Final Report

MRI-COMPATIBLE SKULL-EMBEDDED IMPLANT FOR DIRECT MEDICINE DELIVERY

EN.601.456 Computer Integrated Surgery II

Group 16: Disha Mishra, Henry Noren, Vivian Looi

May 6, 2021

Table of Contents

Introduction

Clinical Motivation

Glioblastoma Multiforme (GBM) is one of the most aggressive types of brain cancer. Currently, the diagnosis of GBM is somewhat of a death sentence for patients because of its malignancy – the median survival time is 14.6 months [1]. There are more than 18,000 new cases in the U.S. per year [2] and the recurrence rate is over 90% [3]. When a GBM is diagnosed, the patient first receives a surgery to remove as much of the tumor as possible. However, total removal is impossible, hence radiotherapy and chemotherapy are often administered to eliminate any potential remaining tumor cells [4]. Unfortunately, over 99% of promising therapies [5] are unable to reach the tumor and be effective due to the blood-brain barrier, which is a barrier that mediates communication between the peripheral and central nervous system.

Patient survival and the standard of care for GBM has been stagnant for the last three decades. Thus, getting past the blood-brain barrier has become one of the most active areas of research, and intersects with the field of neuroplastic surgery: for a skull reconstruction surgery, the collapsed part of the skull is replaced with a 3D-printed, plastic cranial implant that emulates the strength and biocompatibility of the human skull bone. There is a hollow thickness in the cranial implant very close to the brain. Hence, the idea of embedding a cranial implant in this space with functional technology to deliver therapeutics across the blood-brain barrier is of great interest. Once this platform technology is developed, this implant device can be filled with any liquid therapeutics. The target users of this implant device are currently patients with GBM, but eventually it will be applicable to patients with all kinds of chronic neurological diseases such as Alzheimer's and Parkinson's.

Prior Work

This project has been ongoing for several years in the Center for Neuroplastic Surgery Research at Johns Hopkins. Significant progress has been made in the hardware components for the implant device, such as manufacturing all parts of the device with MRI-compatible material, developing the mechanisms for the pumps that deliver drugs from the reservoir through the catheters, and designing the mechanism of power supply for the pumps. A sketch of one of the prototype designs is shown below in Figure 1. The biocompatibility of the prototype was assessed in preliminary swine studies in March-April 2021. The was not be turned on because the purpose of this initial study is to evaluate the structural integrity (i.e. no leakages) and biocompatibility of the device in an animal model.

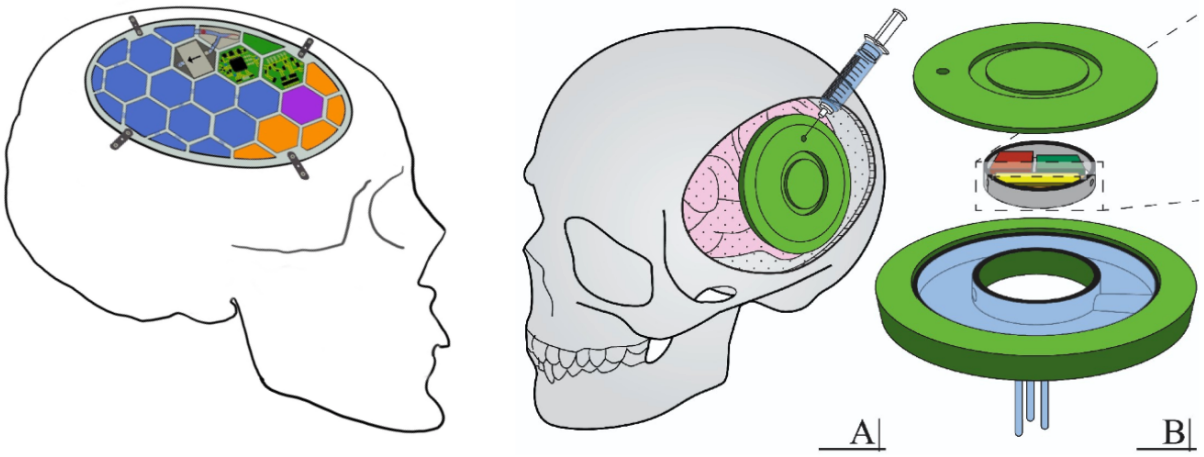

Figure 1: A preliminary prototype of the implant device.

