Go Baby Go: Final Design Report

Margaret Brown, Kaylee Mock, Anna Yrjanson

Client: Bethany Sloane, Go Baby Go Oregon

Advisor: Dr. Tammy VanDeGrift

Instructor: Dr. Andrew Nuxoll

2023 – 2024

University of Portland Capstone

Table of Contents

Executive Summary

In most cases, mobility devices for children, such as manual and power wheelchairs, lack integrated motion-tracking capabilities. Physical therapists and researchers use motion data to analyze how young children interact with mobility devices in natural environments. This data is helpful for many reasons: it helps therapists best teach children how to navigate and interact with their environment and peers, tracks overall device usage to help better understand outcomes, and decreases the caregiver burden of manually tracking device usage. The most accurate motion-tracking devices require reliable cellular or GPS signals, which not all families can access. This inequity has created a need for a motion-tracking device that can be used by any family, regardless of their access to cellular data.

The Adherence Sensor attaches to the back of the Permobil Explorer Mini and most mobility devices. The sensor records critical timestamps, such as the beginning and end of a child's movement. Therapists and researchers can then use this data to determine if device usage is associated with child outcomes and how children navigate their environment in natural settings. The current team's device addresses some of the inconveniences of the previous year's prototype, which helps therapists and caregivers use the device. The improved tracking device provides user interface improvements, including Bluetooth file transfer and LED light indications of low battery power and device idling. The device offers a comprehensive solution for collecting data, particularly in home settings.

Introduction & Problem Statement

Mobility disabilities may impact a child's ability to move, interact with peers, and complete daily activities, influencing their overall development. Understanding the impact that mobility disabilities can have on a child's development is important to moving towards mobility equity and promoting mobility as a human right. One potential solution to moving towards mobility equity is for young children to have access to powered mobility devices.

Using powered mobility devices has improved young children's communication, motor skills, socialization, and autonomy. However, they are often inaccessible for infants and toddlers with disabilities due to the limited availability of devices for children under the age of 3. The options for powered mobility devices in this age group include the Permobil Explorer Mini and modified ride-on toy cars. The current team was motivated to improve powered device's capabilities to help shape the future of people with disabilities and the medical field. While the previous prototype was functional, the new prototype's changes will improve system accuracy, efficiency, and cost. Most importantly, they will enhance the experiences of children with disabilities, caregivers, and physical therapists.

Background / Literature Review

In January 2019, Trexo Robotics created Trexo Home, a therapeutic mobility device that promotes upright walking in children as young as 12 months, as shown in Figure 2 [6]. It features a tablet that can monitor the mobility device's speed, data, and child's progress. Families can purchase the Trexo product for just under $40,000 or lease it for about $1,200 monthly [25]. Similarly, powered wheelchairs such as a Quantum Stretto can cost upwards of $30,000. These products are expensive and often inaccessible to children under the age of 3. There is a need for more accessible devices for young children to promote active mobility in early childhood.

Permobil, a company created to help provide advanced mobility technology for individuals with disabilities, created the Explorer Mini, as shown in Figure 1. The Explorer Mini supports children ages 12-36 months old and has been safety tested and FDA-approved in 2020. The Explorer Mini is powered by a 6 Volt battery, reaches 1.5 miles per hour, and has 5 speed levels. The Explorer Mini device can be used in seated or standing modes. It has developmentally inspired seating to promote freedom of movement in the seat, allowing a child to reach beyond the device and explore their environment. The Explorer Mini has a midline joystick with a 360-degree turning radius for steering that is embedded into a tray. The tray supports a child’s upper extremities to help them maintain an upright position. The entire chair is supported on a four-wheel platform [26]. The Permobil Explorer Mini was listed in TIME Magazine's "Top 100 Best Inventions of 2021" [1].

Figure 1: (Left) Child sitting in the Permobil Explorer Mini

Figure 2: (Right) Child standing in the Trexo Home

The Adherence Sensor, shown in Figure 3, offers essential data insights into a child's interactions with the Explorer Mini device when used in natural environments. By recording critical timestamps of movements and engagement with the Explorer Mini, the sensor assists physical therapists in understanding a child's overall usage of the device. By leveraging the Adherence Sensor's capabilities, physical therapists can provide tailored guidance and support to optimize children's developmental outcomes using powered mobility devices such as the Permobil Explorer Mini.

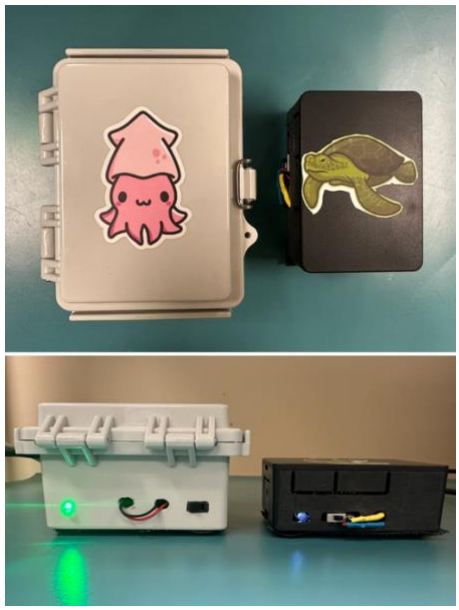

Figure 3: Adherence Sensors side by side (left: current sensor; right: previous team's sensor)

Design Criteria and Considerations

Design Criteria

As detailed in Table 1, The Adherence Sensor ensures continuous data monitoring with user-friendly features for caregivers and physical therapists. It guarantees accurate and secure data storage on a microSD card, which can transmit information via Bluetooth through a discreetly placed button only accessible by physical therapists. Equipped with Velcro strips, the Adherence Sensor case enhances practicality by securely attaching to the Permobil Explorer Mini with minimal force. External features of the case include an on/off switch and an LED indicator with color-coded signals: green for active recording, red for low battery (below 20%), blue for Bluetooth activation, and orange for idle mode (Explorer Mini hasn't moved for over 30 seconds). Seamless integration with the Explorer Mini's design and reproducibility are essential, backed by thorough testing to ensure all functional requirements are met.

Source of Criteria	**Requirements**	**Completed**
Cost	Build an adherence sensor that does not exceed $200.00.	Yes
Security	Encryption and decryption to access child's data.	Yes
Simplicity of Usability	Have the on/off switch and port easy to access.	Yes
	LED display adherence sensor's status.	Yes
	Button to activate Bluetooth.	Yes
	Easy detachment from chair.	Yes
	Appropriate file format for end-user.	Yes
Performance/Reliability	Accurately gather and store the child's movement data.	Yes

Table 1: Design Criteria

Design Considerations

1) Public Health, Safety, and Welfare

The Adherence Sensor adheres to U.S. Food and Drug Administration (FDA) regulations, refraining from altering the Explorer Mini's internal circuitry to ensure compliance with safety standards. The team also complies with general medical research standards by encrypting the device's data. All data on the Adherence Sensor is stored on a microSD card, which does not contain sensitive medical or location information. However, this data will be used in a medical research context which may require reasonable efforts to protect research participant data. Additionally, the Adherence Sensor's circuitry is encased in a sturdy plastic electrical project box, mitigating the risk of electrical hazards and ensuring safety, particularly for the child and caregivers interacting with the device.

2) Global, Cultural, and Societal

As previously mentioned, over 1.3 billion people worldwide are born with disabilities [2]. The team wanted to take steps toward mobility equity for children with disabilities. Creating a user-friendly Adherence Sensor, can help make powered mobility devices more accessible by fully understanding the environment they are used in.

Not everyone has access to strong and stable WiFi and GPS networks. When a physical therapist wants to collect data while observing a child at home or in a non-clinical setting, the device uses Bluetooth data transfer, so physical therapists can receive the data anywhere without risking removing and reinserting the device.

3) Environmental

Environmental consideration played a significant role in shaping and limiting the team's design choices throughout the design process. The production of circuit parts involves using raw materials like metals, silicon, and rare earth elements. However, the extraction of these materials can have severe environmental consequences, including habitat destruction, pollution, and soil degradation, which negatively affect ecosystems and communities. Moreover, the fabrication and manufacturing processes of these parts require substantial energy consumption and involve the use of various toxic chemicals. This energy consumption contributes to greenhouse gas emissions and exacerbates climate change. Additionally, the solvents, acids, and gases utilized in creating the hardware pose significant risks to workers and the environment. Furthermore, the disposal of chips and circuit parts contributes to the generation of electronic waste. Improper disposal practices can release toxic materials into the environment, posing threats to both human health and wildlife. Bearing these environmental impacts in mind, we decided to prioritize the use of the smallest and least wasteful electrical parts and chips possible in our design.

4) Economic

The components and resources required for crafting the sensor are readily available and economical. The electrical parts are compact and priced at just a few dollars each, while the programming can be accomplished using the Arduino Software (IDE) on a readily available device. Throughout the development process, the Adherence Sensor team diligently strived to adhere to the approved budget of $443.73, ensuring that the final product remains affordable. The total cost of the Adherence Sensor has been successfully kept under $200, with the current prototype costing $135. The commitment to cost-effectiveness ensures that the sensor remains accessible to many users while maintaining high-quality performance.

Final Design Overview

Final Hardware Design

Figure 4 demonstrates how the hardware components work together. The Sparkfun ESP 32 IoT RedBoard Microcontroller is the computer and is powered by a 3.7V Lithium Ion rechargeable battery. It gets data from the Adafruit LIS3DH Accelerometer, a Bluetooth button, and the SparkFun RV8803 Real-Time Clock. The ESP 32 Microcontroller shares its status using a colorful LED, and any important information is saved securely on a microSD card and sent through Bluetooth when activated.

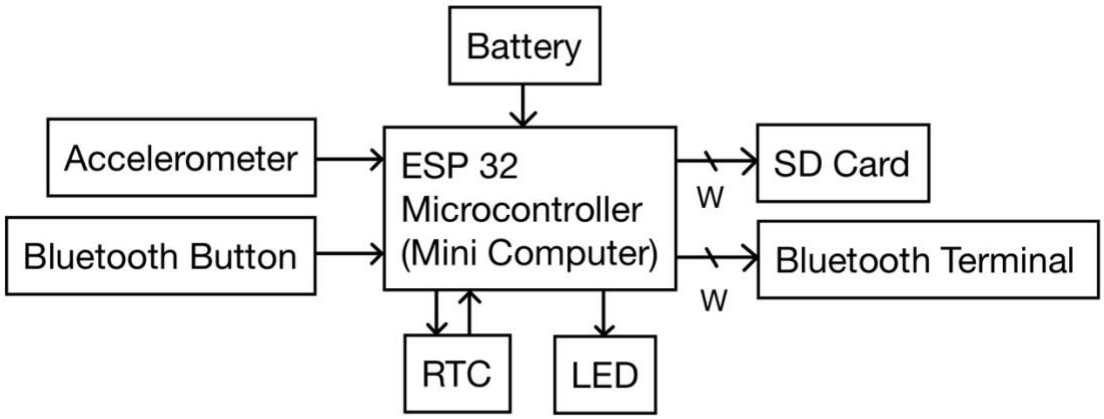

Figure 4: Hardware/Circuitry Final Design

Final User Interaction Design

Figure 5 demonstrates the various interactions the caregiver and physical therapist will have with the device. The caregiver's main interactions with the device involve turning it on or off and charging it. They turn it on using an external switch, and the LED indicates the different statuses: green for active recording, red for low battery (below 20%), blue for Bluetooth activation, and orange for idle mode (Explorer Mini hasn't moved for over 30 seconds). To charge the device, the caregiver uses a Lithium-Ion battery charger. They connect the charger to a USB-A to a Mini-B power cord, then plug the power cord into a USB-A wall charger. After that, they connect the 3.7V Lithium Ion Battery to the charger. Once plugged into a powered wall outlet, the Lithium-Ion battery charges fully, providing enough power to collect data for four weeks.

The physical therapist interacts with the collected motion data after the child has used the Explorer Mini. The data is stored on an SD card, which can be inserted into a computer for transferring to the client manually or through Bluetooth.

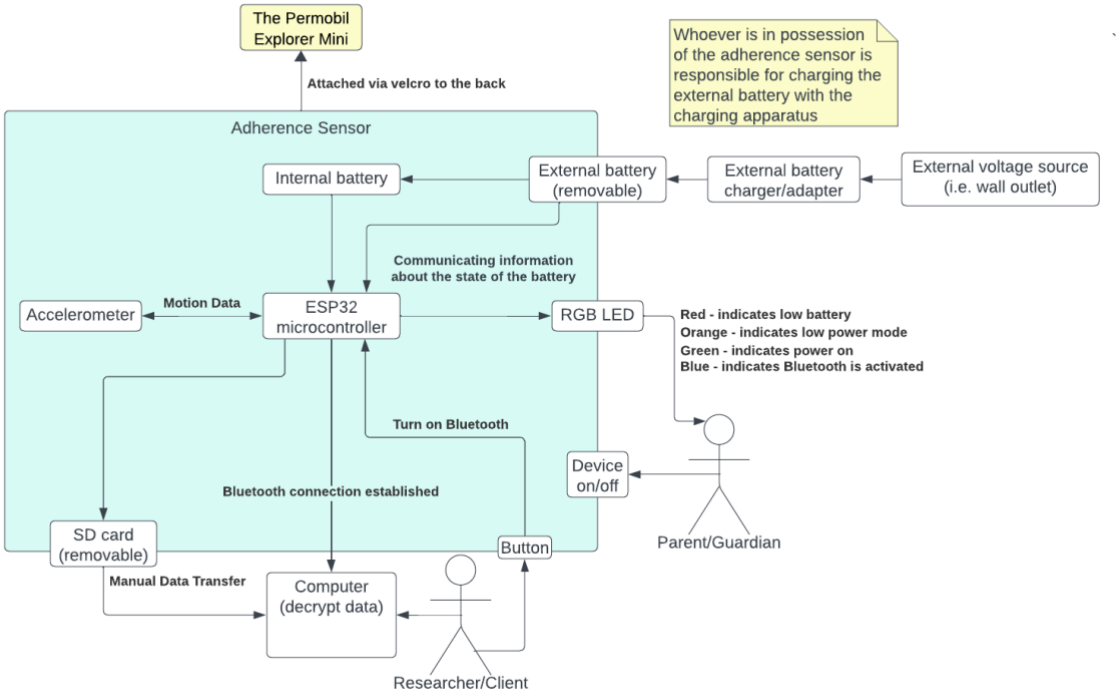

Figure 5: User interaction diagram

Software Final Design

As shown in Figure 17, the Adherence Sensor, programmed in Arduino using C++, featured two essential functions: "setup" and "loop," which can be seen in figure 16. Before the setup function, a variety of variables were declared, encompassing PINs on the ESP32 and electronic components such as the Real-Time Clock Variable (RV8803), Accelerometer Variable (Adafruit_LIS3DH), and Battery Variable (SFE_MAX1704X). Additional variables included readings for the X, Y, and Z axes, acceleration calculations, the number of detections, a Boolean variable for motion detection, and the threshold of change to be considered movement in each axis.

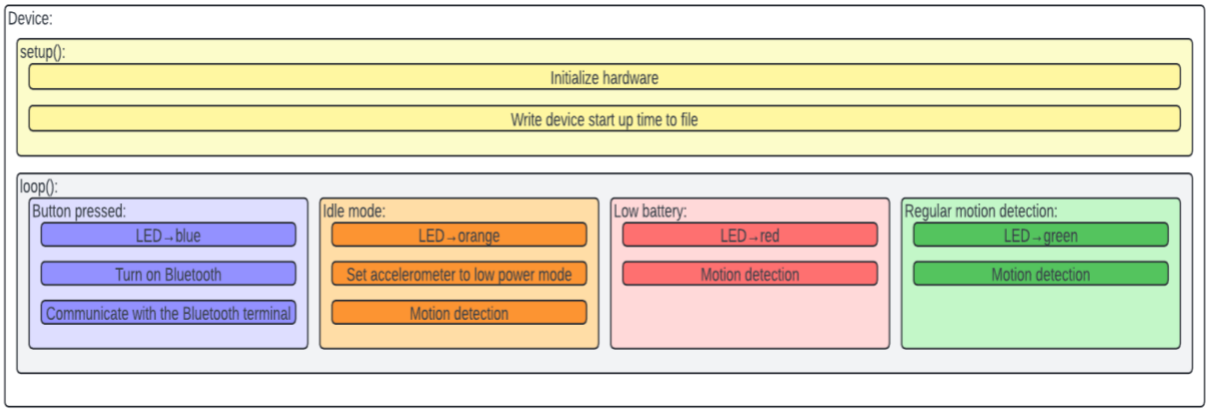

Figure 17: Software Final Design – Device

The setup function served several purposes, including calibrating output pins and electrical components, setting the Inter-Integrated Circuit Protocol to 115200 baud, and calibrating the accelerationdata.txt file while noting the device start-up time. On the other hand, the loop function executed different tasks. It checked if the Bluetooth file transfer button was pressed, initiating the file transfer protocol, and changing the LED color to blue. After a successful file transfer, the loop resumed its regular operations.

Furthermore, the loop function monitored the device's idleness, transitioning the accelerometer into low-power mode and changing the LED color to orange if the device remained inactive for over 30 seconds. The motion detection function was then activated. The loop also checked if the ESP32 Microcontroller's battery level dropped below 20%, signaling the LED to turn red, followed by the execution of the motion detection function. In cases where none of these conditions applied, the program ran the motion detection function while setting the LED color to green.

As shown in Figure 18, the decryption program is an executable called “decrypt-acceleration.exe.” This program uses a specific decryption protocol to decrypt the “data.txt” file line by line. It then prints the output of each line to a new file in the same file called “decrypted.txt.”

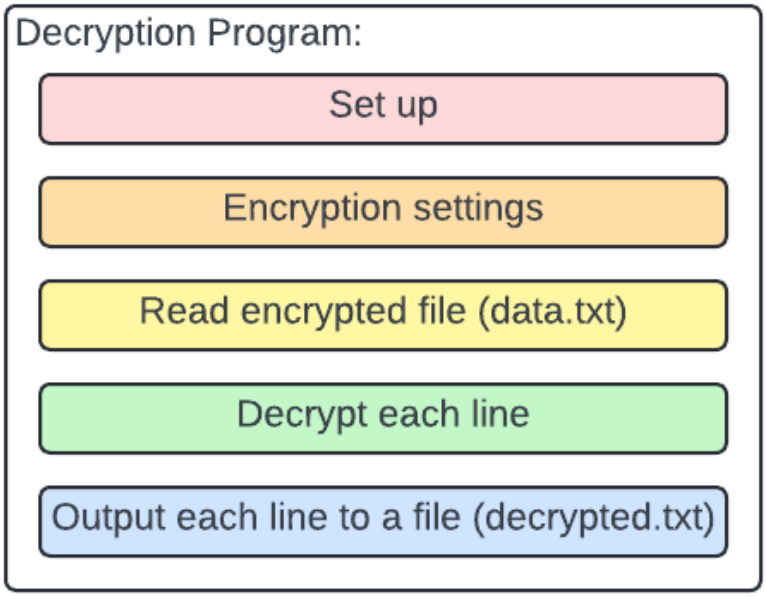

Figure 18: Software Final Design – Decryption

Bluetooth Terminal Functionality

The Bluetooth feature simplifies the data transfer process. Once the client presses the button on the device, they can connect their laptop to the device via Bluetooth and receive the encrypted data. The device must then be connected to the Bluetooth Serial Port app using the "Open Link" button. Once successful, a user can enter the word "help" to display a list of useful commands, as shown in Figure 6.

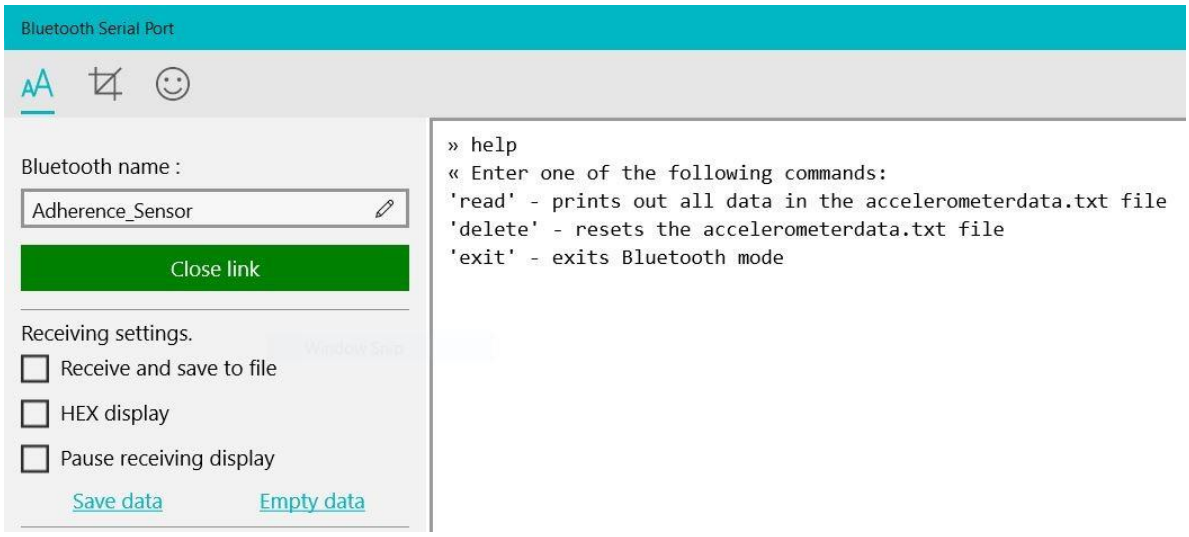

Figure 6: "help" Command

When the “read” command is entered, the data from the Adherence Sensor will appear in the terminal encrypted, as shown in Figure 7.

