance with Table 1:

Mode	State	LED Color
General Use	No Power / OFF	No Light
General Use	Normal Operation	Blue - Solid
General Use	Booting/Initializing	Alternating Orange/Yellow
General Use	Battery Low	Red - Blinking
General Use	Battery Charging	Blue - Blinking
General Use	Error - Non-critical	Orange - Solid
Critical Error	Error - Critical	Red - Solid
Configuration	Machine Learning Classifier - Training	Breathing (purple/blue/green/yellow)
Configuration	Machine Learning Classifier - Recording Data	Breathing (orange/red/yellow)
Configuration	Machine Learning Classifier - Data Collection Done	Green - Solid

Table 1: LED State Indicator

Temperature Sensor [3 points]

In consideration of the environment and use case envisioned for PSYONIC, extra safety systems and feedback greatly improves the reliability and safety of the product. This is where a temperature sensor comes in handy. The sensor will communicate via I^2C with the microcontroller and send back the temperature of the control board. This will act as a fail safe in the event a patient accidentally picks up or spills a hot item or liquid on to the control board, or if the motor controllers overheat.

Requirements: stream temperature data over I^2C *and have the microcontroller safely shut down during overheating. A temperature sensor with a tolerance equal to or greater than ±1°C and an operating range from at least –20°C to 80°C must be selected.*

Requirement	Verification
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
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.
Simulate safety shutdown	1. Set shutoff temperature to 37° C for testing purposes. 2. Apply a hot air gun or hair dryer to achieve the 37° C. 3. Verify that the LED displayed a solid red color indicating a critical error. 4. Verify that no voltage is output from the motor controllers with a multimeter or oscilloscope.

Risk Analysis

The biggest risk posed to the success of this project is in respect to the low-level programming inherent in introducing a new Microcontroller architecture. Core libraries for interfacing with the motor controller, encoder, LED, temperature sensor, and EMG board must all be created on top of the I^2C protocol. Additionally, higher-level code will also need to be written to make use of the input from the EMG board, process it into actionable commands, and act as a fabric to work with the motor controller/encoder pair to actuate the motors.

Debugging is inherent when creating low-level drivers, and data sheets can be vague at times. These both dramatically increase the time spent developing and verifying correct operation. The ARM architecture and the toolchain we will be utilizing is new to us, so a learning curve will also be present in transferring skills we might otherwise be proficient in (for other microcontrollers) into those necessary for the success of this project.

An additional risk comes in the fact that the steps for applying conformal coating and/or potting to the circuit board designs must be completed *after* the designs have been created, manufactured, assembled, tested, and verified. Both methods of adding water resistance make it difficult to debug or replace parts in the circuit board after the coating has been applied. Although the application process is fairly easy, the waterproofing must be dealt with *after* the rest of the hardware aspects have been completed, which sidesteps the modularity and concurrency strived for in the rest of our project layout.

Schematic

Main Hand Board

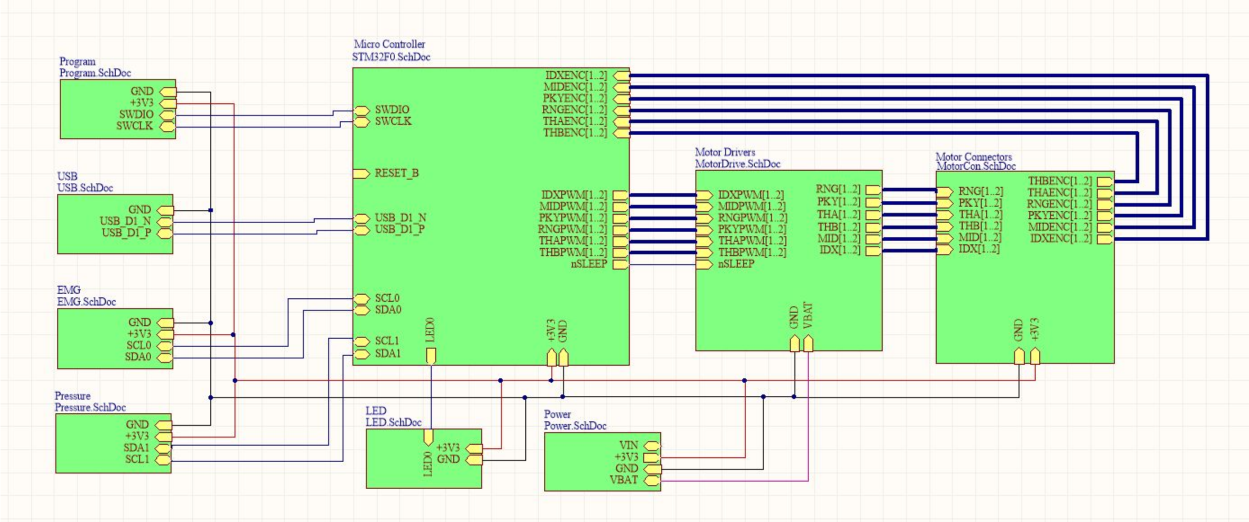


Figure 7: High level overview schematic of the handboard, thicker traces denote a wire bundle (Larger image attached see Appendix A)

