AUGMENTED REALITY INFORMATIONAL SYSTEM (ARIS) FOR ASTRONAUTS

FINAL PRODUCT REPORT

Oussama Ourich (EE-Co-Lead) Felice Gabardi (EE) Kalvin Fonseca (EE)
Maryam Aminu Mukhtar (CE-Team Lead)
CM/TM : Tomas Materdey

University of Massachusetts Boston | 2022-2023

Augmented reality Informational system (ARIS) for Astronauts

Engin 491- Spring 2023

AM. Maryam, O. Oussama, G. Felice, and F. Kalvin

***Abstract*— The Artificial Reality Assistant is designed to give astronauts the ability to access information about their body signatures such as temperature, pulses, oxygen level, and acceleration with the use of sensors. Stored data from past evaluations will be used to compare if there are any abnormal changes and to classify the activity. This will require machine learning to create a math model using the data to predict and compare the status and activity of the user. The Head Mounted Display (HMD); HoloLens will be capable of displaying the collected consumables and body signature data. It will have a hand gesture feature that will track the motion of the glove for interaction and identify the feature name, in addition to the distance to it and the consumables needed to get there. The project will use two Head Mounted Displays (HMDs) that should be able to communicate with each other using a sensor hub and share information. An RF communication network will be designed using a Zigbee protocol to allow the different systems to communicate with each other. The data will come from a sensor node that will comprise the different sensors, which will be sent to the sensor hub for processing before getting sent to the HoloLens from a server to be displayed.**

***Index Terms*—Head Mounted Display (HMD), User Interface (UI), HoloLens, Sensor Hub, Sensor Node, XBee modules, Zigbee protocol.**

I. PROBLEM DEFINITION

The AR Assistant is designed so that astronauts can gain the ability to access information about their body signatures such as body temperature, pulses, acceleration, oxygen levels, and H_2O levels with the use of sensors. This will be done within the sensor node which incorporates sensors for each requirement. The sensor node will be connected to the sensor hub via ZigBee which should transfer the data over to the access point in this case the raspberry pi 4. The access point serves to connect the sensor hub to the HoloLens. The data will be recorded and will be saved within the library so it will be compared to the predictions that are going to be created using MATLAB calculations of how the body should be. Machine learning will create math models for the comparisons to determine the status of the user such that if there is a slight deviation then it will be considered either “normal-” or “normal +” depending on if it is a high or a low deviation. If it’s a major deviation, then it will be classified as “abnormal”. The data will be displayed on the HoloLens of the HMD which will be modified using the Unity software to organize the hub. An eye gaze feature will be used to track the movement of the glove and to analyse the feature. Such features should be named accurately and displayed on the HoloLens. Also, the distance between the actual feature and the user will be determined by comparing their coordinates using a GPS module that will be attached to the sensor hub mentioned earlier. With the distance calculated then, we will determine the consumables (water and O2 levels) necessary to complete the walk. All this data will then be transferred to the second set of HMDS using the ZigBee to be communicated between the two sensor hubs to allow them to monitor their state. The communication should have at least 3 ft to exhibit the experience of a space mission of two astronauts. The second AR assistant will be able to carry out the same functions as the one initially tested.

II. BACKGROUND AND NECESSITY

NASA has been cooperating with international space agencies for the better part of a decade now and has been able to push science experiments to a new frontier, exploring the potential for long-term habitation in space inside the International Space Station (ISS). There are many maintenance requirements onboard the ISS that must be performed on the exterior of the hull; these procedures are called space walks. Spacewalks are the most dangerous aspect of life aboard the ISS. NASA has

funded a project they are calling “Side-Kick” where they desire a Head Mounted Device that has the capability to track the vital systems an astronaut would need to monitor while performing their maintenance on spacewalks. These vital systems include heart rate, oxygen levels, water levels, battery life, the distance between themselves and the target, speed, and the ability to determine if the rate of change of any of these values is normal or abnormal. NASA believes a well-developed sensor hub to connect to their already contracted Microsoft Holo-Lens will be paramount to the implementation of Sidekick and will drastically increase the safety and trainability of astronauts going forward.

III. REQUIREMENTS

This project is working towards designing an Artificial reality (AR) assistant for astronauts to give them the ability to easily monitor and access information about their body responses, consumables, feature names, and distance. To meet the requirements of the design, the system will comprise a Head Mounted Display (HMD); HoloLens 2, a sensor hub utilizing a Raspberry Pi, sensor nodes that will operate using an Arduino Nano, pulse, temperature, pressure, water, battery, accelerometer, and GPS sensors/modules to collect the necessary data and ZigBee modules to create the communication network.

a. CUSTOMER REQUIREMENTS

Our customer requirements are:

I. A sensor node that is integrated on a PCB will connect to a hub utilizing a Raspberry Pi that is capable of handling real-time data analysis to report pulses, body temperature, and acceleration.
II. The hub must be capable of using Machine Learning (ML) to make a math model from prior data of the variation of the level of physical activity vs the pulses.
III. The hub will need to be able to utilize the math model data of running speeds, accelerations, and body response (pulse changes) and report a normal change vs abnormal change and suggest adjustment via Head Mounted Display (HMD).
IV. The HMD (HoloLens 2) will acknowledge astronaut eye movement and finger motions; while wearing gloves and accurately provide the sensor data and information such as feature name (a building on the UMB campus), its distance, and how much O_2, H_2O, and the battery will be needed to get there from the current location via 2 alternative paths.
V. The two hubs, one per astronaut must be able to communicate and share information on respective HMDs.
VI. The communication between the two hubs will be a variable or single-frequency network (RF)
VII. Each hub will be powered through a battery that has a Battery Management System (BMS) and each full unit with a battery must not weigh more than 4 lbs.

b. ENGINEERING REQUIREMENTS

Our engineering requirements focus on the performance, functionality, reliability, and energy of the system. The system will be designed with the sensors having a performance accuracy of 95%, and the ability to use the created ML model to provide consumables data in real-time, the HMD should recognize eye movement and display data, and accurately identify the astronaut status as normal or abnormal. The AR assistant should be operational 99.9% of the time, have a data transmission time of 180sec between devices, and be consistent at responding to commands, identifying feature names, and determining the distance plus paths between features and the user. In addition to that, the system needs to have an average power consumption of 20 W between the HMD and microcontroller, a battery line of 5-6 hours, and the full unit with a battery must not weigh more than 4 lbs.

