EN 601.456 Computer Integrated Surgery II
May 1, 2022

Development of a Wearable Intracranial Pressure Monitoring Device

Project Final Paper

Members: Ese Bowry, Marissa Hsu
Mentors: Joshua Liu, Dr. Chad Gordon, Dr. Mehran Armand

I. Abstract

This project is working on the creation of a wireless, wearable intracranial pressure (ICP) monitoring device for monitoring pressure inside the skull. This device will aid physicians in continuously analyzing a connected patient's intracranial pressure levels. This project is based as a proof-of-concept design to test the feasibility of a wireless implantable device.

II. Introduction

ICP monitoring methods usually involve the placement of a catheter through the skull that monitors the patient's intracranial pressure for a limited period of time. This type of monitoring is commonly done at the hospital or clinic and must be overseen by trained professionals. Modern ICP devices have worked to create a more compact design consisting of an implantable sensor and a connected monitor that can aid in long-term ICP measurements. Our project aims to advance ICP telemedicine techniques by creating a more compact design with an implantable pressure sensor that can wirelessly transfer ICP measurements to a monitor.

Since there is no prior work on this project done in our mentor's lab, this project is a proof-of-concept based project where our central goals are comprised of:

- Prototyping a wearable monitoring device that can be embedded in a cranial implant for measuring a patient's intracranial pressure
- Transferring pressure sensor data to a remote PC/smartphone via Bluetooth
- Live-tracking and feedback analysis in-app

Figure 1 provides a visual of how all the components of this project would come together to represent a wireless ICP monitoring device.

Figure 1. Planned integration of all components on this project. Included is the implantable monitoring device wirelessly transferring (via. bluetooth) pressure measurements to a software application.

By developing a compact monitoring device and specialized ICP monitoring iOS application, our project can verify the potential of creating a new ICP monitoring device that is not only user-friendly but can also advance current ICP telemonitoring techniques.

III. Background

ICP monitoring methods are typically done in hospital or clinic settings overseen by healthcare specialists. There are two ways of monitoring intracranial pressure:

- Placing a small hollow tube or catheter into the ventricle
- Inserting a small hollow device between the skull and brain

Once the device is inserted in the brain, it is able to measure the compression within the skull and involves some drainage system to act as a cushion. These drainage systems are considered to be standard devices for ICP monitoring. An ideal monitor is accurate, reliable and easy to use as this is an invasive process.

There is an example of long term ICP monitoring involving attempts to improve the diagnosis and subsequent treatment of idiopathic intracranial hypertension (IIH). For example, one trial used a NEUROVENT® P-tel System for long term ICP monitoring, researchers were able to hypothesize potential adjustments to IIH guidelines.

IV. Technical Summary

A. Hardware

To begin prototyping a wearable monitoring device that could be embedded in a cranial implant, Group 13 utilized hardware components from Adafruit Industries. Since this is a

proof-of-concept project, these hardware components were chosen to simulate the implementation of an electronic platform, Bluetooth transceiver, and pressure sensor. Our goal with these hardware components was to simulate a monitoring device's capabilities to read pressure sensor data.

a. *Hardware combination setup*

All hardware components were purchased from Adafruit Industries. We have used the Adafruit Flora Wearable Electronic Platform as the main board for all connections between the components as well as the main power source for the device. An Adafruit MPRLS Ported Pressure Sensor is used as the pressure sensor that records and outputs pressure value readings. Lastly, an Adafruit Flora Bluefruit LE is used as the bluetooth transceiver that wirelessly transfers data from the sensor to a personal computer or phone. The below figure shows an outline of the hardware components.

a. b. c.

Figure 2. Hardware Components

a. Adafruit Flora Wearable Electronic Platform.
b. Adafruit MPRLS Ported Pressure Sensor.
c. Adafruit Flora Bluefruit LE.

We utilized a pressure pump (units mmHg) to test various pressure values. This pressure pump allowed us to verify that the MPRLS Pressure Sensor was detecting the same pressure value as the pressure pump was reading. See the figure below for our final hardware combination setup. (Note: the MPRLS Pressure Sensor is shown not connected to the FLORA platform. However, it can be seen that through the blue, gray, purple, and white wires is where the MPRLS sensor gets connected.)