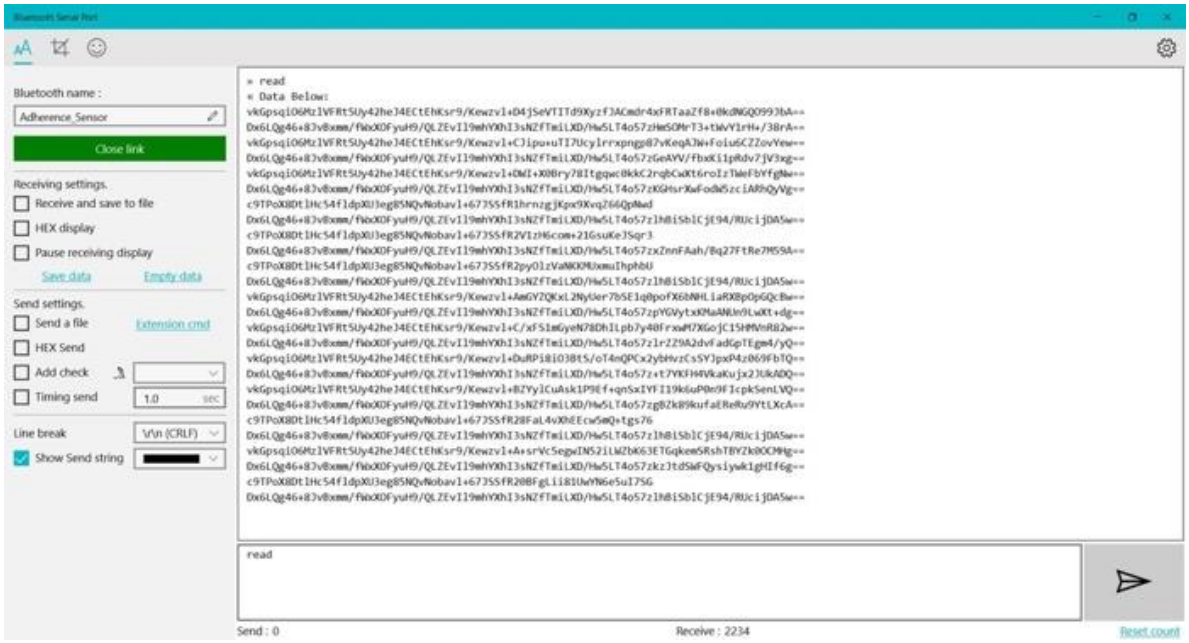

Figure 7: “read” Command

When the “delete” command is entered, the device clears the encrypted data from the file and provides a response to the user, as shown in Figure 8.

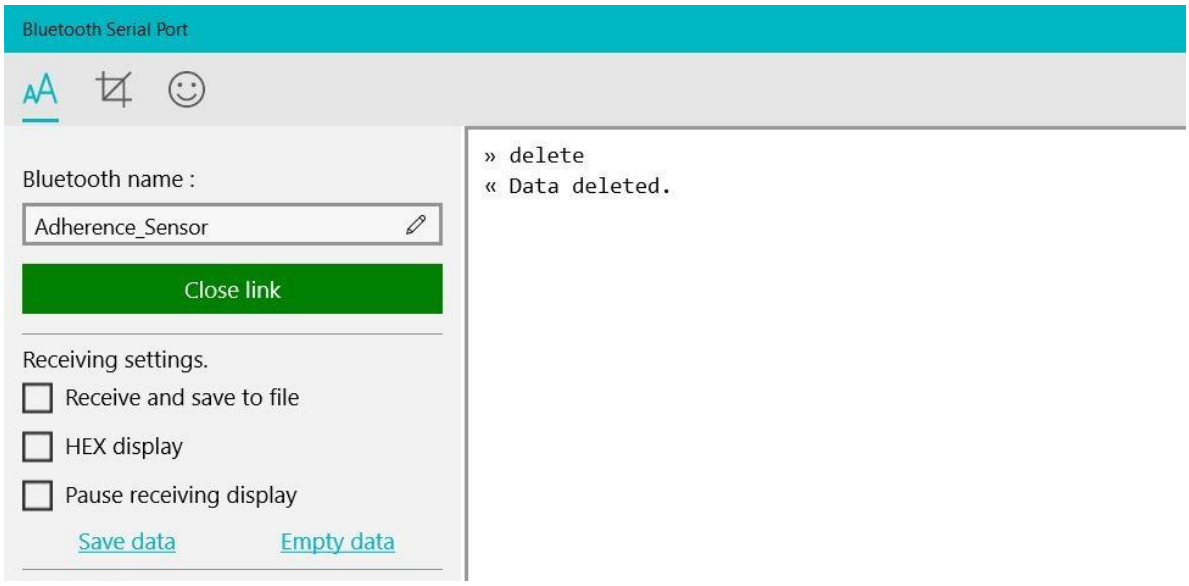

Figure 8: “delete” command

When the user is finished using the device, the “exit” command will deactivate Bluetooth mode, preparing the device to resume data collection, as shown in Figure 9.

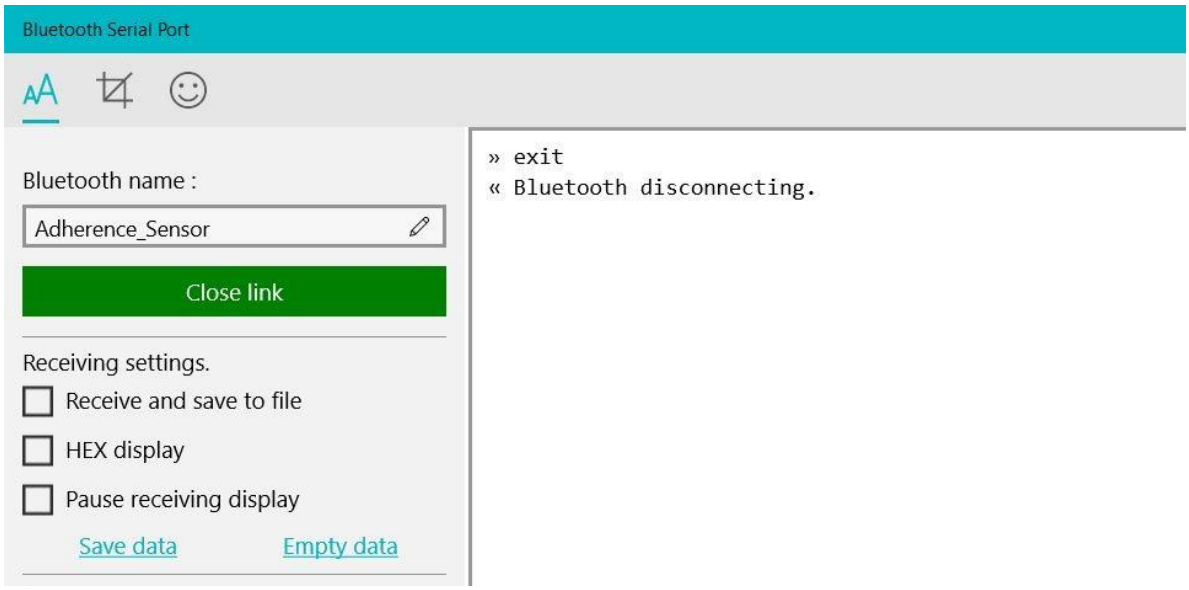

Figure 9: “exit” command

After the user has sent the “read” command and saved the data into a file named “data.txt,” as shown in Figure 9, they will move that file into the folder where the decrypt-acceleration.exe file is located, as shown in Figure 10.

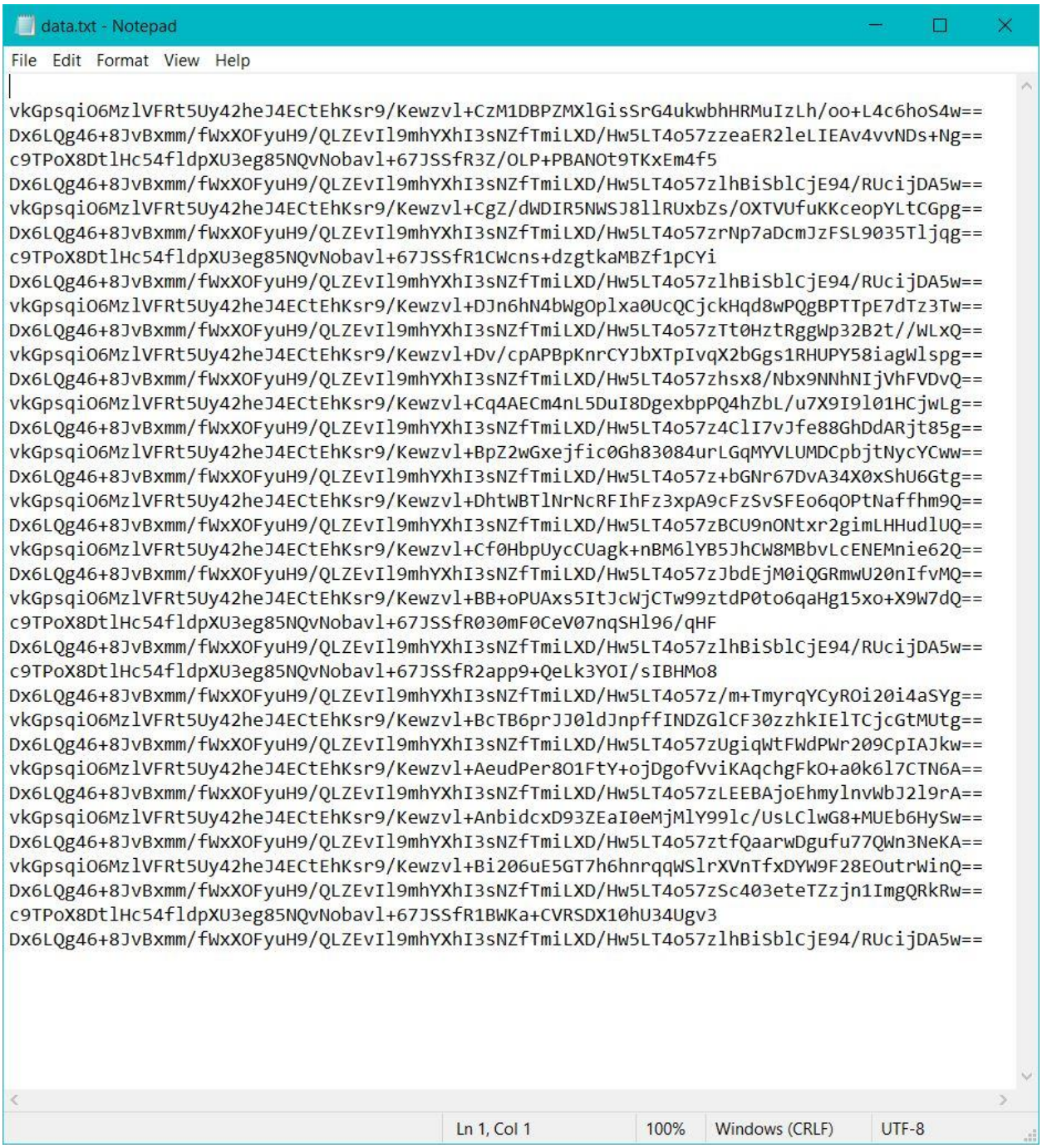

Figure 10: Encrypted file data.txt

Decryption Program Functionality

As seen in Figure 11, double-clicking on “decrypt-acceleration.exe” will create a new file titled “decryption.txt.” This decrypts the contents of “data.txt” in the specific file format the client requested, as seen in Figure 12.

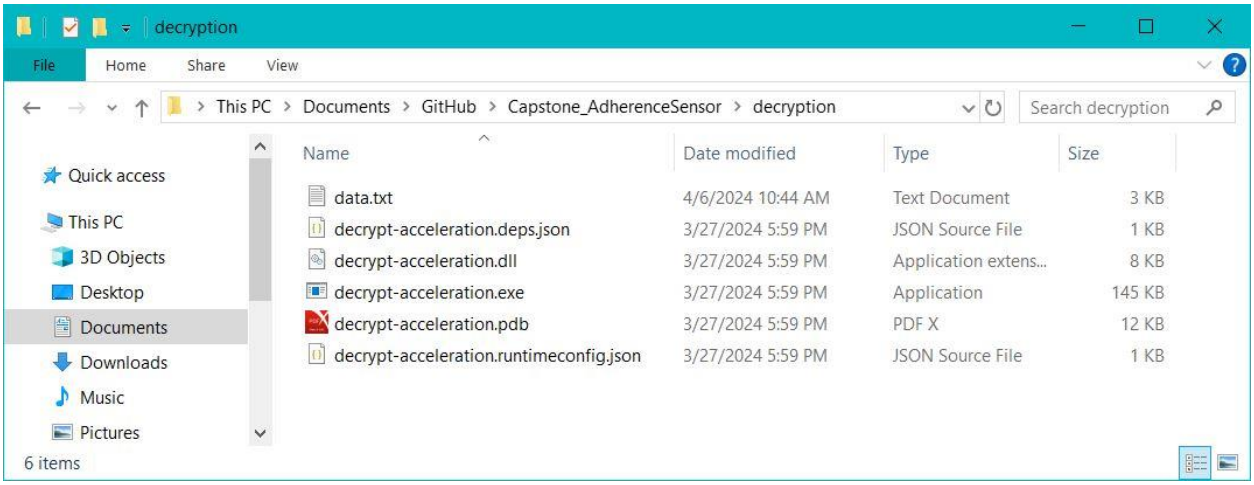

Figure 11: The decryption folder containing “decrypt-acceleration.exe”

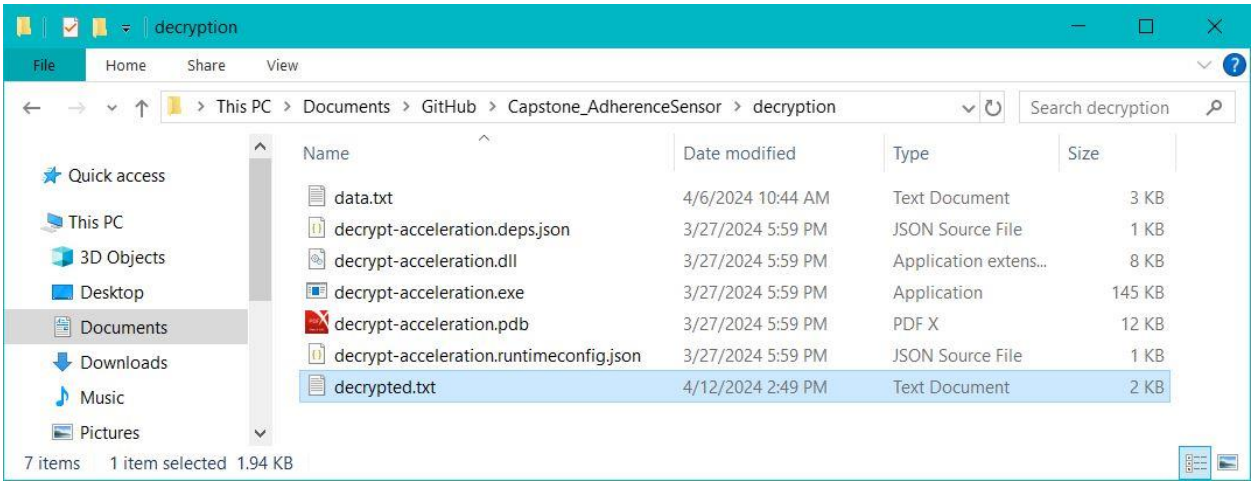

Figure 12: The decryption folder containing the decrypted “decrypted.txt” file

Figure 13 shows a decrypted text file containing the time stamps of when the device has turned on, when the device has moved, and when the device has stopped moving. The device does not collect data about the direction of motion, but that motion has occurred.

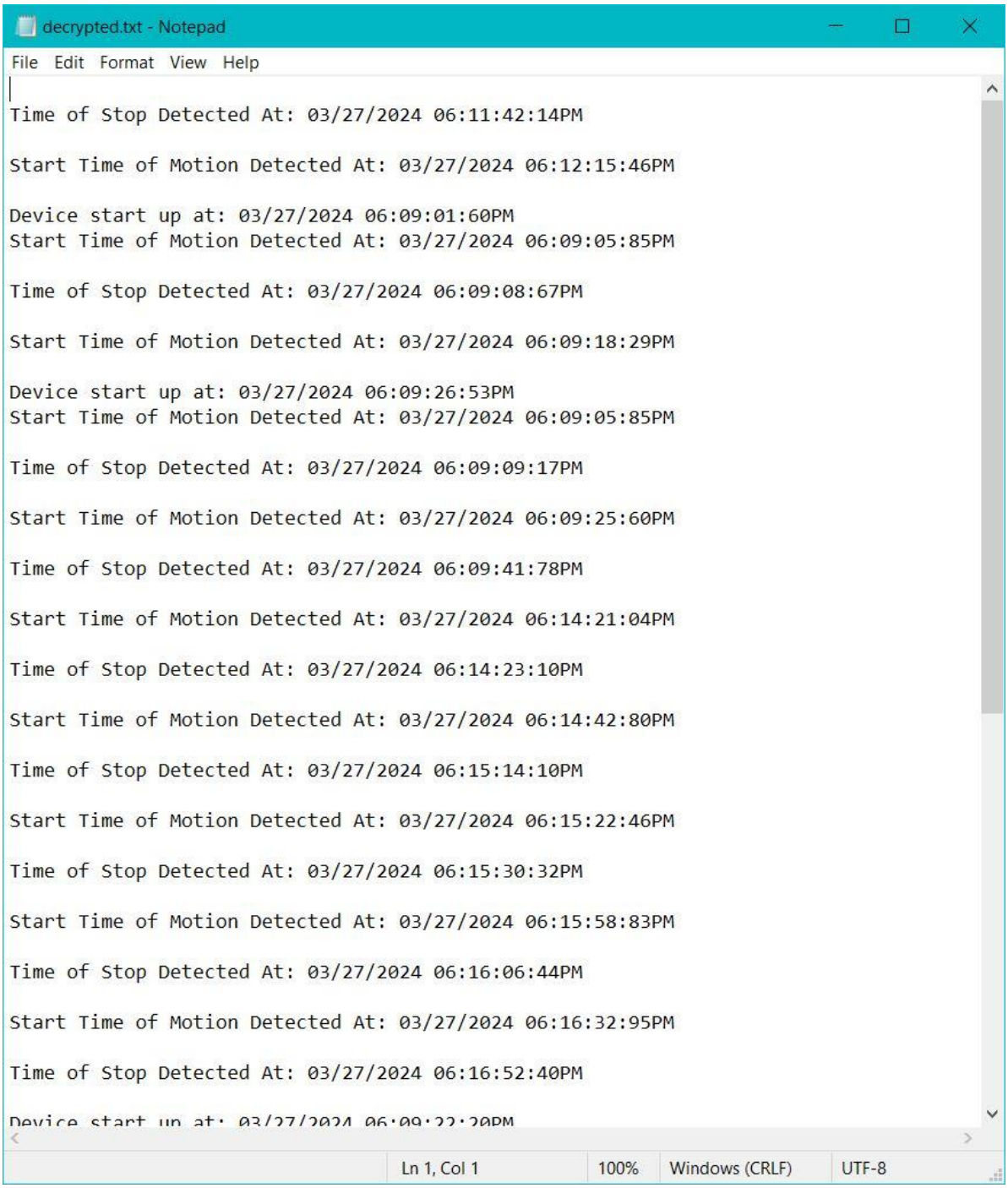

Figure 13: Decrypted file decrypted.txt

Comparison to Previous Prototype

Figure 14 is a pair of side-by-side pictures of the outside of the two versions of the product. The version on the left is the most recent version of the device, and the version on the right is the previous version. From the outside of the device, the user can interact with the on/off switch and observe the RGB LED for information about the device's internal components.

