GNG2101 Deliverable D Report

Eye Gaze Camera Cover

Submitted by
[TEAM IDENTIFIER: B13, Team 13]

[Marwan Mashaly, 300228296]

[Brandon Yeung, 300250806]

[Wilt Moise, 300241883]

[Abd Ennour Souit, 300256956]

[Anna Brimacombe-Tanner, 300245481]

[Ian Mondoux, 300247598]

October 9th 2022
University of Ottawa

Table of Contents

List of Figures

List of Tables

Introduction

This report covers various topics relevant to the project deliverable D. First we discuss our client meeting 2, which gives a summary of the meeting, feedback on our solutions, and answers to our questions needed to further progress in this project. We discuss the designs of our IOS and Windows applications, with illustrations that provide information on the general functionality of the applications, and the multi-step process for both software to initiate synchronization between each other, and ultimately execute the necessary commands to turn the Tobii cameras OFF and ON. Later in the report, we discuss the critical assumptions for the prototypes in production. These assumptions range from access to the hardware and software of the Tobii Dynavox machine, potential compatibility issues between both IOS and Windows applications, and more. Additionally, we provide insight into the feature documentation and demonstration of our IOS and Windows application prototypes. These include the explanation of the general user interface, core functionalities of buttons, and design explanations. We provide further illustrations of the bill of material for this prototype which culminate to a price of 0 dollars. This bill shows all the necessary utilities needed for this stage of development to construct our current prototype. Furthermore, we discuss the prototype's test results and evaluations, illustrating the functionalities of key components of the user interface, and the next steps of implementing and testing. Lastly, we discuss the measures we will be taking for our next client meeting, informing the patient of the current status of the project and an explanation through PowerPoint presentation. Not to mention the list of questions we have prepared to clarify any discrepancies from our side and to further fortify our understanding of the project.

Client Meeting #2

During our client meeting our goal was to present three viable solutions to the client and ask the client for any advice and input on our choices. Following the design process our team was able to analyze the information given to us in the client meetings and through our benchmarking process and advice from clients we were able to come up with a solution to their problem

Feedback on 1st solution

The client enjoyed that this solution was a software based solution as well as requiring no needed hardware to complete this solution as it will be using his phone and Tobii camera. The client really likes the simplicity of our prototype of how our app will look like. She mentioned not having a complicated app with many buttons as it may be hard for the user to operate it. By streamlining our software solution to enable a direct link between devices with clear simple application software the client believed that this could be a long lasting viable solution to optimize his communication system.

Feedback for 2nd Solution

The client liked the creativeness with this solution and was intrigued by the waves used to cancel out the frequency. The client had previously already tried this specific hardware solution and although effective it did not provide the amount of accuracy required for optimal communication.