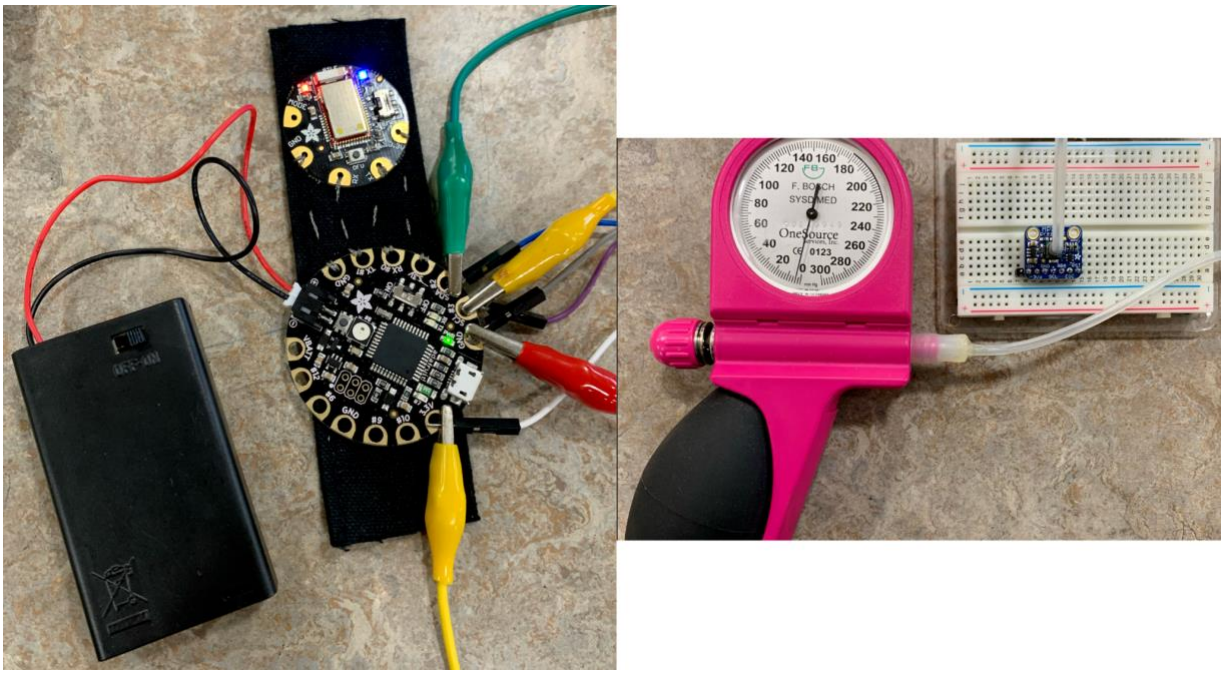

Figure 3. Final Hardware Combination Setup (with pressure pump connected to MPRLS Pressure Sensor shown to the right)

b. *Wired implementation with Arduino IDE and Serial Monitor*

To initially test the transfer of data from the pressure sensor to a personal computer, we first utilized Adafruit's provided testing code implementation for transferring data from the MPRLS sensor. This code implementation utilizes the Arduino IDE and Arduino libraries for both the Adafruit Flora board and the Adafruit MPRLS sensor. The Arduino serial monitor is then used for displaying the data values recorded by the pressure sensor. Figure — captures the serial monitor continuously outputting received pressure values in units of pounds per square inch (PSI).