Goals

As seen in the previous section, there is a need to further develop the software component of the device in order to provide a means of communication between the device and the clinicians. Our ultimate goal is to implement real-time Bluetooth connectivity in the device which would allow the device to report its battery life, power transfer status, drug delivery rate, and reservoir status, as well as allow its delivery rate to be modified through Bluetooth connectivity by clinicians. However, due to time constraints imposed by the semester-long CIS II class, we have identified specific aims to be completed by the end of this class that would set up the foundation for the communication between the device and clinicians using Bluetooth Low Energy (BLE). Our specific aims are to:

1. Implement code to use information from sensing pins to perform flow rate calculations every minute
2. Implement code to use Bluetooth Low Energy (BLE) to
    a. transmit flow rate estimates to clinicians every ten minutes
    b. allow the implant to receive signals to turn the pumps on and off

The completion of the two aims above would be a principal milestone for the software development of the device. Being able to calculate the flow rate of the therapeutics being delivered using information from sensing pins in the implant device allows clinicians to better understand the current medicine delivery status. In addition, once the implant device is able to report its flow rate to clinicians, as well as receive signals to turn the pumps on and off, the efficiency and convenience for real-time control of the device will be significantly enhanced. Ultimately, implementing these two aims could improve the quality of treatment delivered by this implant device.

Technical Approach

Design Requirements

The requirements for our software development are outlined below in Table 1. The requirements have been assigned a phase (1-3) which corresponds to the order of implementation. The phases 1, 2, and 3 roughly match the minimum, expected, and maximum deliverables respectively as discussed later in the report.

Table 1: Code implementation design requirements.

Requirement	Phase
The Runtime shall implement pulse-width modulation (PWM) on two output pins, PWM-L and PWM-R. Each output pin shall correspond to a direction of the implant's pump element.	1
The Runtime shall allow only one output pin to be active at one time. Each output pin shall run for a maximum of sixty (60) minutes before stopping. The Runtime shall then run the operation of the opposite PWM pin (e.g., PWM-R starts after PWM-L stops).	1
The Runtime shall implement two analog input pins for detecting the empty state of the pump, SEN-L and SEN-R. These two pins shall correspond to the PWM pins (PWM-L and PWM-R).	1
The Runtime shall stop operation of a PWM pin when it detects a signal on its corresponding sensing pin (e.g., PWM-L stops when SEN-L detects a reading). The Runtime shall then run the operation of the opposite PWM pin (e.g., PWM-R starts after PWM-L stops).	1
The Runtime shall setup and read the sensing input pins SEN-L and SEN-R using the analog-to-digital converter interface on the board.	1
The Runtime shall record the sensing pin readings in an internal queue buffer.	1
The Runtime shall use information from its sensing pins SEN-L and SEN-R in its estimated flow rate calculation.	1
The Runtime shall record its estimated flow rate calculations in an internal queue buffer.	2
The Runtime shall check for a signal detection on a sensing pin (SEN-L or SEN-R) every one (1) minute.	1
The Runtime shall generate a flow rate reading every one (1) minute.	2
The Runtime shall implement the all on two separate pumps.	1
The Runtime shall dynamically adjust the PWM signal output on a pin based on	3

the flow rate reading and a target flow rate reading.	
The Runtime shall transmit, over BLE, the contents of its flow rate estimates internal buffer every ten (10) minutes.	2
The Runtime shall receive, over BLE, a new target flow rate number, and update its original target flow rate to the newly-received one.	3
The Runtime shall be able to receive a BLE signal to turn itself on and off.	2
The Runtime shall run with as little power draw as necessary.	2
The Runtime shall employ LE Secure Connections for its BLE communications.	3

Software Architecture

A flow chart for the software architecture in which we are implementing our code is shown in Figure 2. The skull-embedded implant contains the nRF52 SoC board which is responsible for the pulse width modulation which enables the pumping action of two pumps that each have two directions (left and right). A pump can only be pumping in one direction at a single time. The nRF52 board remotely communicates with Implantd over BLE to receive signals to turn aspects of the implant state (such as pump A or pump B pumping action) on and off and transmit pump flow data (such as the flow rate estimate) back.