IV. FLOW CHART & BLOCK DIAGRAM OF SYSTEMS

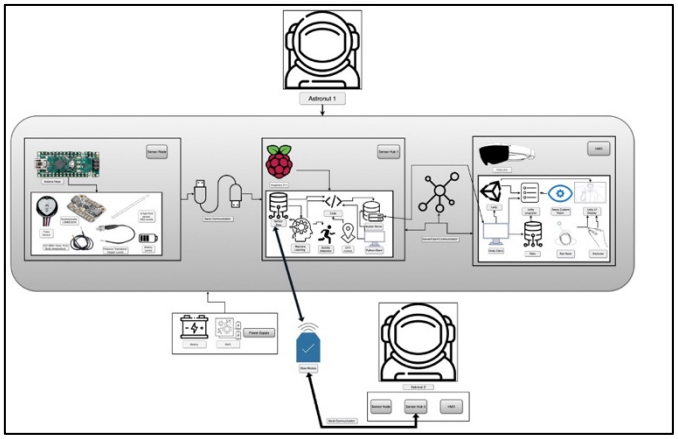


Figure 1: System flowchart diagram

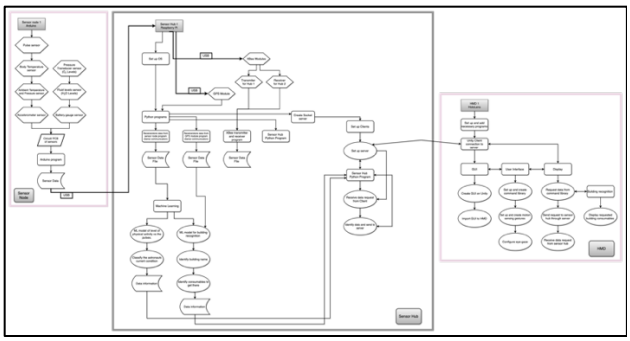


Figure 2: Detailed flow chart diagrams for the sequence of operations

Figures 1 and 2 show the system flow chart diagram and the detailed flow chart diagrams for the sequence of operational steps for our subsystems and system integration that we need to accomplish to execute our project.

V. LIST OF SKILLS AND TOOLS

To design the AR assistant for astronauts to address each of the design and engineering requirements there are skills one or two people in the team already have expertise. The skills that one or no member has expertise in can be found below, we included a list of resources that will be accessed, and the estimated time needed to learn. The team will learn Unity and C# to create a graphical user interface (GUI) to communicate with the HoloLens, Raspberry pi 3 programming to create the sensor hubs, machine learning to create efficient math models, and ZigBee protocols will be studied to create the communication network between the different hubs and the sensor node to the sensor hub.

Rudy
Skills: CAD/Rhino/Fusion360
Skills to learn: Raspberry Pi Plugins (4 weeks), Unity (6 weeks)
Kalvin
Skills: Fusion 360, MATLAB
Skills to learn: Machine Learning (2 weeks- Python), XBee Protocols (2 weeks - YouTube), C# (3 weeks), Unity(6 weeks)
Oussama
Skills: Digital and analog circuit design, MATLAB, LTspice
Skills to learn: Arduino Uno (1 week), CAD (2 weeks), raspberry pi (1 week), Machine learning (3 weeks), Unity (2 weeks), C# (3 weeks)
Maryam
Skills: MATLAB, KiCad (PCB design), Arduino Uno, some experience with python
Skills to learn: Python programming (3 weeks- online course), XBee Protocols (5- hours - online), Unity and C# (3 weeks)

- **Tools:** MS Project, MS Teams, Fusion 360, MATLAB, KiCad, VsCode (Python & C#), Raspberry Pi, Arduino, Unity, HoloLens, XBee modules.

VI. RESOURCES

1. We will need an open space indoors and outdoors to test the HoloLens and sensors.
2. We will need images of UMB buildings to incorporate into our graphical user interface (GUI).

VII. TEAM ROLES

Maryam - Team Lead (Communication Systems and PCB)- She will be overseeing C# and Python Programming. To get the serial communication between the Raspberry Pi and Arduino Nano to receive the sensor data on the hubs. The RF XBee communication between the two sensors hubs and as well as creating the server for the HoloLens communication. In addition to designing the PCB for the sensor node and the HoloLens Interface on Unity for real-time data display.

Oussama - Co-Lead (Sensor node and BMS)- He designs and constructs the sensor node ensuring all sensors are connected on one protoboard and the merged codes are able to accurately display all sensor data on the serial port. In addition to the HoloLens Interface on Unity for real-time data display.

Rudy (CAD file and BMS)- He will oversee creating the power distribution system that will power both the raspberry pi 4 and HoloLens with a singular battery pack and certifying it will be powered for 5-6 hrs and designing the voltage divider and the program to measure battery levels. He is also tasked with creating the CAD file to 3D print the casing necessary for the project to hold all the components.

Kalvin (Machine Learning and Journey logistics)- He will manage the sensor hub ML predictions comparisons to classify the activity with real-time sensor data and indicate the status of the user and have the data ready to be sent to display the status of the user. In addition to

performing the calculations for consumables' depletion information according to the distance; how much 02, H2O, and battery will be needed to get to features/buildings from the current location.

These are just to show who oversees each system and takes the lead. No member will be working on a system independently. Each member will contribute to their part of each system as suggested in the MS project and wherever the system lead or the team believes their skill is needed the most.

VIII. WEEKLY COMMUNICATIONS & DOCUMENT SHARING

The team meets 3 to 4 days a week with Friday being our CM/TM meeting at 10:20 AM. There is a meeting on Monday at 2:00 PM, this is used to reflect on Friday's meeting and what we found from research or what we worked on during the weekend. Our next meeting will be Tuesday and/or Thursday at 5:00 PM depending on the situation and availability, during which we get together, discuss logistics, and work on the project. The documents on SharePoint/MS Teams are all organized into folders that members can easily find and access. With folders dedicated to sensor data, sensor code, previous weekly reports, To Do lists, research links, etc. Everyone can add their findings and data without getting it lost on SharePoint. Our project management correlates with document sharing and group meetings as everyone is contributing in some way. Each member adds research links to the stuff they are assigned to, working on the project at home, doing research and sharing the findings with the team, and performing verification tests for the sensors that are assigned to ensure the accuracy of the sensors.