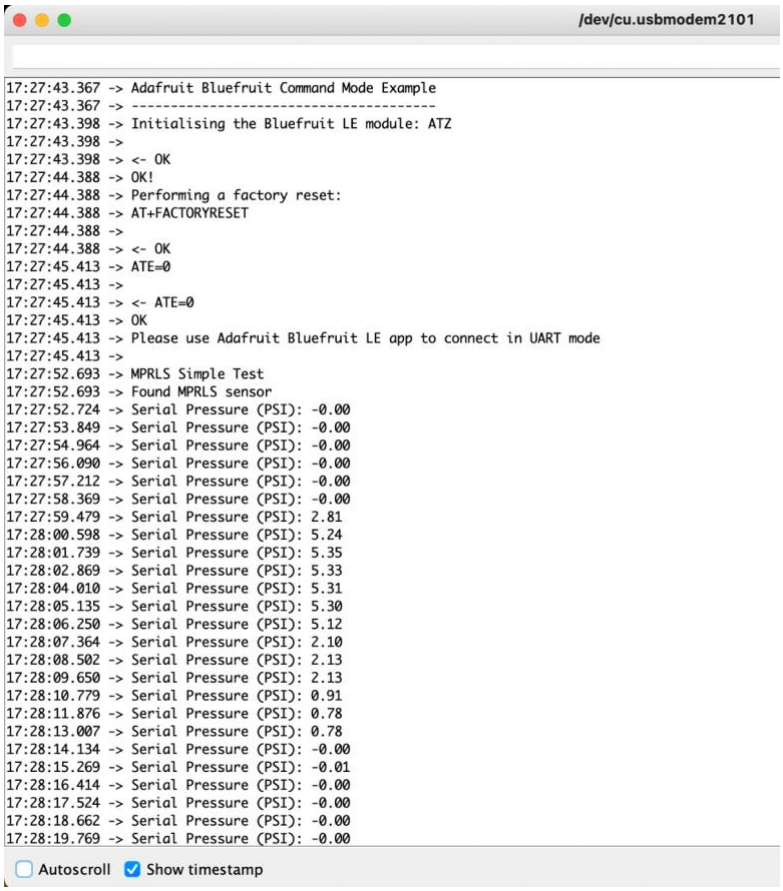

Figure 4. Adafruit-Arduino connection and pressure data as seen on the Arduino Serial Monitor

c. *Wireless implementation with Adafruit's Bluefruit Connect Application*

After verifying the behavior of the wired transfer of pressure data from the MPRLS sensor to a personal computer, we utilized the Adafruit Bluefruit LE as a bluetooth module that could aid in moving to wireless transfer. For code implementation we utilized both Adafruit's code implementation for the MPRLS sensor and Bluefruit UART Command. By combining elements of these codes, we were able to develop a final code implementation that could read in MPRLS sensor data and wirelessly output them to the Adafruit Bluefruit Connect Application on a phone. The Bluefruit Connect App (when the Adafruit Bluefruit LE component is set to "DATA Mode") has a feature to display data to the phone screen (similar to Arduino's serial monitor) and also a feature to continuously update a graph with received data. Figure — captures these two features from the Bluefruit Connect App. The graph in Figure 5 has an x-axis representing sample number and y-axis representing pressure readings in PSI.