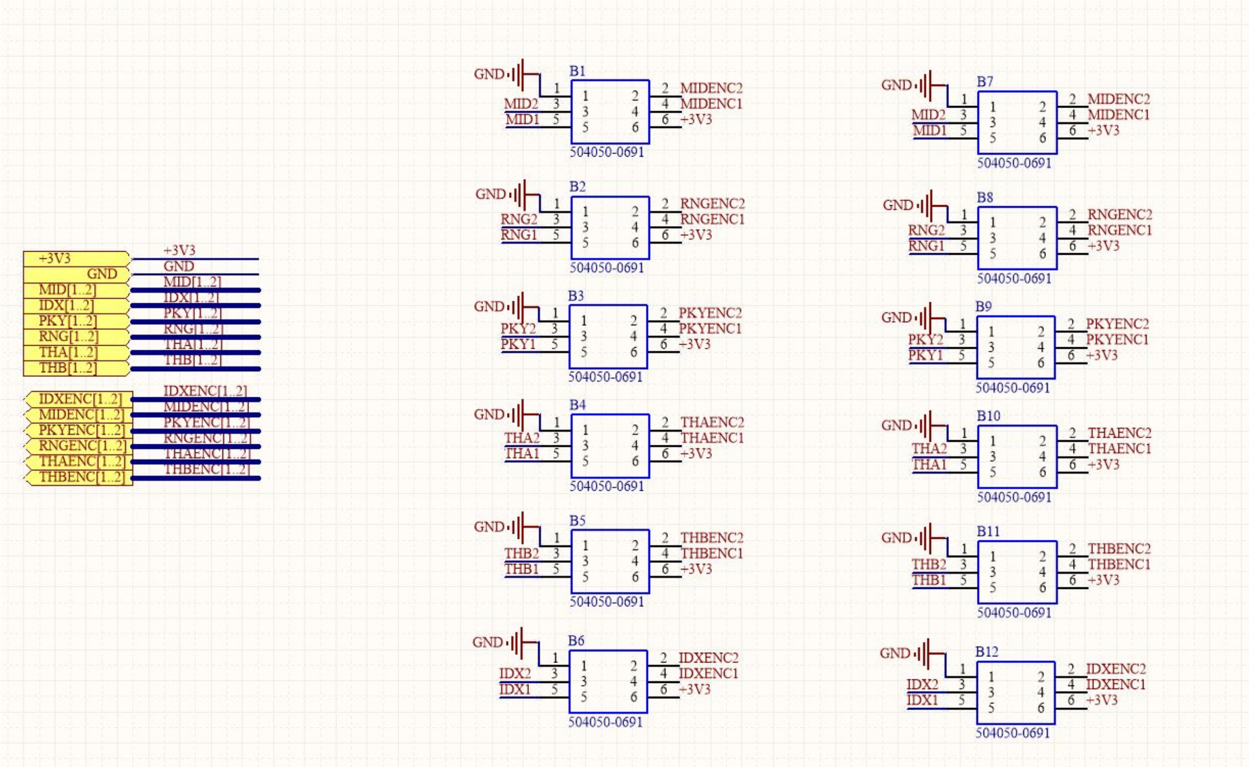


Figure 8: Connectors for motor and encoder board

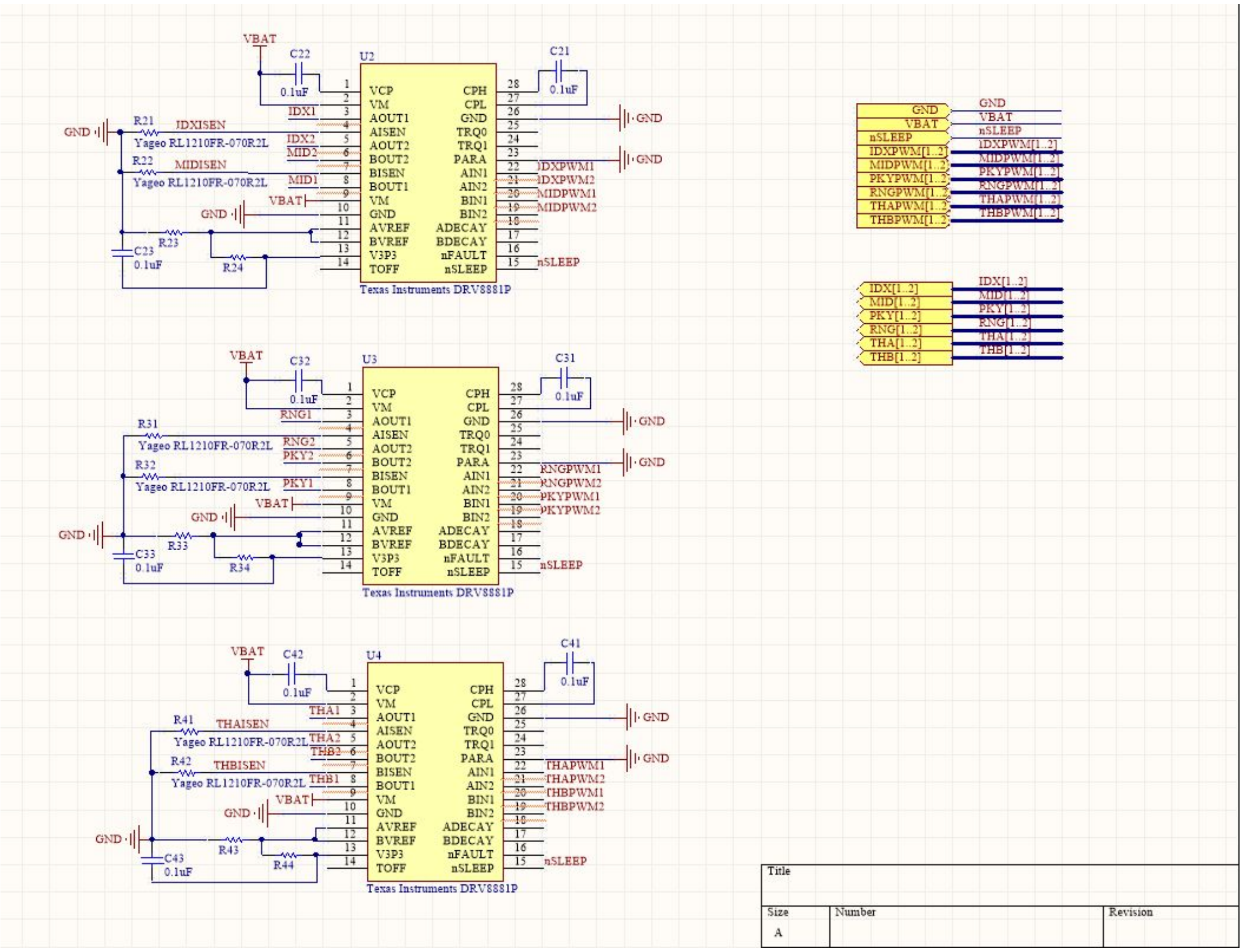


Figure 9: Brushed Motor Drivers

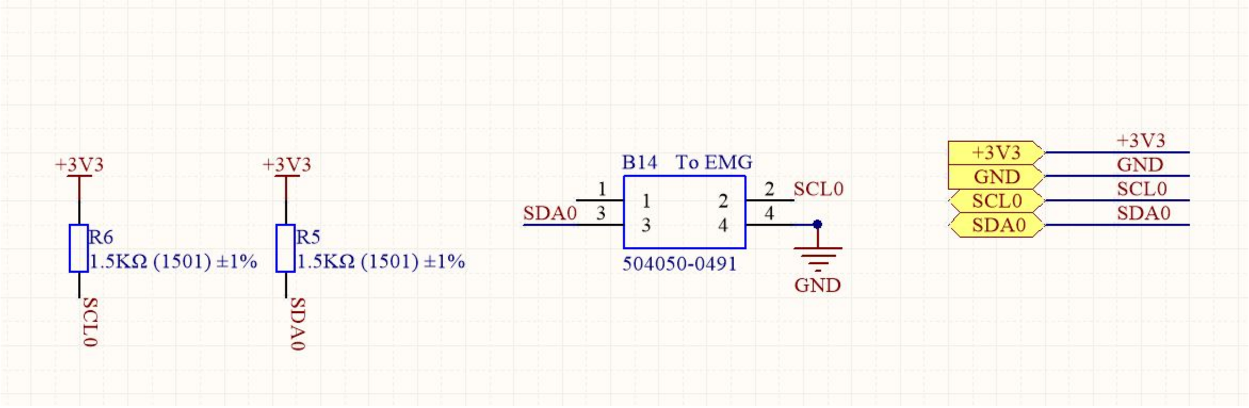