IX. MS PROJECT UPDATED WITH TIMELINES

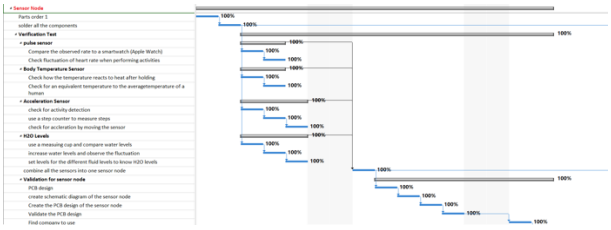


Figure 3: MS project showing the sensor node progress

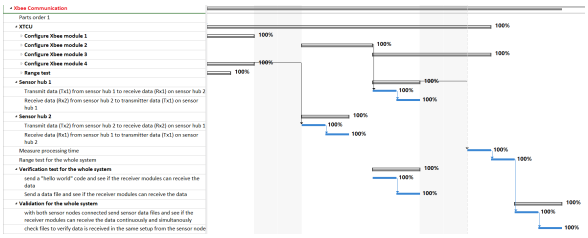


Figure 4: MS project showing the Zigbee communication progress

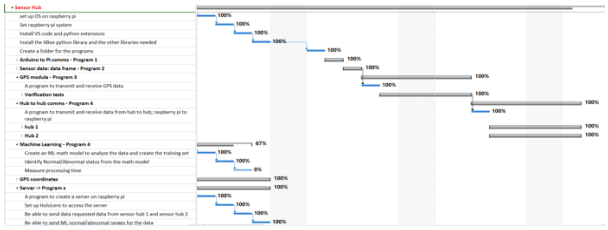


Figure 5: MS project showing the Sensor Hub progress

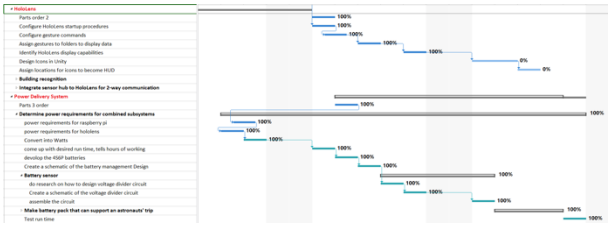


Figure 6: MS project showing the HoloLens and battery management system progress

Figures 3 through 7 show the MS project management used to keep track of our progress throughout the semester. The MS project Gantt chart to the right of the tables on each figure tracks the completion of each main task and each of their subtasks. The main tasks which are highlighted in red in each of the figures are Sensor node, XBee communication, Sensor hub, HoloLens, and Power Delivery System, and each of their subtasks can be seen below. From each figure, each subtask can be seen as a dependency of the task above it. In other words, some tasks such as Machine learning and certain subtasks in the sensor hub got delayed because of configuration issues with the sensor node. As well as issues with configuring the HoloLens hand tracking as an input for our user interface which delayed the HoloLens unity UI creation. However, as can be seen in Figures 1 through 7, most of our main tasks are 100% complete as a result of independent work during the winter break, and most of our focus this semester was working on the power delivery system, finalizing the sensor node as well as the data communication between the sensor hub and sensor

node, as well as putting heavy emphasis on the hub to hub communication and the unity GUI display.

X. CRITICAL TASKS COMPLETION

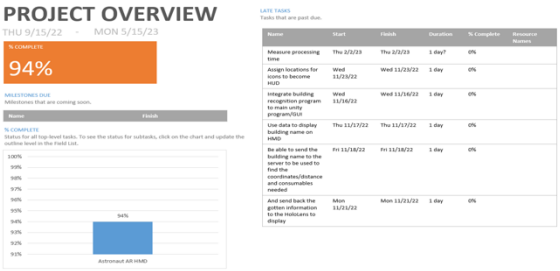


Figure 7: MS project showing the Critical tasks completion

Figure 7 shows the project completion status at 94% as mentioned above, most of our focus was completing the power delivery system, finalizing the sensor node as well as the data communication between the sensor hub and sensor node, and a heavy focus on the Unity UI using the MRTK buttons as HoloLens inputs. We are able to extract data from the sensor node, send it to the sensor hub, and be able to send data from one sensor hub to another, however, the 6% missing from our project is due to the display of data on the HoloLens itself. Due to time constraints, we were not able to finish this part of the project, but most of the concrete tasks and systems have been completed.

XI. PARTS INVENTORY

Qt	H/W	Parts with links	Vendors	Price	Shipping costs
1x	H	HoloLens	Provided to us by the school		
2x	H	Raspberry Pi model 4 (includes wires, batteries, fans)	Amazon	$280	Free Shipping
2x	H	Pulse Sensor	Vetco Electronics	$10	$13.95
2x	H	Temperature and Pressure Sensor	adafruit	$20	Free Shipping
2x	H	Water Tank Gauge	adafruit	$40	Free Shipping
2x	H	Li Batter Fuel Gauge	adafruit	$10	Free Shipping
2x	H	Accelerometer sensor	adafruit	$12	Free Shipping
6x	H	XBee S2C (Zigbee Modules - 802.15.4)	Mouser Electronics	$33	$8.89
2x	H	Adafruit Ultimate GPS Breakout	adafruit	$30	Free Shipping
2x	H	GPS Antenna - External Active Antenna	adafruit	$20	Free Shipping
2x	H	RP-SMA to uFL/u.FL/IPX/IPEX RF Adapter Cable	adafruit	$5	Free Shipping
2x	H	Arduino Nano	Arduino	$25	$4.06
x1	H	Pressure Transducer Sensor (150 Psi)	Amazon	$15	Free Shipping
x2	H	LM35 Analogue Precision Centigrade Temperature Sensor	Amazon	$13	Free Shipping
x4	H	SparkFun XBee Explorer Dongle	SparkFun	$28	Free Shipping
x2	H	SparkFun XBee Explorer Regulated	SparkFun	$12	Free Shipping
x1	H	USB 2.0 Male to Female Extension Plug	Amazon	$8	Free Shipping
x2	H	Battery PCB Protection Board	Amazon	$11	Free Shipping
x2	H	Buck converters	Amazon	$11	Free Shipping
x50	H	18650 Lithium-Ion Battery	18650batterystore	$133	Free Shipping
x1	H	Parallel Nickel-Plated Belt for Batteries	Amazon	$11	Free Shipping
x1	H	Nickel strip for Batteries	Amazon	$6	Free Shipping
x3	H	Lithium-Battery-Plastic-Bracket	Amazon	$13	Free Shipping
x1	H	Resistant-Masking-Soldering-Protecting-Cellphone	Amazon	$10	Free Shipping
x1	H	Boost Charging Module Dual USB 18650 Battery Fast Charger	Amazon	$10	Free Shipping
x1	H	TMP36 Temperature Sensor	Amazon	$15	Free Shipping
x1	W	Sensor Node PCB (Pack of 3)	OSHPARK	$50	Free Shipping
		2 Hubs (One for each HMD)	Total Total x Q	= $898 = $1,605	Budget = $1700
		Parts order 1 = $1102 **Parts order 2** = $177 **Parts order 3** = $276 **PCB** = $50 **Budget** = $1,700 **Budget used** = $1,605			

XII. 5 PROJECT-SPECIFIC SUCCESS CRITERIA (PSSC'S)

The five-project specific success are indicated below:

1) Successful continuous collection of accurate and organized sensor data
2) PCB for Sensor node
3) Machine learning math model from prior sensor data for normal/abnormal prediction and activity detection
4) RF hub-to-hub communications
5) Socket server hub-to-HoloLens communication and Display on HoloLens

The sensor node will be capable of sending continuous and accurate sensor reading to the sensor hub which is then organized and prepared for the use of machine learning. The measurements of H20, pressure, and GPS coordinates are necessary for consumable calculations. The PCB is responsible for keeping all the sensors stable and secured for walks. Machine learning will train and test initial baseline data in which the activity is predicted using body temperature, acceleration, and heartbeat readings. From there the model created will be used to predict the activity of incoming real-time data. The threshold for each activity of each sensor will be established to display the condition of the user. The RF communication modules using the XBee modules allow communication between the two-sensor hub for the exchange of information. Lastly, the socket server created on the sensor hub will be used to communicate with HMD for information to be sent for display on the HoloLens.