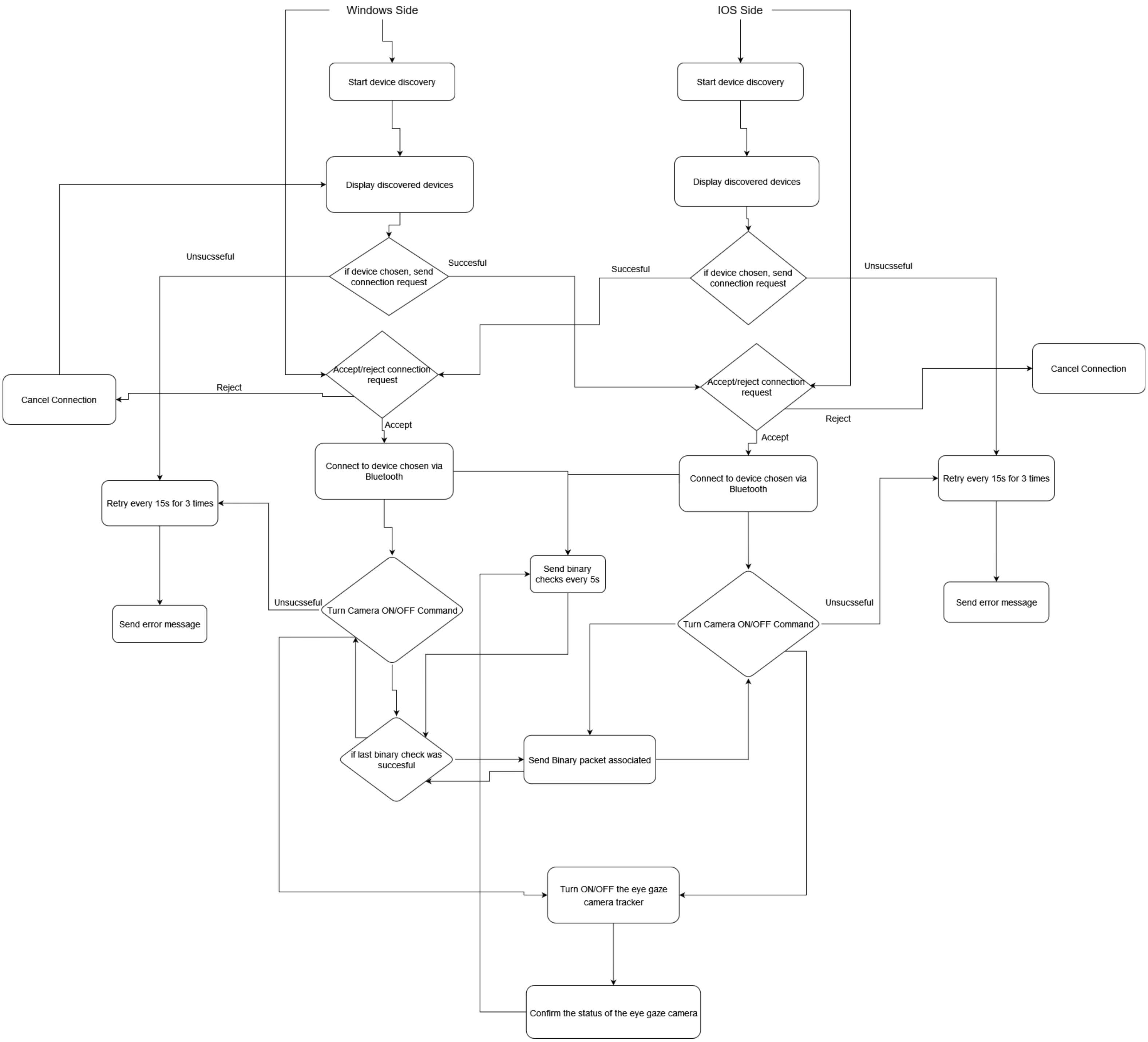
Feedback for 3rd Solution

The client did prefer if this solution was able to turn off and on the application, however we were not yet aware that that was his preference. This solution would be an integrative software and hardware solution that seemed promising to the client based on its creativity and innovation using microcontrollers to distort the infrared rays from distorting the Iphone communication software. We also discovered that the communicative dot was located on his forehead instead of his glasses so the hardware would be unlikely and uncomfortable.

Questions answered

- What is the process to turn off the Tobii camera? The hardware settings are opened and the eye tracker is turned off.
- We saw that there is a developer space for tobii and to get a version of the pdk/sdk we need to contact someone from their team so are wondering if you have those contacts? We were advised by the client to ask the Dynavox rep, Danielle Franklin these questions as she was not sure how to answer this. She also mentioned there is a 30 day free trial for Grid 3
- Is the eye gaze camera controlled by the tobbi device or the grid 3 software or both? The grid 3 software is running on the tobbi device which is just like any windows computer however it is equipped with an eye gaze camera.

Detailed Design



Windows Side
IOS Side
Start device discovery
Start device discovery
Display discovered devices
Display discovered devices
Unsucsseful
if device chosen, send connection request
Succesful
Succesful
if device chosen, send connection request
Unsucsseful
Accept/reject connection request
Cancel Connection
Accept/reject connection request
Cancel Connection
Reject
Reject
Accept
Accept
Connect to device chosen via Bluetooth
Connect to device chosen via Bluetooth
Retry every 15s for 3 times
Retry every 15s for 3 times
Send binary checks every 5s
Unsucsseful
Turn Camera ON/OFF Command
Turn Camera ON/OFF Command
Unsucsseful
Send error message
Send error message
if last binary check was succesful
Send Binary packet associated
Turn ON/OFF the eye gaze camera tracker
Confirm the status of the eye gaze camera

Figure 1: Windows - IOS full application flowchart

Our software design has two major parts which can be classified into two different applications that will be deployed at the different devices of the user. For the most part the two applications will be serving the same purpose when it comes to UI usage and functions however the windows side will have an extra step. Which is to run the powershell scripts in the background to be able to close the eye gaze camera and open it again based on command from either applications. The first thing the program does once it is launched is to start the device discovery and return all devices found that the device can connect with. For user convenience we decided that he can initiate the bluetooth connection from either devices based on what he was using before deciding to do the connection.

The next step would be for the user to select the device he wants to connect to and this should be either his Iphone or his tobii dynavox. In the background, the two devices are going to start the pairing process using bluecove and ios APIs for bluetooth communication. Once the pairing process is complete, the user will get a message and the status will be updated to connected.

On pressing the button to turn the camera ON or OFF the program will check if there is a secure connection. If the connection is secure it will send the specified packets to the respective device to update the status of the camera while simultaneously run the powershell script from the windows app to turn of the eye gaze camera.