Figure 10: Connector and pull-up resistors for EMG I^2C bus

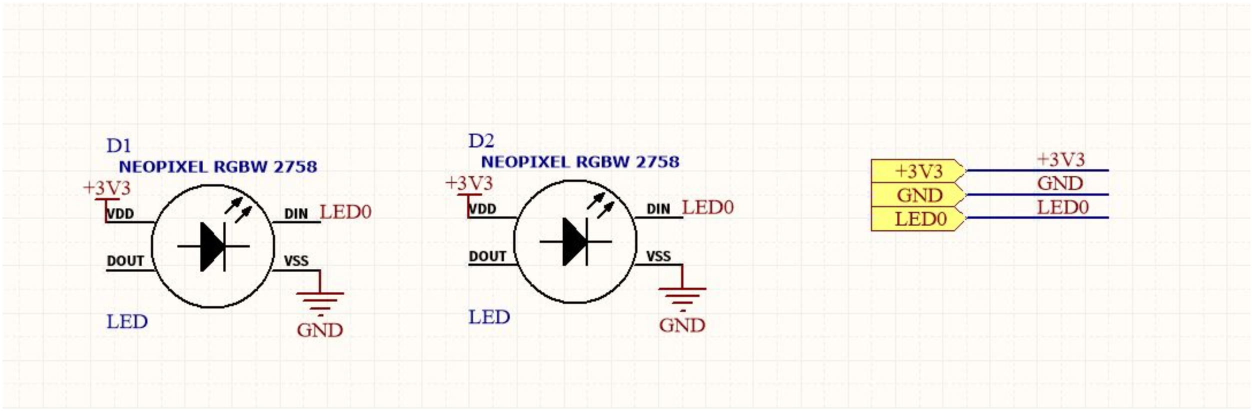


Figure 11: RGB LED module

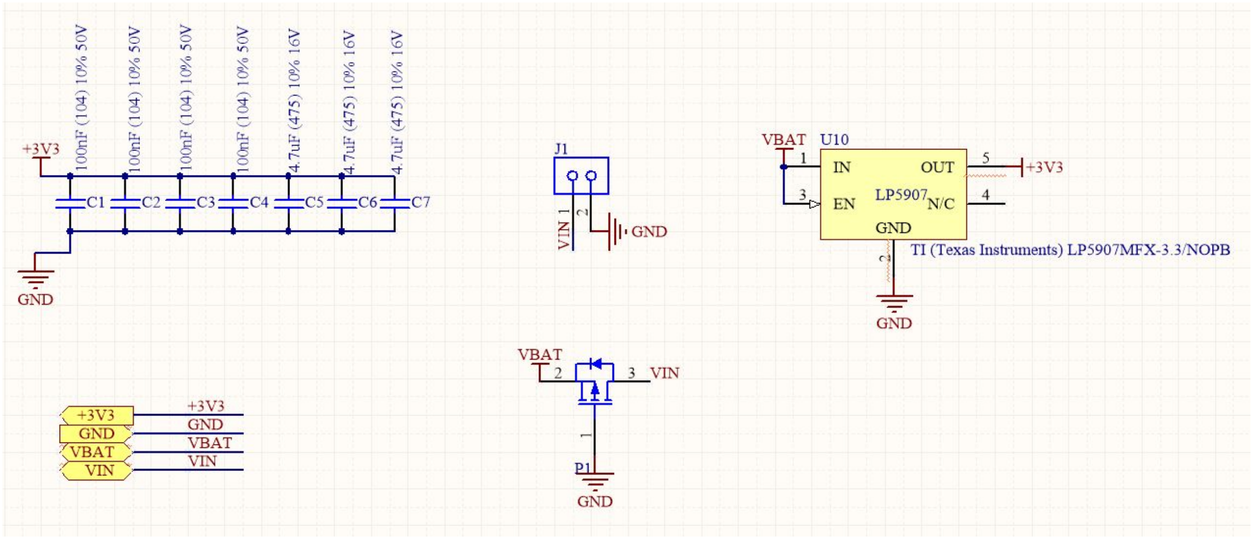


Figure 12: Power connections to board with capacitor bank, 3.3V LDO, and protection mosfet

