INDIVIDUAL PROGRESS REPORT
ECE 445, SENIOR DEISGN

By

Wenjun Sun
PROSTHETIC ARM I/O GROUP

1. Introduction

This project is designed to provide easy access of data and control for sophisticated prosthetic hand developed by Psionic startup. Our goal is to design a fully functional and market ready I/O board that Psyonic can easy patch to their system as a plugin.

1.1 Reasons behind

Currently, Psyonic hand does not have a plug-and-play charging system, they have to use external battery chargers for drones. At developing stage, it is ok for them to use external charging system, but when they are getting ready to push their product to the market, a reliable, easy to use internal charging system is critical. They specifically want to have a charging system that is compatible with USB, because everyone has USB chargers. They do not want a special charger that patients might lose and they need to buy one again. We discussed most commonly used USB 2.0. But this old protocol only provides 5V at 3A which does not meet our requirement of charging speed. We came up the idea of using the newest protocol USB-C to provide power. USB-C PD is capable of delivering up to 20V and 5A of total 100W of power, which exceeds their specification. The trend of USB-C shows most users will have a USB-C device when Psyonic is market ready. The ability of using any USB-C charger from market and charge at fast speed will give Psyonic competitiveness to other prosthetic arm making company since all of them use special charger system.

Psyonic hand also only has wire transfer capability, that means the data can only be transferred with a USB cable attached. That is inconvenient for clinicians to change the parameters or reading essential data from the hand. They need a computer with a USB port and they need to install special software to be able to communicate with the hand. While computer is not really portable and installing special software can take effort. We suggested to use Bluetooth Low Energy system to wirelessly transfer data and controls. So the clinicians can use their phone or tablet to easily download an app and have access to the hand. It eliminates the constrain of cable length and greatly improve the portability by using light weight and small tablet and phones. This will also provide users to be able to read data from the hand such as remaining battery or their daily usage, or even doing a firmware update in the future.

1.2 Group Goal

Byron Hopps and I are dedicated to develop so called “I/O board” for the prosthetic arm system. Our board consists battery management which is capable of charging the battery according to lithium battery charging specifications at fast speed. It provides monitoring API that convert analog data into digital data and send over through data bus to allows hand to know the current, voltage, wattage and temperature of the battery and I/O board itself. It also provides protection mechanism that once hard limit of current or voltage has been reached, the system will cut the connection between battery and outside circuitries.

1.3 Individual Responsibilities:

Byron is an electrical engineering senior who is expert at PCB designs, he is in charge of PCB design. His goal is to design a PCB board that is robust reliable, cost effective. He will be designing

the hardware that provides very high battery charging efficiency and system power provide efficiency. Because it's essential to reduce unnecessary waste of energy from a very limited power capacity from lithium battery. He also will be optimizing the RF design to provide a stable, fast, long range BLE connection.

I am a Computer Engineering who focus on firmware and software design that can seamlessly work with Byron's hardware design. I am designing firmware for microcontroller to control the state of the battery and parameters of charging. I will also be building the interface for BLE to transfer data to the phone or tablet. Design an app on the phone to connect and read, send data to the system. Because Battery Management System chip and USB-C PD chip need to be properly configured to work, I need to make sure the robustness of the firmware to handle any kind of situation. I also need to follow the protocol of those chips to properly set the right register values to configure the hardware into desired state.

2. Project Design Work

2.1 Design considerations and changes

2.1.1 Bus connection reduced

After few weeks into actual design, we realized there are few changes to the original design we can make to make the design more reliable and easier to execute. We decided to switch over using USB to UART bridge and use a mux to bridge the connection from computer USB to the hand instead of using USB to SPI and using microcontrollers to convert SPI to UART to hand. The changes we made greatly reduce the workload and increase stability of the system. On the hardware, there are fewer connections and routing complexity. On the firmware side, we have two fewer bus protocols.

2.1.2 BLE data changed

Due to limitation of BLE bandwidth, we decided to have option to either stream one channel of high bandwidth EMG signal or multiple channels of lower bandwidth, compressed EMG signal. We can let user or researchers to switch the mode through configuration.

2.2 Design Work

The design work was challenging and involving multiple different systems. So I decided to split work into small individual systems. There are four major tasks, NRF51822 BLE firmware design, BMS BQ25700 configuration, USB-C PD FUSB302 configuration, IOS software development.

2.2.1 NRF51822 FIRMWARE

Figure 1 intuitively shows the workflow of configure BLE from startup state into fully functional state.