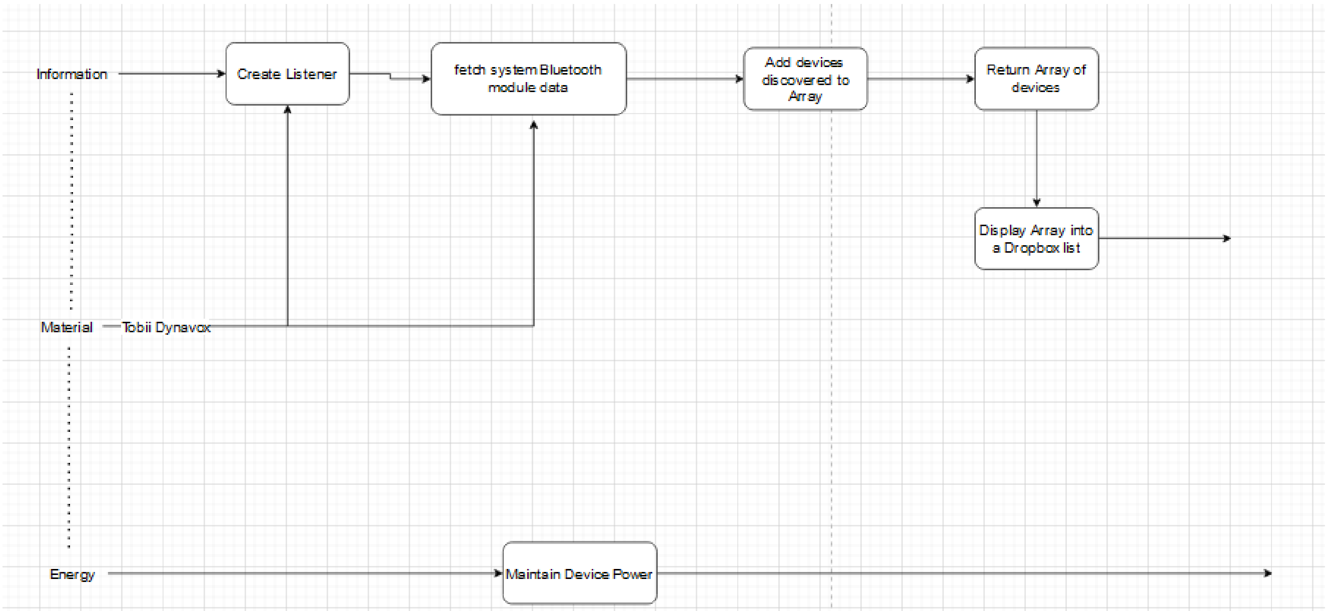

Information
Create Listener
fetch system Bluetooth module data
Add devices discovered to Array
Return Array of devices
Display Array into a Dropbox list
Material
Tobii Dynavox
Energy
Maintain Device Power

Figure 2: Getting Device List Diagram

To fetch the devices we will be using prebuilt APIs to ease the development process and not to reinvent the wheel. Using the API we will create a listener so that when commanded it will fetch all discoverable devices and append them to a list and display the list to the user so he can choose the device he would like to connect. We adopt this methodology of making the user choose the device so that it can be compatible with any other devices in case he wanted to upgrade or change his current device.

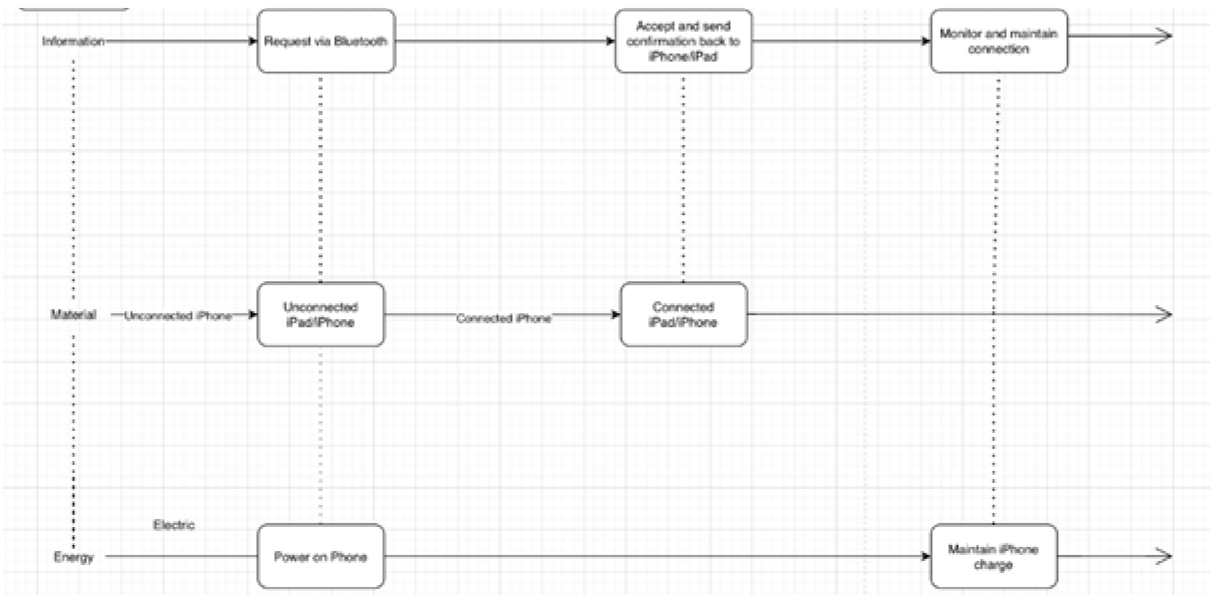


Figure 3: Establish Remote Connection Diagram

Connecting to the device via bluetooth would be the next step once the user selects the device he would like. This will be done through the above process where we would send a connection request using the API and then accepting it through the pop up that will be shown that the respective device would like to make a connection. Most of the communication and validation will be done through the hardware and device system so we do not have to interfere with that.