XIII. DETAIL DESIGN VERIFICATION AND VALIDATION (V&V)

- **Power Delivery**

The battery management system which will ensure that all our subsystems are functioning properly will be tested for proper voltage and current distributions. We will make sure that every system input receives the correct

input current to guarantee that there are no current overflows. We will evaluate the battery pack's ability to maintain its performance over time, withstand various environmental conditions and usage patterns, and operate safely and reliably. In addition to considering the social and environmental impacts of the battery pack design and production. Our V&V plan for the battery pack as a whole is to ensure that it provides voltage to power the critical devices for up to 6 hours and that the voltage drops accordingly until it ceases to charge which should be around 12V for the battery pack.

- **Successful continuous collection of accurate and organized sensor data**

The sensor hub comprises of 5 sensors measuring the pulse of the astronaut, their body temperature, their acceleration, water levels, oxygen levels, and system battery level. An Arduino Uno was utilized to measure the output of each of the sensors, and later an Arduino Nano will be used for the final build along with a PCB to fit all the sensors for a more portable and secure hold. The Arduino Uno and Nano have a built-in analog-to-digital signal converter which is compatible with all the 5 sensors used in this project.

- To measure the pulse, we used a pulse sensor from Adafruit which was easy to configure and implement. The user simply has to put their finger on the LED located on the pulse sensor, and using the code I had sampled, we are able to read the heart rate at around 60-100bpm depending on who was using it.
- As for the body temperature sensor, a DS18b20 was decided after a failed LM35 sensor that was previously used. The DS18b20 came in a steel casing that is waterproof and will allow the user to measure their body temperature by simply holding the device in their hands. The body temperature was able to output the room temperature at the time at around 76 and after holding the DS18b20 for 5 minutes, I was able to receive a reading of 97-99 which we can conclude was accurate as a human body temperature.
- As for the acceleration of the astronaut, we used an LSM6DSOX which outputs x y z values corresponding to the acceleration of the user. Measurements would increase and decrease according to whether the user was running or walking. For example, an output of x = 1.23, y = 0.9, z = 9.8 m/s^2 would be correspondent to walking, while an output of x = 7.74, y = 1.2, z = 9.8 m/s^2 would be correspondent to running because of a drastic increase in the x value that was a result of the user running fast in a straight line.
- To measure the water levels of the astronaut, an etape liquid level sensor is used it provides a resistive output that varies with the level of the fluid. In the experiment, a 650mL graduated cylinder was used to verify and validate the sensor. After calibration, results showed the output varying from 0-2600 ohms or 0-650mL after conversion which concluded its functionality. To measure the oxygen of the astronaut, a pressure transducer will be utilized with an air compressor to measure the PSI level of the tank. This way we are able to save on our budget and provide a proof of concept that we can measure gas levels.
- The air compressor tank allows us to vary the pressure value being received by the pressure transducer, making it easy to validate it. As we moved the valve from 0-130PSi, we were able to see changes from the serial monitor on the Arduino ranging also from 0-130PSi. After verifying and validating each of the sensors, the code of each sensor was combined into one sketch which was able to result pulse readings, body temperature, acceleration values, water levels, and pressure values in the serial monitor of the Arduino platform. The code used can be seen below.

- **Machine Learning**

The predictions of the activity level from baseline data will give accurate labels with the increased training epochs to ensure that the model is well-trained. The test set is used to validate the accuracy of the model and print out any errors that have appeared during the training. Throughout the continuous training of the baseline data within the raspberry pi we can create a model that is 100%. The thresholds for each activity level will be adjusted accordingly to give a more accurate status reading and ensure that the astronaut has the correct indication. The machine learning algorithm as a whole will predict the right activity of the astronaut and the condition they are in with no biases.

Validation data:

Figure 8: Screenshot showing the predicted label for baseline data along with the printed accuracy

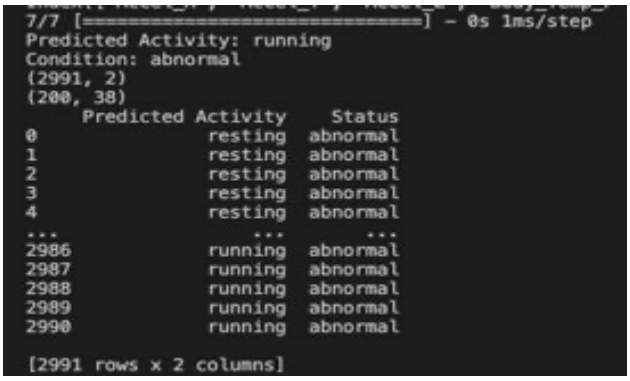


Figure 9: Screenshot showing the predicted activity of the user using the prior model and the prescribed condition of the user

- **RF hub-to-hub communications**

The XBee modules will ensure that our data can be shared between the two hubs. The XBees are set up in a full duplex protocol where data can be transmitted in both directions on a signal carrier at the same time. The performance measurement is to help determine the effective communication range and identify any potential obstacles or interference that could affect the performance of the system. Following the Ethics of fairness, the CSV test will be conducted using the same set of data files for all XBee modules to ensure fairness to all parties involved.