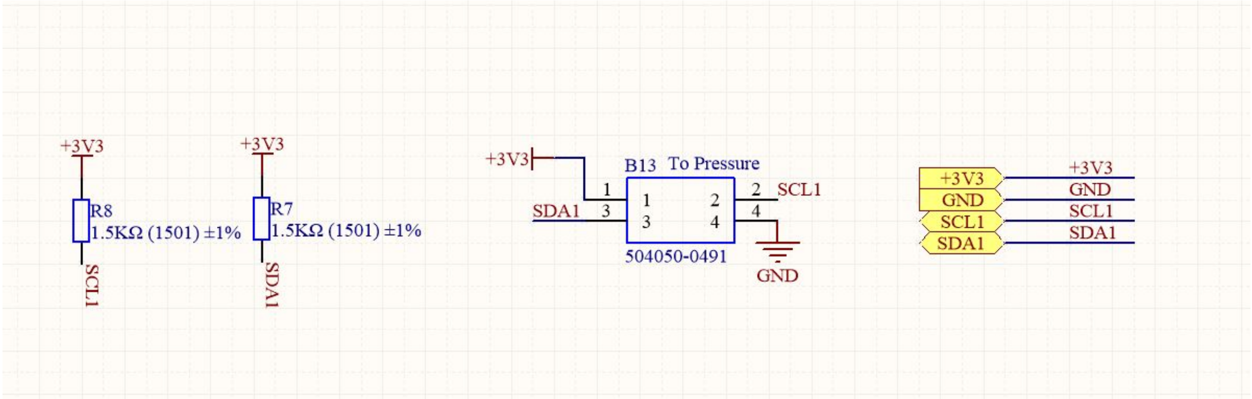


Figure 13: Connector and pull-up resistors for pressure sensor I^{2}C bus

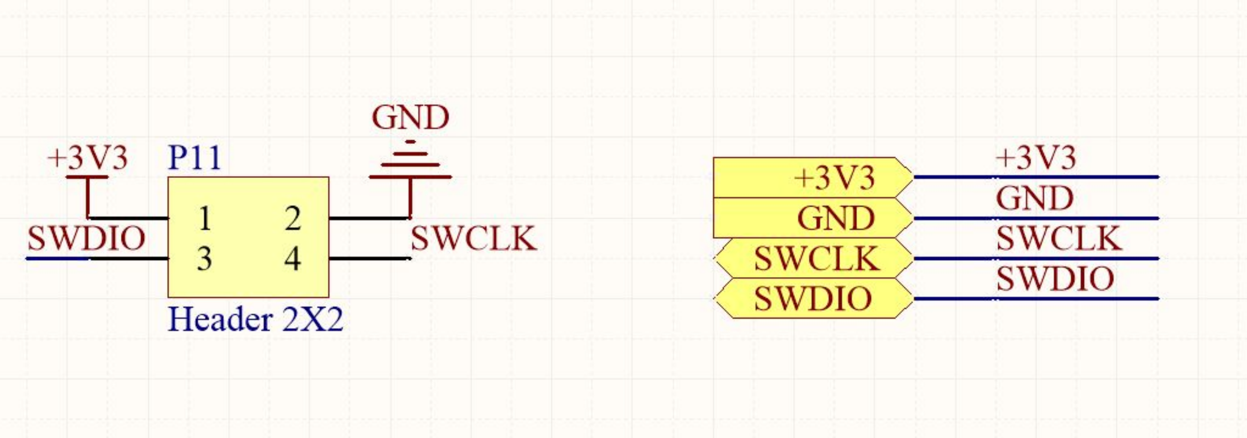


Figure 14: Programming header for the microcontroller

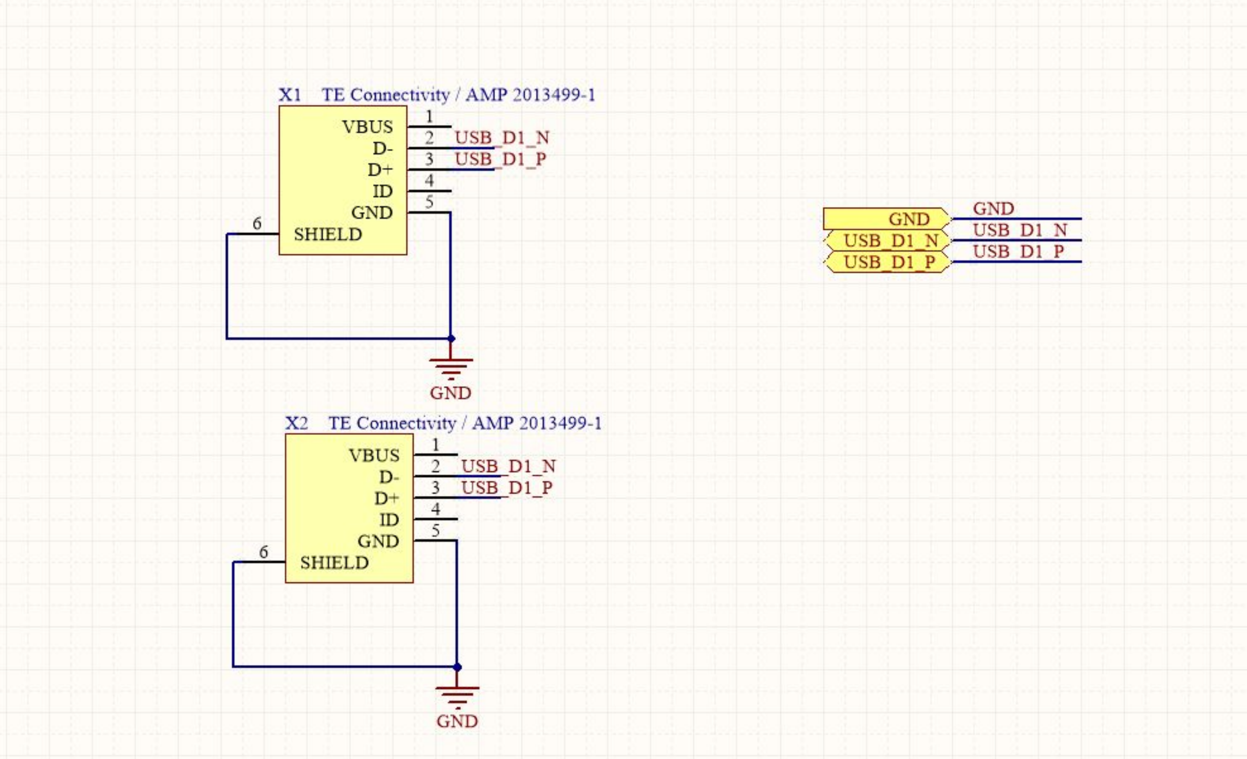


Figure 15: Micro USB type B connector

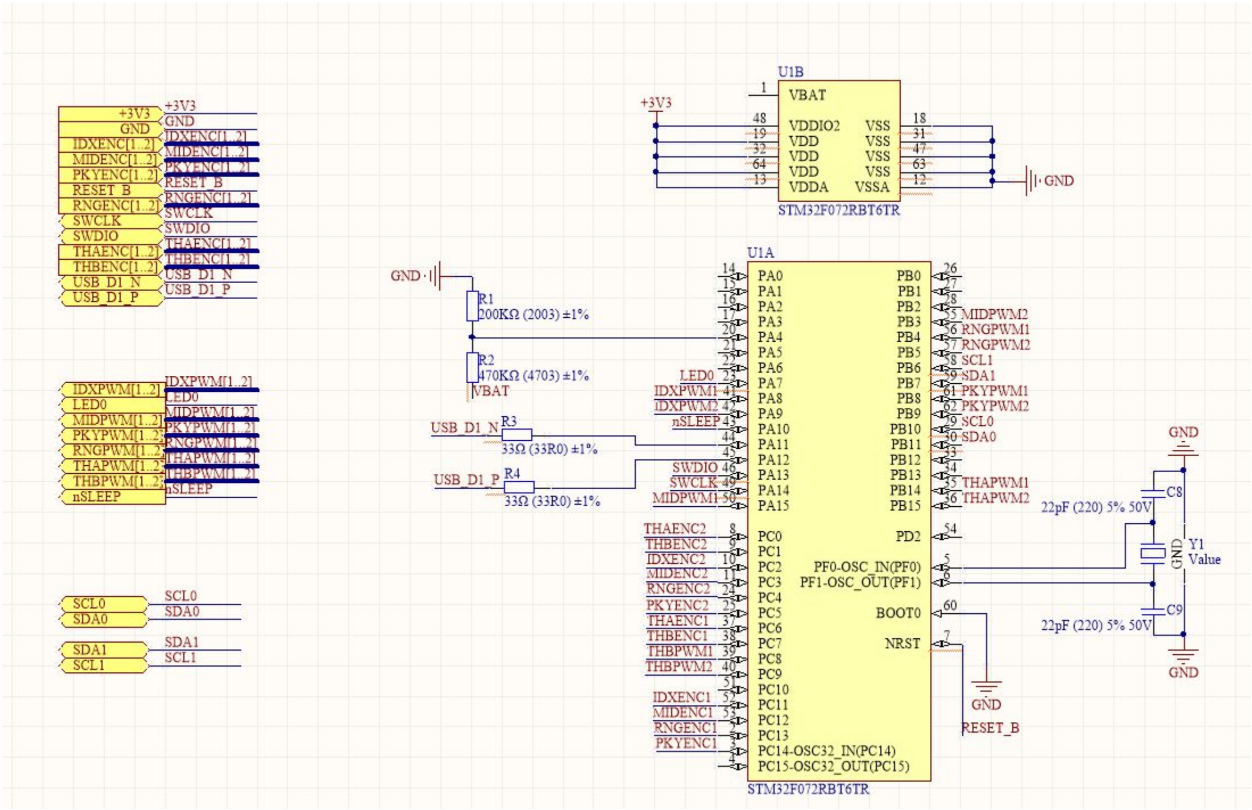


Figure 16: STM32 microcontroller

Motor and Encoder Board

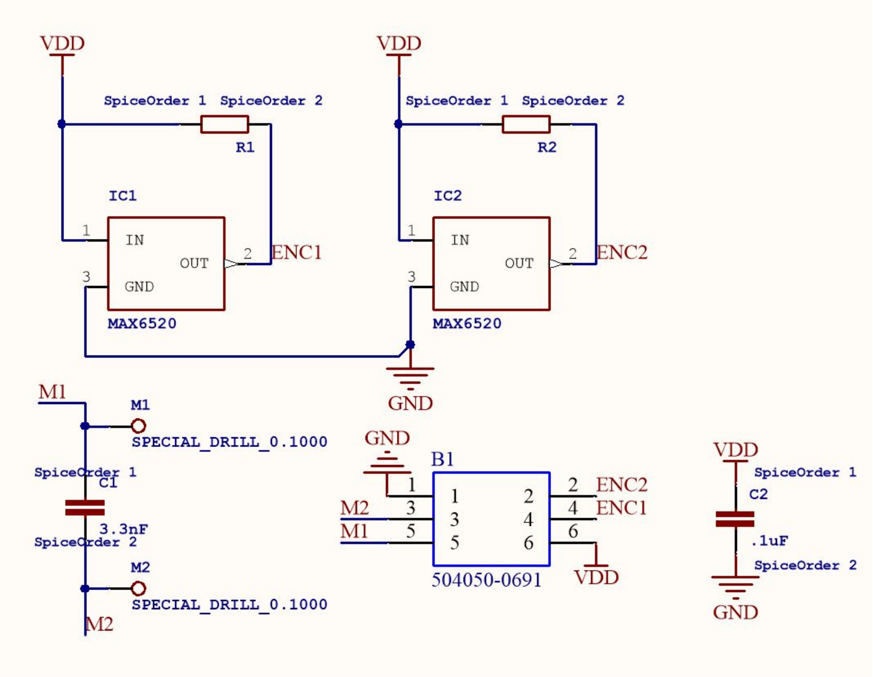


Figure 17: Encoder board schematic with magnetic encoders, capacitors, and connector