Hardware Initialization

The nrf51822 firmware initialize the BLE SOC into a usable state by setting up the PLL internal clocks, I2C bus, GPIO, TIMER, and SYSTICK.

Peripheral initialization:

SOC configure the peripherals such as BMS IC to set the maximum charging/discharging current, maximum charging voltage, minimum discharge voltage, and read battery status through I2C bus. It also configures the USB-C PD IC to a standby state that is ready for connection at any time.

I also set up the interrupt handler so whenever a charger is plugged in, it will inform SOC and set up the input voltage and current for maximum efficiency and power delivery. And whenever a fault or abnormal reading of battery triggers BMS IC, the SOC will do future processing and handling such as report what fault is triggered and whether to restart or reconfigure the system.

Start and configure BLE Stack:

With NRF51822, the Nordic provides SDK and BLE stack called soft device to improve the robustness of BLE system and greatly reduce the workload of programmers. I called the initialization function to start the stack and configure the GAP parameters such as BLE name, UUID, password protection, power consumption, scanning interval, and set up the handlers to handle BLE traffic such as device connect, read/write data.

Start Service and Characteristics:

After device is connected, BLE stack need to broadcast services and characteristics to inform central device (iPhone in this case) that what service it can provide and how to read and write data to the service. After this stage, the initialization of BLE is pretty much completed.

Event Driven Handler:

Once the BLE is initialized the rest is handling service accesses, there are two types of major service, one is write and read. Write command will make SOC to send desired data to the digital bus so it can configure the peripherals such as stimulator, EMG, and hand system. And read command to read data from digital bus such as EMG data and battery percentage, discharge current, etc.