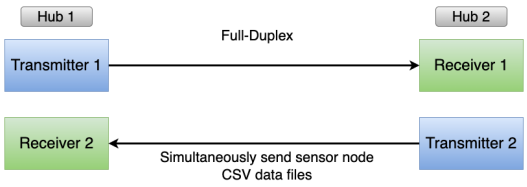


Figure 10: Diagram of how the XBees are set up in a full duplex protocol

XBee Modules Communication Range Test: To verify the RF communication accuracy/reliability between the hubs (XBee modules) by sending and receiving messages indoors and outdoors at different distances. The number of successfully sent and received messages will be observed, as well as the latency, packet loss rate, and signal strength (RSSI) will also be measured. The acceptable accuracy range is 80%. We found that given the XBee specifications of an Indoor range: Up to 60 m (200 ft) and an Outdoor RF line-of-sight range: Up to 1200 m (4000 ft), we were able to get an accuracy of 90% at over 200ft indoors and an accuracy of greater than 80% at almost 400 ft outdoors. This allowed us to understand the effective range of the system and by running tests at different distances both indoors and outdoors, we were able to get a comprehensive picture of the system's performance.

CSV Data Transfer Test: Test the ability of the XBee modules to send and receive CSV data files over a wireless connection. Where with both sensor nodes connected, we send sensor data files and receive the data simultaneously from the receiver modules.

- **Hub-to-HoloLens communication**

The HoloLens must receive the sensor data to display so we designed a Python server and client that can send the data to a Unity C# client that can save the data and display it on the HoloLens.

Socket Server Transmission Test: This is to test the ability of the Python socket server and the client to transmit and receive messages between each other. The successful transmission and reception of messages will be recorded with the test considered successful if messages are transmitted and received without errors. We were able to send simple test messages to the server and clients and receive them.

CSV Data Transmission Test: This is to test the ability of the Python socket server and the client to transmit and receive CSV data files between each other and save the received data to a folder. We were able to successful transmission and reception of CSV data files will be recorded, as well as save the received data with only one hub to a folder. This allows us to send the sensor data CSV files that are generated from the sensors on the sensor node to the HoloLens. The reliability of the data transmission process is a performance measurement we have to keep track of. This can be measured by monitoring the number of data transmission errors or

failures and assessing the overall stability of the transmission process.

- **Head Mounted Display (HMD)**

The head mount display (HMD) is crucial in telling our astronauts crucial information, it will be the difference between life and death in space, so ensuring the functionality of the device is crucial. We will be testing the HMD for an accurate display of the different consumables, testing the functionality of the command library, as well as test the response of the GUI system for an accurate display of feature names and coordinates. All data displayed on the HMD is sent from our sensor's node.

Unity UI Functionality Test: This is to test the Unity user interface (UI) display and programs to ensure they are set up correctly and that the allocated UI elements are functioning properly. This will allow us to display the data on the HoloLens and interact with the interface. We need to verify that the Unity UI elements and scenes are displaying the received data correctly and in the expected locations. We need to ensure that the data transmitted and displayed on the HoloLens interface is accurate and unbiased as it is crucial in telling the astronaut critical lifesaving information.

Unity UI Usability Test: This is to test the usability and functionality of the Unity UI design on the HoloLens emulator. We will look at if the UI elements display data and function as set up. In addition to how easy the UI is for users to learn and use, a low error rate and a high satisfaction rating.

XIV. PROJECT BREAKTHROUGHS AND PROGRESS ON THE 5 PROJECT-SPECIFIC SUCCESS CRITERIA

Power Delivery

So far, we have calculated how much power each of our components needs. The raspberry pi will need a 9V and 2A to function while the HoloLens needs 5V and 3A to function. With that said, we are still deciding on how to approach the BMS. We were essentially thinking of using two battery packs to power both devices, however, we decided it was best to use one battery pack with a buck converter that will allow us to step down the voltage to what is necessary. The buck converter has a step-down voltage range from 1.8V-12V, however, our biggest challenge when designing the BMS is managing the power for a full spacewalk, around 8-9 hours.

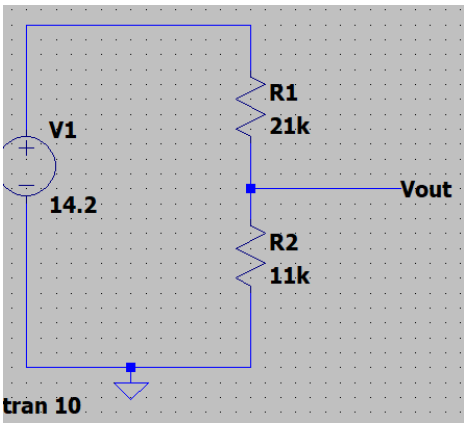


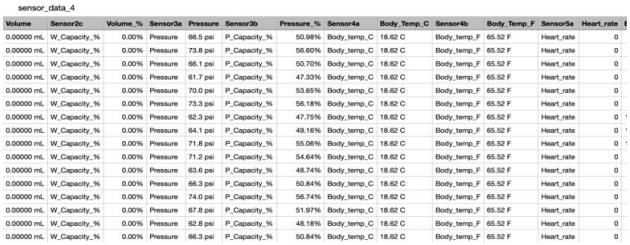
Figure 11: Schematic of the voltage divider to measure our battery's levels on the Arduino

Successful continuous collection of accurate and organized sensor data

After verifying and validating each sensor, the individual code sampled for each sensor was integrated into one sketch which resulted in the output of the 5 sensors mentioned above in the "**S***uccessful continuous collection of accurate and organized sensor data***"** section. Figure 13 shows the individual sensor readings, their names, values, and units. The output was extracted from the Arduino serial monitor and was made into a CSV file. This way it can be separated into columns which can be easier to use for machine learning. It is important to note that not all the values in the figure will appear accurate because nothing was being measured at the time of taking the screenshot. It was simply to show the 5 sensor outputs organized together into one file.