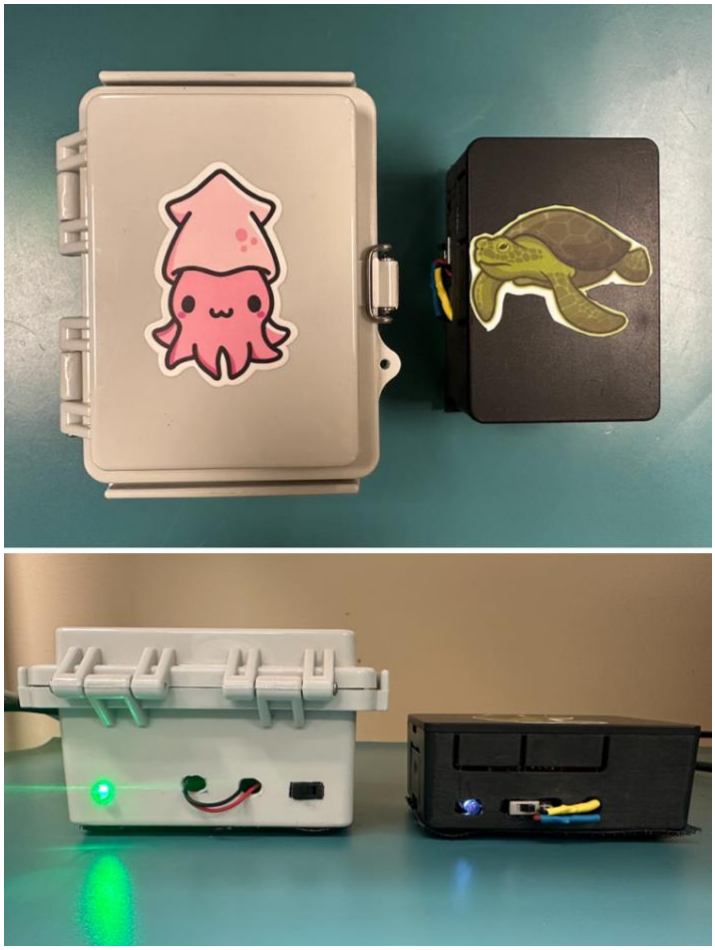

Figure 14: Adherence Sensors side by side (left: current sensor; right: previous team's sensor)

Figures 15 and 16 show the internal circuitry of the current and previous versions of the Adherence Sensor. Both versions feature the SparkFun ESP32 IoT RedBoard, SparkFun RV8803 Real-Time Clock, Adafruit LIS3DH Accelerometer, and a 3.7V Lithium Ion Battery. The current version uses an RGB LED with a Bluetooth transfer button, whereas the previous version has a single-color LED.

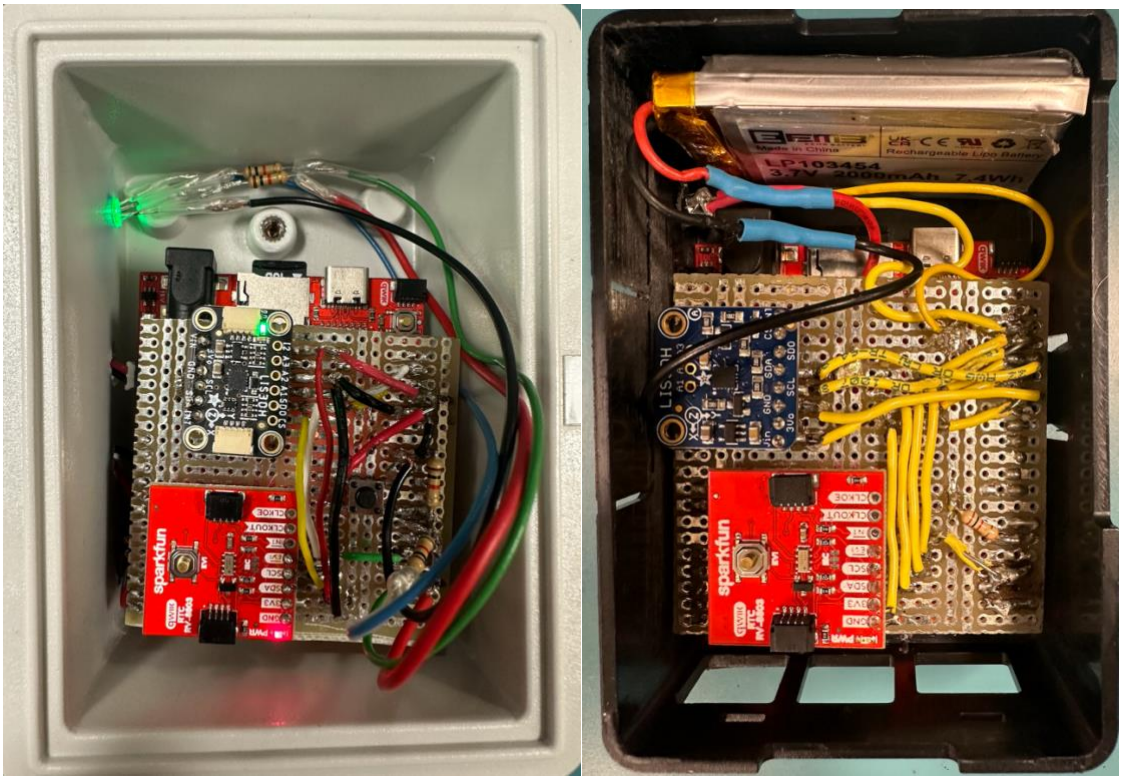

Figure 15: (Left) Current Adherence Sensor Design

Figure 16: (Right) Previous Adherence Sensor Design

Two replicas or prototypes of the previous adherence sensor were produced as a precautionary measure to enhance efficiency. If one replica malfunctioned, it was set aside for any reason, and the other replica assumed the role of the primary sensor. Unfortunately, such an occurrence did transpire, leading to the disassembly of the defective replica. However, salvageable non-faulty parts were repurposed for further testing. The Arduino-based project's troubleshooting and error-checking processes were straightforward and continuous.

Testing Process

Motion Detection Testing & Writing to File

Table 2 below shows the testing procedures for various features implemented in the previous prototype before implementing encryption. The table explains the procedures for testing, recording the device start-up time to the file, the starting motion time to the file, and the ending motion time to the file. These testing procedures required monitoring a computer clock and comparing the times the device recorded and the computer clock time for accuracy. These tests aimed to ensure that the device recorded valuable and accurate data before obscuring the data with encryption.

Feature	Testing Procedure
Recording Start-Up Time to File	Before the team implemented encryption, the team monitored a computer clock and matched device start-up times with times recorded on computer clock for accuracy.
Recording Starting Motion Time to File	Before the team implemented encryption, the team monitored a computer clock and matched start-of-motion times with times recorded using the computer clock for accuracy.
Recording Ending Motion Time to File	Before the team implemented encryption, the team monitored a computer clock and matched device end-of-motion times with times recorded using the computer clock for accuracy.

Table 2: Motion Detection & Writing to a File Testing Procedures

Device Interface & Interaction Testing

Table 3 below describes the testing procedures used to assess the device's interface and interaction functionalities. These tests include Bluetooth mode, idle mode, low battery notification, and the device's normal data collection state.

Feature	Testing Procedure
Bluetooth Mode	After clicking the button, check for a blue LED display and the ability to establish a connection with a compatible Windows laptop. After entering 'exit' in the Bluetooth terminal, the LED would turn from displaying blue to displaying green and continue normal data collection.
Idle Mode	Monitored a computer clock to verify that after 30 seconds of not moving the device, that the RGB LED would turn orange, indicating the device has entered idle mode and the accelerometer has entered power-saving mode. Moving the device while it's in idle mode will cause the device to leave idle mode and present a green LED, indicating that the device is now collecting data normally.
Low Battery	The lithium-ion battery takes several weeks to run out of battery. To test that the LED would display red if the battery hit a low enough threshold, the team set the threshold to a higher value and verified that the device would display the red LED at any given threshold. Note: the threshold is 20% battery remaining. After charging the battery above the threshold, the LED would instead display green or orange, depending if the device can qualify for idle mode or not
Return to Normal Data Collection	The team verified that the device would return to the normal data collection state and the LED would display a green light given that the device no longer has a status that would indicate one of the above conditions.

Table 3: Device Interface & Interaction Testing Procedures

Bluetooth Interface Testing

Table 4 below outlines the testing procedures used to evaluate the device's Bluetooth interface functionalities. These tests cover various features such as connection establishment, command execution, and data manipulation through Bluetooth communication. They serve to validate the reliability of the Bluetooth commands through the interface.

Feature	Testing Procedure
Connection to device after button press	Confirmed that the device is properly connected to the Bluetooth terminal by sending the ‘help’ command to the device.
‘help’ Command	Sending the ‘help’ command to the device after sequences of various other commands presents the text printout expected from the ‘help’ command.
‘read’ Command	The team tested the read command with various amounts of data in the file. Sending the ‘read’ command to the device with data on the device will print the encrypted data from the file to the terminal. Sending the ‘read’ command to the device when the file has no data will present the printout stating that there is no data in the file.
‘delete’ Command	The team tested the ‘delete’ command with various amounts of data in the file. Sending the ‘delete’ command to the device with data on the device will indicate that the file has been deleted. On sending the ‘read’ command would indicate that the file was deleted and that an empty ‘accelerationdata.txt’ file was created and ready for new data collection.
‘exit’ Command	The team tested the ‘exit’ command by running various sequences of the other commands then entering the ‘exit’ command. The team would then confirm that after each sequence that the device would turn off Bluetooth connectivity and resume normal data collection.

Table 4: Bluetooth Interface Testing Procedures

Encryption & Decryption Testing

Table 5 outlines the testing procedures to evaluate the device's encryption and the executable's encryption and decryption functionalities. These tests ensure the accuracy and reliability of the encryption and decryption in their respective programs.

Feature	Testing Procedure
Correct encryption on ESP 32 Microcontroller	Before the team developed a decryption executable, the team would take the data and enter it into an online decryption algorithm for the specific encryption standard the team implemented. The team verified that the encrypted values were correct.
Decryption executable produces correct data	The team would given the decryption executable encrypted files of various lengths and various types of data. The team verified that the decryption algorithm produced the correct values.
Decryption executable produces unique file names	If a file called “decrypted.txt” already exists in the executable’s folder, the executable will create “decrypted2.txt” and count upward. The team tested this by running the same file through the executable and confirmed that the executable would create files with an incremented number appended.

Table 5: Encryption & Decryption Testing

Decisions Regarding Components

The LIS3DH Triple-Axis Accelerometer, manufactured by Adafruit, is a widely used accelerometer supporting both I2C and SPI interfaces. It offers a range of data rate options from 1Hz to 5kHz and features interrupt outputs. With a low current draw of 2uA, it includes functionalities such as freefall detection, tap detection, and tilt orientation detection. Compatible with various microcontroller platforms like Arduino, its price is $4.95 [8].

On the other hand, the SparkFun Triple Axis Accelerometer Breakout—KX132 (Qwiic) is a straightforward, power-efficient accelerometer produced by Kionix. It boasts a 16-bit resolution across three axes, a maximum output data rate of 10kHz, and dimensions comparable to a quarter. This product relies on Arduino IDE and is available for purchase at $14.95 from the SparkFun website.

While both the LIS3DH in the Adherence Sensor and the KX132 share similarities in terms of dimensions and features such as freefall detection and tap detection, the LIS3DH's lower price point keeps the device affordable while still providing essential functionalities.

Limitations

The previous design decisions heavily influenced the current version of the device. The previous team used an accelerometer to detect the start and stop times of motion and attempted to extract directional motion data. However, accelerometers cannot calculate how far the device moves in a certain direction. Accelerometers can only detect whether they are in motion or not. If future teams are interested in increasing the accuracy and usefulness of the data, they should consider including a proximity sensor or a GPS module to assist the accelerometer in collecting directional motion data. However, a version of the device using GPS capabilities would require stable GPS or cellular data that may not be available everywhere, specifically in rural areas.

Similarly, the team was limited to using Bluetooth communication protocols for electronic data file transfer because not every household has a stable Wi-Fi network or cellular service. To implement Bluetooth file transfer with the ESP32 microcontroller, the team used a third-party Bluetooth terminal to establish a connection with the device. The team discovered that whole-file transfer over Bluetooth is not possible, so the team worked within the constraints of the Bluetooth terminal to allow the user to save the information to a file. However, any other information printed on the terminal screen will also be saved to the file. The user must go into the file and delete any extra information that may have been printed to the terminal before the printout from the ‘read’ command. Any solution with whole-file transfer would require a stable Wi-Fi or cellular signal, which may not always be available.

Process Outcomes

Throughout development, the team developed various valuable skills, encompassing technical proficiencies crucial for engineering roles and project management skills. Diligent note-taking was imperative for effective communication of progress with the client, streamlined preparation for the final report, and provided comprehensive documentation to benefit future project teams. The team's client wanted to minimize the number of face-to-face meetings, entrusting the team with substantial responsibility for the project's manifestation.

Securing access to the previous year's work presented challenges, as the client had actively dispatched the device for research with a family. Resourcefulness was paramount throughout periods when neither the previous device nor the ordered device parts were available, compelling the team to utilize available resources to advance the project. Moreover, discrepancies and gaps in the documentation required for constructing the prototype complicated the assembly of circuitry essential for the functionality of the previous code in data collection on the device. Consequently, the team had to derive the circuit design through visual inspection of the previous year's device, as opposed to relying on documentation.

Despite these challenges, the team has accomplished all their development goals and feels they have earned a sense of pride in their achievement. The team has navigated obstacles, independently troubleshot unforeseen challenges, and collaborated effectively.

Conclusions & Recommendations

Based on the agreed-upon and project development milestones, the Go Baby Go: Adherence Sensor team met the development requirements for the 2023-2024 academic year.

The Adherence Sensor contributes to the growing technological field of developmental data analytics. The device will help pediatric physical therapists like Dr. Bethany Sloane analyze how toddlers interact with mobility devices like the Permobil Explorer Mini when they're not there. However, the Adherence Sensor can also be used with other mobility device used by older children and adults, like power and manual wheelchairs.

The user interface and interaction changes make the Adherence Sensor easier for all users to understand. The low battery and idle mode LED indicators let caregivers know when the device is close to dying so they can charge the device before it gives out. Similarly, idle mode lets caregivers know when the child hasn't interacted with the device so they can either encourage the child to move the Explorer Mini or know when the child is no longer engaged with the Explorer Mini. The green LED lets the parent know that the device is functioning and the child is interacting with the Explorer Mini regularly. The Bluetooth terminal helps the physical therapist receive data from the device without needing to access the internal circuitry of the Explorer Mini. These user interaction changes make the Adherence Sensor easier for caregivers and physical therapists to work with.

Encryption and decryption broaden the device's applications, including research data collection and analytics. The embedded encryption and convenience decryption executable preserve the convenience of the Adherence Sensor while also opening more doors for research and discovery in pediatric physical therapy.

The current team recommends that future teams explore new sensors that can measure directional distance, such as ultrasonic distance and infrared proximity sensors [28][29]. The team should also explore possibilities for implementing cellular and GPS data for distance tracking. The current team also suggests developing a custom Bluetooth terminal and interface to decrypt and customize data storage for improved ease of use. A previous University of Portland Go Baby Go capstone team developed a Go Baby Go car for a child to use. A future Adherence Sensor team could work synchronously with a Go Baby Go car development team to embed analytics into a car for more accurate direction motion data collection.

Acknowledgments

We are grateful to Bethany Sloane and Go Baby Go Oregon for trusting us to improve on the Adherence Sensor. We appreciate the faculty and Shiley School of Engineering for their support and invaluable resources. Lisa Bassett, Dr. Aziz Inan, Dr. Andrew Nuxoll, and Dr. Tammy VanDeGrift have been incredible resources, mentors, and instructors, and their guidance and patience are appreciated immensely. We would like to thank the previous Go Baby Go capstone team, comprising Ashlee Mei Balignasay, Kyler Mento, and Tara Peterson, whose development and collaboration facilitated our design process.

Glossary

Explorer Mini: The Permobil Explorer Mini provided by the Oregon Health & Science University team.

Device: the hardware designed to track and record movement.

ESP32 Microcontroller: a small, specialized electronic component that computes and manages simple devices, making them work by following specific instructions.

References

[1] Charlton, Haley. "Plummer, Permobil's Explorer Mini Recognized in TIME Magazine's Top 100 Best Inventions of 2021." Belmont Unviersity News & Media. Belmont University, April 8, 2022. https://news.belmont.edu/plummer-permobils-explorer-mini-recognized-in-time-magazines-top-100-best-inventions-of-2021/#:~:text=The%20Explorer%20Mini%20was%20listed,that%20positively%20impact%20the%20world.

[2] "Disability." World Health Organization. WHO, March 7, 2023. https://www.who.int/news-room/fact-sheets/detail/disability-and-health#:~:text=Key%20facts,earlier%20than%20those%20without%20disabilities.

[3] "Go Baby Go - Design Document (Ver. 1.0)." Microsoft Word. Microsoft, April 25, 2023. https://upedu-my.sharepoint.com/:w:/r/personal/petersot24_up_edu/_layouts/15/Doc.aspx?sourcedoc=%7BCEB0018C-C5F9-4FEC-A064-951111659539%7D&file=Go%20Baby%20Go%20-%20Design%20Document%20(ver.%201.0).docx&action=default&mobileredirect=true.

[4] "Go Baby Go." Go Baby Go Oregon. Go Baby Go, April 25, 2023. https://gobabygooregon.org/home.

[5] "Modified Ride-on Toy Car User's Manual." Google Docs. Google, Accessed September 28, 2023. https://docs.google.com/document/d/1YicK4_H787xEOGDYQX43jDBuXEcOHK2fsZFHY6uRAy8/edit.

[6] Nikitina, Dina. "Trexo Home Puts Kids on Their Feet and Helps Them Walk." Abilities. Trexo Robotics, Accessed September 25, 2023. https://www.abilities.com/community/trexo.html.

[7] "Pediatric Adaptive Equipment." Adaptive Specialties. Adaptive Specialties. https://www.adaptivespecialties.com/pediatric-adaptive-equipment.aspx.

[8] "Adafruit LIS3DH Triple-Axis Accelerometer (+2g/4g/8g/16g)." Adafruit Industries, Unique & Fun DIY Electronics and Kits. Adafruit, https://www.adafruit.com/product/2809.

[9] "SparkFun Triple Axis Accelerometer Breakout - KX134 (Qwiic)." SparkFun Electronics. SparkFun, Accessed October 5, 2023. https://www.sparkfun.com/products/17589.

[10] "SparkFun Real Time Clock Module - RV-8803 (Qwiic)." SparkFun Electronics. SparkFun, Accessed October 5, 2023. https://www.sparkfun.com/products/16281.

[11] "Lithium Ion Polymer Battery - 3.7v 2500mAh." Adafruit Industries, Unique & Fun DIY Electronics and Kits. Adafruit, Accessed October 5, 2023. https://www.adafruit.com/product/328.

[12] “Amazon.com: Tenergy 2 Pack, Fire Retardant Lipo Bags, Battery Bags for Charging and Storage, 5.5x3.5x2 inches Each, material tested to meet UL94 Standard.” Amazon, Amazon.com. https://a.co/d/41Whqd9.

[13] “USB Lilon/LiPoly charger – v1.2.” Adafruit Industries, Unique & Fun DIY Electronics and Kits. Adafruit, Accessed October 5, 2023. https://www.adafruit.com/product/259.

[14] “Amazon.com: Amazon Basics USB-A to Mini USB 2.0 Fast Charging Cable, 480Mbps Transfer Speed with Gold-Plated Plugs, 3 Foot, Black.” Amazon, Amazon.com. Accessed October 5, 2023. https://a.co/d/aCNqv4V.

[15] “microSD Card – 1GB (Class 4).” SparkFun Electronics. SparkFun, Accessed October 5, 2023. http://sfe.io/p15107.

[16] “Amazon.com: YETLEBOX Waterproof Electrical Box with Mounting Plate 200x150x100mm, IP67 Junction Box Dustproof Clear Cover Plastic DIY Electric Project Enclosure Box Grey 7.9”x5.9”x3.9”.” Amazon, Amazon.com. Accessed October 5, 2023. https://a.co/d/4wDrJ3R.

[17] “SparkFun IoT RedBoard – ES32 Development Board.” SparkFun Electronics. SparkFun, Accessed October 5, 2023. http://sfe.io/p19177.

[18] “LED – RGB Addressable, PTH, 5mm Diffused (5 Pack).” SparkFun Electronics. SparkFun, Accessed October 5, 2023. https://www.sparkfun.com/products/12986.

[19] “LED – Assorted (20 pack).” SparkFun Electronics. SparkFun, Accessed October 5 2023. https://www.sparkfun.com/products/12062.

[20] “Amazon.com: TUOFENG 24 awg Wire Solid Core hookup Wires-6 Different Colored Jumper Wire 30ft or 9m Each, 24 Gauge Tinned Copper Wire PVC (OD: 1.46mm) Hook up Wire Kit.” Amazon, Amazon.com. Accessed October 5, 2023. https://a.co/d/eGDl9JQ.

[21] “Qwiic Cable – Grove Adapter (100mm).” SparkFun Electronics. SparkFun. Accessed October 5, 2023. http://sfe.io/p15109.

[22] “Amazon.com: USB Plug, USB Wall Charger 3 Pack, GiGreen Dual Port Electrical Cube 5V 2.1A Charging Block USB Outlet Plugs Compatible iPhone 11 XS X 8 7, LG V30 G8, Samsung S20 S10+ S9 S8 Note 9 8, Moto G6.” Amazon, Amazon.com. Accessed October 5, 2023. https://a.co/d/2o6K0pf.