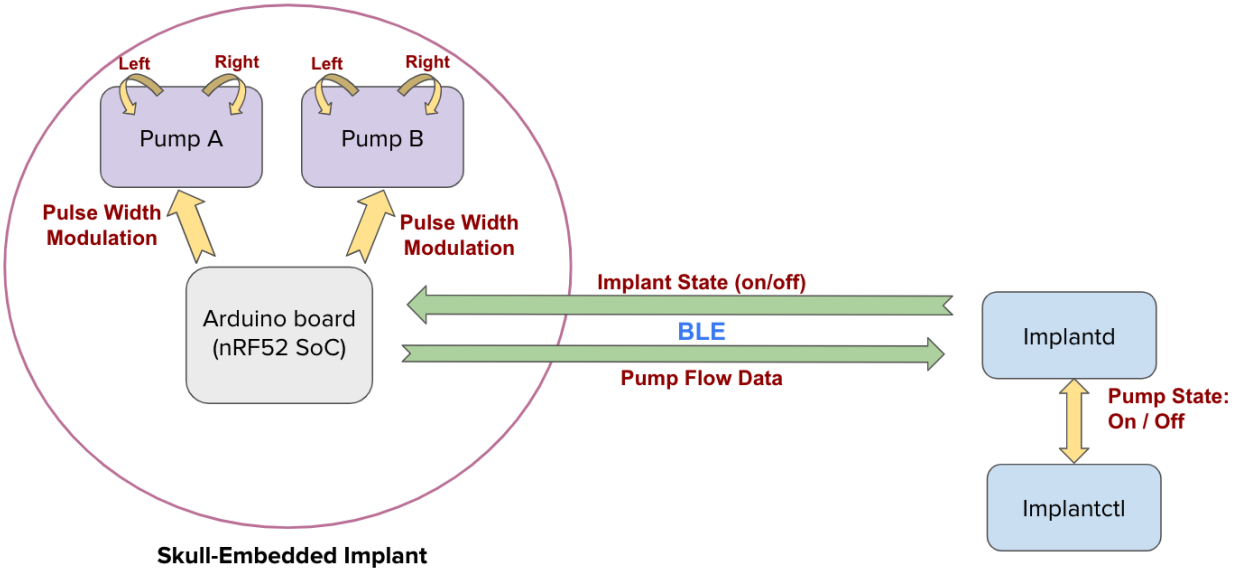

Figure 2: Software architecture flow chart.

Implementation

Documentation of the code discussed in this section is provided in the Appendix.

Double pump functionality

Our implementation accounts for two pumps in the skull-embedded implant, `pump_a` and `pump_b`. The pumps run independently of each other, so one can run while the other is turned off and direction switches within the pump can occur without affecting the other. The pumps are "turned on" by implementing a pulse width modulation that initiates the pumping action in the implant. The sensing pin of the pump collects a volume reading every 1 minute. If the volume reading passes a predefined threshold, then the pump switches directions. The maximum amount of time in which the pump can run in one direction is 60 minutes. We implemented this by capping the number of times the function `pump_loop()` can be called on a pump without switching and then forcing the pump to switch directions once this number of iterations is reached.

Flow rate calculations

The flow rate is calculated for each pump every 1 minute. The flow rate is calculated from the current and previous volume readings from the sensing pin and the amount of time that has passed between readings. When a reading is collected from the sensing pin, it is pushed onto a queue within the pump variable. To make an estimate of the flow rate at that given time, the difference between the current reading and previous reading is computed and then divided by the amount of time (seconds) that has passed between these readings, as shown below:

Equation 1: $\text{FlowRate}_1 = (\text{reading}_0 - \text{reading}_1) / (\text{time}_0 - \text{time}_1)$

These flow rates collected every minute are stored in a queue for each pump. Every 10 minutes, when it is time to transmit a flow rate estimate to the user over BLE, these rates are averaged, then popped from the queue so that a new set can be pushed on. The frequency at which readings, flow rate calculations, and transmission of estimates to the user occur is easily modifiable and is subject to change in the future to meet the needs of clinical translation.

Remote Communication with BLE

Remote communication between the nRF52 board and the nRF Connect mobile app is achieved using BLE Peripheral. A BLE Peripheral Service, `doublepumpService`, is instantiated with a UUID, set as the advertising service for this instance of BLE Peripheral, and added as an attribute. Five BLE Characteristics are created and added as attributes to BLE Peripheral. There are two BLE Int Characteristics, one for each pump, that turn the respective pump off (0) and on (1). There is an

additional BLE Int Characteristic that turns a builtin LED light on and off, used only for testing purposes. Lastly, there are two BLE Double Characteristics, one for each pump, which are read only and send the estimates of the respective pump's flow rate to the user every 10 minutes. The Arduino `loop()` function checks if the user is connected, and if so collects new values written by the user for the pump (a and b) and LED characteristics. The only actions occurring while connection is established is the collection of new Characteristic values and updating of respective pump states to preserve power and use as little energy as possible. If new values are written the pump states are modified before disconnecting and then calls `pump_loop()` with the updated values. Every 10 minutes, the average flow rate estimate is made accessible to the user from the rate BLE Double Characteristics (0.00 if the pump is turned off).

Testing Plan

We performed software quality control using a verification and validation approach on our code.

Verification

We have developed tests to verify that the code we wrote meets the specified design requirements for 1) the pumps, and 2) BLE connection.

Pump functionality

As mentioned in the Technical Approach section, we implemented code to use information from sensing pins in the pumps to determine the action of the pumps and perform flow rate calculations every minute to be displayed. The tests listed below were used to determine whether the pumps exhibit expected behaviors:

Table 2: Tests used to verify the functionality of pumps.