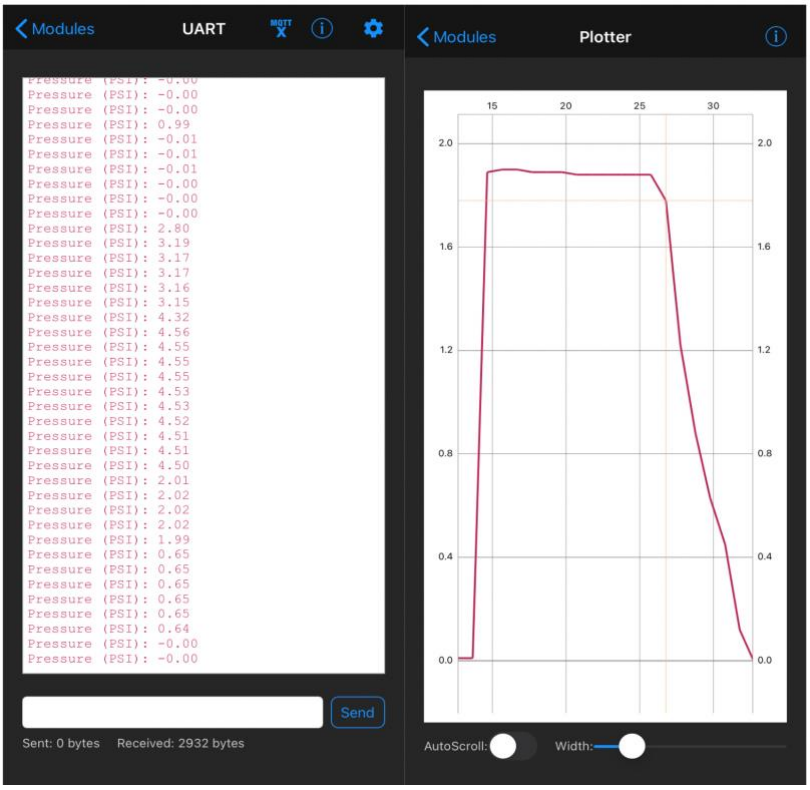

Figure 5. Display monitor and graph as seen on Bluefruit Connect Application

For this project, we are also developing a specialized iOS software application that can read in pressure sensor data and analyze the data according to Intracranial Pressure Monitoring conditions. Future work on this project will not be using the Adafruit's Bluefruit Connect Application, but this section did aid us in understanding the background on how the hardware and software would be connected and which libraries might be necessary to combine all the working hardware components in the software implementation.

d. Results

To test the hardware setup, the pressure pump was set to various pressure values while the output value received from the MPRLS Pressure Sensor was recorded and compared. We have found that the output from the MPRLS Pressure Sensor is accurate within a 0.05 PSI error. We take the slight discrepancy in the expected and actual pressure values to be representative of the inability to fully zero the pressure pump or the state of the tube that connects the pressure sensor to the pressure pump. However, the error was deemed small enough to still accept the pressure sensor as a workable component of this hardware setup. In addition, we needed to include a manual calibration to the pressure readings as seen on the pressure pump as the pressure pump does not start at 0 mmHg. The table below

contains comparisons of a few expected and actual pressure values from the pressure sensor and pump.

Pump (mmHg)	Adjust (-3mmHg)	Expected (PSI)	Arduino (PSI)
50	47	0.908	0.89
100	97	1.876	1.88
150	147	2.843	2.8
200	197	3.809	3.75

Table 1. Comparisons of expected and actual pressure values

B. Software

To begin creating a minimally functional iOS app that connects with our Bluetooth monitoring device, Group 13 utilized Adafruit and Arduino documentation. As this project has no previous code, we were building an app completely from scratch. Similarly to getting familiar with the hardware components, we also needed to be familiar with the latest Adafruit documentation with Swift. The documentation for the Adafruit Bluetooth app is best suited for Android apps and or much older versions of Swift. The goals associated with the software project:

a. *Sign-in/sign-up screens*

Since we are dealing with patients' medical data, we need a way to ensure that the information is only available to certain people. Although people tend to abandon apps where some form of sign-in is required, we decided that we needed to protect our user's information to the best of our ability. We were able to create a functional login and sign-up page that loads automatically when the app is started. The buttons allow the user to distinguish themselves as a previous member or a new member and move to the next page.

b. Fully connected Bluetooth app

While our intentions were to make a fully connected app, we were unable to achieve our goal. However, I will be discussing the steps that we took to connect our Swift code, Arduino code, and Adafruit devices.

In order to work with the Swift interface, we utilized the Core Bluetooth framework for the app to communicate with Bluetooth-equipped low energy technology. Within the CBCentral Manager, we need to be able to discover and connect to peripherals in order to control how the hardware's information is displayed. There are certain services that control how the device behaves and characteristics to show more details of the service

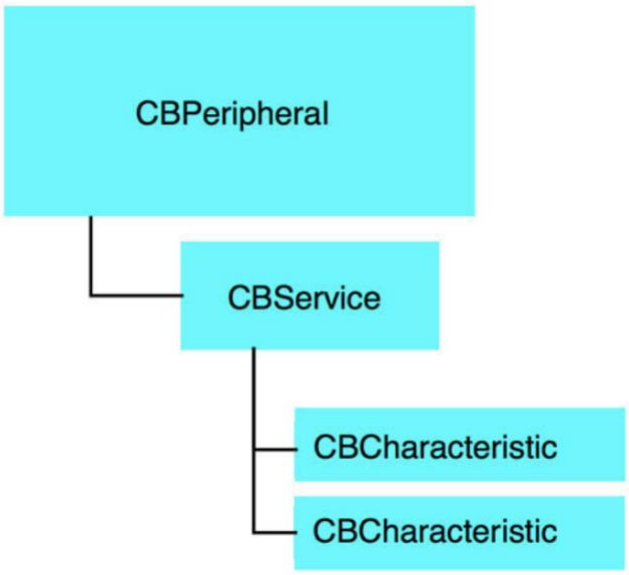

Figure 6. Hierarchy of the CBPeriphal[4]

After the CBCentralManager is created, the system needs to be continuously scanned for peripherals. When a peripheral is discovered, it can now be recognized and stored as advertisement data. The peripheral's services can be discovered and the central manager is able to handle and filter services with our didDscoverService(). The value of the characteristic can be read and written as well.