[23] "Amazon.com: Gikfun Solder-able Breadoard Gold Plated Finish Proto Board PCB DIY Kit for Arduino (Pack of 5PCS) GK1007." Amazon, Amazon.com. Accessed October 5, 2023. https://a.co/d/hW9Bqlz.

[24] "Amazon.com: E-Projects 10EP5121K00 1k Ohm Resistors, ½ W, 5% (Pack of 10)." Amazon, Amazon.com. Accessed October 5, 2023. https://a.co/d/8MjtlmN.

[25] "Trexo Home Pricing." Trexo Robotics, October 5, 2023. https://www.trexorobotics.com/trexo-home-pricing/.

[26] Explorer Mini. Accessed October 6, 2023. https://www.permobil.com/en-us/products/power-wheelchairs/permobil-explorer-mini.

[27] "IOT Redboard ESP32 Development Board Hookup Guide." IoT RedBoard ESP32 Development Board Hookup Guide - SparkFun Learn. Accessed October 6, 2023. https://learn.sparkfun.com/tutorials/iot-redboard-esp32-development-board-hookup-guide/all.

[28] SparkFun. 2017. "Ultrasonic Distance Sensor - HC-SR04 - SEN-15569 - SparkFun Electronics." Sparkfun.com. 2017. https://www.sparkfun.com/products/15569.

[29] Adafruit. n.d. "IR Distance Sensor Includes Cable (10cm-80cm)." www.adafruit.com. https://www.adafruit.com/product/164.

[30] Medicaleshop, "Permobil Explorer Mini Power Mobility Device ," *Medicaleshop*. https://www.medicaleshop.com/permobil-explorer-mini.

[31] TS02-66-100-BK-160-LDR-D CUI Devices - DigiKey. Accessed April 19, 2024. https://www.digikey.in/en/products/detail/cui-devices/TS02-66-100-BK-160-LCR-D/15634325.

[32] SLW-1276864-4A-D CUI Devices – DigiKey. Accessed April 19, 2024. https://www.digikey.com/en/products/detail/cui-devices/SLW-1276864-4A-D/21259972?utm_adgroup=&utm_source=google&utm_medium=cpc&utm_campaign=PMax%20Shopping_Product_Low%20ROAS%20Categories&utm_term=&utm_content=&utm_id=go_cmp-20243063506_adg-_ad-__dev-c_ext-_prd-21259972_sig-CjwKCAjw5v2wBhBrEiwAXDDoJT8dQrq9Gsg00p51xXpCEVUHULj6fdptYoj0v5vlf3bPNxaXZi7YAxoCV2IQAvD_BwE&gad_source=1&gclid=CjwKCAjw5v2wBhBrEiwAXDDoJT8dQrq9Gsg00p51xXpCEVUHULj6fdptYoj0v5vlf3bPNxaXZi7YAxoCV2IQAvD_BwE.

[33] "Nisuoien 15 Sets Strips with Adhesive Heavy Duty - Strong Double Sided Tape with Hook and Loop - 1 x 4 Inch: Amazon.Com: Industrial & Scientific." NISUOIEN 15 Sets Strips with Adhesive Heavy Duty - Strong Double Sided Tape with Hook and Loop - 1 x 4 Inch: Amazon.com: Industrial & Scientific. Accessed April 19, 2024. https://a.co/d/645u9FY.

[34] 61301611121 Headers, Male Pins – DigiKey. Accessed April 19, 2024. https://www.digikey.com/en/products/detail/w%C3%BCrth-elektronik/61301611121/4846854?utm_adgroup=&utm_source=google&utm_medium=cpc&utm_campaign=PMax%20Supplier_W%C3%BCrth%20Elektronik&utm_term=&utm_content=&utm_id=go_cmp-19901677914_adg-_ad-__dev-c_ext-_prd-4846854_sig-CjwKCAjw5v2wBhBrEiwAXDDoJYodzm2xohPtwGktXcx32vi1BeqABGPL8hrR9cpjyXuNKM2E1F4jKxoCu0YQAvD_BwE&gad_source=1&gclid=CjwKCAjw5v2wBhBrEiwAXDDoJYodzm2xohPtwGktXcx32vi1BeqABGPL8hrR9cpjyXuNKM2E1F4jKxoCu0YQAvD_BwE

Appendix A: Roles and Responsibilities

Collective:

- Rotated roles as team lead.
- Completed all documentation and presentations.
- Managed troubleshooting and testing.

Margaret Brown:

- Encryption and decryption programming
- Bluetooth file transfer programming and protocol

Kaylee Mock:

- Manage and research hardware components.
- Solder adherence sensor circuitry.
- Construct adherence sensor case.
- UP alumni liaison.

Anna Yrjanson:

- Budget/Purchasing manager.
- Industry advisor liaison
- LED programming
- Battery-level programming
- Accelerometer idle-mode programming

Appendix B: Budget

Project Materials/Supplies (Individual Items Under $500)

Adafruit LIS3DH Triple-Axis Accelerometer (x2) [8]	$9.90
SparkFun Triple Axis Accelerometer Breakout – KX134 (x2) [9]	$43.90
SparkFun Real Time Clock Module – RV-8803 (Qwiic) (x2) [10]	$35.00
Lithium Ion Polymer Battery – 3.7v 2500mAh (x2) [11]	$29.90
Tenergy Fire Retardant LiPo Bag (Set of 2) [12]	$11.99
USB Lilon/LiPoly Charger – v1.2 (x2) [13]	$25.00
Amazon Basics USB-A to Mini USB 2.0 – 3ft (x2) [14]	$12.64
microSD Card – 1GB (x2) [15]	$11.00
YETLEBOX Waterproof Electrical Box with Mounting Plate (x2) [16]	$35.98
SparkFun IoT RedBoard – ESP32 Development Board (x2) [17]	$59.90
LED – RGB Addressable, PTH, 5mm Diffused (5 Pack) (x2) [18]	$7.00
LED – Assorted (20 Pack) (x2) [19]	$3.95
TUOFENG 24 awg Wire Sold Core Hookup [20]	$14.99
Qwiic Cable – Grove Adapter (x10) [21]	$16.00
USB Wall Charger, GiGreen Dual Port Electrical Cube 5V (Set of 3) [22]	$13.99
Gikfun Solder-able Breadboard Gold Plated Finish Proto Board (Set of 5) [23]	$11.98
E-Projects 10EP5121K00 1k Ohm Resistors (Set of 10) (x2) [24]	$10.82
Subtotal:	$352.94

Miscellaneous

Shipping Total Cost	$15.79
Cost Overage	$75.00
Subtotal:	$90.79

Estimated Total: $443.73

Budget Justification

The previous capstone team had an estimated budget of $470.32 and underestimated their costs by around $100. Our team will purchase enough items to create two prototypes: one prototype that will remain at the University and another that can be tested by researchers at OHSU. Each of these prototypes can be produced for around $150 each. The University of Portland will be handling the costs of any necessary parts.

Appendix C: Arduino Program

```
/*
Go Baby Go Adherence Sensor Master Code
Written by Ashlee B, Kyler M, Tara P 4/26/2023
Adapted by Margo B, Kaylee M, Anna Y 4/26/2024
*/

// Bluetooth Libraries
#include <BTAddress.h>          // Part of BluetoothSerial
#include <BTAdvertisedDevice.h>  // Part of BluetoothSerial
#include <BTScan.h>             // Part of BluetoothSerial
#include <BluetoothSerial.h>     // Part of BluetoothSerial

// General Libraries
#include <SPI.h>                                    // Communication with Serial Peripheral Interface (SPI) devices (ex. microcontroller and other circuits)
#include <Wire.h>                                   // I2C communication between chips
#include <SparkFun_RV8803.h>                        // Read and set time on the RTC
#include <Adafruit_Sensor.h>                        // Required library for all Adafruit Unified Sensor libraries
#include <Adafruit_LIS3DH.h>                        // Configures and communicates data from Adafruit LIS3DH accelerometer (uses Unifed Sensor Library)
#include <SparkFun_MAX1704x_Fuel_Gauge_Arduino_Library.h>  // Used to monitor external battery level
#include <AES.h>                                    // 128-bit CBC for encryption
#include <AESLib.h>                                 // AES implementation (128-bit CBC) for low-memory conditions
#include <CTR.h>              // Part of the crypto libraries
#include <Crypto.h>           // Rhys Weatherly's crypto library
#include "arduino_base64.hpp"  // Turns AES encryption into base64
#include "FS.h"               // ESP 32 File System library
#include <SD.h>               // Potential Problem: the duplicate libraries
#include "SPI.h"              // Part of the SPI library

//Pins
#define EVI 14     // Timestamp pin
#define R_LED 4    // Red LED pin
#define G_LED 13   // Green LED pin - changed from 16 to 13
#define B_LED 17   // Blue LED pin
#define BUTTON 25  // Button input pin

// Bluetooth enabling definition
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run 'make menuconfig' to and enable it
#endif

// Bluetooth serial enabling definition
#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif

// Electronic component variables
RV8803 rtc;                          // Real Time Clock variable - uses the class RV8803
Adafruit_LIS3DH lis = Adafruit_LIS3DH();  // LIS3DH Accelerometer variable
SFE_MAX1704X battery;                // External battery object, used to initiate low battery mode
double battery_percent;              // Holds the current battery percentage
int low_battery_threshold = 20;      // Threshold for low battery indication
```

These numbers identify which pin numbers on the ESP 32 Microcontroller the hardware components are connected to.

```
// Bluetooth Variable
BluetoothSerial SerialBT;

float accel;
int detect = 0;                // Number of no-motion detections
bool det = false;              // Indicates a new motion was detected
float numReadings = 50.0;      // Used to average the accelerometer readings
const float gravity = 9.80665; // Earth's gravity in m/s^2

// Hold the previous averaged readings for each axis on the accelerometer
float last_x;
float last_y;
float last_z;

// Hold the current averaged readings for each axis on the accelerometer
float x;
float y;
float z;

// Read state of the button press
int buttonState = 0;

// Idle Mode: variables that measure the time since the last detected motion
unsigned long stop_motion;                      // Time when motion stopped
unsigned long current_time;                     // Current time
lis3dh_dataRate_t current_rate = lis.getDataRate(); // Gets the current accelerometer data collection mode (for low power mode)

// AES object used for encryption
AESLib aesLib;

// The accelerationdata.txt file
File file;
```

These values are involved in determining if the child moved the cart or not

```
void setup() {
  pinMode(R_LED, OUTPUT);
  pinMode(G_LED, OUTPUT);
  pinMode(B_LED, OUTPUT);
  pinMode(BUTTON, INPUT);
  pinMode(EVI, OUTPUT);  // Connecting to the Real-Time Clock
  Wire.begin();          // I2C addresses begin
  Serial.begin(115200);  // Set baud rate

  //RTC initialization
  if (rtc.begin() == false)  // If the RTC cannot be found or started
  {
    Serial.println(F("Device not found. Please check wiring. Freezing."));  // Prints to the serial monitor that the RTC cannot be found
    while (1);  // NEED INFORMATION: why are they using an infinite loop?
  }
  Serial.println(F("RTC online!"));  // Prints to the serial monitor that the RTC was found

  if (rtc.setToCompilerTime() == false) {
    Serial.println(F("An error has occurred in setting the RTC to compiler time"));
  }

  rtc.setEVIEventCapture(RV8803_ENABLE);  //Enables the Timestamping function

  Serial.println(F("LIS3DH test!"));
  if (!lis.begin(0x18)) {               // If the LIS3DH accelerometer cannot be found or used, follow this condition
    Serial.println(F("Couldnt start"));  // Prints to the serial monitor that the accelerometer cannot be found
    while (1) yield();                   // NEED INFORMATION: why are they using an infinite loop AGAIN? What is yield?
  }
  Serial.println(F("LIS3DH found!"));  // The LIS3DH accelerometer is found

  // Set up lis and init readings
  lis.setRange(LIS3DH_RANGE_4_G);

  delay(10);   // Delays by 10 milliseconds
  lis.read();  // Takes a reading from the accelerometer

  //orientated on the back of the permobil
  x = -lis.z;  // Takes the x-axis reading, using a negated z-axis accelerometer reading
  y = lis.y;   // Takes the y-axis reading
  z = -lis.x;  // Takes the z-axis reading, using a negated x-axis accelerometer reading

  // Check if SD is connected correctly
  if (!SD.begin()) {                        // If the SD card cannot be found, follow this condition
    Serial.println(F("Card Mount Failed"));  // Prints to the serial monitor that the SD card cannot be found
    return;                                  // Returns without further action
  }
  uint8_t cardType = SD.cardType();  // Saves the type of SD card to an unsigned 8-bit integer

  if (cardType == CARD_NONE) {                // If there is no SD card, follow this condition
    Serial.println(F("No SD card attached"));  // Prints that there is no SD card attached
    return;                                    // Returns without further action
  }
```

These identify the specific hardware pins to send information to or receive information from

```
  Serial.print("SD Card Type: ");      // Prints this statement to the serial monitor
  if (cardType == CARD_MMC) {          // If the card is a MultiMediaCard, follow this condition
    Serial.println(F("MMC"));          // Prints that the card is a MultiMediaCard (MMC) to the serial monitor
  } else if (cardType == CARD_SD) {    // If the card is a standard SD card, follow this condition
    Serial.println(F("SDSC"));         // Prints that the card is a standard SD card to the serial monitor
  } else if (cardType == CARD_SDHC) {  // If the card is a high capacity SD card, follow this condition
    Serial.println(F("SDHC"));         // Prints that the card is a high capacity SD card to the serial monitor
  } else {                             // If the card is of another type, follow this condition
    Serial.println(F("UNKNOWN"));      // Prints that the card is of an unknown type of the serial monitor
  }

  // Initialize the MAX17043 battery object
  if (battery.begin() == false) {
    Serial.println(F("Can't find MAX1704X Battery"));
  }

  // For low battery mode
  battery.quickStart();                        // Restarts the MAX1704X to createa. more accurate guess for the SOC
  battery.setThreshold(low_battery_threshold);  // An interrupte to alert when the battery reahes 20% and below

  //axis zeroing to eliminate false positives
  x = 0;
  y = 0;
  z = 0;

  // Procedure at device start-up
  String startTime = RTC();  //timestamp
  Serial.println();
  file = SD.open("/accelerationdata.txt", FILE_APPEND);  //open file.txt to write data
  if (!file) {                                           // If the file cannot be opened, follow this condition
    Serial.println(F("Could not open file(writing)."));  // Prints that the accelerometer.txt file cannot be opened
  }

  else {                               // If the file can be opened, follow this condition
    file.println();                    // Adds a new line to the file
    file.print("Device start up at: ");  // Writes this to the bottom of the file
    file.print(startTime);             // Prints the start-up time to the bottom of the file
    file.println();                    // Prints another new line to the bottom of the file
    file.close();                      // Closes the file
  }
}
```

```
void loop() {
  // Records whether the button has been pressed
  buttonState = digitalRead(BUTTON);
  current_time = millis();

  // Gets the battery percent
  battery_percent = battery.getSOC();

  /*
   * Condition 1: Bluetooth File Transfer
   * The user presses the button to connect the sensor to a Bluetooth compatible device to perform various functions:
   * - Read data off accelerationdata.txt
   * - Delete data off accelerationdata.txt
  */

  // Check for button press
  if (buttonState == 1) {

    // Bluetooth Transfer Set-Up
    SerialBT.begin("Adherence_Sensor");                       // Sets the name of the device
    Serial.println(F("The device can connect to Bluetooth"));
    setColor(0, 0, 255);                                      // Set LED color to blue

    while (1) {
      if (SerialBT.available()) {                       // If there is data coming in (from Bluetooth)
        String command = SerialBT.readStringUntil('\n');  // Read string until new line
        command.trim();                                 // Remove leading and trailing whitespace

        // Help Command: prints command information
        if (command == "help") {
          SerialBT.println("Enter one of the following commands:");
          SerialBT.println("'read' - prints out all data in the accelerometerdata.txt file");
          SerialBT.println("'delete' - resets the accelerometerdata.txt file");
          SerialBT.println("'exit' - exits Bluetooth mode");
        }

        // Read Command: prints all text data on the accelerationdata.txt file
        else if (command == "read") {
          file = SD.open("/accelerationdata.txt", FILE_READ);  // Opens the file to print to the Bluetooth terminal
          if (file) {                                          // Checks if accelerationdata.txt is readable
            if (file.peek() < 0) {                             // Checks if accelerationdata.txt is empty
              SerialBT.println("File is currently empty.");     // Prints warning that the file is empty
            } else {                                           // If accelerationdata.txt is not empty
              SerialBT.println("Data Below: ");  // Prints all data from the file to the serial monitor
              while (file.available()) {         // Prints the data from the file line-by-line
                SerialBT.write(file.read());     // Writes the line to the Bluetooth terminal
              }
            }
            file.close();  // Closes the accelerometerdata.txt file
          }
          else {  // If the file cannot be opened
            Serial.println(F("ERROR: Cannot open file"));
            SerialBT.println("ERROR: Cannot open /accelerationdata.txt; File Size: " + String(SD.open("/accelerationdata.txt").size()));
          }
        }
```

This ‘if’ block has the instructions for the commands used in the bluetooth terminal

```
      // Delete Command: deletes existing data on the accelerationdata.txt file
      else if (command == "delete") {
        SD.remove("/accelerationdata.txt");        // Deletes the accelerationdata.txt file from the SD card
        if (SD.exists("/accelerationdata.txt")) {  // Checks that the file stil exists
          Serial.println(F("Delete functionality did not work"));
        } else {                                                // If the file successfully deletes
          file = SD.open("/accelerationdata.txt", FILE_APPEND); // Resets the accelerationdata.txt
          file.close();                                         // Closes the accelerationdata.txt file
          SerialBT.println("Data deleted.");                    // Prompts the user to reset the Adherence Sensor
        }
      } else if (command == "exit") {                 // Protocol for the exit command
        SerialBT.println("Bluetooth disconnecting."); // Prompts the user to reset the device to normal collection
        delay(5000);
        SerialBT.end();                               // Disconnects the bluetooth from the serial monitor
        break;                                        // Exits to the main loop() function
      }

      // Invalid Command: prompts the user for a valid command
      else {
        SerialBT.println("Invalid command: " + command);
      }
    }
  }
  // If the user (incorrectly) leaves the device on after exiting, the following will execute before exiting this condition
  buttonState = 0;       // Returns the button variable to the "not-pushed" state
  setColor(0, 255, 0);  // Sets the LED color back to green
}

/*
 * Condition 2: Idle Mode Check
 * Puts the accelerometer into low power mode if the cart has been idle for longer than 30 seconds
 * Conducts regular motion detection in low power mode
*/

else if (current_time - stop_motion > 30000) {
  setColor(255, 30, 0);                          // Sets the color to "orange-ish"
  lis.setDataRate(LIS3DH_DATARATE_LOWPOWER_5KHZ); // Sets the accelerometer to low-power mode
  motionDetection();                             // Resumes motion detection
}

/*
 * Condition 3: Low Battery Check
 * Checks if the battery level is below 20% and displays a red LED
 * Conducts regular motion detection in full power mode
*/

// Checks if battery is below 20 percent
else if (battery_percent < 20) {
  setColor(255, 0, 0);  // Sets the LED color to red
  motionDetection();    // Runs the motion detection

  // Print the battery percentage
  Serial.print("Battery Percentage: ");
  Serial.print(battery_percent);
  Serial.println("%");
}
```

This ‘else if’ block is used for idle mode

This ‘else if’ block is used for low battery mode

```
/*
 * Condition 4: Regular Motion Detection
 * Sets the LED color to green and conducts regular motion detection
*/

else {
  setColor(0, 255, 0);                    // Sets the LED color to green
  lis.setDataRate(LIS3DH_DATARATE_10_HZ);  // Sets the accelerometer to normal data collection
  motionDetection();                      // Runs the motion detection
}
```

This 'else' block is used for normal data collection

```
/*
 * void motionDetection
 * Reads accelerometer data and uses a threshold of change to determine if the child is moving
*/

// Accelerometer: Uses a delta scheme to detect movement
void motionDetection() {
  accelRead();        // Takes 50 accelerometer readings and averages them
  Serial.println(x);  // Prints the x value, which is influenced by vertical gravity

  // Threshold of change in each axis to be considered a detected movement
  float move_tholdX = 0.1;
  float move_tholdY = 0.1;
  float move_tholdZ = 0.1;

  // If there is a significant change in all three axes, create a timestampe to record to SD card, and indicate the start of a new motion
  if (abs(last_x - x) > move_tholdX && abs(last_y - y) > move_tholdY && abs(last_z - z) > move_tholdZ && det == false) {
    digitalWrite(EVI, HIGH);    // trigger EVI pin
    delay(20);                  // wait for a second
    digitalWrite(EVI, LOW);     // turn off EVI pin output
    Serial.print("Detected ");  //Debug purposes
    det = true;                 // Indicates a new motion was detected
    stop_motion = millis();
    sd_Start();  // Writes to the file when the motion was detected
  }

  // If there is no longer significant change in all three axes add to the detection counter
  else if (move_tholdX > abs(last_x - x) && move_tholdY > abs(last_y - y) && move_tholdY > abs(last_z - z) && detect < 3 && det == true) {
    detect += 1;
  }

  // If after 300ms there is no more significant change, the time of stopped motion will be printed to the SD card
  else if (move_tholdX > abs(last_x - x) && move_tholdY > abs(last_y - y) && move_tholdY > abs(last_z - z) && detect == 3 && det == true) {
    digitalWrite(EVI, HIGH);   // trigger EVI pin
    delay(20);                 // wait for a second
    digitalWrite(EVI, LOW);    // turn off EVI pin output
    Serial.print("Stopped ");  //Debug purposes
    sd_Stop();                 // Prints the time the motion stopped
    det = false;               // Resets the det boolean to false
    detect = 0;                // Resets the detect counter
    stop_motion = millis();    // Detects the time motion stops
  }

  else {        //if the data goes above the thresholds over reset detect counter
    detect = 0;  // Resets the detect counter
  }

  // Give time between readings
  delay(100);  // 100 milliseconds between readings
}
```