Tests	Date Completed	Pass / Fail
Pump switches directions at max iterations	03/25	Pass
Pump switches directions at min threshold	03/25	Pass
Pumps A and B function independently of each other	04/01	Pass
No action when both pumps turned off	04/01	Pass
Performs flow rate calculation at expected intervals	04/05	Pass

The first two tests are used to ensure that the pump switches its pumping direction (left/ right) when its pumping action is completed (when the maximum runtime is reached or when the reservoir is empty). With this functionality tested, we then tested that the two pumps exhibit this correct behavior when the code is run on Pump A and B simultaneously. In the fourth test, we confirmed that the "turn off" command in the code works for both pumps. Finally, we tested that the flow rate calculations in each pump are done at the expected intervals such that clinicians

would be updated with the current medicine delivery rate regularly. The passing of these five tests indicates that our code meets the specified design requirement for the pump's functionality. Detail descriptions and significance of these tests can be found in the test plan documentation in the Appendix.

BLE connection and control

As discussed in the Technical Approach section, the BLE connection should be implemented such that flow rate estimates can be transmitted to connected devices, and that the pumps can be turned on or off as instructed by signals sent from connected devices. The tests listed below were used to determine whether the pumps exhibit expected behaviors:

Table 3: Tests used to verify the Bluetooth connection and control of pumps.

Tests	Date Completed	Pass / Fail
BLE Peripheral Connection	04/05	Pass
Turn LED on/off	04/12	Pass
Turn Pumps A and B on/off	04/12	Pass
Receives pump A rate at expected intervals over BLE	04/27	Pass
Receives pump B rate at expected intervals over BLE	04/27	Pass
Correct behavior upon BLE Peripheral disconnection	04/27	Pass

These tests were performed by connecting to the board with the nRF Connect mobile app. The first two tests are used to ensure that the BLE Peripheral Connection can be established. After ensuring that the connection is established, we then tested that the two pumps exhibit the corresponding behavior when they are instructed to be turned on or off using the nRF Connect app. In the fourth and fifth test, we confirmed that the flow rate calculated for each pump can be displayed on the connected device at the expected intervals. Finally, we made sure that the functioning of the pumps are not affected by disconnection from BLE Peripheral. The passing of these 6 tests indicates that our code meets the specified design requirement for the BLE connection and control. Detail descriptions and significance of these tests can be found in the test plan documentation in the Appendix.

Validation

We hope to validate that our system fulfills its intended role in the implant for clinicians, i.e. if the system in its current state could be useful in a clinical setting. While there are no concrete plans to integrate our code in the current prototype of the implant and test it in swine studies, we expect to receive qualitative feedback from our project mentors on the system and its clinical usability. This would provide us guidance for our next steps to maximize the utility of the BLE functionality established for a clinical setting in swine studies in Summer or Fall 2021.

Management Summary

Team Members / Mentors

Team Members

- Disha Mishra (dmishra4@jhu.edu): 3rd Year BS/MSE Candidate, Biomedical Engineering
- Henry Noren (hnoren1@jhu.edu): 3rd Year BS/MSE Candidate, Biomedical Engineering
- Vivian Looi (nlooi1@jhu.edu): 2nd Year BS Candidate, Computer Science

Team Mentors

- Dr. Chad Gordon: Director, JHU Neuroplastic & Reconstructive Surgery
- Dr. Avi Rubin: Technical Director, JHU Information Security Institute
- Dr. Nathan Scott: Senior Design Professor, JHU Mechanical Engineering
- Dr. Mehran Armand: Principal Faculty, JHU Applied Physics Laboratory

Primary Project Supervisor

- Tushar Jois: 3rd Year PhD student, Computer Security

Meeting Schedule and Organization

Meetings:

- Weekly Neuroplastic Surgery Laboratory meetings on Monday 10AM
- Weekly meetings with Tushar: Friday 4:15PM
- Biweekly meetings for group: Monday 9PM, Thursday 10AM

Programs Used:

- Communication using Facebook Messenger, Email, and Slack
- Updating and Sharing code using GitHub (Private repository)
- Writing Reports and Documentation and Uploading onto CIS Wiki page

Final Distribution of Tasks

All class-related deliverables (presentations, reports, etc) were divided equally amongst the members. For the actual project deliverables, the pump code was divided up as follows:

- Henry: Flow rate calculations and BLE peripheral set up
- Disha: Pump loop, pump sensing and BLE peripheral set up
- Vivian: Pump direction switches, setting up test plans

Progress Evaluation

Dependencies

Table 4: Outline of dependencies for the project

Dependency	Need	Contingency	Status	Deadline	Completion
nRF52 development kit	Need Bluetooth low energy board for pins	Could start some code with existing pins	Complete	2/22	2/20
Remote virtual machine	Connect to the implant and test pins	Code on GitHub, have Tushar test in person	Complete	2/26	2/28

Discussion of Dependencies and Delays

The two dependencies in the table were resolved fairly close to the deadlines set, which meant that our project did not have any significant setbacks due to external machinery or arrival delays. The reason that the remote virtual machine was set up a few days later than expected was because the team had some troubles obtaining pins and setting up the virtual machine over the previous weekend that could not be resolved until Monday, requiring some extra days for setup. The project overall did not have any severe dependency-related setbacks.