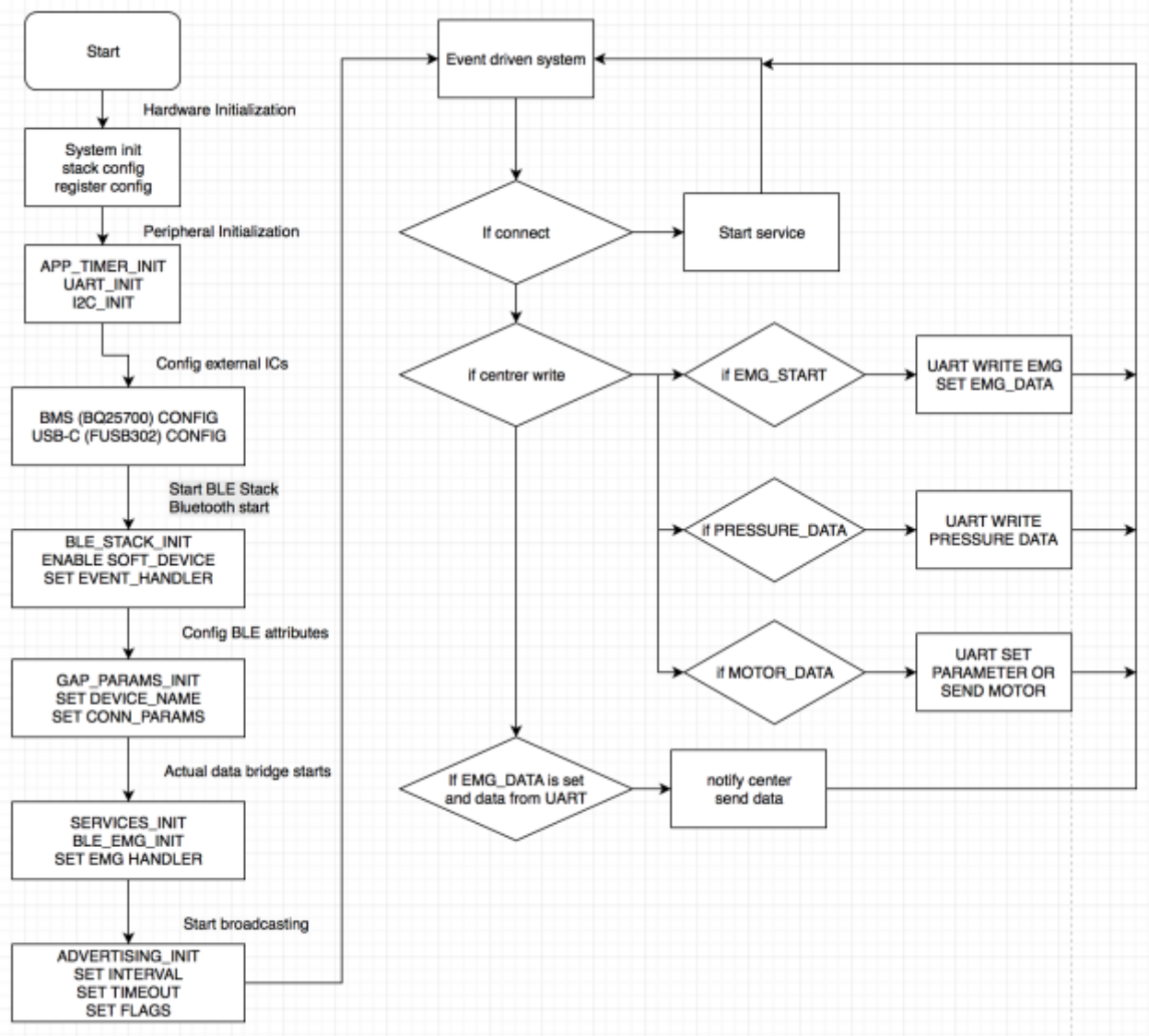

Figure 1. BLE work flow

2.2.2 USB-C PD

USB-C PD protocol is the most complex USB protocol, it enables one smart charger have multiple voltage and current output to suit the needs of its device. We want to fit all the chargers on the

market and select whatever voltage and current combination we want according to the information provided by the charger. In order to do that, I used combination of logic analyzer and Oscilloscope to understand how USB-C PD protocol works from physical layer. The logic analyzer is also used for debugging and verification of physical layer.

The USB-C plug in detection is done by pull down resistor (Rd). Once USB power supply detects Rd on the CC line, it will start support 5V on VBUS. After power supply supports 5V on VBUS, it starts sending power capability package to device through CC line.

The firmware for FUSB302 then initialize and configure the interrupt, so SOC will receive an interrupt after cable and power supply is attached. Then it will check which CC line is connected and set that CC line as data line. It then will read the data from the CC line and convert from bits into useful data such as supply capability of voltage and current. The SOC gets the data and through an algorithm to select the highest wattage and highest voltage according to the situation.

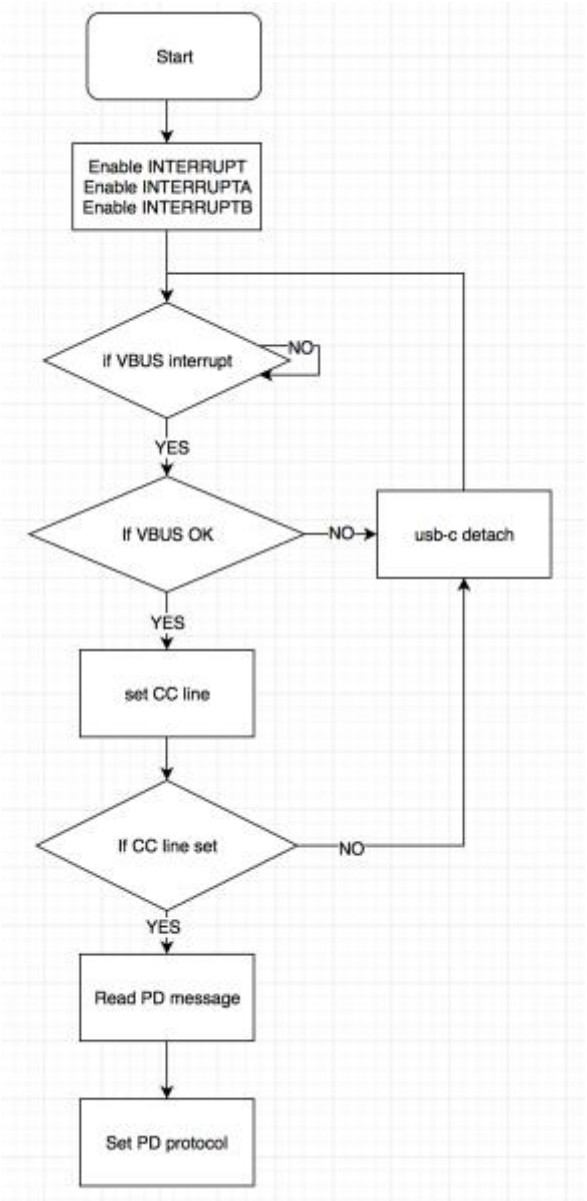

Figure 2. USB-C PD configuration flowchart

2.2.3 BQ25700

BQ25700 is a Battery Management Chip, it works with USB-C PD because its wide input voltage range, it supports buck, boost, buck-boost mode to make sure it charges batter under any normal USB voltage. It also provides battery charging current and voltage through registers.