The motionDetection function records when a movement starts and ends

```
/*
 * String RTC
 * Creates a Real-Time Clock timestamp for recording the current time
*/

String RTC() {
  String time;  // The string used to save the timestamp data
  //RTC timestamp
  if (rtc.getInterruptFlag(FLAG_EVI)) {
    rtc.updateTime();  // Grabs an updated time from the compiler
    rtc.clearInterruptFlag(FLAG_EVI);
    String currentTime = rtc.stringDateUSA(); // Get the current date in mm/dd/yyyy format
    String timestamp = rtc.stringTimestamp(); //Get the timestamp

    // Saves the date string and the time string
    time = currentTime + " " + timestamp;

    Serial.print(time);  // Prints the date and time to the serial monitor
    //end debug
  }
  return time;  // Returns the timestamp string
}

/*
 * void accelRead
 * Takes 50 accelerometer readings and averages them
*/

//read from the accelerometer function puts data through moving average filter
void accelRead() {
  sensors_event_t event;  // new sensor event

  // Sums the 50 samples for each axis
  float x_sum = 0;
  float y_sum = 0;
  float z_sum = 0;

  // Moving Average Filter- Take average of 50 samples for each axis
  for (int i = 0; i < numReadings; i++) {
    lis.getEvent(&event);  // Grabs the specific accelerometer data

    x_sum += (-event.acceleration.z);  // Adds this specific x-axis instance to the sum, using the negated z-axis accelerometer data
    y_sum += event.acceleration.y;     // Adds this specific y-axis instance to the sum
    z_sum += (-event.acceleration.x);  // Adds this specific z-axis instance to the sum, using the negated x-axis accelerometer data
    delay(2);                          // Delays for 2 milliseconds
  }

  // Assigns the previous readings to the 'last' versions of the specific axes
  last_x = x;
  last_y = y;
  last_z = z;

  // Divides the readings by 50 (because of the 50 readings), and assigns these values to their respective axis readings
  x = x_sum / numReadings;
  y = y_sum / numReadings;
  z = z_sum / numReadings;
}
```

This RTC function takes a timestamp from the Real-Time Clock

The accelRead function averages accelerometer readings used in determining if the cart is moving

```
/*
 * void sd_Start
 * Prints the timestamp of a started motion
 */

void sd_Start() {  //prints to sd card starting

  String startTime = RTC();  // Grabs the initial timestamp using the RTC helper function
  Serial.println();          // Adds a new line in the serial monitor
  String start_encrypt = "Start Time of Motion Detected At: " + startTime + "\n";  // String for the start time
  encryptAndWrite(start_encrypt);
}

/*
 * void sd_Stop
 * Prints the time of stopped motion to accelerationdata.txt
*/

void sd_Stop() {  //printing to the SD card

  String stopTime = RTC();  //timestamp
  Serial.println();         // Adds a new line

  // The acclerationdata.txt file could be read
  String stop_encrypt = "Time of Stop Detected At: " + stopTime + "\n";
  encryptAndWrite(stop_encrypt); // Appends this to the bottom of the file when motion is stopped
}

/*
 * void encryptAndWrite
 * Encrypts data and writes to the accelerationdata.txt file
*/

void encryptAndWrite(String data) {
  file = SD.open("/accelerationdata.txt", FILE_APPEND);  //open file.txt to write data
  if (!file) {                                           // If the accelerationdata.txt file couldn't be read, follow this condition
    Serial.println("Could not open file(writing).");     // Prints to the serial monitor that the file couldn't be opened
  }

  else {
    data = encrypt(data);          // Encrypts the data
    String data2 = decrypt(data);  // Decrypts the data
    file.println(data);     // Prints encrypted data to the file
    Serial.println(data);   // Prints the encurpted file to the serial terminal
    Serial.println(data2);  // Prints the decrypted data to the serial terminal
    file.close();  // Closes accelerationdata.txt file
  }
}
```

sd_Start and sd_Stop both use the encryptAndWrite function to write the encrypted motion start and stop times to the file

encryptAndWrite uses the encryption standard to print the encrypted data to the file

This report does not include the encryption and decryption functions in the Adherence Sensor program to maximize security.

Appendix D: User Manual

Device Operation Basics

How to Open and Close the Device

As shown in Figure 1, to open the device, unhook the latch by pulling up on the large metal tongue. This will unlatch the “locking” mechanism and allow you to pull the top lid away from the container to reveal the internal components. Note: the lid itself can be tight and may require more force than anticipated to remove.

Figure 1: The device open.

To close the device, hook the latch by using the small metal loop around the plastic groove identified in Figure 2 and push down on the large metal tongue. Similar to opening the device lid, closing the lid also may require more force than anticipated to close. Figure 3 shows the device with the latch mechanism correctly close and the lid fully closed.

Figure 2: The metal hook on the device latched.

Figure 3: The device closed.

Attaching and Detaching the Device

The correct orientation of the device is squid upwards, as shown in Figure 4.

Figure 4: The correct orientation of the adherence sensor.

The back of the device should have two strips of Velcro that aligns with the back of the cart, as shown in Figure 5.

Figure 5: Back of the adherence sensor and chair/cart

With the device in the right orientation and aligned with the velcro on the chair/cart, the device can be pressed on, as shown in Figure 6.

Figure 6: Attaching the adherence sensor.

Turning the Device On and Off

As shown in Figure 7, Flip the switch from one side to another to turn the device on or off. The RGB will turn on or off based on whether the device is powered on or not.

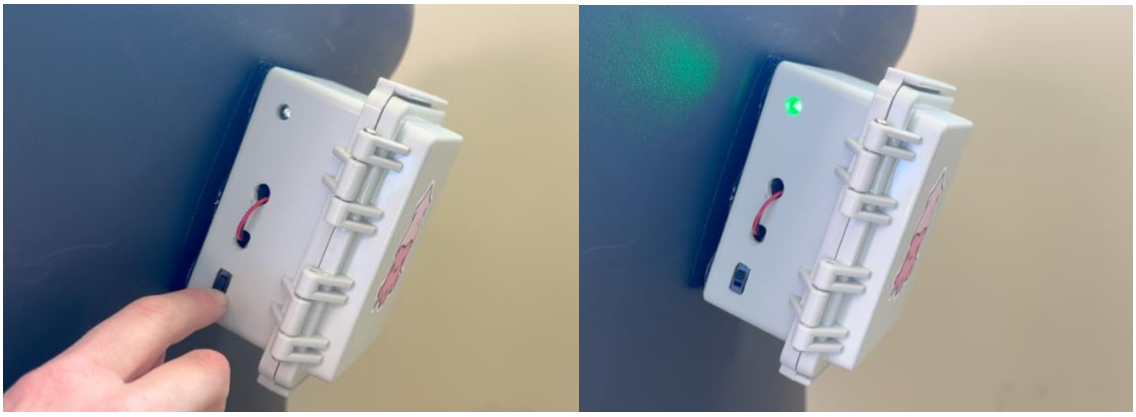

Figure 7: Flipping the switch to turn on the Adherence Sensor

Charging the Device

As shown in Figures 8-10, The charger has three components: a USB-A charger block, a cord with a USB-A and Mini-USB end, and a Mini-USB lithium-ion battery charger.

Figure 8: (left) USB-A charging block

Figure 9: (center) USB-A/Mini-USB cord

Figure 10: (right) Mini-USB lithium-ion charger

As shown in Figure 11, the USB-A end of the cord is inserted into the charging block, and the Mini-USB end of the cord is inserted into the lithium-ion charger at the “DCIN” port.

Figure 11: Fully assembled lithium-ion battery charger

As shown in Figure 12, the cord attached to the lithium-ion battery inserts into the “BATT” end of the lithium-ion charger. Once the cord is fully inserted and the power starts moving into the battery, the “CHRG” LED will turn orange to indicate the battery is charging. Once the battery finishes charging, the “DONE” LED will turn green to indicate the battery is at full charge.

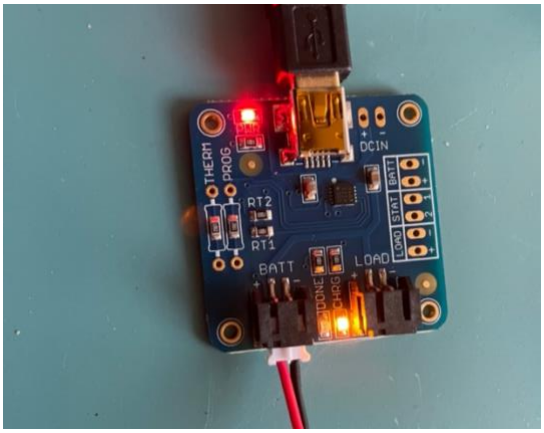

Figure 12: The lithium-ion battery charging, as indicated by the orange “CHRG” light

Lithium-ion batteries naturally degrade in quality and the length of charge they can hold over time. To get the most out of your battery and decrease the risk of damage from battery fires, follow the best-practice suggestions below:

- Charge your battery at standard room temperature: 68-77°F
- Keep the battery between 20-80% charge at all times; do not charge the battery to 100% or let the battery percentage fall under 20%
- If you do not plan to use the battery for over a month, store the battery at 50% charge to preserve battery integrity
- If you notice the battery decreasing in capacity, consider replacing it with a new battery to prevent battery fires
- As shown in Figure 13, charge the battery in a lithium-ion charging bag to prevent a battery fire from causing property damage

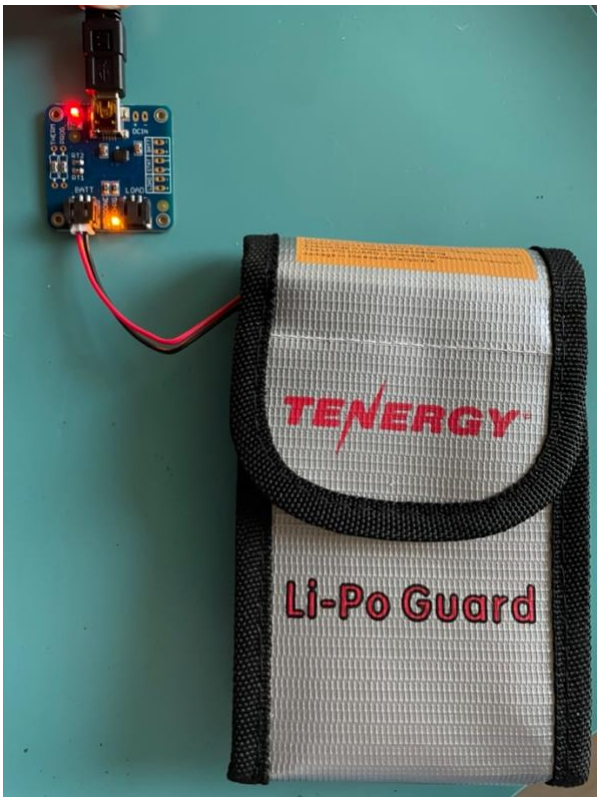

Figure 13: Lithium-ion battery charging in a charging bag

Manual Data File Transfer

As shown in Figure 14, to access the “accelerationdata.txt” file stored on the SD card, remove it from the ESP32 and insert it into an SD card reader connected to a computer. The SD card should automatically appear with its file, particularly the “accelerationdata.txt” file. If it doesn’t appear automatically, you can locate it by opening the File Explorer application, navigating to the “This PC” tab, scrolling to the “Devices and Drives” section, and clicking on the “SD Card (D:)” icon.

When finished using the SD card, it’s crucial to eject it properly to maintain its integrity. This can be done by right-clicking on the “SD Card (D:)” in File Explorer and selecting “Eject.” Alternatively, you can click on the “^” symbol in the bottom right of the taskbar to reveal hidden icons, then click on the icon resembling a flash drive with a checkmark next to it labeled “Safely Remove Hardware and Eject Media.” Choose “- Eject SD Card (D:)” from that menu that appears.

After ejecting, your computer should display a message confirming it’s safe to remove the hardware. At this point, you can take out the SD card from the reader and reinsert it into the ESP32 SD card slot. You’ll know it’s properly reinserted when you feel a clicking sensation.

Figure 14: Safely inserted microSD card into a compatible SD card port

Bluetooth Terminal

Computer Requirements

The app requires Windows 10 to download and run.

How to Download

To open the Microsoft Store, click on the “Start” icon in the bottom left corner of the screen or press the “Start” key between the “fn” and “alt” keys. Next, click on the search bar on the left side of the taskbar, type in “Microsoft Store,” and press the “Enter” key.

To download an app from the Microsoft Store, click on the search bar at the top of the Microsoft Store window and enter “Bluetooth Serial Port,” as shown in Figure 15, then press enter. As shown in Figure 16, choose between the free trial version and the $3.99 permanent version. Click the “Install” button on the screen and wait for the app to finish downloading, depicted in Figure 17. Once downloaded, you can access the app from the Windows “Start” menu using the search bar.

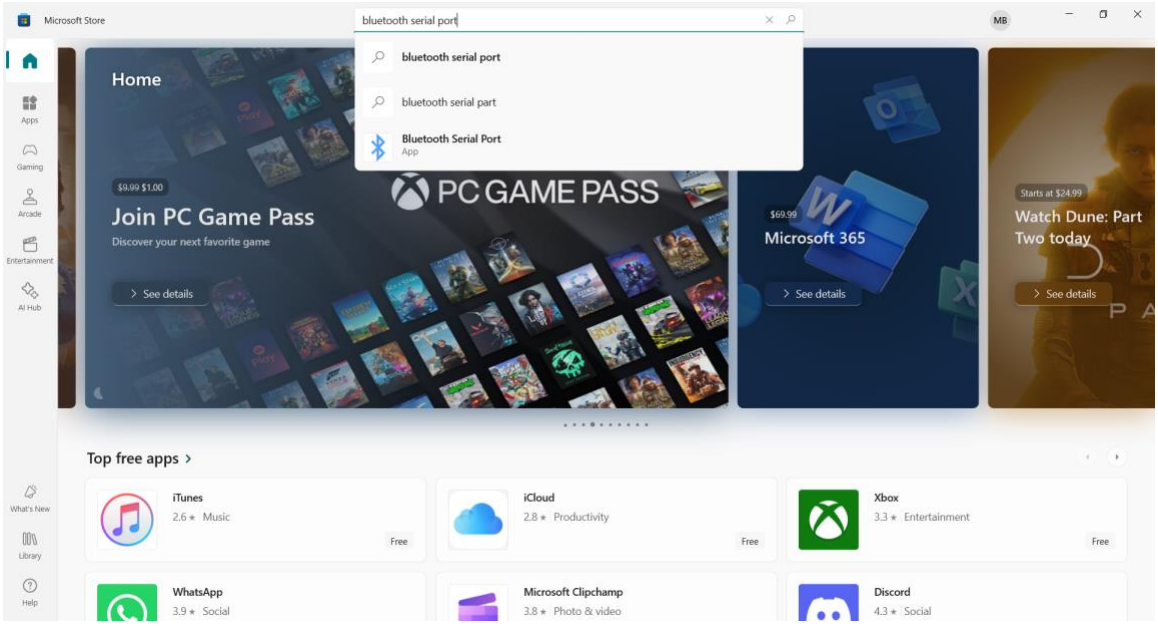

Figure 15: Microsoft Store with “Bluetooth Serial Port” entered into the primary search bar

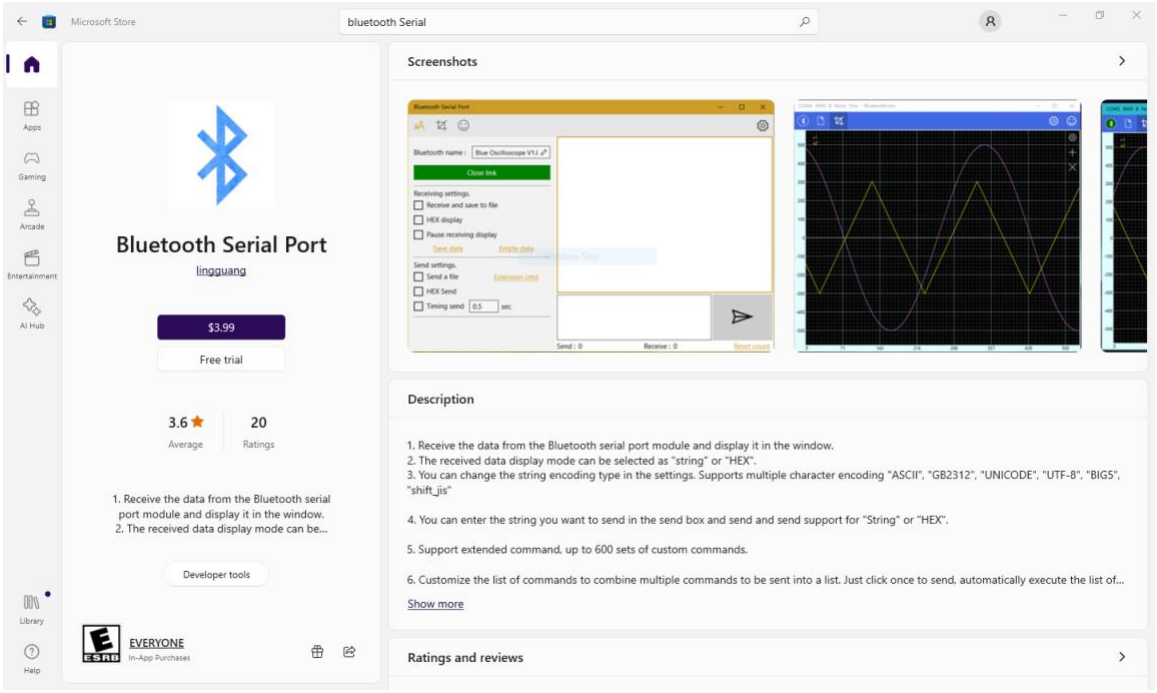

Figure 16: Purchase page for the Bluetooth Serial Port application

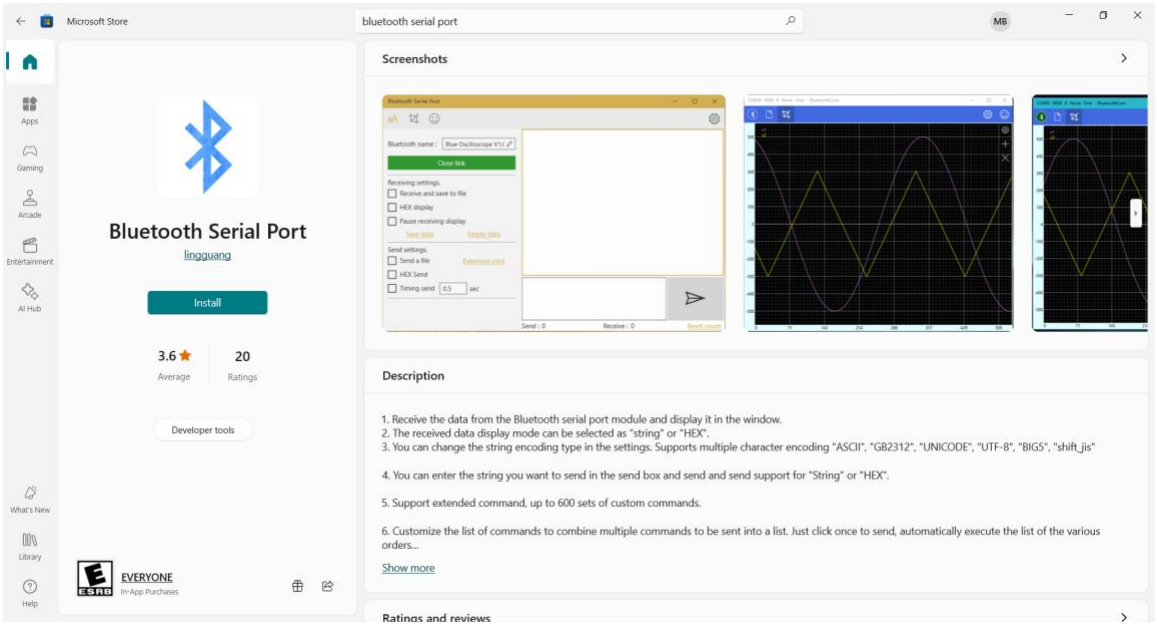

Figure 17: Installation page for the Bluetooth Serial Port application

Pairing the Device with the Computer

To begin, ensure that the device is powered on, then open it using the latch located on the right side and press the black Bluetooth button. Hold the button until the LED turns blue to confirm that Bluetooth is activated, as shown in Figure 18.