Key Activities and Deliverables

Table 5: Minimum, expected, and maximum key activities along with their corresponding deliverables.

	Activity	Deliverable	Target Date	Actual Completion Date
Minimum	Implement code that only allows one pin to be active at one time in Runtime	- Documentation of code and test results demonstrating performance	3/3	3/5
	Set-up two analog “sensing” pins and supporting code to sense empty state of the pump	- Secured pins on the implant - Documentation of code and performance results	3/11	3/15
	Implement code to record signal detections from pins and time between direction reversals	- Documentation of code and performance - Accuracy tests and test results	3/25	4/05

Expected	Implement code to use information from sensing pins, to perform flow rate calculations every minute	- Documentation of implementation and math used for flow rate calculations - Testing procedures for accuracy. Results of testing	4/07	4/20
	Implement code to use BLE to: 1) transmit flow rate estimates to clinicians, 2) receive signals to turn pumps on and off	- Working bluetooth implementation - Code documentation. -Testing procedures and testing results.	4/21	4/30
Maximum	Implement code that allows implant to receive and update to new target flow rate numbers given by clinician	- Code documentation. - Testing for signal reception and flow rate updating accuracy. Testing results	5/01	Not Yet Completed
	Employ low energy secure connections: patient privacy	- Documentation of code	5/13	Not Yet Completed

Table 5 lists the key activities and their respective deliverables. The deliverables are classified as either minimum, expected, and maximum deliverables. Minimum deliverables are associated with activities to be completed in Phase 1, expected deliverables correspond to activities in Phase 2, and maximum deliverables correspond to activities in Phase 3. The phases of the project were previously outlined in Technical Approach.

Discussion of Deliverables and Delays

As can be seen from the table, only the minimum and expected deliverables were completed in the time frame of this semester. The general reason for this is because the completion dates of the first few activities took longer than expected due to issues in debugging of the code. For example, while completing the implementation of recording signal detections, we ran into an error in which the pumps were not stopping and going to sleep when certain thresholds were reached. After debugging and talking to our mentor, we realized that this error was occurring because of a for-loop error where we were essentially stuck in an infinite loop at one threshold. Due to these time consuming errors and the need to debug early on, we fell behind in our expected deliverables.

In the expected activities, we had to learn how to code and implement bluetooth low energy (BLE), which required more reading and understanding of documentation than we had accounted for. This understanding of documentation and code increased the time taken to properly implement the bluetooth connectivity, setting the team behind on the expected deliverable dates. However, because we were significantly behind and knew that the maximum deliverables would likely not be achieved, we decided to add a few more tests to our testing plan for more robust testing of the implemented code, meaning that though we were not able to meet our maximum deliverables, we still took time to more thoroughly test our code. These tests can be seen in the Testing Plan section above.

Next Steps

As this project will be continued, the next steps will be to start on the maximum deliverables so that the implant can receive user-specified values for target flow rates over BLE and update the pumps accordingly. It will also be important to create a testing plan for how to properly and accurately test that the target flow rates are being updated correctly as currently the flow rates are only confirmed through the serial monitor output in the Arduino. It may be interesting to add a flow meter attached directly to the pump so that an external flow monitor can measure and confirm the flow rate values that are displayed on the serial monitor. This added measure will also confirm that the sensor pins are sensing the correct values and are working as expected. In order to add this flow monitor in for testing purposes, dependencies such as price and compatibility / usability may become important to consider. It may also be a good idea to implement an LED set-up with the pumps to visualize when pumps turn on and off or switch directions as again the only visual indication is through the serial monitor as of right now.

In a more general and broader perspective, the implant has just undergone a swine study and is scheduled to perform another one in the following months. The major goal of the next study will be to implement the first maximum deliverable and test that the code can modify the flow rates through bluetooth communication and update the pump settings in a real swine. We can also assess key performance metrics, such as power consumption and connection stability and strength, of BLE in these future swine studies. Collecting data on the power consumption of the BLE connection and pumping action is important because it will allow us to determine the lifespan of the implant before charging/replacement of the MRI-compatible battery is necessary. The stability and strength of the BLE connection can be measured by observing the received signal strength indication (RSSI) score and percent of successful BLE advertisements at varying distances between the implant and connected mobile app. These tests on power draw and BLE connection will provide key insights on improvements to be made before subsequent iterations of swine studies and eventual clinical translation to humans.