Operating Modes, States, and Initialization

The software for this project has several modes of operation, dictated by the use cases PSYONIC has provided. In general, there are several major modes that are required:

1. **Off**

   This is the default mode for when power is not applied.
2. **Initialization**

   Several steps must be taken when power is first applied. The microcontroller enters a bootup sequence and begins execution of a startup routine. That startup sequence checks the status of sensors, initializes communication with the EMG subsystem, and finds the relative position for the motor encoders.
3. **General Use**

   This mode is would be perceived as "normal" to the user. While in this state, the control board takes input from the EMG subsystem and executes the given actions (i.e. extend a finger).
4. **Configuration**

   This mode is primarily for use in a diagnostic setting in order to customize the hand for a given patient. During this mode, the prosthetic arm will be hooked up to a computer and the EMG board will act as a data acquisition device. A machine learning classifier (not in the scope of this project) will record data and run training procedures. The purpose of this mode with respect to our project is simply to pass data along to the EMG subsystem and display a status on the LED.
5. **Critical Error**

   This state does not describe a true "operational" mode, but is present on state transition diagrams in an effort to show the implementation of exception handling for critical errors (i.e. thermal overruns).

Figure 18 below visualizes in a flowchart form the startup sequence and state transformations expected to occur when power is initially applied to the hand control board:

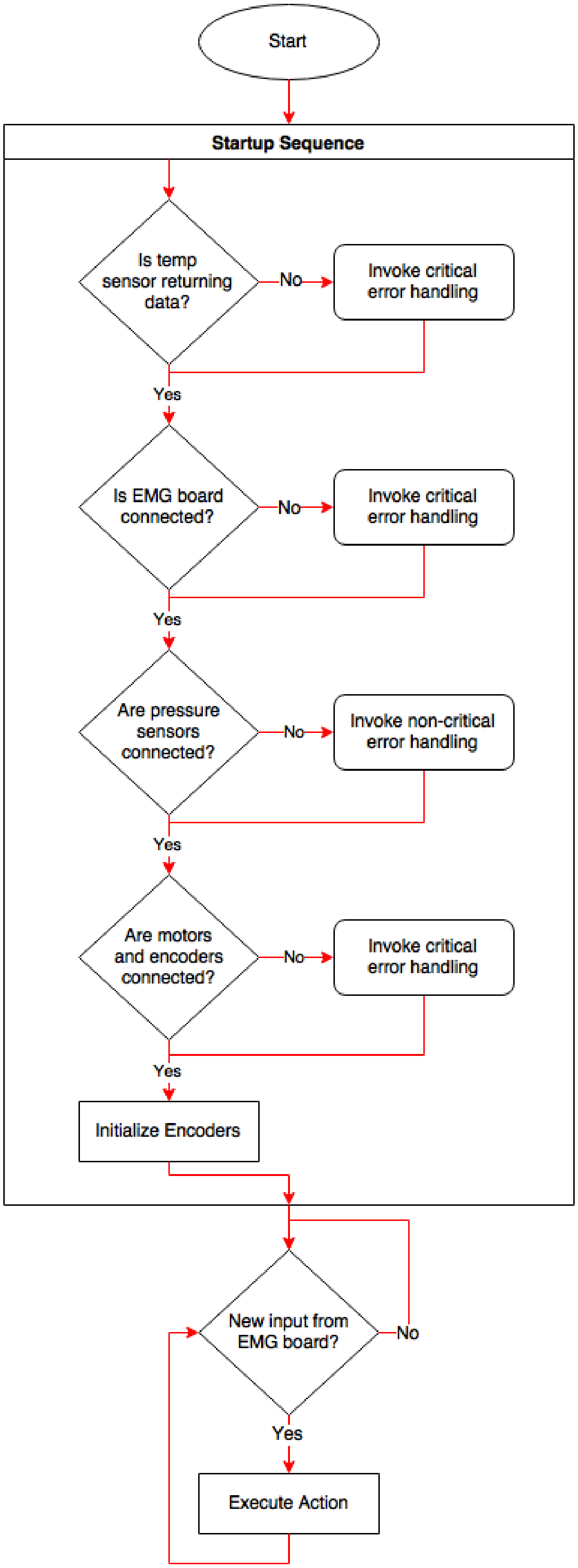

Start
Startup Sequence
Is temp sensor returning data?
No
Invoke critical error handling
Yes
Is EMG board connected?
No
Invoke critical error handling
Yes
Are pressure sensors connected?
No
Invoke non-critical error handling
Yes
Are motors and encoders connected?
No
Invoke critical error handling
Yes
Initialize Encoders
New input from EMG board?
No
Yes
Execute Action

Figure 18: Initialization and Boot Sequence

As mentioned, there are several major modes of normal operation for our project and a critical error handling state. The transition between these modes are shown on Figure 19 as a finite state machine diagram:

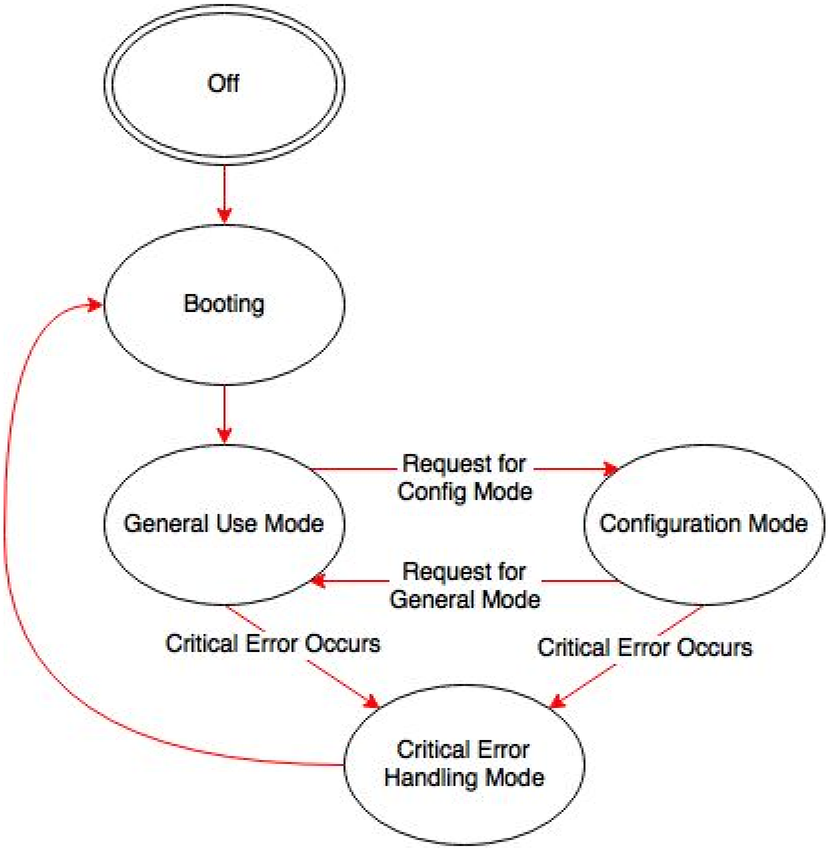


Figure 19: FSM for Operating Modes

There is a separate state machine that operates outside the realm of the aforementioned modes. This is in an effort to describe the functionality of non-critical error handling that will be exhibited in our code. For example, a motor or pressure sensor being disconnected will represent a non-critical error -- an error that might affect operation of the device, but does not affect the ability for other functions to run. This non-critical error handling state diagram is shown below in Figure 20.