After writing the code to make our app discoverable, we created a connection with the Arduino code to set up the communication between the two types of software.

c. *Specified pressure warnings pop up on the app*
Our goal was to create a pop-up screen when a reading was below a certain threshold so that the patient could take note in effort for a longer telemonitoring experience.

C. Combined Workflow

The following figure shows the workflow intended for this project. When integrating the hardware with the software, the hardware will first output the ICP readings via a bluetooth connection which will then be read in by the iOS software application. The iOS application can then interpret and display the ICP measurements for viewing by both the patient and the physician.

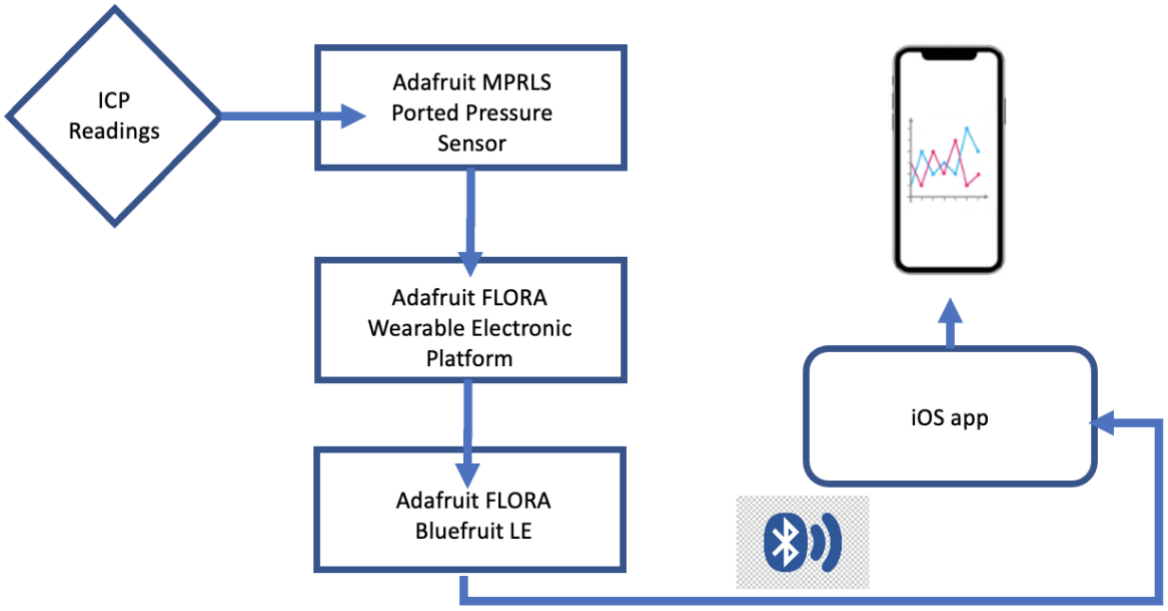

Figure 7. Hardware and Software Integration Workflow

V. Management summary

As members of Group 13, Ese Bowry and Marissa Hsu worked through most of the semester, focusing on the hardware and software components separately. These two components were initially done separately to ensure that the hardware and software setups were operating successfully on their own before integrating them together. In the last few weeks, the integration of the hardware and software components was worked on together.