References

[1] Tamimi AF, Juweid M. Epidemiology and Outcome of Glioblastoma. In: De Vleeschouwer S, editor. Glioblastoma [Internet]. Brisbane (AU): Codon Publications; 2017 Sep 27. Chapter 8.

[2] Ostrom, Q.T., et al., CBTRUS statistical report: Primary brain and central nervous system tumors diagnosed in the United States in 2006-2010. Neuro Oncol, 2013. 15 Suppl 2: p. ii1-56.

[3] Central Brain Tumor Registry of the United StatesCBTRUS statistical report: primary brain and central nervous system tumors diagnosed in the United States in 2004–2007 www.cbtrus.org. Accessed June 28, 2012

[4] Hottinger AF, Stupp R, Homicsko K. Standards of care and novel approaches in the management of glioblastoma multiforme. Chin J Cancer. 2014 Jan;33(1):32-9. doi: 10.5732/cjc.013.10207. PMID: 24384238; PMCID: PMC3905088.

[5] Solid lipid nanoparticles for skin and drug delivery: Methods of preparation and characterization techniques and applications - ScienceDirect: https://www.sciencedirect.com/science/article/pii/B9780128162002000153

[6] Gordon, Chad. Magnetic Resonance Imaging Compatible, Convection-Enhanced Delivery Cranial Implant Devices and Related Methods. CraniUS®, 2020.

Appendix

Pump Driver Code Documentation

ble_peripheral.ino

Description

Pump driver program to monitor the state of both pumps (pump A and pump B) and perform flow rate calculations every minute. Depending on thresholds or number of iterations set, the pumps will change directions or turn off as required. Bluetooth Low Energy is also implemented for transmission of flow rate calculations to and receiving signals for turning off pumps from connected bluetooth devices.

Imported Libraries

- `<SPI.h>` and `<BLEPeripheral.h>`
    - Both of these libraries are interconnected:
        - SPI.h is the serial peripheral interface which is a synchronous data protocol used for communication with one or more peripheral devices
        - BLEPeripheral.h is an arduino library for creating custom bluetooth low energy (BLE) peripherals. Allows and enables communication between the arduino board and a bluetooth device
        - The BLEPeripheral library allows for setup of the bluetooth connectivity and the SPI allows for communication
- `<QList.h>`
    - Template class that provides lists with the functionality of a queue, which is how the flow rate readings will be stored in this code

Structs

- `pins`
    - Stores two integer variables:
        - `int pwm` = (pulse width modulation), which is a method of reducing the average power delivered by an electrical signal. The pwm integer values refer to either the `PWM_A or PWM_B` depending on the pump this pin is referring to.
        - `int sensor =` sensor configurations of a pin, which refers to the definitions `SENS_A_L, SENS_A_R, SENS_B_R,` or `SENS_B_L,` depending on which pump (A or B) the pin is referring to and which direction the medication is being pumped (L or R).
    - method of reducing the average power delivered by an electrical signal, and sensor configurations of a pin.
- `pump_info`
    - Stores and defines:

- `char id` = the identification of the pump (either 'A' or 'B')
- `char state =` stores if the state of the pump is on or off. Refers to definitions `STATE_ON_A, STATE_OFF_A, STATE_ON_B, or STATE_OFF_B` depending on the pump id
- `char msg =` stores the starting pump state and keeps it unchanged for comparison later in the code
- `char iter` = number of iterations / times the pump_loop has been checked (meaning the number of times the pump state has been checked and has stayed constant)
- `int pwm_size =` stores the pwm for the particular pump that the id refers to
- `direction dir =` stores the direction that the medication is being pumped
- `pins left =` stores the pwm and sensor for the left direction
- `pins right =` stores the pwm and sensor for the right direction
- `QList<int> readings =` queue that stores current and previous sensing pin readings

Major Variables

- `pump_a :` A pump_info struct defined according to the configurations of pump A.
- `pump_b :` A pump_info struct defined according to the configurations of pump B.
- `blePeripheral` : Instance of type BLEPeripherial
- `doublepumpService` : Instance of BLE type service with UUID
- `pumpA_Characteristic` : Read and write BLEIntCharacteristic where 0 = pump off, 1 = pump on
- `pumpB_Characteristic` : Read and write BLEIntCharacteristic where 0 = pump off, 1 = pump on
- `led_Characteristic` : Read and write BLEIntCharacteristic (0 = pump off, 1 = pump on) for testing purposes only
- `rateA_Characteristic` : Read only BLEDoubleCharacteric for the pump A low rate estimate transmitted to the user at set intervals
- `rateB_Characteristic` : Read only BLEDoubleCharacteric for the pump B low rate estimate transmitted to the user at set intervals
- `flowRates_a` : Queue of doubles containing the pump A flow rates since the last transmission of the average back to the user
- `flowRates_b` : Queue of doubles containing the pump B flow rates since the last transmission of the average back to the user

Functions

- `void pump_loop(pump_info *pump)`
  - Checks if the pump is off. If so, it will exit the loop and keep running. If the pump is on, it will check thresholding and iteration numbers, and once certain thresholds

are hit, the pump will change directions. If the readings and iterations are below threshold values, the pump will continue running in its current direction.