Next, navigate to the “Settings” app and access the “Devices” section, followed by the “Blueooth & other devices” subsection, as shown in Figure 19. Toggle the button under “Bluetooth” to turn it on. Then, click the “Add Bluetooth or Other Devices” button.

In the “Add a Device” window, select the “Bluetooth” option to display other Bluetooth-enabled devices, such as mice and keyboards. Locate and click on the device labeled “Adherence_Sensor”.

Once the screen indicates “Your device is ready to go!” you can close the window, indicating that your computer is now connected to the device. If you deactivate Bluetooth on your computer, power off the device, or disconnect it from your computer, you should find the device listed under “Other devices” as “paired”.

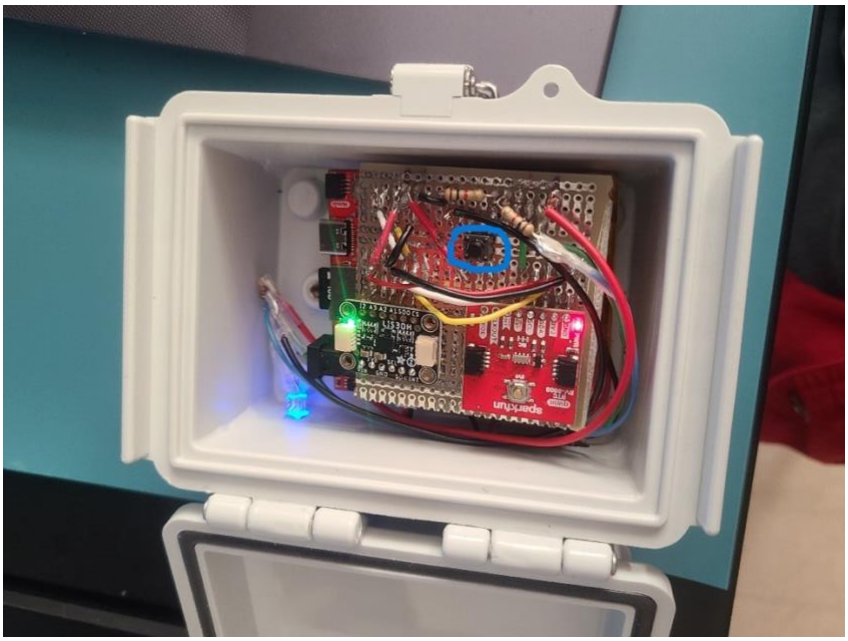

Figure 18: The Bluetooth button (circled in blue) and the blue LED on (bottom left corner)

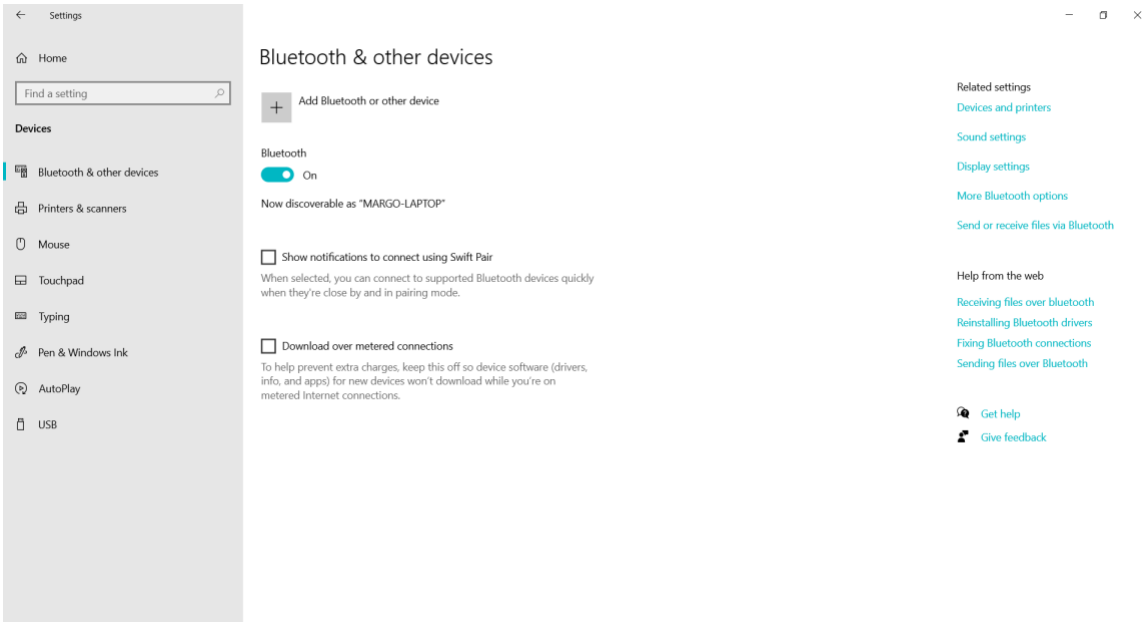

Figure 19: Settings' Bluetooth & other devices menu

Pairing the Device with the Bluetooth Terminal

After establishing a connection with the computer, navigate to the “Start” menu and open the Bluetooth Serial Port app. If an “Error” window appears upon opening the app, you can safely ignore it by clicking the “Close” button on the right.

In the app window, locate the dropdown menu titled “Bluetooth name:” at the top right corner. Click on it and select the “Adherence_Sensor” option from the list, as shown in Figure 20.

Next, click on the “Open link” button. A small message confirming “link success” will appear, indicating that you can now successfully send commands to the device.

Figure 20: “Adherence_Sensor” selected from the Bluetooth dropdown menu with “Open link” circled

How to Send Commands to the Device

Within the Bluetooth Serial Port app, locate the large textbox positioned at the bottom of the window to input commands. When typing commands, refrain from pressing the “Enter” key. Instead, click the arrow icon in the bottom right corner to send the command.

Pressing “Enter” inadvertently adds an extra line to your command, potentially leading to the device returning an “Invalid command” message.

Several commands are available:

- “help” provides a brief description of the functionality of all other commands (Figure 21)
- “read” retrieves the encrypted contents of the file stored within the device (Figure 22)
- “delete” clears the file stored within the device (Figure 23)
- “exit” instructs the device to deactivate Bluetooth mode (Figure 24)

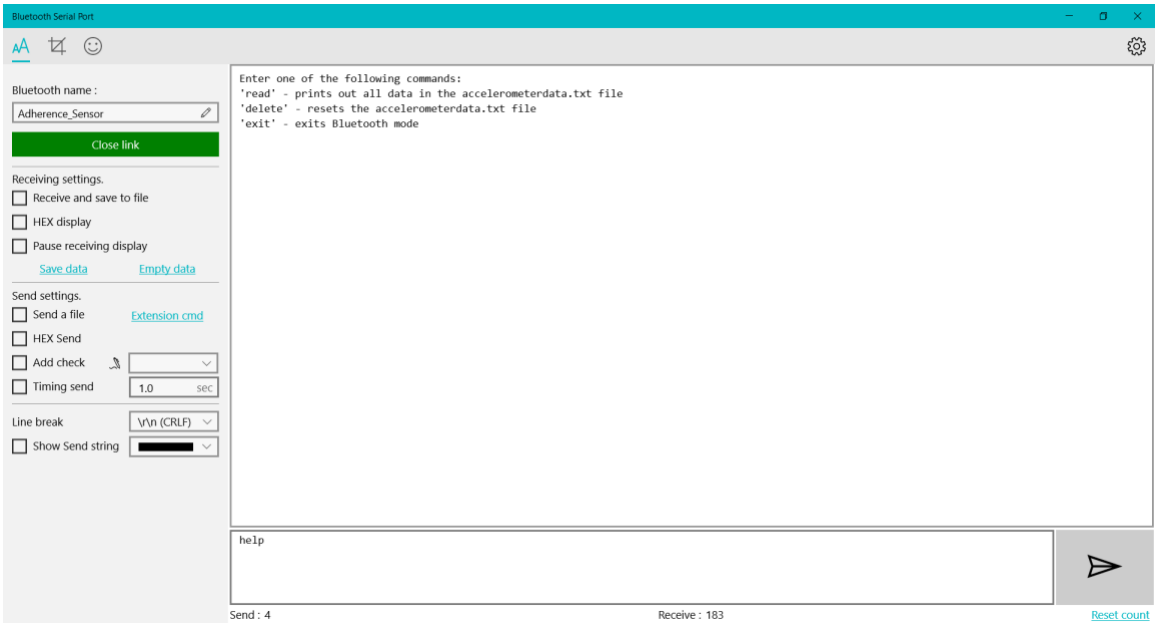

Figure 21: “help” command providing a brief description of all other commands

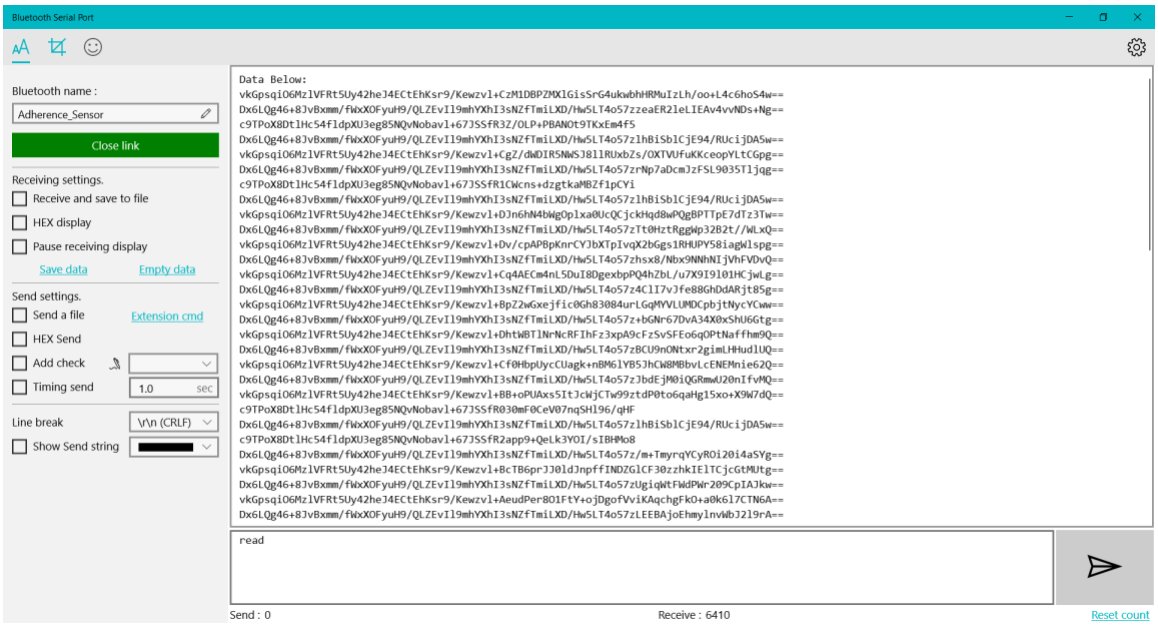

Figure 22: "read" providing a printout of all data on the device

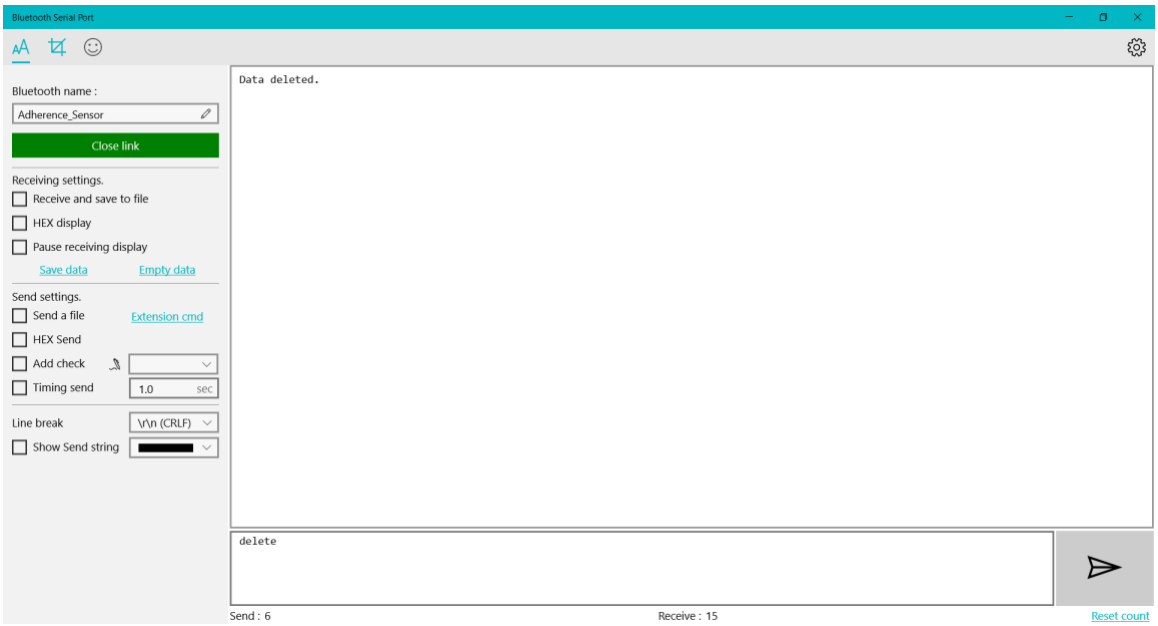

Figure 23: "delete" confirming that all data has been deleted from accelerationdata.txt

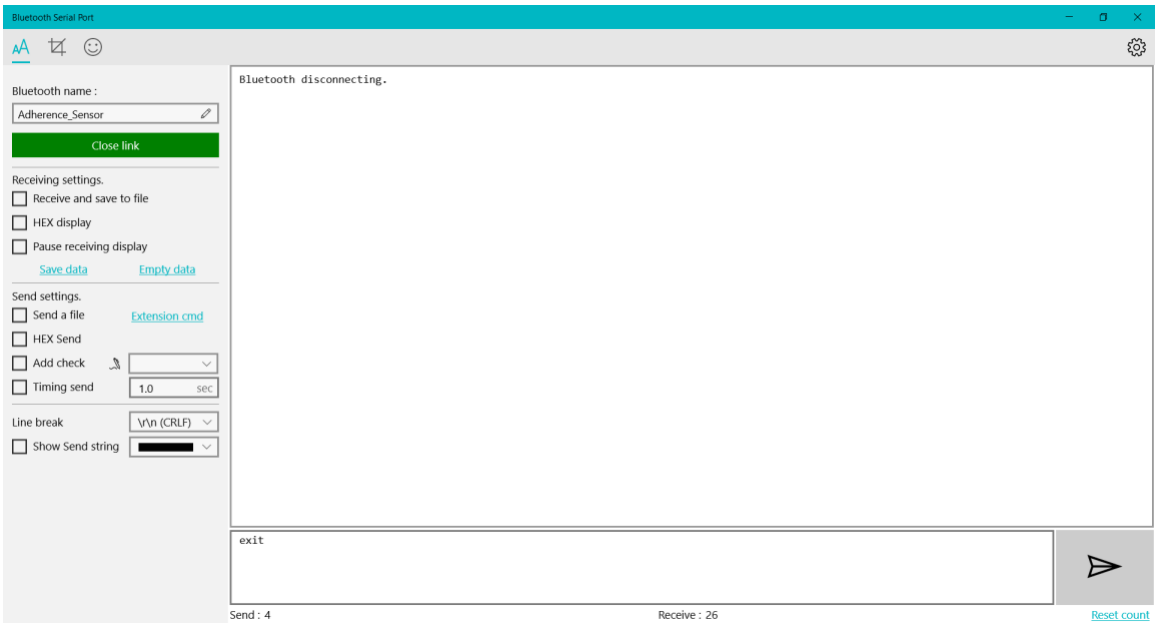

Figure 24: "exit" disabling the Bluetooth on the device

Saving Data to a File

As shown in Figure 25, after using the “read” command to retrieve the data, press the “Save data” button circled in blue. A prompt will appear to locate the folder to save and enter the file's name, depicted in Figure 26. Delete any extra command prompts that may be saved to the top of the file. The “Save file” functionality will save any information displayed in the white box upon clicking.

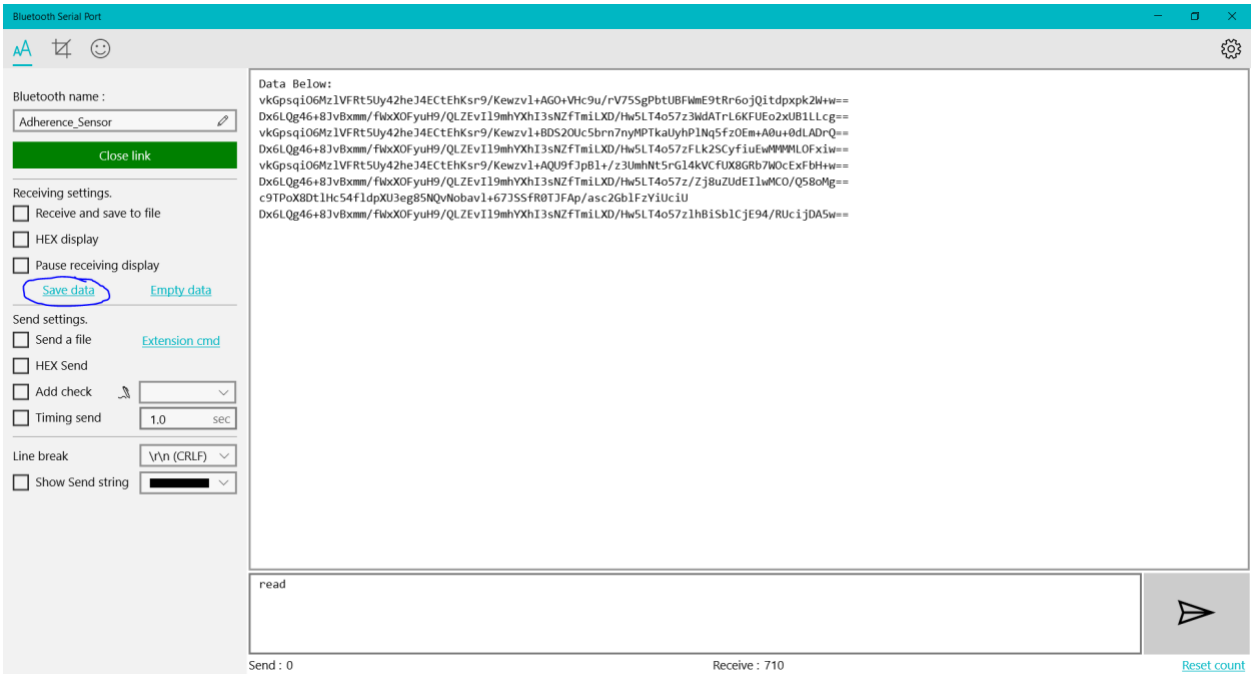

Figure 25: The “Save data” button circled in blue with encrypted data printed in the terminal.

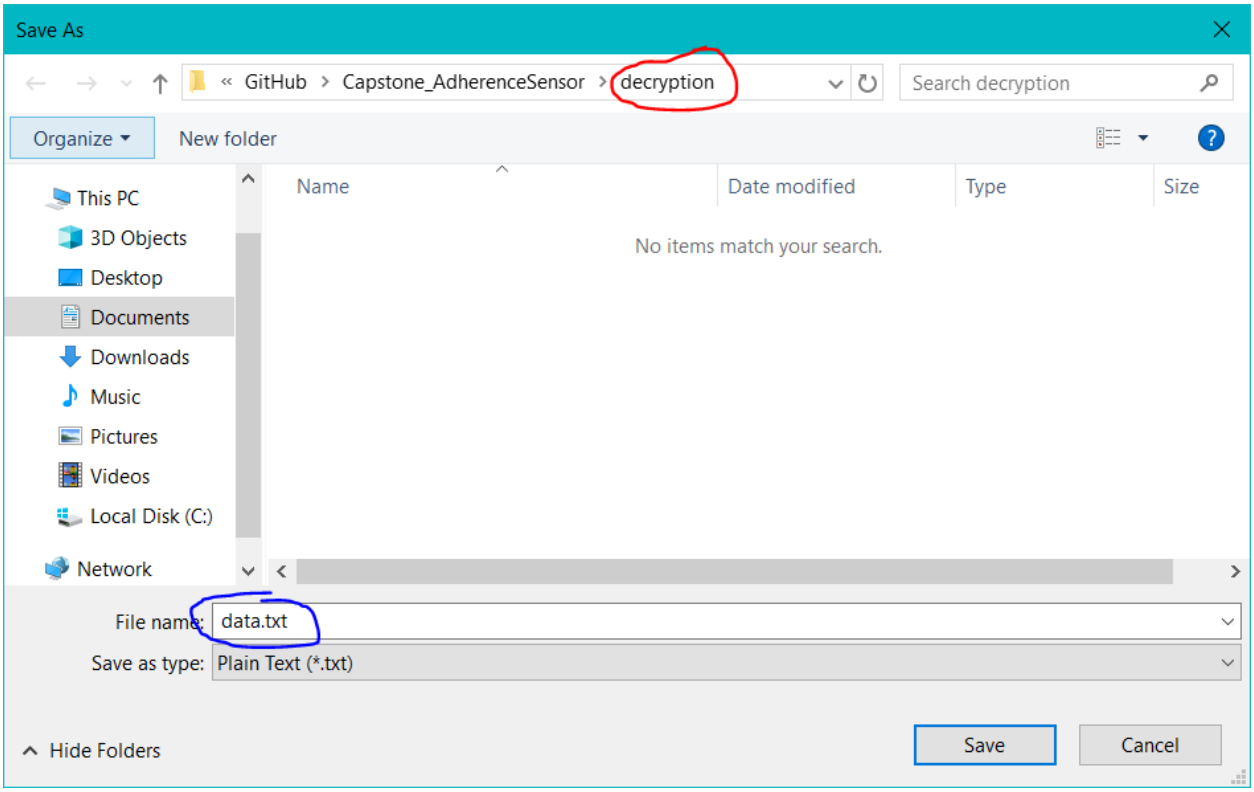

Figure 26: Saving the file as "data.txt" to the folder "decryption."