To get BQ25700 to work is pretty easy, after we obtain USB-C charger profile, we set max charging current according to the USB-C charger. After that we set max charge voltage at 4.2V per cell according to battery specification. We enable charging by set charge_option0 register. We enable protection on overvoltage, under voltage and overcurrent, once those conditions met, it will cut connection between battery and the rest of the system. Then it enables protection interrupts so when the protection is triggered, it will report the condition to SOC so we can do further handling. After that, we enable ADC to read voltage and current of battery. And it repeats itself infinitely to provide battery information to the system.

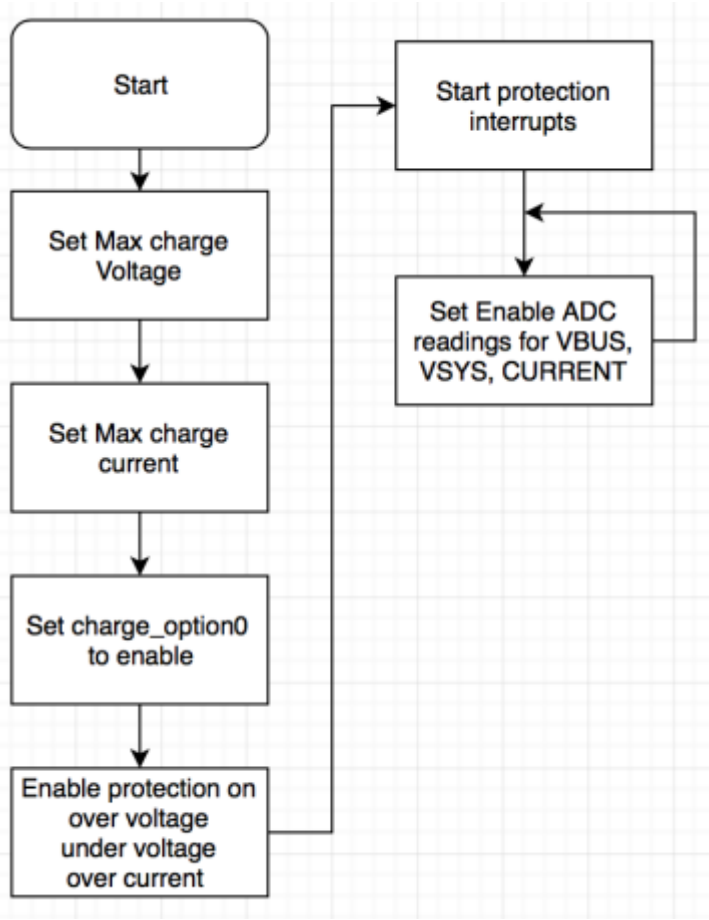

Figure 3. BQ25700 configure Flowchart

2.2.4 IOS software

With hardware and firmware, we still need software on iPhone to control the data. It consists two parts, the background BLE handling part and front-end user interface to provide visual insights. Figure 5 shows the UI layout of whole app.

The BLE core consists two parts, CBCentralManager and CBPeripherialManager. CBCentralManager starts BLE hardware on iPhone. It controls BLE hardware to scan active BLE

devices around the phone. It controls BLE to connect to the device. After device is connected, the CBPeripheraialManager starts and copy the hardware BLE into a structure variable and act as an object. It reads the services and characteristics and have ability to write, read the data from device. It provides APIs for easy access. We have timers to read RSSI (signal intensity) periodically. The front-end mostly made of sliders and buttons from default UI object. One critical part is the plot of EMG data, it requires customized plotting function. I was able to find a chart framework that fits our needs. We have data stored into array and whenever data comes in from BLE I/O board, we update the array and update the plot. Because the fresh rate is high enough, user will see it as a continues plot. Figure 4 shows the rough version of the plot