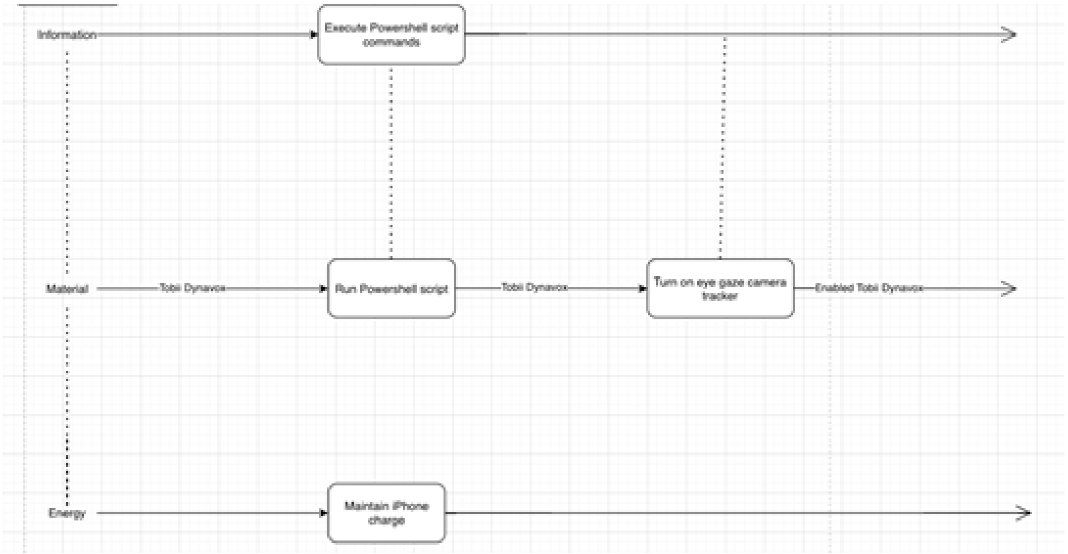


Figure 4: Turn on the Eye Gaze Camera Tracker

Once the user establishes his connection and wants to turn the camera ON or OFF certain conditions like device connection, packet message, and current state will be checked. If all conditions were true, the windows application will execute the powershell script. The purpose of this powershell script is to call the settings of the eye gaze camera to close and open the camera upon request. The assumption that this is feasible since the settings rely on powershell to interact with the hardware layer. However this assumption will not be possible to test during this prototype as we need access to tobii dynavox software development kit and the specific contacts to get this were not provided in a timely manner to ensure its completion during this prototype.

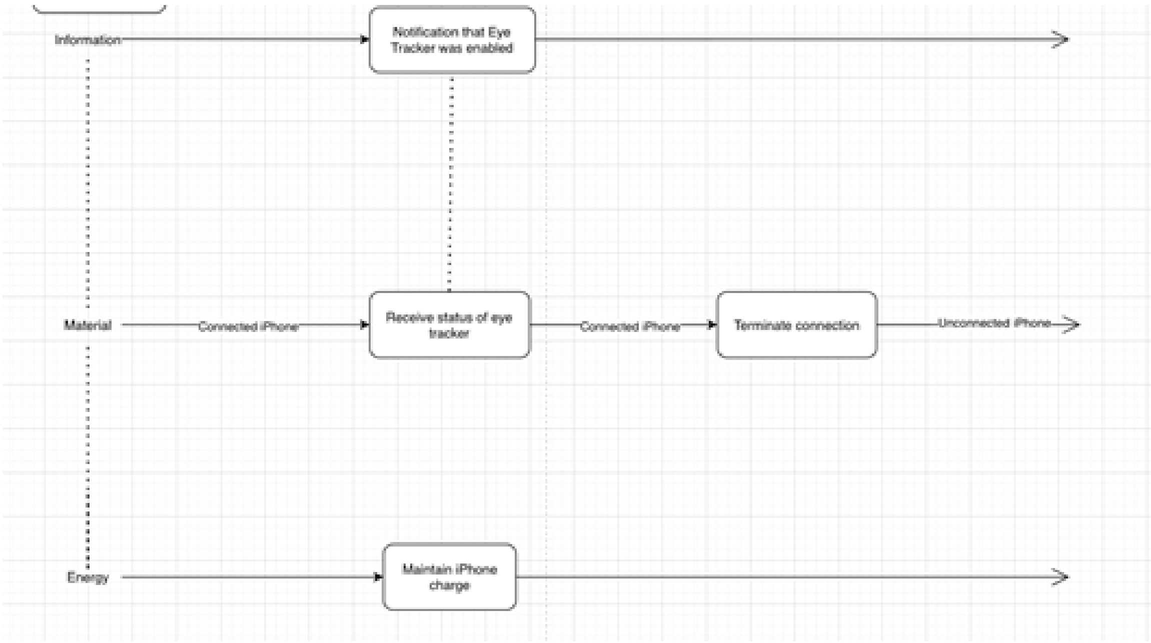


Figure 5: Confirm Status of Eye Gaze Camera Diagram

After executing the powershell script we will do one last check before the application goes into a stagnant state where it only checks for updates every second by sending empty binaries to ensure the connection is still persistent. This check will be to confirm that the state of the eye tracker camera has changed and this will be done through the tobii dynavox software development kit as it provides certain APIs that we can call to do this check.

Critical Assumptions

The following assumptions derived from the research of our prototype will be split into three sections, each of a critical functionality level.

Downloading software onto Tobii DynaVox.

The first assumption focuses on the access levels of the Tobii Dynavox software and the ability to download third-party software within it. Tobii Dynavox is a reputed software company known for generating, manufacturing, and distributing speech-generating devices across the globe. As such, an enormous company such as Dynavox needs to maintain the safety and protection of their software from any outside user. This consideration is factual for any company since giving open access to confidential software to users will cause massive damage to that organization financially and ethically. By being software giants, permitting access to the software that is their principal source of income, to six serendipitous university students is not plausible. Since access to Tobii software would mean the capability of tinkering with source code, adding and removing scripts, and entrance to confidential scripting secrets, tactics, and practices making it even less likely to be granted a permit to the software. During the second client meeting, we asked the client whether it was possible to download third-party devices onto the Tobii Dynavox device, and the reply was a hesitant yes, however that may stem from the lack of software knowledge the client has, since no software company with a logical development department would grant access to confidential software to outsiders. The client instructed us to email an individual she is acquainted with at Tobii DynaVox regarding software accessibility from a third-party such as ourselves. Therefore while waiting for a reply from the Dynavox team, further analysis and research must be conducted for an alternative.

Compatibility between Windows and IOS applications

The second assumption focuses on the compatibility between Windows and IOS applications. Ultimately, when the IOS device sends a signal to the Tobii DynaVox device to establish a connection. As such, triggering the signal required for running the PowerShell scripts added onto the client's computer is purely speculative. Initiating a prompt from the IOS device to send over to the Tobii is effortless, but how the Tobii device extrapolates the IOS encrypted signal is the core of the issue. Trying to find a medium that both devices can comprehend when sending signals back and forth to stabilize a connection will require further research on this domain. Furthermore, the PowerShell script should alter the current state of the EyeGaze Camera Tracker, enabling it from the command line with administrator privileges, however, that is assuming the script can indeed grant administrator privileges. Doing so will allow us to create modifications on a computer that will positively affect the user. These privileges will grant us access to change security settings, make software modifications, and access all files on the computer. However, being able to get admin privileges from an instantiation on an opposite IOS device proves troublesome since we are using two completely separate pieces of software to execute this command. The IOS device will have to account for the language barrier through specific alterations to the code. The device receiving this command will not be a similar IOS device signal sent, but rather a Windows computer running third-party software.