Deleting Data from the Device

As shown in Figure 27, sending the delete command will delete data from the “accelerationdata.txt” file and will display a confirmation that data was deleted. Attempting to read data from an empty file will display the message “File is currently empty.”

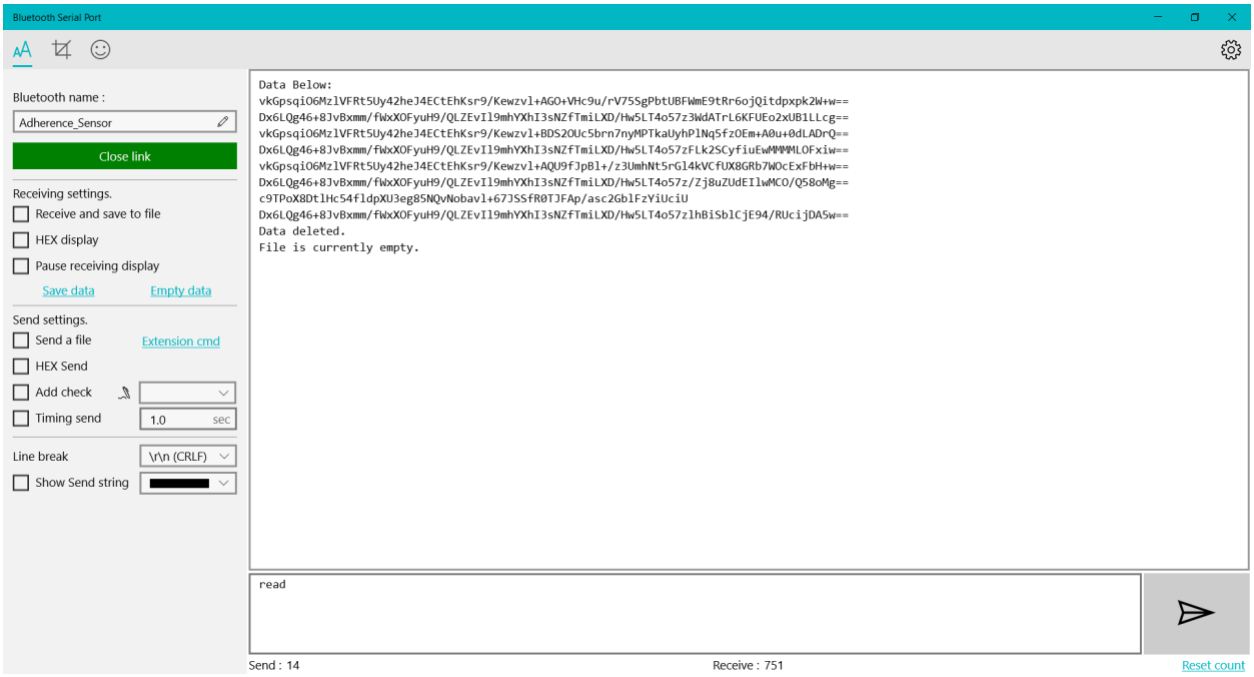

Figure 27: Attempting to read from an empty file gives the response, “File is currently empty.”

Exiting Bluetooth Mode

Enter the “exit” command into the terminal, and the Bluetooth capabilities will be disabled. The computer will no longer have a Bluetooth connection with the Adherence Sensor.

Decrypting the Data

Downloading “decrypt-acceleration.exe”

Open your internet browser and go to the GitHub link provided: https://github.com/brownmar24/Capstone_AdherenceSensor.git

As shown in Figure 28, On the main page of the “Capstone_AdherenceSensor” repository, look for the green dropdown button titled “<> Code” and click on it (as shown by arrow 1). From the dropdown menu that appears, select “Download ZIP”. This will download the entire project as a ZIP file named “Capstone_AdherenceSensor-main” (as shown by arrow 2).

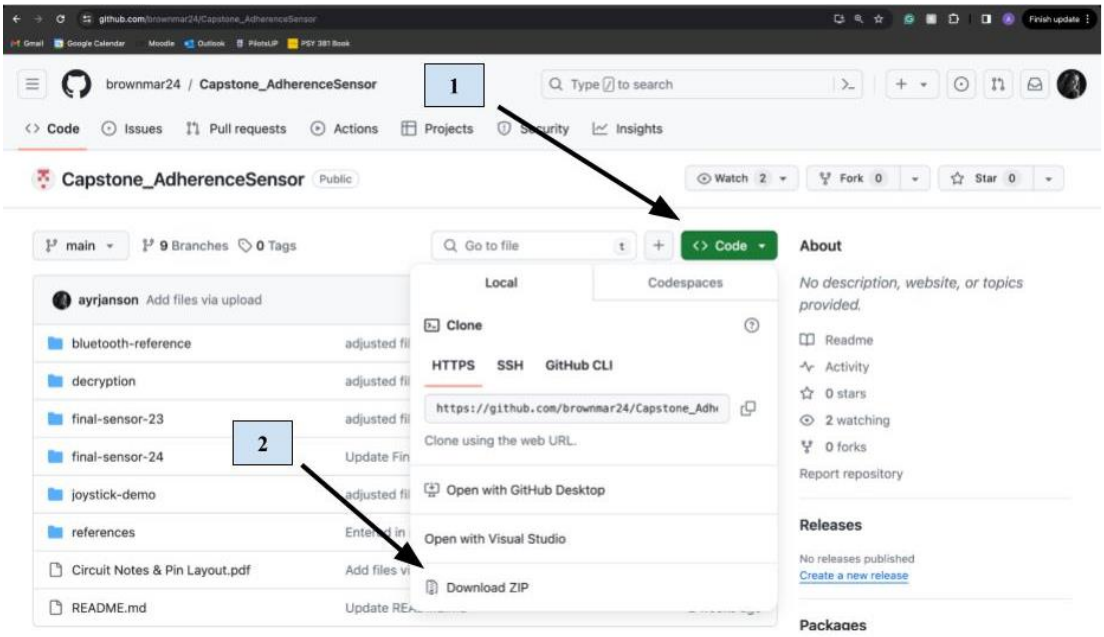

Figure 28: The “Capstone_AdherenceSensor” repository with the “<> Code” dropdown menu

As shown in Figure 29, double-clicking on the ZIP file will unzip the repository contents into the same location where the ZIP file was located. After extracting, locate the “decryption” folder (identified by arrow 3). Move the “decryption” folder to your desired location on your computer; it can work separately from the rest of the repository.

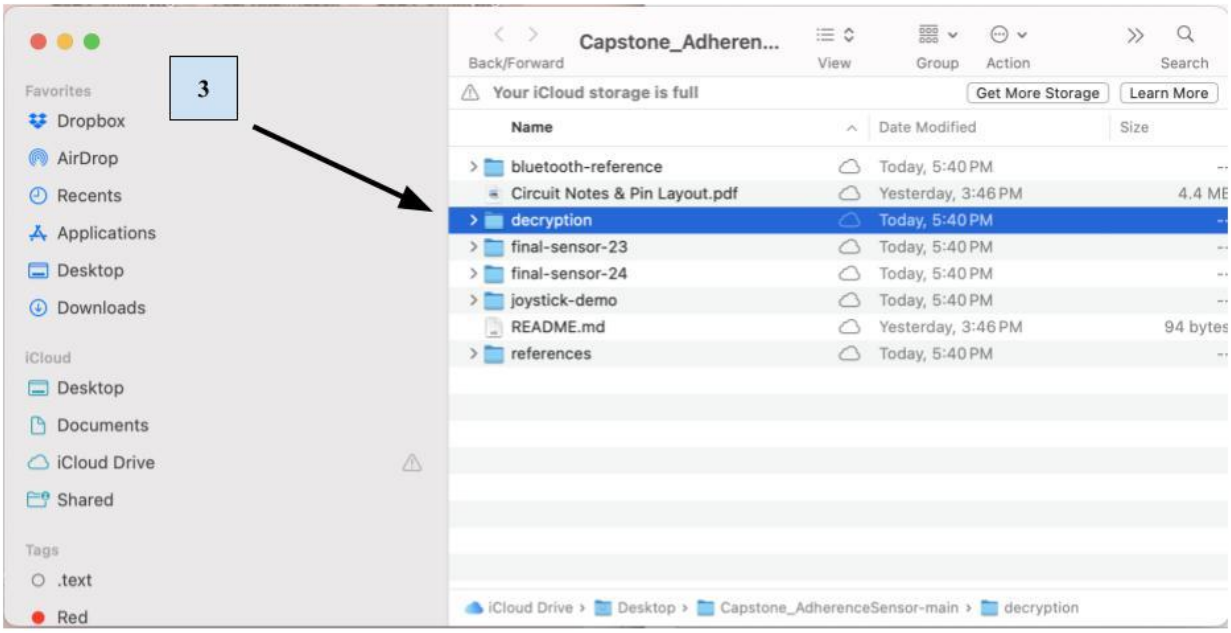

Figure 29: The “Capstone_AdherenceSensor-main” unzipped folder with the “decryption” folder

Using “decrypt-acceleration.exe”

As shown in Figure 30, to use “decrypt-acceleration.exe” for decryption, place the “data.txt” file (identified by arrow 4) in the same folder as the executable. Then, double-click on “decrypt-acceleration.exe” to run the decryption program (identified by arrow 5).

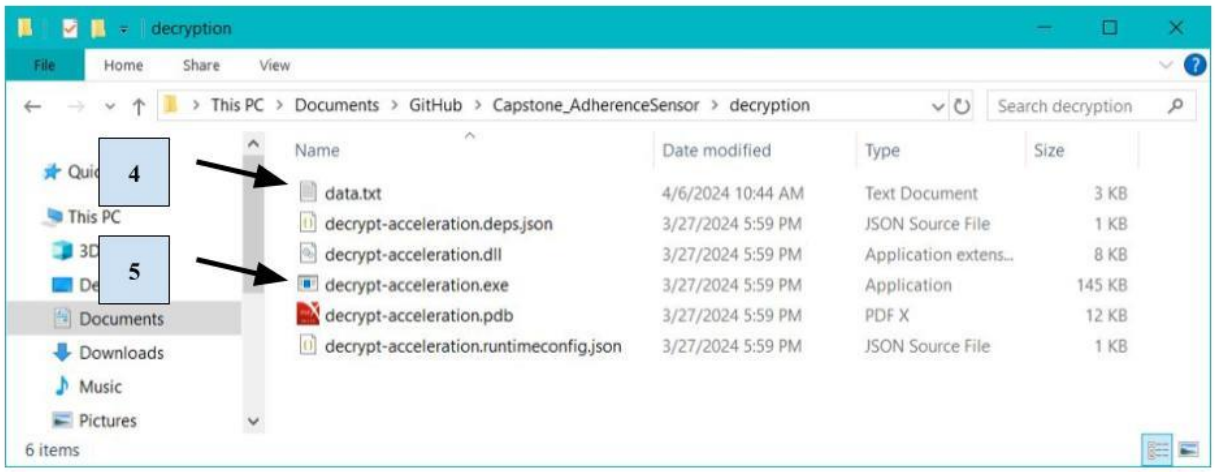

Figure 30: The decryption folder containing “data.txt” with encrypted motion data

As shown in Figure 31, the executable will then produce the decrypted data in the same folder (as identified by arrow 6).

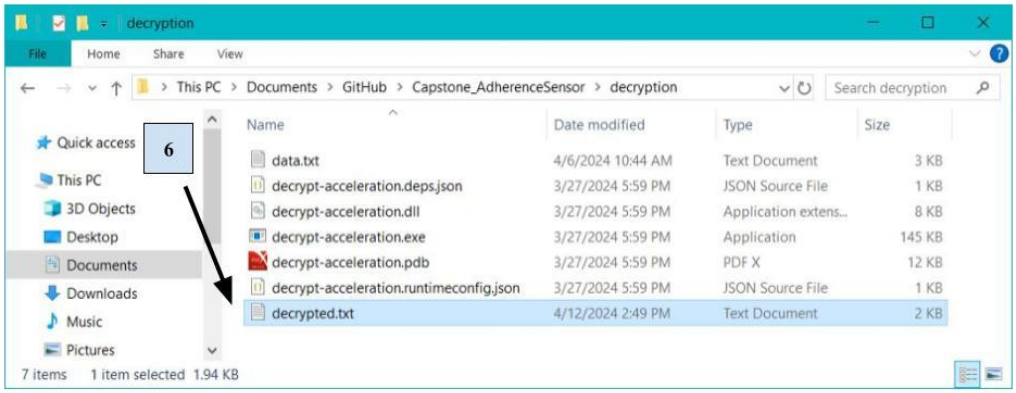

Figure 31: The decryption folder with the decrypted motion data in “decrypted.txt”

The decrypted.txt should contain human-readable text like the text in Figure 32 below:

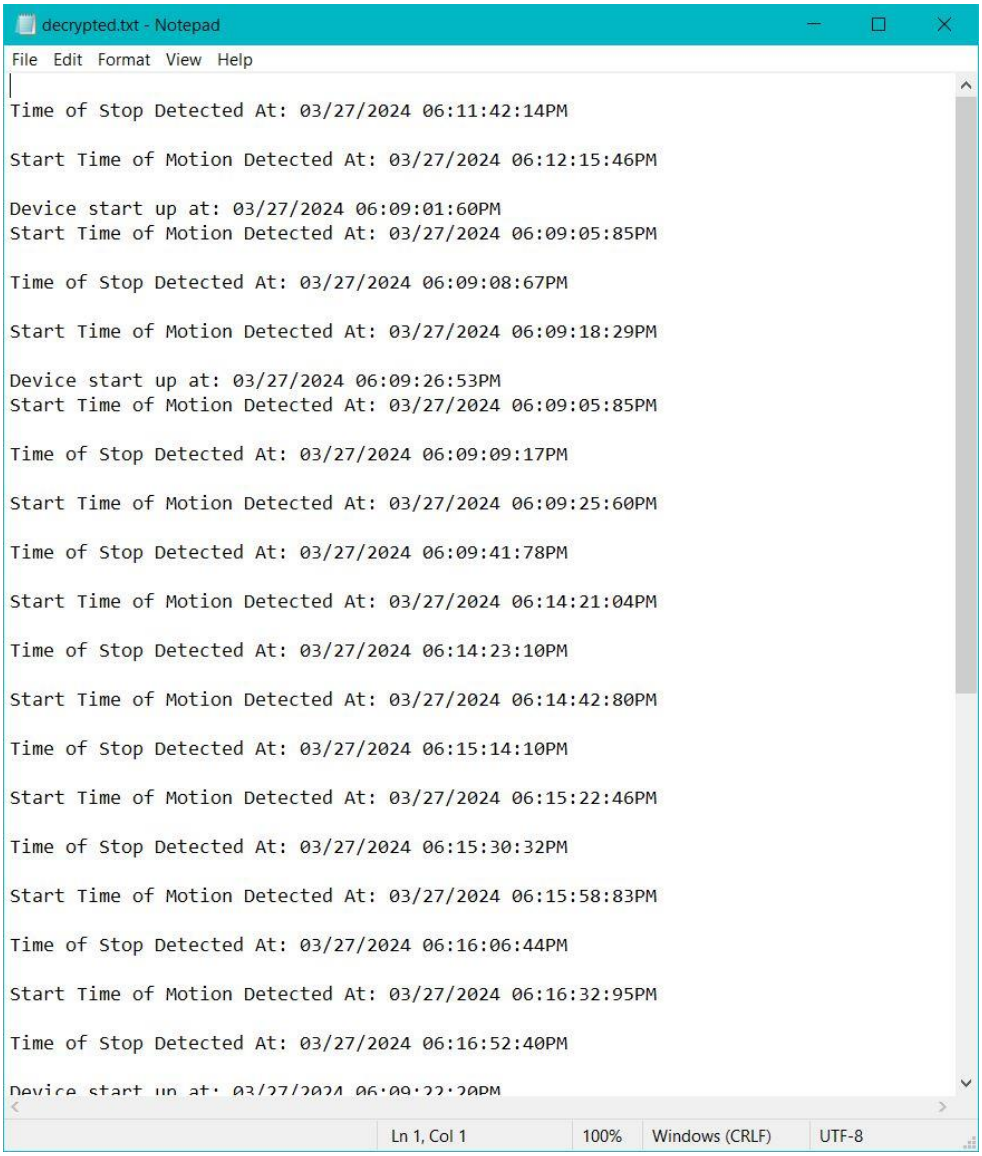

Figure 32: An example file of decrypted motion data

Troubleshooting Errors During Decryption

If an error occurs during execution, verify that no additional text besides the encrypted data is in the file. Commonly, the “read” command used to request the data will appear at the top of the file. Delete any human-readable text in the encrypted data file. Figure 33 shows the common error message received when the executable attempts to decrypt any non-encrypted text.

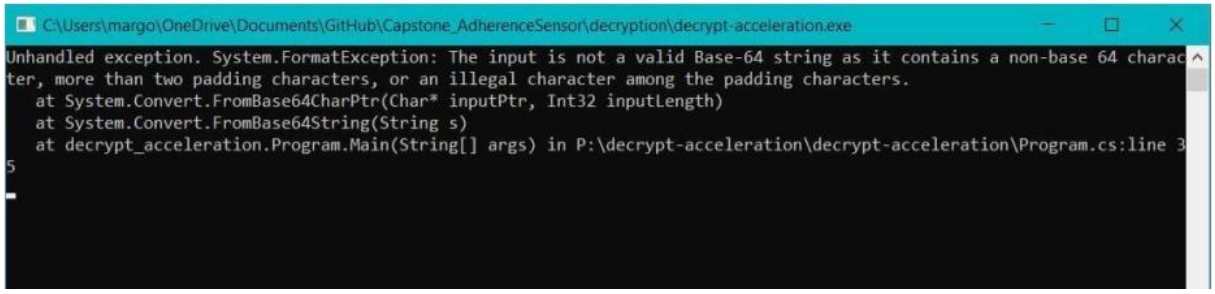

Figure 33: Error received when attempting to decrypt a file with non-encrypted text included

Understanding the Data

The decrypted.txt file has three types of recordings: device start-up timestamps, start-of-motion timestamps, and end-of-motion timestamps. These timestamps do not record any directional data information.

Figure 34 shows a decrypted motion data file with all three types of recordings.