Figure 4. EMG data plot UI

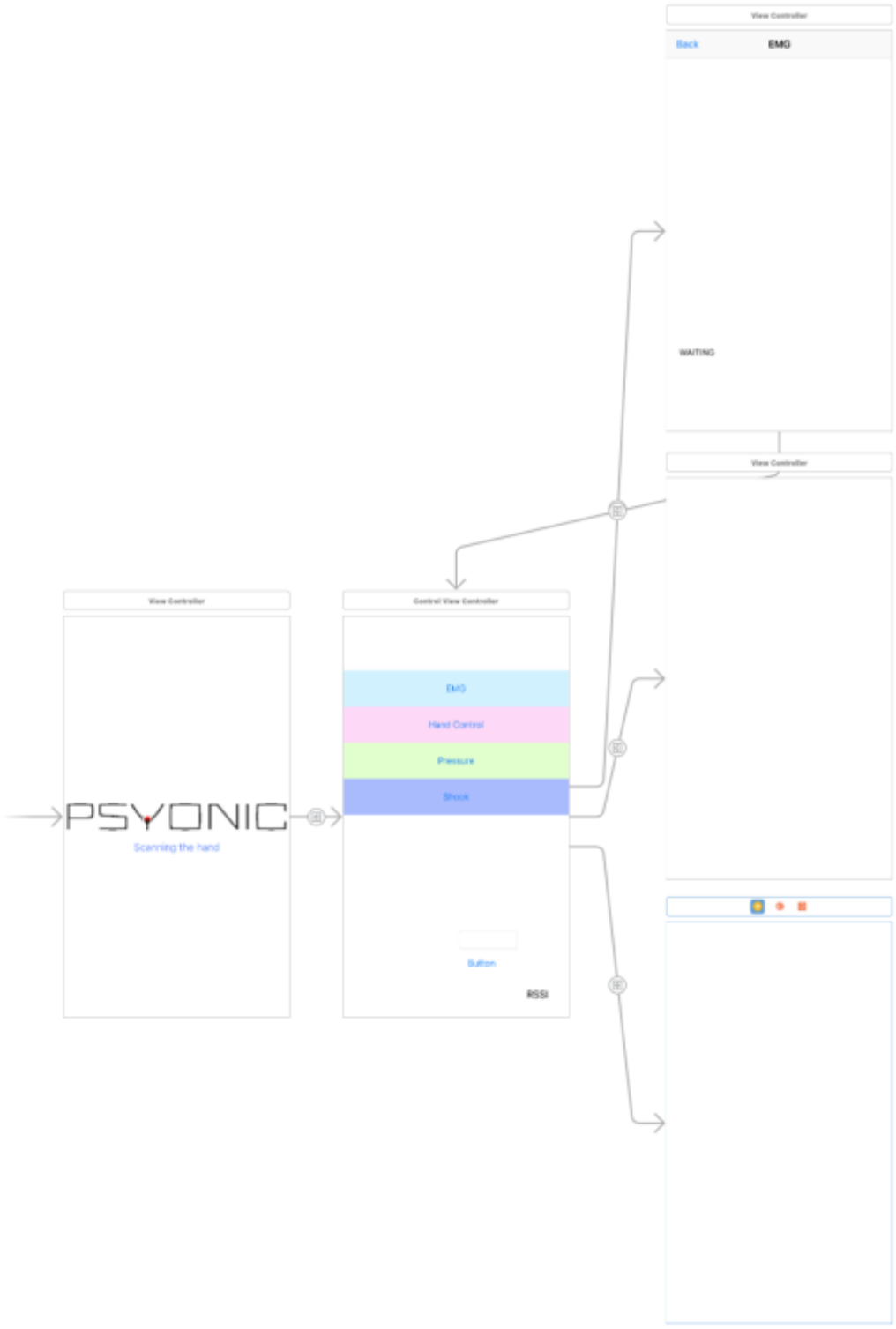

Figure 5. UI design of whole app

2.3 Testing/Verification

2.3.1Testing:

Test Date	Test item/Goal
11/10	Test all major USB-C chargers that come with popular devices such as Macbook Pro, Surface book2, chrome book, Google pixel phone and tablets that has USB-C chargers. That we can read charger capability as voltage and current and choose desired combination.
11/13	Test the BQ25700 battery management IC with continues voltage input from 5V to 25V and different charging current from 1A to 3A. Test battery short protection, battery over current protection, over voltage, and under voltage protection by using simulated battery with power supply and electric loads.
11/17	Test IOS code will report BLE status and read RSSI at any condition, handling connecting and disconnect events. Write and read data from BLE I/O board
11/19	Modular testing, if iOS app is able to read battery status through BLE, and read EMG data from UART bus and send control command through UART bus to control the stimulator and hand movement
11/25	Whole system testing, basically combines BLE, charging function, USB-C configuration and iOS app and find out the bugs produced in the process of integration.
11/27	To have system patched to psyonic hand and test it with real people.