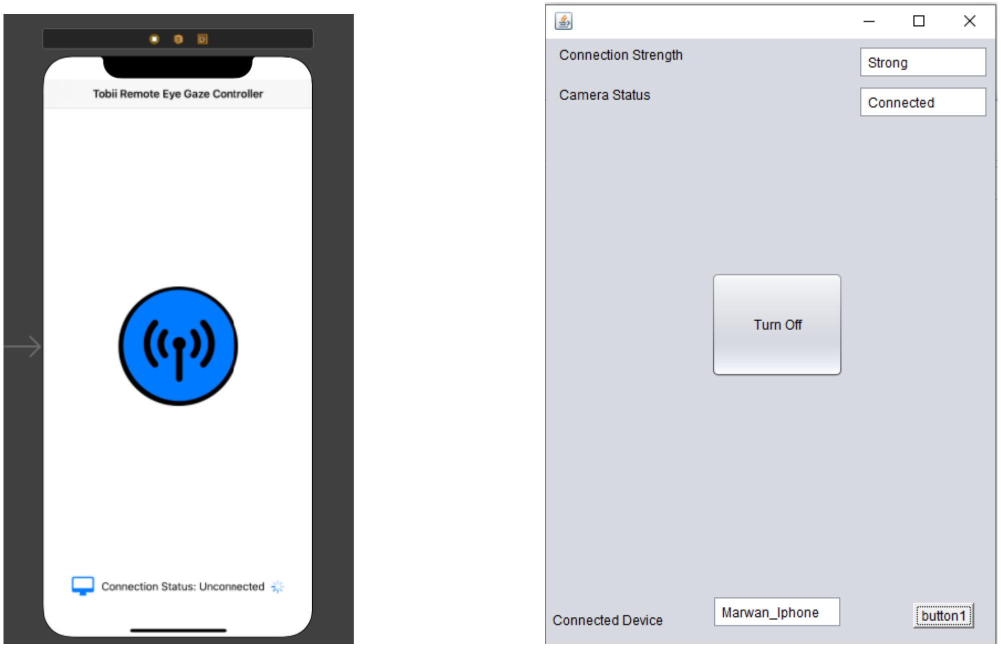
Bluetooth levels stabilization

The third assumption centralizes the connectivity of the devices in use to the same Wifi with stable Bluetooth frequencies in the given area. When the mobile application sends a 2.4 gigahertz radio frequency signal to the Tobii device to synchronize the two devices together, however, we are not accounting for external interferences that can destabilize the frequencies. Utilities such as microwaves can destabilize the connection along with wireless devices in the surrounding vicinity that can disrupt Bluetooth connections. As such, we will assume that the patient will not be using his devices near any Bluetooth disrupters since accounting for such a nuisance would require sending signals with a plethora of alternating frequencies that would avoid such disruptions which are beyond the scope of this project. Lastly, we require the Tobii DynaVox device to conduct tests since this device serves as the medium to run the Windows application, and without access to this device, all approaches are purely speculative. Without access to this piece of hardware, tinkering with the flexibility of the user privileges and testing the extent of the device's capabilities will further allow us to wander the realm of speculation and assumption.

One can only hypothesize for so much without actual hardware and software access. To properly test software, access to the hardware and software of the Tobii Dynavox must be given to our team for further progress to be attained. Overall, the aforementioned assumptions will be analyzed, after gaining access to the Tobii DynaVox device to test said hypotheses, and we will continue to operate under the assumption that our methods are potential tautologies until proven otherwise.

Prototype Demonstration and Feature Documentation

Our current prototype implements mainly front-end UI features while the back-end features remain a work in progress. We had decided that getting the interactive part of the project out of the way would allow us to better focus on the back-end technical aspects of the project due to the heavily abstracted nature of the Bluetooth and sockets programming implementations. The current status of our prototype is depicted in the two following images:



Connection Status: Unconnected

Connection Strength Strong

Camera Status Connected

Connected Device Marwan_Iphone

Figure 6: Prototype of the two Applications

The image on the right is the current rough prototype of the mobile application to be installed on our client's device. The image on the left is the second part of our prototype which is required for communication between our client's Windows Desktop and the mobile device.

Mobile Application

The front-end of the mobile application can be broken down into three main elements. Firstly, the button, the main focus of the application as well as the part which our client would be interacting with the most. Secondly, the connection status, demonstrating to what device the application is currently connected to as well as depicting the current quality of the connection. Finally, the settings button would allow the altering of various attributes of the Remote Eye Gaze Controller app.