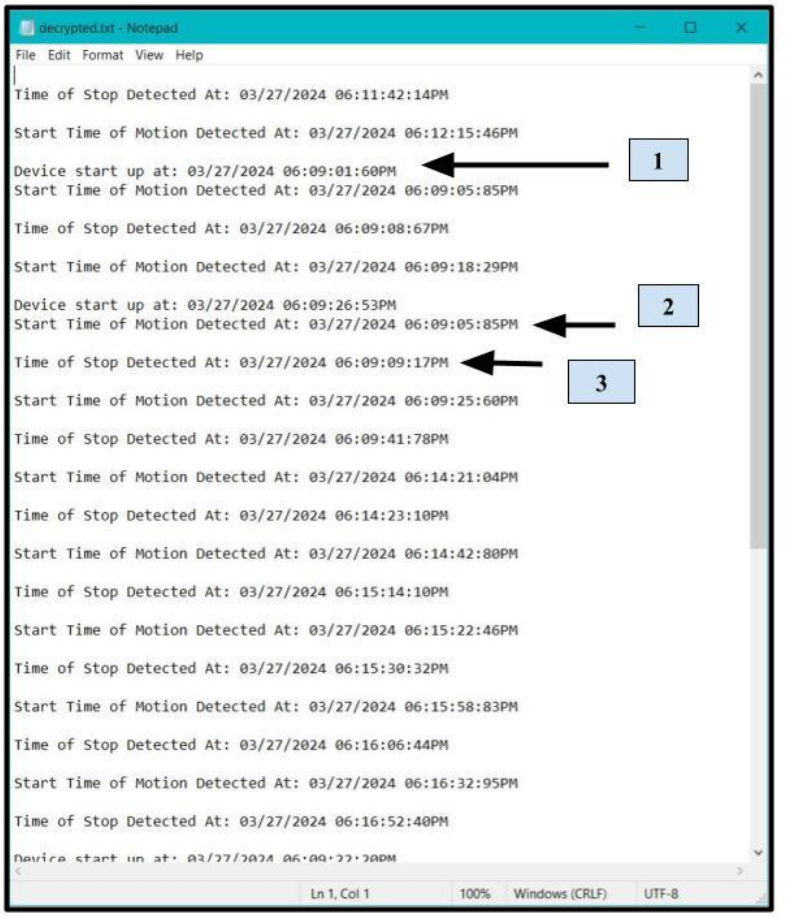

Figure 34: Displays device start-up times and motion timestamps

Types of Data Recordings

As shown by arrow 1, when the device is turned on, the device prints the following to the text file:

“Device start up at: MM/DD/YYYY hh:mm:ss:ms(A/P)M”

As shown by arrow 2, when the device detects that the child has started moving, the device prints the following to the text file:

“Start Time of Motion Detected At: MM/DD/YYYY hh:mm:ss:ms(A/P)M”

As shown by arrow 3, when the device detects that the child has stopped moving, the device prints the following to the text file:

“Time of Stop Detected At: MM/DD/YYYY hh:mm:ss:ms(A/P)M”

Where:

- MM – Month
- DD – Day
- YYYY – Year
- hh – hour (12-hour clock)
- mm – minute
- ss – second
- ms – millisecond
- (A/P)M – AM or PM

Appendix E: Hardware Guide

Parts List

Part	Quantity
Adafruit LIS3DH Triple-Axis Accelerometer [8]	1
SparkFun Triple Axis Accelerometer Breakout – KX134 [9]	1
SparkFun Real Time Clock Module – RV-8803 (Qwiic) [10]	1
Lithium-Ion Polymer Battery – 3.7v 2500mAh [11]	1
Tenergy Fire Retardant LiPo Bag [12]	1
USB Lilon/LiPoly Charger – v1.2 [13]	1
Amazon Basics USB-A to Mini USB 2.0 – 3ft [14]	1
microSD Card – 1GB [15]	1
Casing/Box	1
SparkFun IoT RedBoard – ESP32 Development Board [17]	1
LED – RGB Addressable [18]	1
USB Wall Charger, GiGreen Dual Port Electrical Cube 5V [22]	1
Breadboard	1
1k Ohm Resistors [24]	4+
Button [31]	1
Switch [32]	1
Pin headers [34]	1+
Velcro strips with adhesive [33]	2+
Electrical Wires	1+

*Purchase links can be found in references.

*The quantities column has the minimum number of individual parts needed (not packs or sets). Some quantities have a range or a “+” because it depends on product type and how efficiently the circuit is connected. For instance, two strips of Velcro are needed on the outside of the adherence sensor casing, but some Velcro strips are sold in a roll, so the quantity is “2+”. For the pin headers case, the quantity is “1+” because some pin headers are sold with 5-12 pins, but they can be broken up and it depends on what pins the creator wants to be plugged in to other parts. Many circuit elements require a resistor of at least 1k Ohm to be in series with it, but depending on how efficient the circuit is built, the number of resistors can vary. The less resistors/elements/wires used, the better!

*The breadboard used was donated, we were unable to find the same one online, but any solderable breadboard can be used.

Steps

*Steps do not need to be done in the order they are listed in.

*Please analyze the datasheets (found on the part websites) of all circuit elements prior to assembly.

Figure 1 is a photograph of the internal circuitry of the current iteration of the Adherence Sensor.

Figure 1: Current Adherence Sensor Internal Circuitry

As shown in the Internal Circuitry Diagram (Figure 2, insert the microSD card into the microSD card slot. Then, connect “SD Det” to the back of the ESP32. Solder EVI on the Real-Time Clock to pin 14 of the ESP 32. Then, connect all the SCLs, SDAs, Power pins, and GND pins together on the ESP32, Real-Time Clock, and the accelerometer.

To create the on/off switch, first source a switch with 3 prongs and cut off one of the edge prongs. Then, cut the red wire on the Lithium-Ion Battery. Then, connect and solder one of the cut red wires to one prong and the other cut red wire to the other prong, making sure that they don’t touch.

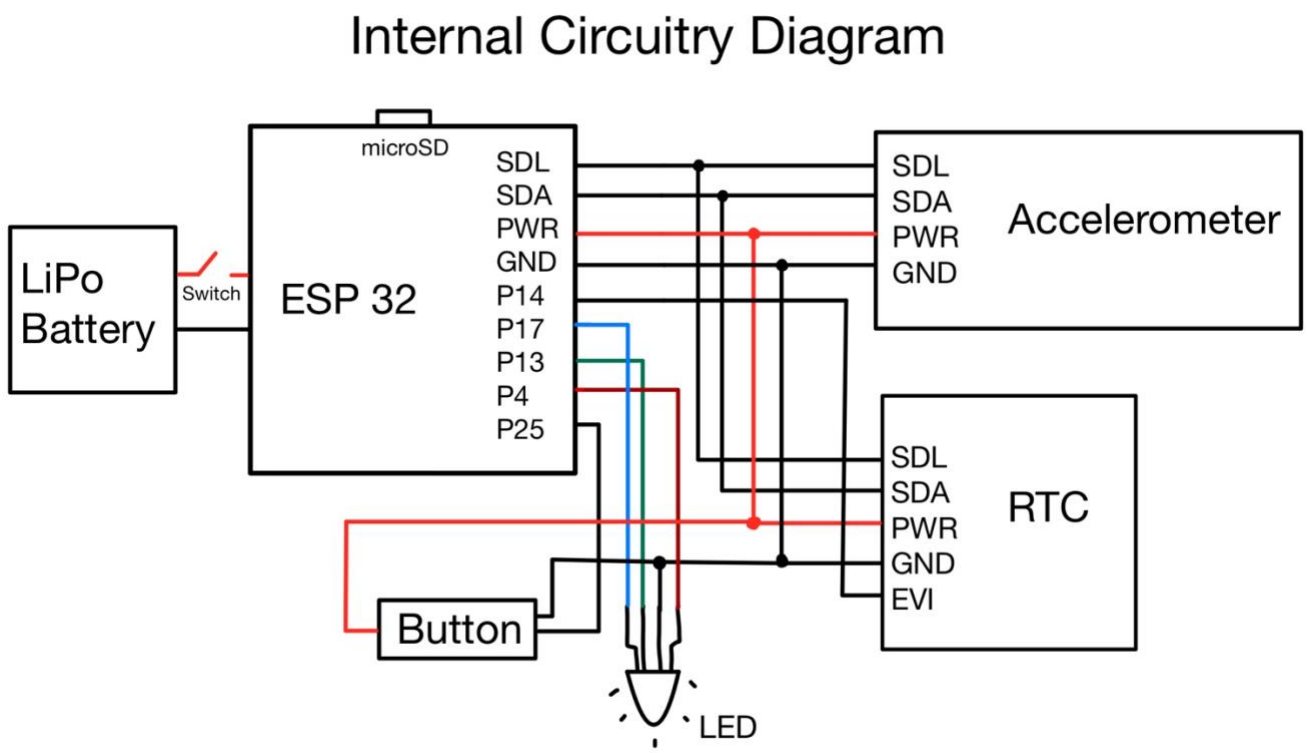

Figure 2: Connections between circuit elements.

The entire circuit must be oriented specifically to ensure correct motion detection. Match the X, Y, and Z axes on the accelerometer to the two previous devices (X should be pointing down, and Y should be pointing right, as seen in Figure 3 below).

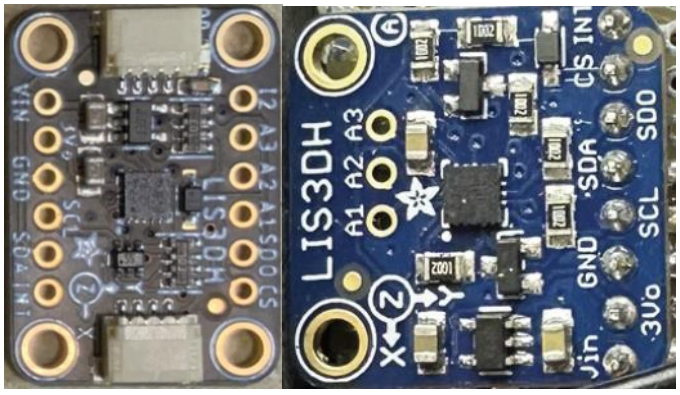

Figure 3: Correct accelerometer orientation.

The RGB LED, shown in Figure 4 below, has each pin specified for a color input or grounding. Each RGB input pin (all except GND) must have a 1kΩ (or greater) resistor connected in series with each leg. Then, each of the pins must be connected to their corresponding ESP32 ports as designated in the Arduino program.

1. DIN is connected to pin 17 on the ESP32 (Blue)
2. VDD is connected to pin 13 on the ESP32 (Green)
3. GND to Ground on the ESP32
4. DOUT is connected to pin 4 on the ESP32 (Red)

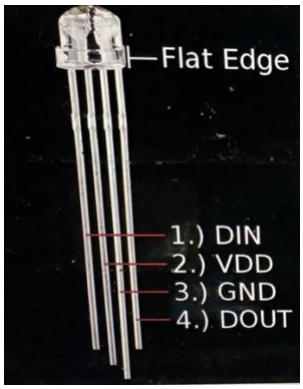

Figure 4: Clear Addressable 5mm LED diagram.

Figure 5 shows an example of a button that is connected to a microcontroller. One half of the button must be connected to power (red wire in Figure 5), the other half must be connected to the resistor, pin 25 (green wire), and ground (black wire). A 1kΩ (or greater) resistor must connected with it in series.

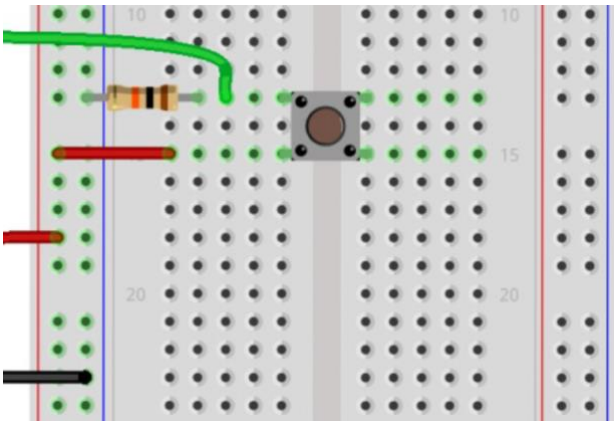

Figure 5: Example of a button connected to a microcontroller.

As shown in Figure 6, the current iteration of the Adhesion Sensor has a firm plastic box encasing its electronic components. A decoration or image should be on the outside of the case to indicate the correct orientation of the Adherence Sensor, previously achieved with ocean-themed

stickers, as shown in Figure 6. Additionally, on the back of the case, Velcro straps, each at least an inch wide, should be attached for mounting purposes, as depicted in Figure 7.

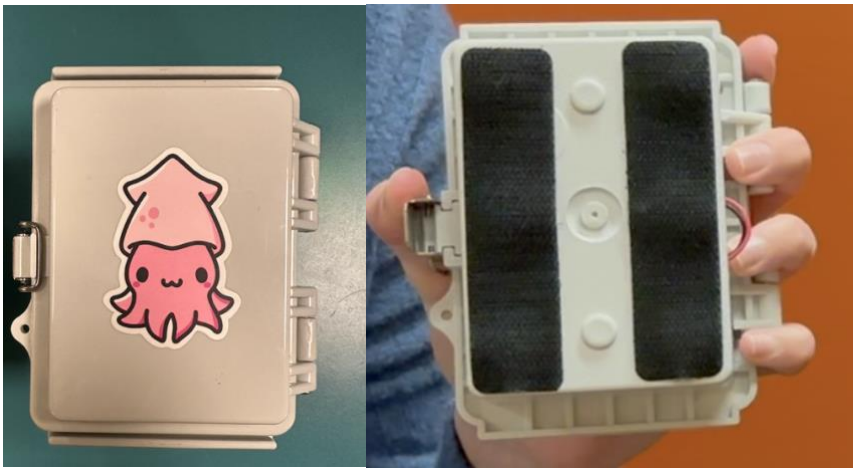

Figure 6: Photograph of the plastic box encasing the Adherence Sensor electronic components

Figure 7: Velcro straps on the back of the case.

The size of the casing or box will be determined by the size of the circuit (or vice versa). As seen in previous cases, the casing should feature a relatively childproof covering made of durable material, typically hard plastic. As seen in Figure 8, there need to be four holes hard enough to accommodate a switch, RGB LED, and Lithium-Ion Battery power cord.

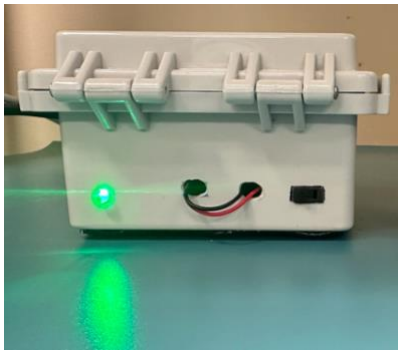

Figure 8: Side view of the Adherence Sensor featuring components requiring holes in the sensor

Arduino Uploading Guide

To access the Adherence Sensor repository on GitHub, use the provided link: https://github.com/brownmar24/Capstone_AdherenceSensor

Once on the repository's main page, click on the green "<> Code" button, then select "Download ZIP" from the drop-down menu at the bottom at the list. Extract the folder form the ZIP file, creating a new folder called "Capstone_AdherenceSensor-main" containing all the necessary files. Open this folder, locate the file named "FinalSensor24," and open it. Inside, find the file named "FinalSensor24.ino" and click on it to automatically open the Arduino IDE. Table 1 shows the specific libraries to download from the Arduino library search.

Library Listing	How to Download	Other Notes
<BluetoothSerial.h>	BluetoothSerial by Espressif	• Go to Tools>Board>Boards Manager • Search for "esp32" by Espressif Systems and download
<BTAddress.h>	See <BluetoothSerial.h> other notes	Included in esp32 by Espressif
<BTAdvertisedDevice.h>	See <BluetoothSerial.h> other notes	Included in esp32 by Espressif
<BTScan.h>	See <BluetoothSerial.h> other notes	Included in esp32 by Espressif
<SPI.h>	None	Pre-installed on Arduino
<Wire.h>	None	Pre-installed on Arduino
<SparkFun_RV8803.h>	"SparkFun_RV-8803_Arduino_Library" by SparkFun	
<Adafruit_Sensor.h>	"Adafruit Unified Sensor" by Adafruit	
<Adafruit_LIS3DH.h>	"Adafruit_LIS3DH" by Adafruit	
<SparkFun_MAX1704x_Fuel_Gauge_Arduino_Library.h>	"SparkFun_MAX1704x_Fuel_Gauge_Arduino_Library" by SparkFun	
<AES.h>	"AESLib" by Matej Sychra	
<AESLib.h>	"AESLib" by Matej Sychra	
<Crypto.h>	"Crypto" by Rhys Weatherly	
<CTR.h>	See <Crypto.h> How to Download	Included with "Crypto" library by Rhys Weatherly
"arduino_base64.hpp"	"base64_encode" by dojyorin	
"Fs.h"	None	Included in esp32 by Espressif
<SD.h>	SdFat – Adafruit Fork by William Freiman "SD" by Arduino and SparkFun	
"SPI.h"	None	Pre-installed on Arduino

Table 1: Required libraries needed for the Adherence Sensor program

As identified in Table 2, the compilation settings are specific to the constraints of the Adherence Sensor. In the tools menu, select the following settings.

Tool Setting	**Value**
CPU Frequency	240MHz (WiFi/BT)
Core Debug Level	None
Erase All Flash Before Sketch Upload	Disabled
Events Run On	Core 1
Flash Frequency	80MHz
Flash Mode	QIO
Flash Size	4MB (32 Mb)
Arduino Runs On	Core 1
Partition Scheme	Huge APP (3MB No OTA/1MB SPIFFS)
PSRAM	Disabled
Upload Speed	115200

Table 2: Tools menu settings needed to compile the Adherence Sensor program

As shown in Figure 9, to set the board and port that you will upload the device to, select the dropdown menu pointed to by arrow 1. From there, click “Select other board and port…” Using the search bar, select the option to use the “SparkFun ESP32 IoT RedBoard.” Then, connect the adapter cord from your computer to the ESP32. The ESP32 has a USB-C port for uploading data. Once a connection between the computer port and the ESP32 is established, the USB serial port should appear under the port menu. Select the available USB port.

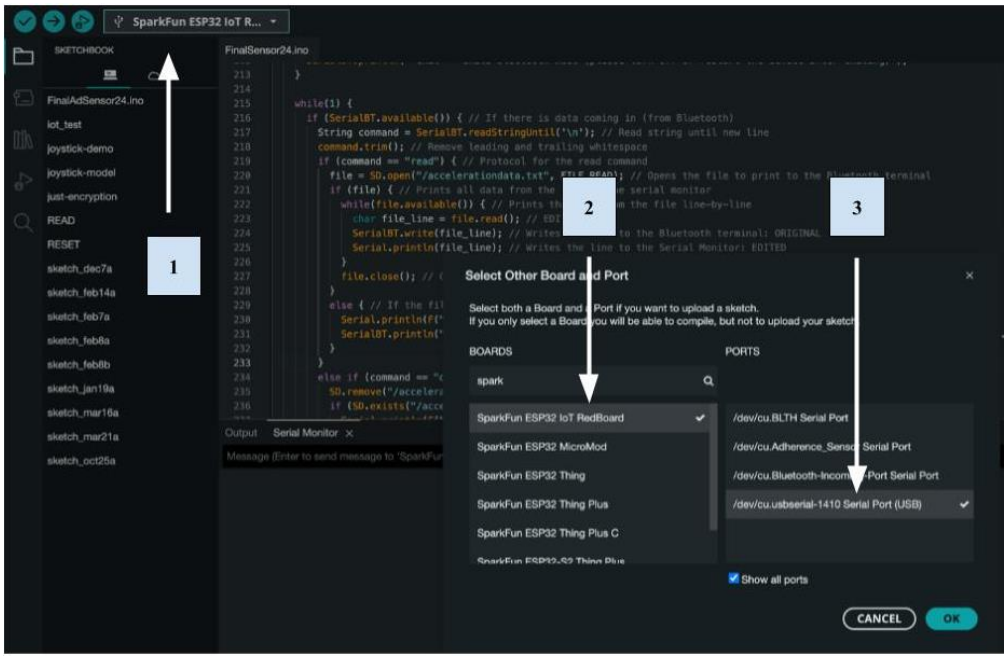

Figure 9: Board and port selection menu within Arduino IDE

As shown in Figure 10, the button pointed to by arrow 1 will compile the program and check for errors. The button pointed to by arrow 2 will compile and upload the program to the sensor through the specified port.

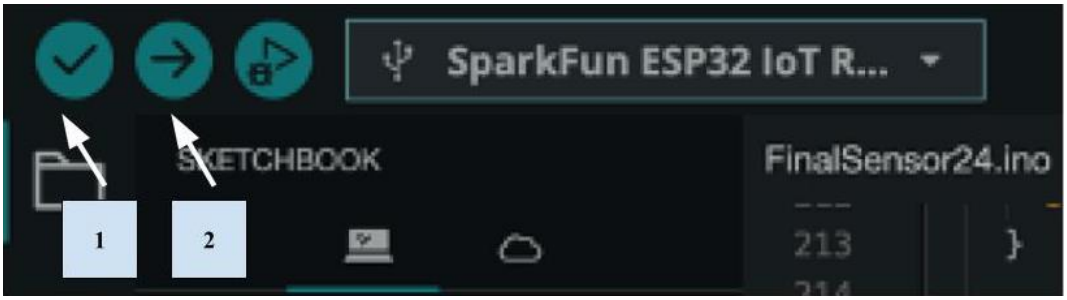

Figure 10: The buttons used to compile and upload the Adherence Sensor program to the device

From here, you should be able to upload the Adherence Sensor program to the ESP32 and have a working Adherence Sensor.

Appendix F: Anna Yrjanson's Contributions

Professional Development

Anna coordinated meeting times and objectives throughout the project, ensuring productive discussions and goal-oriented development. She meticulously tracked the progress of her teammates and provided support and guidance where needed. She also developed an Arduino programming reference guide and a comprehensive budget request form. Anna also accepted the team leader role for sprints 1, 4, and 5. In the last week of development, she also created the framework and content requirements for the manufacturing and user manuals for the device.

Arduino Programming

Anna was pivotal in Arduino programming, contextualizing the previous code base to align with the project requirements. She implemented the multi-color LED conditional display and restructured the loop function for improved efficiency with the new functionality. She also conducted a thorough restructuring of the loop function.

Bluetooth and Encryption

Anna led research on Bluetooth interface design and successfully integrated AES 128 encryption into the system. She also worked on a C# decryption program that decrypts the data file on a double-click.