2.3.2 modular testing:

We need test whole system around 11/25/2017, that when a charger plugs in, the USB-C PD chip works and BQ25700 will enable charging on the battery. We have iOS app to provide selection from different functionality such as EMG data streaming, motor control, data configuration, and execute stimulator using app. When we over load the system or over charge the battery using external power supply, the BQ25700 should cut the connection between battery and the system. User should be able to read the battery voltage or capacity percentage through the iOS app.

2.3.3 Results:

We have USB-C working with google pixel charger, macbook pro charger and a popular USB-C charger from china. All of them behaves almost the same because they need to stick with USB-C protocol. We are able to read power capability from all of them and select power combinations we want.

We have BQ25700 BMS working with current lithium battery Psyonic has, it reads the battery voltage and current within 5% of error from a calibrated multi-meter. It charges battery from 5V to 20V continuously and stops charging at maximum voltage set by firmware.
We have EMG data plot working as desired, it plots out the data points fast and accurately.
We have connection between iOS app and I/O board reliably
We have ability to communicate with I/O board through BLE protocol by reading and writing from the I/O board.

2.3.4 Contingency plans for failed tests

With failed SPI data bus from USB to EMG, we decided to use UART bus instead.
We cannot get accurate temperature reading from BQ25700 chip, so we switching to using an external temperature sensor
If BLE stack doesn't work reliably, we have other BLE stack protocols we can switch and test
If BQ25700 is not reading the correct data from ADC, we can do calibration for individual chips through software algorithms such as compensation and filtering.
If certain parts not working or meet desired specification, we can look for similar IC to replace.

3. Conclusion

The workload of designing software/firmware is very heavy, it's easy to write few lines of code and configure some registers to make one chip work temporary, but it's not as easy like connecting few wires together to make a system work. Without operating system, each task and timing need to be carefully and thoughtfully considered. USB-C PD is so advanced that there are only few solutions on the market for research use, I had to manually goes though the USB protocol white paper and figure out the configuration from scratch. As now, the whole system is working as desired on the separate hardware system, but I can see the integration and debugging can take a long time.

For the remaining task, it's essential to have an integrated hardware design on one single PCB. After we have the actual PCB, everything firmware wise can be port over to the actual hardware and we can do final testing and integration. Right now, we are waiting for the PCBs to come in. We are hoping we can get our PCB before thanksgiving and finish up the hardware before thanksgiving ends, and have whole system up and running before end of November of 2017. After whole system works, we need to test the robustness of system by working with Psyonic to run test on actual patients for certain period of time.

References

[1]FUSB302 USB-C Controller, datasheet, Fairchild semiconductor, Inc., 2015 Available at: https://www.fairchildsemi.com/Assets/-Featured-Collateral/Datasheet-Briefs/FUSB302_PB_Final.pdf

[2]NRF51822 SOC, datasheet, Nordic semiconductor, Inc., 2010. Available at: http://infocenter.nordicsemi.com/pdf/nRF51822_PS_v3.1.pdf

[3]BQ25700 BMS IC, datasheet, Texas Instrument, Inc., 2017. Available at: http://www.ti.com/lit/ds/symlink/bq25700a.pdf

[4]USB-IF Technical White Papers, protocol specification, web page, IEEE 1394, available at: http://www.usb.org/developers/docs/whitepapers/. Accessed November 2017.

[5]SoftDevice BLE stack, protocol stack specification, Nordic semi, Inc. available at: https://www.nordicsemi.com/eng/Products/S132-SoftDevice, Accessed November 2017

[6]iOS Bluetooth programming guide, guides and sample code, Apple, Inc. Available at: https://developer.apple.com/library/content/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/AboutCoreBluetooth/Introduction.html, Accessed November 2017

Appendix A: Abbreviations

Term	Symbol or abbreviation
Battery management system	BMS
Integrated Chips	IC
System On Chip	SOC
Universal Serial Bus Type-C Power Delivery	USB-C PD
Electromyography	EMG
Pull down resister for USB-C PD	Rd
Configuration Channel	CC
Generic Access Profile	GAP
Generic Attribute Profile	GATT
Universally Unique ID	UUID
Relative signal strength indicator	RSSI
Printed circuit board	PCB