Connection Button

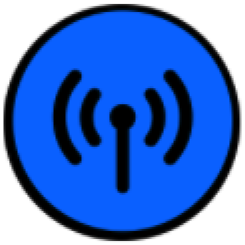


Figure 7: Connection Button

Design

The design of the button was based around the needs of the client established during our previous client meetings. As previously stated, our client is a GMFCS5 level patient who relies heavily on eye tracking software for just about every aspect of their daily life. Through our own testing of eye tracker software as well as from the input of the client, we concluded that 3 things were to be kept in mind when implementing the connection button; the position, the size, and the color. The eye tracking software the client relied on to control the mobile application was far less accurate than a human hand. Unlike the tracking software on the client's desktop which tracked the client's eyes using infrared light, the software used on the iphone was more of a workaround. The system utilizes a head mouse, a sort of tracking mechanic where the user places some sort of reflective dot on their head and the phone's camera captures their head movements. This means that the head mouse was more or less purely dependent on the user's head orientation, only keeping rough direction of where the user's eye's may be directed in. Additionally, our client also had less neck mobility than a non GMFCS5 patient. Consequently, we decided to make the connection button a large button placed in the central point of the screen with no surrounding interactive components. This allowed us to off-set the inaccuracies of the head mouse.

Current Functionality

Although the back-end components required for establishing a solid and secure connection between the mobile application and a desktop device are still a work in progress, we have managed to implement a couple of quality of life features thus far.

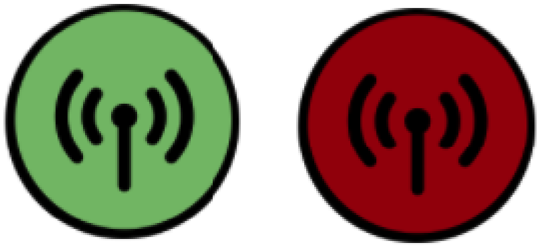
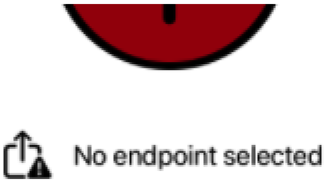


Figure 8: Connection Button Status

The first feature which we implemented within the connection button allows for it to change its color when clicked. Since our client is not interacting with the phone physically, they would lack any form of tactile feedback in the form of vibrations. In order to let the user know that a connection was attempted, we decided that our button would turn green on success and turn red on failure. Reasons as to why the connection failed would then be displayed under the button, whether it be the endpoint's bluetooth being turned off, no device being selected to connect to, etc .



No endpoint selected

Figure 9: Status Warning

The second feature which we implemented relates to how the head mouse interacts with the button. The head mouse implements a mechanic known as “dwell”, meaning that when interacting with the phone, the user is performing less of a click, and more of a press and hold. The time duration for this hold is defined as “dwell time”. Thus we decided to provide a meter depicting how long the button has been held down and the amount of time required for the click to be completed.

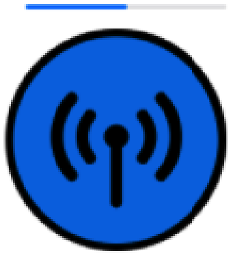


Figure 10: Status Dwell

Bottom Navigator

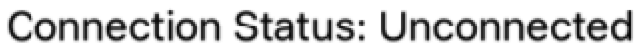
Design

The design philosophy behind the bottom navigator is based almost purely on communicating information quickly and effectively. It is composed of 3 parts; a device button which opens up a bluetooth panel where the user can then switch their selected device, the connection status depicting which device the mobile application is currently connected to, and a status bar measuring the current connection quality.

Current Functionality

At this point in time, the bluetooth panel to select which device to connect to is still a work in progress. However, the connection information and connection quality are completed aside from the back-end reliant component.

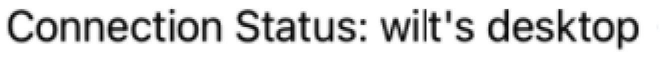
Connection Information



Connection Status: Unconnected

Figure 11: Connection Status

The connection information label is currently capable of displaying whatever device is currently connected to the mobile application. When a device is connected, the "Unconnected" is altered to display the current connected device's bluetooth name.



Connection Status: wilt's desktop

Figure 12: Connection Status With Laptop

(Not a demo of a successful bluetooth connection, just the ability to pass device information to the label)

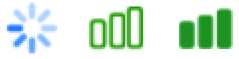


Figure 13: Quality of Connectivity

The final piece of information displayed by the bottom navigator is a connection measurement scale. Prior to connection, the scale simply displays an animated pending emoticon, signifying that it is still waiting for a valid connection to be established. The middle image symbolizes that the device is successfully connected but that the transfer of information may be faulty due to an unstable connection. Finally, the rightmost image demonstrates when the connection is solid and that the remote control should be successful. The measurement of the connection is dependent on the percentage of packets lost and received in the communication between the mobile application and the connected device.

Windows Application

The windows application has yet to have any implemented features and is purely a non-functional UI design. Our design philosophy for the windows application is more or less the exact same as the mobile application's. This program is meant to be launched within Grid3 to allow for the mobile application to remotely control the Tobii Dynavox device and its interactive components are designed solely around the scenario that our client may need to terminate the connection from their desktop.