- `double get_rate(pump_info *pump)`
    - Uses the current and previous readings in each pump to obtain an average flow rate calculation. Calculated as $\text{FlowRate}_1 = (\text{reading}_0 - \text{reading}_1) / (\text{time}_0 - \text{time}_1)$.
- `void setup()`
    - Sets up and starts bluetooth low energy (BLE) peripheral by creating a universally unique identifier (uuid) for the pump. Also sets up pumpA and pumpB bluetooth read and write characteristics that allow for bluetooth to be able to read and (eventually in further implementations) write values to the pump systems.
- `void loop()`
    - Connects the pumps / implant to bluetooth and if pumps are on, sends flow rate values to bluetooth at certain intervals.
    - Runs both pumps simultaneously by using `pump_loop` for each pump.
    - Every minute, uses `get_rate` to obtain flow rate calculations for each pump and sends the average of rates every 10 minutes through bluetooth.

Test Plan Documentation

Pump switches directions at max iterations

Description

We have defined a maximum runtime for the pump in each direction ((left or right), which is given by `NUM_INTERVALS x CHECK_INTERVAL x TIME_SEC`, where

- `NUM_INTERVALS` is the maximum number of intervals a pump should run in each direction,
- `CHECK_INTERVAL` is the duration between each time the sensor reading of the pump is polled,
- `TIME_SEC` is the unit of time used to define `CHECK_INTERVAL`.

We wish to ensure that the pump does not continue pumping beyond the set maximum runtime in one direction, and hence we expect that when the maximum number of iterations is reached for the pump in one direction, it switches to pumping in the other direction.

Significance

The maximum runtime for each direction of the pump is calculated based on the capacity of the reservoir and the pumping rate of the pump, and therefore set to ensure that

1) the pump delivers drug in the reservoir not only to one side, but both sides that the pump reaches
2) the pump does not continue pumping when the reservoir is empty

Therefore, it is important to ensure that the pump does not continue pumping beyond the set maximum runtime in one direction.

Testing Method

We output the pump name, pumping directions, and the number of iterations a pump has run on the serial monitor in Arduino for every iteration. Hence we can run the pump in one direction for the set maximum runtime, and check if the number of iterations run by the pump in one direction before it switches direction equals to the set `NUM_INTERVALS`.

Pump switches directions at min threshold

Description

We have defined a minimum threshold for the reading of the sensor in one direction of the pump at which the pump should stop pumping, hence we expect that the pump switches to the other direction when the sensor reading in one direction is less than the set threshold.

Significance

The minimum threshold for the reading of the sensor in each direction of the pump is defined to indicate the empty state of the pump, indicating that delivery of drug via that direction has been completed. Therefore, it is important to ensure that the pump does not continue pumping when delivery in that direction has been completed, and switches to pump in another direction to continue with drug delivery.

Testing Method

We output the sensor value detected by the pump in one direction on the serial monitor in Arduino for every iteration. Hence we can run the pump in one direction and check if it switches direction when the sensor value is lower than the set threshold.

The two pumps function independently of each other

Description

We expect that regardless of the configurations of the two pumps, they function expectedly in accordance to their own configurations without interacting and hence influencing the functioning of each other.

Significance

The two pumps in the skull-embedded implant are supposed to pump drugs independently to different brain regions via their corresponding catheter. Hence it is important to make sure that the two pumps function properly independent of each other.

Testing Method

We set equal and different starting configurations (starting in the left of right direction) and constant values, such as maximum runtime and minimum threshold for each pump, and we tested that they exhibit their corresponding expected behaviors based on the output messages on the serial monitor in Arduino.

No action when both pumps turned off

Description

We expect that when the code instructs the pumps to be turned off, any pumping action is stopped until it is turned on again.

Significance

Since we use the “turn off” instruction to stop the operation of pumps at the appropriate conditions, such as when the maximum iterations or minimum sensor reading threshold has been reached, it is important to ensure that the pumps stop their operation when they are instructed by the code to be turned off such that they are functioning as intended.

Testing Method

We attached the pumps to an LED such that the on/off state of the LED corresponds to that of the pumps. We also print the statement "pump X is turned off" on the serial monitor of Arduino when the pump is turned off. Hence, we observe that the LED attached to the two pumps respectively turns off, and see the aforementioned statement on the serial monitor when the pump is instructed in the code to be turned off.