Time	Sensor1a	Amb_Temp	Sensor1b	Accel_X	Sensor1c	Accel_Y	Sensor1d	Accel_Z	Sensor2a	Resistance	Sensor2b
2023-03-25 22:52:49	Amb_Temperature	25.82 C	Accel_X	0.27 m/s^2	Accel_Y	0.29 m/s^2	Accel_Z	9.98 m/s^2	Resistance	56727.69 ohms	Volume
2023-03-25 22:52:49	Amb_Temperature	25.84 C	Accel_X	0.28 m/s^2	Accel_Y	0.30 m/s^2	Accel_Z	9.99 m/s^2	Resistance	56727.69 ohms	Volume
2023-03-25 22:52:50	Amb_Temperature	25.86 C	Accel_X	0.28 m/s^2	Accel_Y	0.29 m/s^2	Accel_Z	9.98 m/s^2	Resistance	56727.69 ohms	Volume
2023-03-25 22:52:51	Amb_Temperature	25.85 C	Accel_X	0.28 m/s^2	Accel_Y	0.29 m/s^2	Accel_Z	9.98 m/s^2	Resistance	56727.69 ohms	Volume
2023-03-25 22:52:52	Amb_Temperature	25.86 C	Accel_X	0.28 m/s^2	Accel_Y	0.29 m/s^2	Accel_Z	9.99 m/s^2	Resistance	63093.42 ohms	Volume
2023-03-25 22:52:52	Amb_Temperature	25.80 C	Accel_X	0.28 m/s^2	Accel_Y	0.30 m/s^2	Accel_Z	9.98 m/s^2	Resistance	56727.69 ohms	Volume
2023-03-25 22:52:53	Amb_Temperature	25.86 C	Accel_X	0.27 m/s^2	Accel_Y	0.27 m/s^2	Accel_Z	9.99 m/s^2	Resistance	56727.69 ohms	Volume
2023-03-25 22:52:54	Amb_Temperature	25.81 C	Accel_X	0.28 m/s^2	Accel_Y	0.29 m/s^2	Accel_Z	9.99 m/s^2	Resistance	63093.42 ohms	Volume
2023-03-25 22:52:54	Amb_Temperature	25.83 C	Accel_X	0.27 m/s^2	Accel_Y	0.29 m/s^2	Accel_Z	9.99 m/s^2	Resistance	63093.42 ohms	Volume
2023-03-25 22:52:55	Amb_Temperature	25.82 C	Accel_X	0.28 m/s^2	Accel_Y	0.29 m/s^2	Accel_Z	9.98 m/s^2	Resistance	56727.69 ohms	Volume
2023-03-25 22:52:56	Amb_Temperature	25.80 C	Accel_X	0.28 m/s^2	Accel_Y	0.28 m/s^2	Accel_Z	9.99 m/s^2	Resistance	56727.69 ohms	Volume
2023-03-25 22:52:56	Amb_Temperature	25.84 C	Accel_X	0.28 m/s^2	Accel_Y	0.28 m/s^2	Accel_Z	9.98 m/s^2	Resistance	63093.42 ohms	Volume
2023-03-25 22:52:57	Amb_Temperature	25.80 C	Accel_X	0.29 m/s^2	Accel_Y	0.30 m/s^2	Accel_Z	9.97 m/s^2	Resistance	63093.42 ohms	Volume
2023-03-25 22:52:58	Amb_Temperature	25.79 C	Accel_X	0.28 m/s^2	Accel_Y	0.28 m/s^2	Accel_Z	9.99 m/s^2	Resistance	56727.69 ohms	Volume
2023-03-25 22:52:59	Amb_Temperature	25.85 C	Accel_X	0.28 m/s^2	Accel_Y	0.28 m/s^2	Accel_Z	9.98 m/s^2	Resistance	56727.69 ohms	Volume
2023-03-25 22:52:59	Amb_Temperature	25.82 C	Accel_X	0.28 m/s^2	Accel_Y	0.28 m/s^2	Accel_Z	9.99 m/s^2	Resistance	63093.42 ohms	Volume

Figure 12: Sensor node data extracted from Arduino into a CSV file showing all individual sensor readings, names, and units



sensor_data_4

Volume	Sensor2c	Volume_%	Sensor3a	Pressure	Sensor3b	Pressure_%	Sensor4a	Body_Temp_C	Sensor4b	Body_Temp_F	Sensor5a	Heart_rate
0.00000 mL	W_Capacity_%	0.00%	Pressure	66.5 psi	P_Capacity_%	50.98%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	73.8 psi	P_Capacity_%	56.60%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	66.1 psi	P_Capacity_%	50.70%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	61.7 psi	P_Capacity_%	47.33%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	70.0 psi	P_Capacity_%	53.65%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	73.3 psi	P_Capacity_%	56.18%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	62.3 psi	P_Capacity_%	47.75%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	64.1 psi	P_Capacity_%	49.16%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	71.8 psi	P_Capacity_%	55.06%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	71.2 psi	P_Capacity_%	54.64%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	63.6 psi	P_Capacity_%	48.74%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	66.3 psi	P_Capacity_%	50.84%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	74.0 psi	P_Capacity_%	56.74%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	67.8 psi	P_Capacity_%	51.97%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	62.8 psi	P_Capacity_%	48.18%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0
0.00000 mL	W_Capacity_%	0.00%	Pressure	66.3 psi	P_Capacity_%	50.84%	Body_temp_C	18.62 C	Body_temp_F	65.52 F	Heart_rate	0

Figure 13: Sensor node data extracted from Arduino into a CSV file showing all individual sensor readings, names, and units

Machine learning math model from prior sensor data for normal/abnormal prediction and activity detection

With the use of neural network programming, we were able to combine 3 datasets for running, walking, and resting. With the combined dataset it is split into training and testing sets which will be used to build the model using Keras sequential consisting of three layers. We were able to successfully print out predicted labels for 3000 data points with 100 percent accuracy. The model is evaluated using the testing set from the baseline data. After we load the live data and remove any unnecessary data to fit into the model such as the other sensor reading that isn't necessary for the activity detection. From there we were able to print out labels for each incoming data point. The thresholds for each sensor were put into place for each activity and are created to determine the status indication with the use of if statements and count. With that, we were also able to print out the status for each line of data as well as save the data to a CSV file.

Journey logistics

The journey logistics will determine the amount of 02, H2O, and battery needed to get to features/buildings from the current location. The distance formula will take in GPS coordinates of the current location of the user and the coordinates for each feature. The closet feature will be printed out as well as the time and consumables needed to make that trip. Machine learning can be integrated into the journey logistics where the status and activity level of the astronaut will alter the number of consumables needed for their trip.

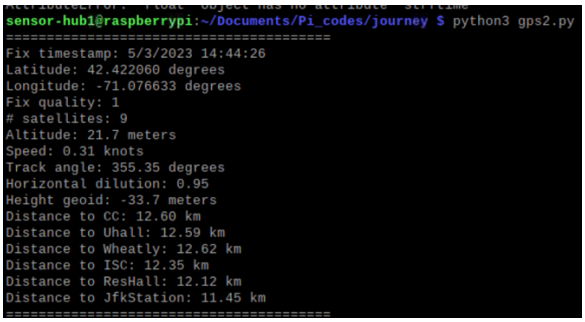


Figure 14: Journey logistics using GPS showing distance to buildings from the current location

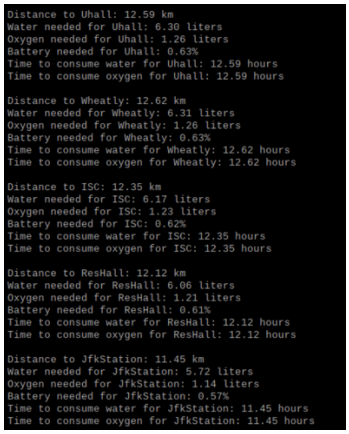


Figure 15: Journey logistics using GPS showing consumables needed to get to buildings from the current location

RF hub-to-hub communications

Two XBee dongles will be used with the XBee module attached and placed on both sensor hubs. This will allow for full duplex communication, in other words, our astronauts will be able to communicate with each other simultaneously. In our case, we want to request any vital information from our body as well as the other astronauts.