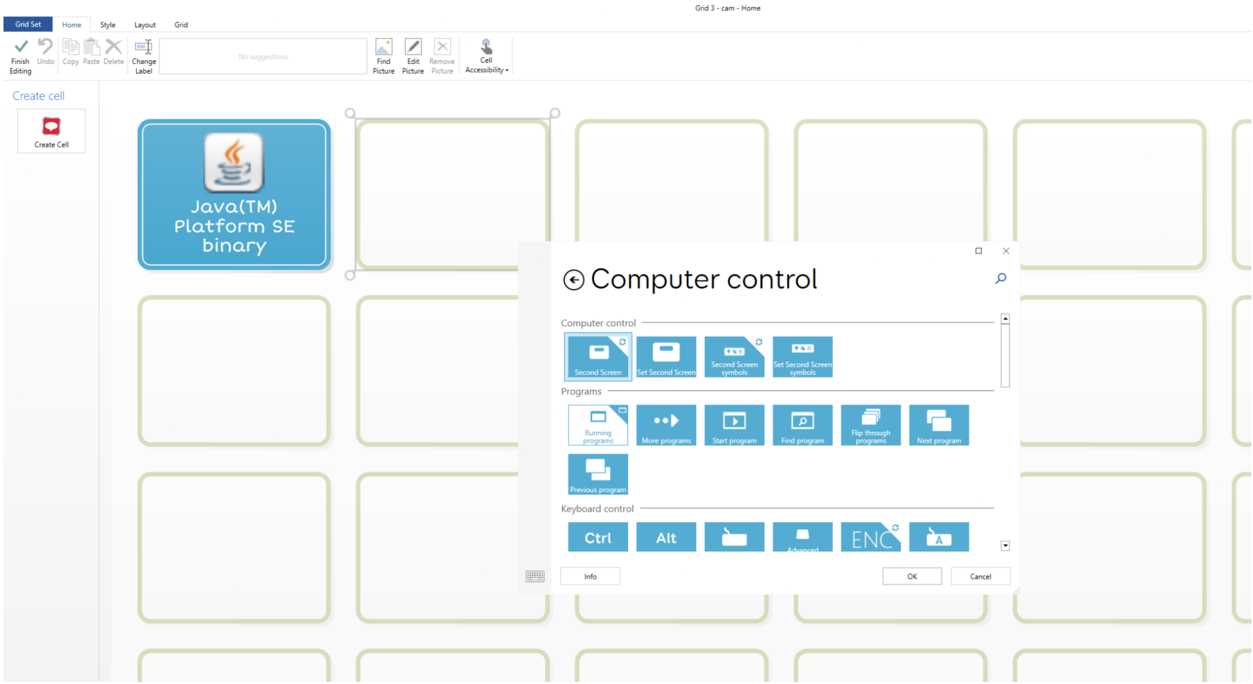


Figure 14: Grid3 Developer Platform

BOM

This Bill of Materials concerns the software required to create our program that will accomplish our client's mission. Each part of this table is what showcases the specifics units of measure of each item, how many of each is needed, the link to find it, and the overall cost (with and without taxes).

Table 1.0: BOM

Item Name	**Description**	**Units of Measure**	**Quantity**	**Unit Cost ($)**	**Extended Cost ($)**	**Link**
Grid 3 (Trial)	60 day trial version of the program Grid 3	N/A	1	0	0	https://thinksmartbox.com/download-grid-3/
Java	Scripting language for the software	N/A	1	0	0	https://www.java.com/en/
Xcode	App development	N/A	1	0	0	https://developer.apple.com/xcode/
Blue Cove	Bluetooth API for Java	N/A	1	0	0	http://www.bluecove.org/
Swift	Programming language	N/A	1	0	0	https://developer.apple.com/swift/
Apache NetBeans	Development Environment for applications	N/A	1	0	0	https://netbeans.apache.org/
MakerLab 3D Printers	3D printers found in makerspace	mm	1	0	0	https://www2.uottawa.ca/faculty-engineering/spaces/richard-labbe-makerspace

Total Product Cost (No Taxes or Shipping Included $)	0
Total Product Cost (With Taxes and Shipping $)	0

Prototype Testing and Evaluation

The following are the components that were discussed, tested, and evaluated within the IOS application. Each will mention the component's core functionality test, along with the expected and actual results achieved. These tests are not consistent with the metrics created in PB B since those metrics are conclusive metrics for a working product, as such in this deliverable, the tests conducted solely depend on the user interface of the IOS application.

Table 1.1: Prototype Testing and Evaluation

Component for testing	Test	Expected Value	Accepted Value
ON/OFF button of IOS application	Capability to click the button while surrounding UI features remain functional	Fully functional while not glitching and disrupting functionality of other aspects of the UI	Fully functional while not glitching and disrupting functionality of other aspects of the UI
Settings button of IOS application	Capability to click the button while surrounding UI features remain functional	Fully functional while not glitching and disrupting functionality of other aspects of the UI	Fully functional while not glitching and disrupting functionality of other aspects of the UI
Connection status bar of the IOS application	Capability to view device connected with the application	Display demonstrating what device the application is connected to as well as illustrating the quality of the connection	Display demonstrating what device the application is connected to as well as illustrating the quality of the connection
Bluetooth Tethering	Being able to send data packets from IOS device	Cannot Test	Cannot Test
IOS and Windows compatibility	If Windows application can receive signal sent by IOS application	Cannot Test	Cannot Test

Stable Bluetooth connectivity	If connection between Windows application and IOS application remain stagnant for long periods of time	Cannot Test	Cannot Test

ON/OFF button of the IOS application

The goal of the ON/OFF button for the scope of this project is the ability to click the button while the remaining UI features stay functional. This functionality implies that across the user interface if the user clicked a button, it should not disrupt other aspects of the UI, and should solely execute the command initiated by the respective button clicked. During the first trial run, after the basic user interface was created along with its principal buttons such as the ON/OFF button and the settings button, when clicking the ON/OFF button, ON/OFF would slightly change position from its central location to the right of the screen. The developer space used to build this application is named Xcode, a space used to program applications on the IOS environment. As such, further analysis of the functionalities created on Xcode was analyzed, and ultimately after restarting the computer, and trying the trial once more, the malfunction did not occur. This malfunction seemed to stem from the Xcode software, however, this occurrence was not a repeating one. Ultimately, after conducting more trials and even restarting the computer, the button was clickable and did not disrupt any other core functionality of the user interface proving the first test successful.