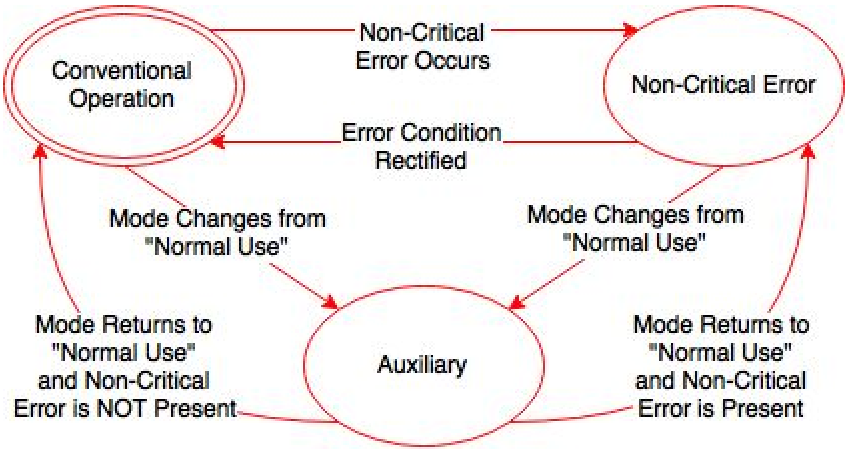

Conventional Operation
Non-Critical Error Occurs
Non-Critical Error
Error Condition Rectified
Mode Changes from "Normal Use"
Mode Changes from "Normal Use"
Mode Returns to "Normal Use" and Non-Critical Error is NOT Present
Auxiliary
Mode Returns to "Normal Use" and Non-Critical Error is Present

Figure 20: FSM for Non-Critical Error Handling

The “Conventional Operation” state is enacted whenever the board is in the “General Use” mode or the “Configuration” mode. The “Non-Critical Error” state is entered whenever a non-critical error occurs while the board is in “General Use” or “Configuration” mode. Any other modes “Critical Error”, “Off”, or “Booting” transition the machine into the “Auxiliary” state.

Development and Project Costs

We estimated a fixed development cost by estimating a pay rate of $40 per hour, multiplied by two people working. We expect to work 12 hours per week, across 14 weeks in order to finish this project. Multiplying this figure by 2.5 gives a fixed cost of:

$$2 \cdot \frac{\$40}{hr} \cdot \frac{12\ hr}{wk} \cdot 14\ weeks\ \cdot 2.5\ =\ \$33,600$$

For the purposes of calculating a per-board cost, we created a bill of materials showing the price of parts when not buying in bulk, as seen below:

Name	Model #	Units	Unit Cost	Total
Temperature sensor	MCP9800A5T	1	$1.26	$1.26
6-pin Molex Pico-Lock Female Header	504050-0691	6	$0.36	$2.16
6-pin Molex Pico-Lock Male Assembly	504051-0601	6	$1.17	$7.02
4-pin Molex Pico-Lock Female Header	504050-0491	4	$0.15	$0.60

4-pin Molex Pico-Lock Male Assembly	504051-0401	4	$1.13	$4.52
Molex Pico-Lock Cable Crimp Tab	504052-0098 (Cut Strip)	100	$0.11	$10.50
100nF capacitor (104)	81-GCG21BR91H104KA3L	4	$0.10	0.25
4.7uF capacitor (475)	81-GRM035R60G475ME5D	3	$0.10	0.42
22pF capacitor (220)	581-02013A220GAT2A	2	$0.10	$0.20
AVX Interconnect / Elco Ceramic Capacitors 4V 10uF	06034D106MAT2A	9	$0.38	$3.42
RGB LED	WS2812	1	$0.50	$0.50
Molex header pin	22-28-4022	2	$0.10	$0.20
Vishay Mosfet	SISS23DN-T1-GE3	1	$0.93	$0.93
200K resistor (2003)	603-AC0603FR-07200KL	1	$0.24	0.1
470K resistor (4703)	603-RC0603FR-07470KL	1	$0.10	$0.10
33 resistor (33R0)	603-RC0603FR-0733RL	2	$0.10	$0.20
1.5K resistor (1501)	71-CRCW0603-1.5K-E3	4	$0.10	$0.40
Yageo current sense .2 ohm resistor	RL1210FR-070R2L	6	$0.59	$3.54
Vishay Dale resistor 47K resistor	CRCW06034K70JNEC	6	$0.10	$0.60
Micro Controller	STM32F072RBT6TR	1	$4.44	$4.44
Texas Instruments motor driver	DRV8881PPWPR	3	$3.64	$10.92
TI 3.3v LDO	LP5907MFX-3.3/NOPB	1	$0.36	$0.36
Micro USB Connector	2013499-1	1	$1.78	$1.78
PCB Cost		5	$1.00	$5.00

Table 2: Bill of Materials

Parts Cost: $\$59.42$

Summing the fixed development costs with the estimated part expenditure for a single board (our goal) gives a total project cost of:

Total Cost: $\$33,600 + \$59.42 = \$33,659.42$

The point of our project is to implement the control board, however we are making a modification to the encoder board that is present in every finger on the prosthetic device in order to change the 6-pin connection used (at the request of PSYONIC). As such, we are not counting the entire bill of materials for the encoder board towards our total project cost, however we have counted the cost of the new connectors.