Range test

To understand the effective range of the system we performed a range test for the XBee S2C modules with Zigbee firmware and ensured that it is working as expected. Each test provided information about the system's performance at different distances and can help identify any potential issues or limitations in the system. In our range test, we looked at several metrics, including latency, accuracy, packets sent, Tx error, packets received, packets lost, the Rx timeout, Tx interval, and RSSI. By running tests at different distances both indoors and outdoors, we were able to get a comprehensive picture of the system's performance and identify any

potential issues that may arise in different environments. While we had limitations on measuring and stayed within the 0ft to over 200 ft range, we expected to see a gradual decrease in performance as the distance increased, with a higher likelihood of packet loss and lower RSSI values at longer distances. A lower RSSI value may indicate issues with signal strength or interference. RSSI is a measurement of how well your device can hear a signal from an access point or router. It's a value that is useful for determining if you have enough signal to get a good wireless connection. the closer to 0 dBm, the better the signal is.

The indoor range test on the XBee modules showed that the XBee S2C modules with ZigBee firmware are capable of transmitting data accurately up to the 231ft range with an accuracy of over 80% with latency increasing. This indicates that the signal propagation delay increased with distance, leading to a longer latency. However, the latency values were still relatively low throughout the range test, suggesting that the XBee S2C modules with Zigbee firmware can maintain a fast and reliable wireless connection over long distances. We found that with an increase in packets, the latency increased. At a distance of 0ft, the latency was 6.9min, to send 300 packets at a distance of 0ft. However, beyond that range, there are more errors and packet losses, indicating that the signal strength is getting weaker. The test also revealed that the RSSI decreases as the distance increases, which is typical behavior in wireless communication systems.

Hub 1- Tx1_xbee1 -> Rx1_xbee2

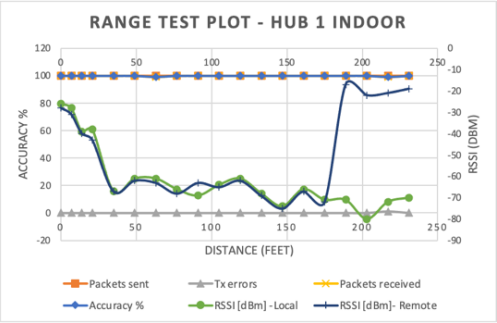


Figure 16: Plot for indoor range test for hub 1 (Tx1_xbee1 -> Rx1_xbee2) showing accuracy, packets sent, Tx error, packets received, packets lost, and RSSI

Hub 2- Tx2_xbee3 -> Rx2_xbee4

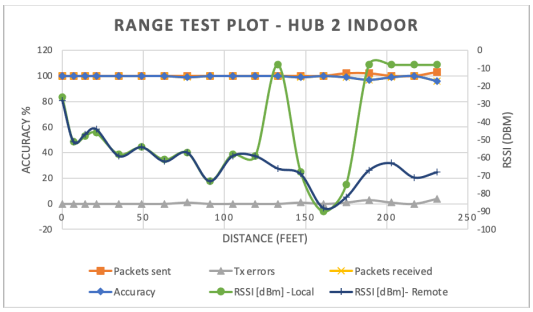


Figure 17: Plot for indoor range test for hub 2 (Tx2_xbee3 -> Rx2_xbee4) showing accuracy, packets sent, Tx error, packets received, packets lost, and RSSI

The outdoor test measured the accuracy of a wireless communication system at different distances. At closer distances (0-64 ft), the accuracy was consistently high (above 99%), with minimal latency and no packets lost. However, at distances beyond 79 ft, the accuracy started to drop, with some packets lost and higher latency. The drop in accuracy could be attributed to signal interference or attenuation caused by obstacles or other wireless devices operating on the same frequency. The data suggests that the wireless communication system's performance is affected by distance and environmental factors and that further testing and optimization may be necessary to improve its accuracy and reliability over longer distances and at different locations.

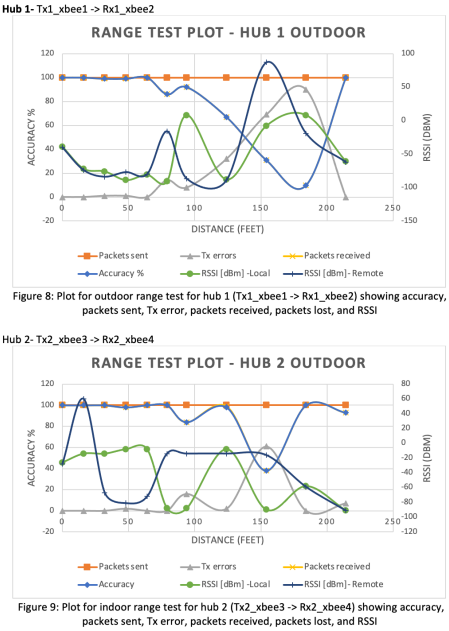


Figure 18: Plot for outdoor range test for the hubs showing accuracy, packets sent, Tx error, packets received, packets lost, and RSSI

We found that the system generally performed well at shorter distances, with a high level of accuracy, minimal packet loss, and good RSSI values. However, as the distance increases, we see a decrease in accuracy and an increase in packet loss, indicating that the system may struggle to maintain reliable communication over longer distances. The furthest we measured was from the University Hall entrance to the Campus Center lawn at ~400 feet. The system showed a latency of 2.2 minutes and an accuracy of 92% and 82%. While no packets were lost, there were 8 transmission errors. The system's RSSI values indicated weaker signal strength at the remote end, possibly due to distance or obstacles. The results suggest that the system can transmit data over a long distance, but with reduced accuracy and increased latency, which could be improved with further testing and optimization.

Hub-to-hub communication

With both sensor nodes connected, we were able to send sensor data files and receive the data simultaneously from the receiver modules continuously and save it to a CSV file. The code sets up two XBee devices to transmit and receive data over a wireless network. It opens serial ports for communication and defines two node identifiers for the transmitter and receiver XBee devices. The code discovers the remote devices, sets up data callbacks, and reads a CSV file, sending it in chunks to the receiver XBee device. The code waits for three minutes before sending the next file. The code also sets up a data received callback to receive data from the remote device and writes it to a CSV file in chunks. The code creates a new CSV file after every 200 chunks of data received. Finally, the main function is called to execute the code.