Settings button of IOS application

Similar to the ON/OFF button, the Settings button concentrates on the capability to click the button while surrounding UI features remain functional. After testing the ON/OFF button, the time came to test the Settings button. The expected outcome of this experiment was a fully functional settings button that was able to intake click commands while not glitching and disrupting the functionality of other aspects of the user interface. When testing the button for the first time, the button was functional and did not disrupt other features within the application. We tried different variations of clicking for the next trials, which involved rapidly clicking on the button, clicking and holding, and more simple clicking strokes. These variations of clicking did not diverge in functionality and ultimately did not disrupt the overall user interface. Finally, the Settings button did not malfunction and instead produced quality results.

Connection Status Bar of the IOS application

Lastly, the final component capable of testing is the connection status bar. The functionality of this bar focuses on the ability to view potential devices connected with the application. This functionality implies a display demonstrating what device the application is connected to while mentioning the unconnected or connected status of the bar with further illustrations of the quality of the connection. Since the scope of the deliverable does not require the application to connect with the Tobii device or any computer, the status bar shows unconnected, as its status clearly demonstrates that while not connected to a device, it will befit the necessary scenario in terms of display. As such, it passes the test case of showing unconnected as its status for not being connected to an external device. Lastly, connectivity status is an indicator of the functionality of one of the sub-components which is the quality of the connection. For now, the status connectivity can pass information to the quality of connection label indicating that it is not connected to any device, since that is the default state of the device, and it's the only state for the time being. To conclude, the status bar is capable of displaying its singular connectivity status while also showing its singular quality of connectivity for being always unconnected therefore, we will focus on being able to illustrate multiple statuses and quality of connectivity for the two remaining edge cases for future deliverables.

Bluetooth Tethering, IOS and Windows compatibility, and Stable Bluetooth connectivity

The last three components all share similar expected and accepted values which are of incompletion. First is Bluetooth tethering which in essence focuses on being able to send data packets from the IOS device, however, this aspect of the prototype cannot be tested since it is outside the scope of this deliverable. Likewise for the IOS and Windows compatibility which considers if the Windows application can receive signals sent by the IOS application to execute certain commands, however, just like the previously mentioned case, this functionality focuses on back-end design which was not the focus for this deliverable. Lastly is the stable Bluetooth connectivity between both cross-platform applications, which suggests whether the connection between Windows application and IOS application remains stagnant for long periods of time without interruption from external sources. The scope of this report only focuses on UI functionalities, and not back-end and cross-platform synchronizing. Therefore, all the before-mentioned components were not tested for this project deliverable. Furthermore, we are still missing access to the Tobii DynaVox device to conduct more accurate tests. Without the Tobii Dynavox device, all tests conducted between the IOS application and any other Windows computer would not meet conclusive standards, since we are supposed to trigger a PowerShell script to give us access to shutting the Tobii device's cameras on or off. To finalize, we prioritized the user interface aspect of the applications for the scope of this deliverable which implies the inability to test back-end functionalities currently. And without access to the Tobii device, we can only conduct tests up to the preliminary level. Therefore on future deliverables, we will focus on back-end integrations and gaining access to further resources such as the Tobii device to ensure satisfying results.

To conclude, the user interface tests proceeded as planned although bumps were encountered, the team was diligent and critical in solving all issues encountered. Therefore, the next step of this process is integrating back-end functionalities to the buttons and commencing trials for connection of the Windows application and the IOS application, however, without access to the Tobii DynaVox device, only preliminary trials can be conducted, as we will not have conclusive answers as to whether these applications synchronize the same way if not tested using the Tobii device. We will use the limited-time free access of Grid3 to commence separate trials with a different computer in order to solidify the aforementioned approaches and conduct preliminary tests while waiting for access to the Tobii device.

Plans for Next Client Meeting

In our next client meeting, we would like to demo our first prototype and get feedback from them. As the first prototype revolves around building the UI and ensuring it will serve the purpose of what we need. It will be important getting the client's feedback on how they feel on using the app. After showing them the two apps we built, we will explain their current functionalities and a more technical explanation of how it should act. After giving a brief explanation we will go into providing more information as to what we plan to do next which would be building the back-end phase. Given our explanation of the next stage we will take note of their feedback and what they would like to add or change in the current and the next steps. Lastly, we will ask them the following questions to improve our understanding and ensure that we meet their expectations.

- In the case of a failure to shutdown the eye gaze camera how would you like the software to react?
- Are you satisfied with the current UI and would you like to finalize it?
- Is there a particular aspect of the UIs you would like altered?
- Have you thought of any additional functionalities you might desire that are in the scope of this project?
- May we receive access to the Tobii DynaVox device?

Therefore, our next client meeting will illustrate a detailed presentation illustrating our progress, findings, and questions. We will also give a sneak-peak at our user interface for the IOS application in hopes to receive feedback and satisfy client expectations. Throughout we will take notes of the discussion, while also recording the conversation. Overall, we will enter the discussion with an open mind with the reminder to learn even more than our previous encounter, empathizing with the client in hopes of further fortifying our resolve for the completion of this task.

Conclusion

The second client meeting provided important new information and introspection of the clients specific needs, likes and restrictions. It was insightful to understand the details we can include to make the solution as effective as possible. Specifically the details on our first idea, which enabled us to see how the complete software based solution would be the most helpful and user friendly for our client. In our second idea, we further learned of the specifications and preferences in terms of camera accuracy, on and off applications and prior experience with the proposed hardware. In our third idea we revisited the placement of the reflective tracking dot, deferring from where we originally were told it was located, allowing us to refocus our attention on a more streamline software solution.

Wrike Snapshot Link

https://www.wrike.com/frontend/ganttchart/index.html?snapshotId=F1g6Evks84Ta46jQEa0XMcxcEMCVKT4n%7CIE2DSNZVHA2DELSTGIYA