Tolerance Analysis

The velocity feedback is given by an encoder board that is implemented on a pcb solder to the back of the motors. It utilizes the hall effect to sense magnet disturbances which are caused by a six position rotor attached to the shaft of the motor. The orientation of the sensors allow us to grab data from both of the sensors by utilizing the edge of the signal output.

$$(\# \, region) \, * \, (\# \, counted \, points) \, = \, 6 \, * \, 2 \, = \, 12 \, CPR$$

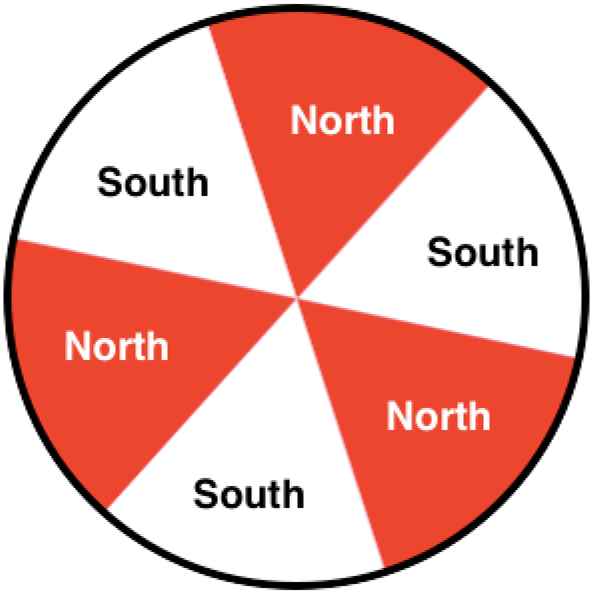


Figure 21: Illustration of Rotor

The motor these are paired with, have key specs of 320 RPM with 20 mA at no load, 30 oz-in (2.2 kg-cm) and 1.6 A at stall at 6v. This low speed is due to an attached gearbox with a gear ratio of 100.37:1.

$$(RPM) \, * \, (Gear \, ratio) \, = \, 320 \, * \, 100.37 \, = \, 32118.4 \, RPM$$

This new calculated value is the theoretical speed at which the motor is operating at and therefore what the rotor magnet is spinning at.

$$(RPM) * (CPR) \;=\; 32118.4 * 12 = \; 385420.8\; CPM$$

Since motors can have inconsistent results due to the load applied, lifespan of motor, etc. a tolerance of 10% is applied to give a reasonable operating range. Since we are moving to a new architecture and code base, this leads us to figuring out what frequency will we have to poll for data.

$$(CPM) \;*\; tolerance \;= 385420.8 \;*\; 1.1 \;=\; 423962.88\; CPM$$

Converting this number to hertz to see the frequency of data collection.

$$CPM/60 \;=\; 423962.88/60 = 7066.048\; Hz$$

Rounding this number to a nice whole number gives us approximately 7KHz of data to parse through. Compared to the operating frequency of our microcontroller chosen (48MHz), there is a relatively large amount of cycle time at our disposal to accurately read the speed and interpret position.

Schedule

Week	Caleb	Daniel
10/2/2017	Mock Design Review Finish Design Document	Mock Design Review Finish Design Document
10/9/2017	Design Review Setup STM32 programming environment and review PSYONIC's current codebase Verify schematics and order PCBs	Design Review Finish PCB layout for the control boards Order parts
10/16/2017	Setup Doxygen and create skeleton for each function, listing appropriate input and output	Plan any architectural changes to PSYONIC's current function list
10/23/2017	Assemble control board Interface with LED and temp sensor via	Assemble encoder board(s) Interface with motor controller to drive

	I2C	motor via input from microcontroller
10/30/2017	Implement initialization sequence, normal operation mode, and learning mode.	Implement communication with EMG subsystem over I2C
11/6/2017	Implement critical error handling Apply conformal coating	Create test cases for operation modes Test thermal shutdown
11/13/2017	Test operation with other PSYONIC subsystems	Verify project meets PSYONIC requirements
11/20/2017	Work on Final Paper	Work on Final Paper
11/27/2017	Mock Demo	Mock Demo
12/4/2017	Demonstrate Project Finish Final Paper	Demonstrate Project
12/11/2017	Project Presentation	Project Presentation

Table 3: Weekly Schedule Breakdown

Ethics and Safety

The PSYONIC prosthetic arm ***is*** intended to be a medical device and, as such, carries a large selection of ethical and safety concerns. Our project is a contribution to a larger end product that is slated to go through FDA certification before being sold. As such, every aspect of our design choices must be influenced by the concern for patient safety and security. Subsections 1 and 5 of the IEEE Code of Ethics [1] make direct reference to the health and safety of people, as well as the directive of applying technology and understanding potential ramifications, respectively. These directives speak directly to the importance of the design choices we make.

The hardware design of our control board will essentially be in the palm of a patient’s [albeit prosthetic] hand. As such, safety systems need to be in place to protect the patient from any sort of electrical shock, unanticipated motor movements, overheating, or other unforeseen harms. With that said, the device still needs to strike a balance of affordability and flexibility while maintaining those rigorous standards.

The first aspect, dealing with preventing electrical shock, is handled in our design via several methods. The first of which is utilizing low-voltage signals (less than 15v for our purposes) throughout the entirety of our control board design. Additionally, implementing best practices such as properly sized ground planes on the circuit board can help absorb any unintentional over-current from accidental electrical shorts. The implementation of conformal coating is an additional step we are taking to protect both the patient and the medical equipment from accidental liquid spills and dust. Conformal coating works by adding a water-resistant [typically] silicone layer across the entirety of the circuit board components, which acts as a barrier between metal and any liquids.

Given patients will have no sense of temperature coming from their prosthetic hand, the possibility of unintentionally picking up a hot object is inherent. A temperature sensor is being positioned near the microcontroller on the control board towards the center of the design such that extreme temperatures can be quickly noticed. These high temperatures can act as safety stops, sending an interrupt to the microcontroller to fail safely (de-energize motors) and notify the patient via LED of a potential thermal-related error. Additionally, the application of conformal coating or potting has the ability to be used as a thermal heat sink, effectively aiding in thermal protection for our circuit board.

We are testing several types of interlocking 4-pin connectors to use between our control boards and motor control boards. These connectors will provide more robust and secure ways to send signals to and from other subsystems without the possibility of a patient accidentally unplugging a device by hyperextending a finger or otherwise physically impacting the prosthetic device.

The software and firmware side of this project is not to be underestimated from a safety and ethics perspective. The first concern is that the PSYONIC device has little in the way of mechanical fail safes for the finger actuators. Due to the nature of their design, the primary method for mechanical safety is the limit on stall torque that the brushed motors can give. Previous examples of medical devices, notably the *Therac-25*, that lack hardware interlocks and rely solely on software requirements have taught us that appropriate software design directives must be taken in order to prevent harm to patients [2].

The EN/IEC 62304 standard defines software practices that are necessary to promote patient safety. To be specific, our project would be under the scrutiny attributed to Class B software, which is defined as having the potential for nonserious injury [3]. That classification carries with it the expectation that appropriate documentation will be present such that each software unit functionality is sufficiently stated and can be unit tested by continuous integration systems. This will be achieved by concretely defining functions such that their purpose is very specific, with any additional complexity to be completed at a more abstract level. Defining the expected input and output in each function documentation will allow PSYONIC to pursue unit testing to meet FDA requirements more easily in the future.

Additionally, we will be auditing documentation and supplementing definitions for Software of Unknown Provenance, which represents supplementary libraries that are used. Just because libraries or other code bases have worked in the past does not inherently mean they are standards compliant or to be trusted, as the aforementioned Therac-25 showed.

The other primary concern with any medical device is the security of patient data. The Healthcare Insurance Portability and Accountability Act (HIPAA) dictates efforts that must be extended to protect patient confidentiality. To expand on this, the HIPAA Privacy Rule [4] specifies that PSYONIC would be a Covered Entity, which means they are a stakeholder in securing Protected Health Information (PHI). In order to approach these concerns with best practices in mind, our goal is to reduce attack vectors that could disclose PHI. As such, every aspect of code implemented for our control board will be done in such a way that no patient-specific identifiers or settings will have to be stored in memory. Instead, any settings pertaining to the patient will only be present on the EMG board and those will drive whatever input is given to the control board. The Health Information Technology for Economic and Clinical Health Act (HITECH Act) sets forth requirements about what constitutes a breach of confidential data, and the further specifies that these events must be responsibly disclosed [5]. By removing the potential for patient-specific information to be stored in our project's scope, we limit the potential for that data integrity to be breached.