Performs flow rate calculations at expected intervals

Description

Flow rate calculations will be performed for each pump to determine the current medicine delivery rate. As the expected intervals are set within the code, it is expected that the calculations will be performed every minute. If the intervals of calculations are not as expected, the code will be reviewed and changed

Significance

It is important that the calculations are done at regular intervals so that when bluetooth functionality is enabled later on, the calculations will be completed, updated, and sent at reliable intervals to clinicians.

Testing Method

This functionality will be tested by running the code and manually timing when the flow rate calculations appear on the serial monitor in Arduino.

BLE Peripheral Connection

Description

Bluetooth Low Energy (BLE) is the connection that allows flow rates to be transmitted wirelessly. As the BLE connection is largely dependent upon the code initialization and app functionality, we expect that the connection is successful. If the BLE connection is unsuccessful then both the code and app will have to be checked to ensure that they are functioning properly.

Significance

It is important that BLE is properly connected in order for clinicians to eventually be able to use the bluetooth to obtain flow rate calculations and alter the pump settings / states as required.

Testing Method

The BLE Connection will be checked using the app “nRF Connect”. The app will connect to the bluetooth and once connected will display a “connected” message. This same “connected to bluetooth” message will be displayed on the serial monitor of the Arduino, confirming that the connection was successful.

LED is turned on/off with BLE connection

Description

The builtin LED pin on the board can be controlled using bluetooth as further validation of the BLE connection. It is expected that when the nRF Connect app is connected to BLE central, if the Characteristic is set to “0” the light will be off and if it is set to “1” the light will be on.

Significance

This test is a very simple way to validate the connection to BLE and ability to write to a Characteristic using the nRF Connect app. It provides a visual depiction of the connection and writing to a Characteristic which follows a similar process to the turning on/off of the pumps as tested above.

Testing Method

The board will be connected to BLE central on the nRF Connect app. Before writing a new value, the LED light should be off. Write a “1” to the LED Characteristic and verify the light turns on. Write a “0” to the LED Characteristic and verify the light turns on.

Pumps are turned on/off using BLE connection

Description

Both Pump A and Pump B states can be controlled using bluetooth. It is expected that if the nRF Connect app turns the state off (which is done by setting the BLE Characteristic associated with either pump to “0”), the pump will turn off and stop pumping. If the expected outcome does not occur, there is likely an issue with the BLE connection and the above test will be performed again.

Significance

It is important that the pumps can be turned on and off through BLE connection because clinicians will need to be able to control the pump states for various tasks. The most common reason that clinicians will need to turn off the pumps is so that the patients can take an MRI scan without running pumps interfering with the scan’s results.

Testing Method

The pump states and their changes can also be monitored using the nRF Connect app. The pump in question (either A or B) will be turned off using the app by setting the BLE Characteristic associated with the pump to “0”. The serial monitor on the Arduino will then be checked for the “pump X is turned off” message which confirms that the pump was turned off using bluetooth connectivity.

Receives pump A rate at expected intervals over BLE

Description

The calculated flow rate for pump A is to be transmitted to the user over BLE every 10 minutes, or for testing purposes at some specified shorter interval that is easier to observe behavior. The estimate for the flow rate is the average of the previous rate calculations performed since the rate was last transmitted. When running pump A, it is expected that the pump A rate Characteristic is set to a double value. When pump A is off, it is expected the value will be read as 0 at these same time intervals.

Significance

One of the most important aspects of the code’s functionality in the context of the entire medicine delivery project is the ability to monitor medicine infusion data. This is accomplished through the read only characteristic for the pump flow rate estimates, which have an updated value set at a constant time interval. This interval is default set to 10 minutes, but can be easily modified to monitor the flow rate more or less frequently.

Testing Method

Connect to the board using nRF Connect App and turn pump A on. Verify that a double value is read by the pump A rate Characteristic at the specified time intervals. Turn the pump A off and verify that a 0.0 double is read at these intervals to represent no current medicine infusion.

Receives pump B rate at expected intervals over BLE

Same test as above, but with pump B.

Correct behavior upon BLE Peripheral disconnection

Description

When disconnecting from BLE Peripheral by disconnecting from central using the nRF Connect App, all other actions should continue to run in the loop function. If the pumps were turned on during the connection, they should continue to run, and the rate estimates should be sent to the user at specified intervals as discussed above. When disconnected, the user should no longer be able to write new values to the Characteristics.

Significance

This functionality is important because as little action as possible is desired while connected to BLE. Connecting to BLE the entire time would waste power and consume energy at an unsustainable rate for this application, so connecting and disconnecting with core functionality continuing to run is a necessary.

Testing Method

Connect to BLE with nRF Connect App, write values to any or all of the pump and LED Characteristics, disconnect and then observe the behavior in the Serial monitor or board. The pumps should continue to run and light continue to function once disconnected.