Ese Bowry was responsible for the following software tasks:

- Creation of a sign-in/sign-up screen to ensure patient safety
- Fully connected iOS software that connected to our hardware set-up

Marissa Hsu was responsible for the following hardware tasks:

- The initial combination of the Adafruit Flora Electronic Platform, Adafruit Flora Bluefruit LE, and Adafruit MPRLS Ported Pressure Sensor Breakout
- Optimization of hardware combination setup
- Wireless implementation of hardware set up with Adafruit's Bluefruit Connect application

Group responsibilities:

- Debugging hardware and software integration
- Attending weekly meetings with mentor Joshua Liu
- Making all presentations and reports needed for the CIS II course
- Updating the project wiki
- Maintaining code documentation on project Github

Group 13's Project Mentor Joshua Liu was responsible for the following:

- General project guidance
- Attending weekly meetings with Group 13

The planned deliverables for this semester are shown in the table below:

	Deliverable
Minimum	• Functional Monitoring Device using Adafruit FLORA Wearable Electronic Platform • iOS App for ICP monitoring • Documentation
Expected	• ICP monitoring with specified pressure (high) warnings • Creation of cranial implant for holding ICP monitoring device (Fabrication - 3D printing) • Feasibility Trial • Documentation
Maximum	• Usability testing • Submitted Conference Paper

The minimum and components of the expected deliverables will be met during the course of CIS II this semester. Due to complications of interpreting and implementing an iOS application in Swift based on Adafruit's Android documentation for their Bluefruit Connect Application, we were met with major delays in the final hardware and software integration. This further delayed the 3D printing of a cranial implant for the ICP monitoring device. Although we were able to conduct feasibility trials on the wireless implementation with Adafruit's Bluefruit Connect Application, we were not able to conduct a feasibility trial within the constraints of a cranial implant. The maximum deliverable was decided to not be expected to be reached in the course of this semester due to inability to design and print the cranial implant.

As our group struggled specifically with the Swift software code implementation, it is expected that the next steps for this project will be to continue working towards establishing a stable connection between the Swift code and Adafruit Bluefruit. Following the submission of this final report and our final presentation day, we will continue to work at debugging the software code. Assuming the Adafruit Bluefruit can be connected to the software application, the next steps will be to design the cranial implement and perform feasibility trials on the system as a whole.

Although there were many setbacks with faulty hardware wire connections and software connections, our group has learned a lot from this project. This group had experience prior to this project working with Arduino components and Swift coding separately. Therefore this project broadened our skills to be able to combine Arduino hardware components with Swift code. Establishing a hardware and software connection without the aid of previously developed code with their associated libraries proved to be a challenge for our group, but nonetheless provided us

with new skills for working through both hardware and software bugs. In addition, upon reading from our reading list, we also learned a lot of new information on ICP monitoring. We not only explored the current hospital ICP monitoring systems but also explored how various companies have shifted to developing more compact ICP monitoring systems. If ICP telemonitoring techniques are to broaden to monitoring for long-term periods or to comfortably incorporate these systems into a patient's daily life, reading on these companies' designs broadened our perspective of the current innovations in telemedicine.

Although the members of this semester's CIS II project will not be continuing this project, the work done will be handed off to the next interested students.

VI. Technical Appendices

A. Documentation for all code: https://github.com/mhsu17/ICP_Project
    a. Included Folders:
        i. Adafruit_Bluefruit: code associated with the "Wireless Implementation with Adafruit's Bluefruit Connect Application" section of this project
        ii. Swift_code: code associated with the iOS application of this project

VII. References

[1] Velazquez Sanchez, V.F., Al Dayri, G. & Tschan, C.A. Long-term telemetric intracranial pressure monitoring for diagnosis and therapy optimisation of idiopathic intracranial hypertension. *BMC Neurol* 21, 343 (2021). https://doi.org/10.1186/s12883-021-02349-8

[2] J, P. (2021, May 27). Intracranial Pressure Monitoring Device and its critical applications. Retrieved February 17, 2022, from https://www.researchdive.com/blog/intracranial-pressure-monitoring-device-and-its-critical-applications

[3] Mitchell, K. S., Anderson, W., Shay, T., Huang, J., Luciano, M., Suarez, J. I., Manson, P., Brem, H., & Gordon, C. R. (2020). First-In-Human Experience With Integration of Wireless Intracranial Pressure Monitoring Device Within a Customized Cranial Implant. *Operative neurosurgery (Hagerstown, Md.)*, *19*(3), 341–350. https://doi.org/10.1093/ons/opz431

[4] Beaton, T. (n.d.). *Build a bluetooth app using swift 5*. Adafruit Learning System. Retrieved May 1, 2022, from https://learn.adafruit.com/build-a-bluetooth-app-using-swift-5?view=all#scanning-for-peripherals