Finally, our concerns extend to the ensurance that PSYONIC will be able to effectively use the deliverables we give them, be it software, hardware, or both. This extends to the potential for our control board designs and codebase to be used in both the current hand implementation, as well as future designs. In order to achieve this, our design and code-base will be made as general as possible. For example, we will strive for our board layout (which is currently for a right hand) to be developed in such a way that the layout can be flipped to accommodate a left hand. Likewise, our codebase will strive to make all functions general, flexible, and modular in such a way that it can be easily utilized for left-hand operation, or be extended in the future. This is especially important if PSYONIC intends to add features or functionality. It also addresses best practices aforementioned pertaining to the pursuit of FDA approval.

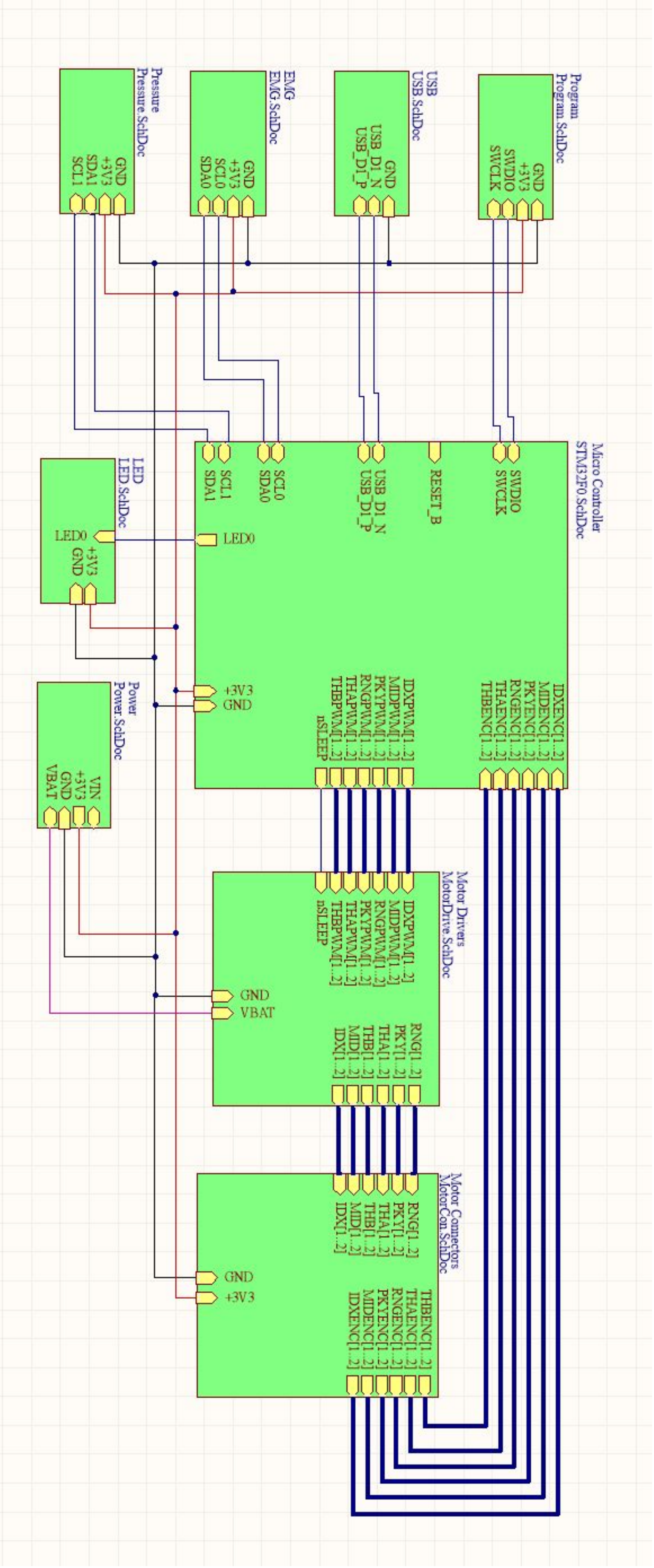
References

1. IEEE.org, "IEEE IEEE Code of Ethics", 2016. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 21- Sep - 2017].
2. N. Leveson, *Medical Devices: The Therac-25**. University of Washington. [Online]. Available: http://sunnyday.mit.edu/papers/therac.pdf. [Accessed: 21- Sep- 2017].
3. 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: 21- Sep- 2017].
4. U.S. Department of Health and Human Services Office for Civil Rights, "HIPAA Administrative Simplification - Regulation Text, 45 CFR Parts 160, 162, and 164 (Unofficial Version, as amended through March 26, 2013)", 2013. [Online].

https://www.hhs.gov/sites/default/files/ocr/privacy/hipaa/administrative/combined/hipaa-simplification-201303.pdf. [Accessed: 21- Sep- 2017].
5. Department of Health and Human Services, "Federal Register Part II, 45 CFR Parts 160 and 164 Breach Notification for Unsecured Protected Health Information; Interim Final Rule", 2009. [Online]. https://www.gpo.gov/fdsys/pkg/FR-2009-08-24/pdf/E9-20169.pdf. [Accessed: 21- Sep- 2017].

In addition to the cited references above, we would like to formally recognize PSYONIC for providing the CAD renderings shown in Figures 2 through 6.

Appendix A


Pressure
Pressure.SchDoc
GND
+3V3
SDA1
SCL1
EMG
EMG.SchDoc
GND
+3V3
SCL0
SDA0
USB
USB.SchDoc
GND
USB_D1_N
USB_D1_P
Program
Program.SchDoc
GND
+3V3
SWDIO
SWCLK
LED
LED.SchDoc
LED0
+3V3
GND
Micro Controller
STM32F0.SchDoc
SDA1
SCL1
SDA0
SCL0
USB_D1_N
USB_D1_P
RESET_B
SWDIO
SWCLK
LED0
+3V3
GND
IDXPWM[1..2]
MIDPWM[1..2]
PKYPWM[1..2]
RNGPWM[1..2]
THAPWM[1..2]
THBPWM[1..2]
nSLEEP
IDXENC[1..2]
MIDENC[1..2]
PKYENC[1..2]
RNGENC[1..2]
THAENC[1..2]
THBENC[1..2]
Power
Power.SchDoc
VIN
+3V3
GND
VBAT
Motor Drivers
MotorDrive.SchDoc
IDXPWM[1..2]
MIDPWM[1..2]
PKYPWM[1..2]
RNGPWM[1..2]
THAPWM[1..2]
THBPWM[1..2]
nSLEEP
GND
VBAT
RNG[1..2]
PKY[1..2]
THA[1..2]
THB[1..2]
MID[1..2]
IDX[1..2]
Motor Connectors
MotorCon.SchDoc
RNG[1..2]
PKY[1..2]
THA[1..2]
THB[1..2]
MID[1..2]
IDX[1..2]
GND
+3V3
THBENC[1..2]
THAENC[1..2]
RNGENC[1..2]
PKYENC[1..2]
MIDENC[1..2]
IDXENC[1..2]