Hub 1

- Transmit data (Tx1) from sensor hub 1 to receive data (Rx1) on sensor hub 2
- Receive data (Rx2) from sensor hub 2 transmitter (Tx2) on sensor hub 1

Hub 2

- Transmit data (Tx2) from sensor hub 2 to receive data (Rx2) on sensor hub 1
- Receive data (Rx1) from sensor hub 1 transmitter (Tx1) on sensor hub 2

Hub-to-HoloLens communication

We create a server on the raspberry pi that our HoloLens can communicate with through a Unity Client.

Python Client: The code defines a SensorClient class that establishes a connection to a server using a socket. It has two methods to get the filenames of two CSV files to be sent to the server. The code sends each file in chunks to the server using the sendall method of the socket. The filename, length of the filename, and data of the file are combined into a single message and sent to the server. The code prints the number of bytes sent for each file and a success or error message. The code continuously sends the files to the server every 180 seconds.

Python Server: This code implements a Python server that listens for incoming connections from Python clients, it receives files from the Python client and saves them to specific directories based on the filename prefix (Hub1 or Hub2). It creates a server socket and listens for incoming connections from clients. Once a client connects, it creates a new thread to handle that client's requests. When a client sends a file, the server receives the filename, data length, and file data. It then saves the file to the appropriate directory with a file lock. Finally, it sends the file data to all connected clients using a broadcast method.

C# Client: This is a C# script for a Unity client that connects to the server at a specified IP address and port, receives files in binary format from the server, and saves them as CSV files in specified directories. The script creates two directories to save CSV files, one for each hub. It then starts a new thread to handle the client connection and receives data from the server in a loop. The received data includes the file name, the file length, and the file content, which are read from the network stream using the BinaryReader class. The script then checks the file prefix to determine which folder to save the file to, combines the folder path with the file name, and saves the file using the File.WriteAllBytes method.

We were able to successfully accomplish the task of transmitting sensor data CSV files from the Python client to the Server and then to the Unity C# client every 3 minutes, while also generating a new file after every 200 sets of data read from the sensor node. This interval is precisely the time required to read 200 sets of data from the sensor node, and a new file is generated every 200 sets of data. The Unity Client can receive data and save the Hub 1 and Hub 2 data files appropriately in Hub1 or Hub 2 folders to then be used to display on the interface.

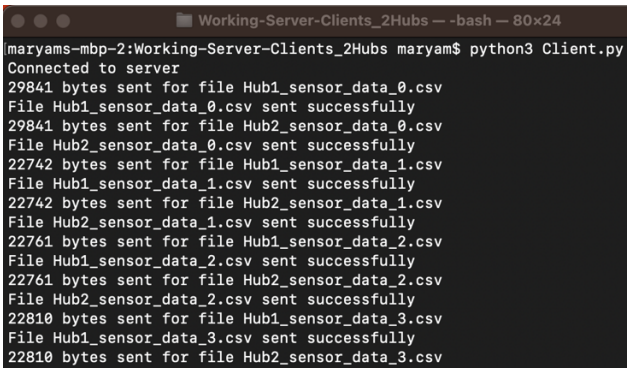


Figure 19: Python Client accurately sending the two hubs csv files

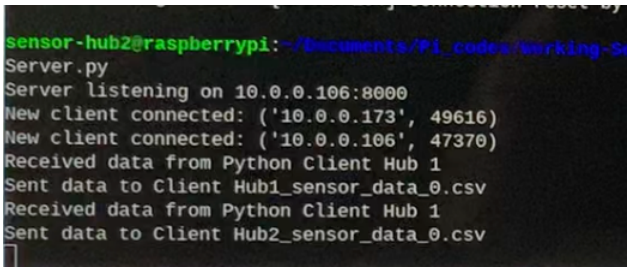


Figure 20: Python Server accurately saving and sending the data files to the Unity Client

Display on HoloLens

The HoloLens UI on Unity is in progress. We were able to set up Unity with the Microsoft mixed reality toolkit (MRTK) as it provides a cross-platform input system that allows for HoloLens hand tracking and interactive functions. We added a dropdown menu and buttons that display CSV data according to which label you picked and use a button to switch scenes and check on the other astronaut and see his data. We were able to get it functional with the MRTK and use hand gestures to make it an interactive User Interface and click on the buttons using the hand tool. The system starts out in scene 1; astronauts 1, we can click the drop-down menu using the cursor on the different options and display the assigned data from the CSV file on the vital stats text box. Next to it we have the 5 buttons, and it also can display the assigned data when the button is pushed to the appropriate text box. The check on Hal button takes us to a new scene to check on astronaut 2. The interface works just like the first scene, and we can use the check on Mer to get back to the original astronaut's scene. Due to time constraints, we were not able to add the journey logistics information, finish integrating all the subsystems and implement the interface on the HoloLens.

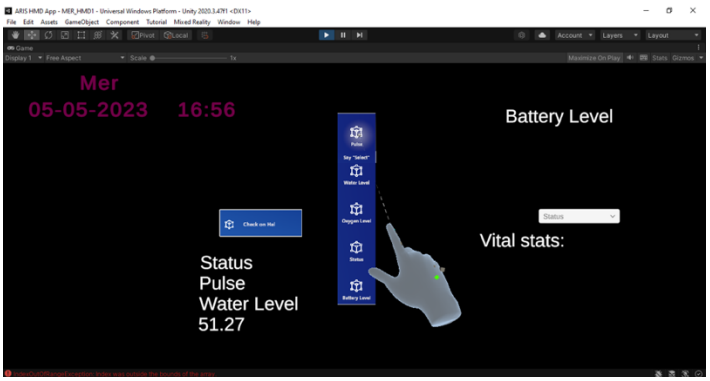


Figure 21: In progress HoloLens UI

XV. PCB FILE READY FOR PRODUCTION

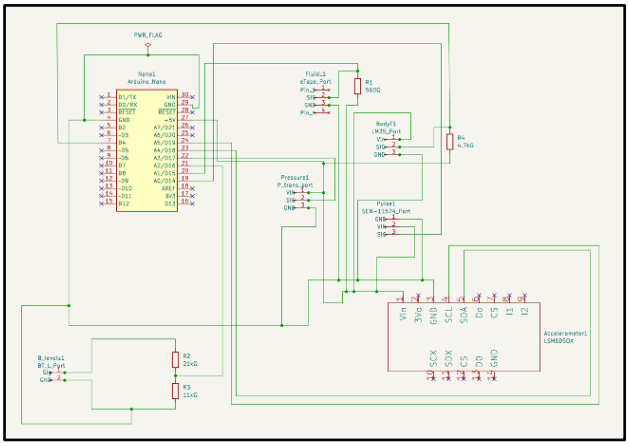


Figure 22: Schematic of the Sensor node PCB

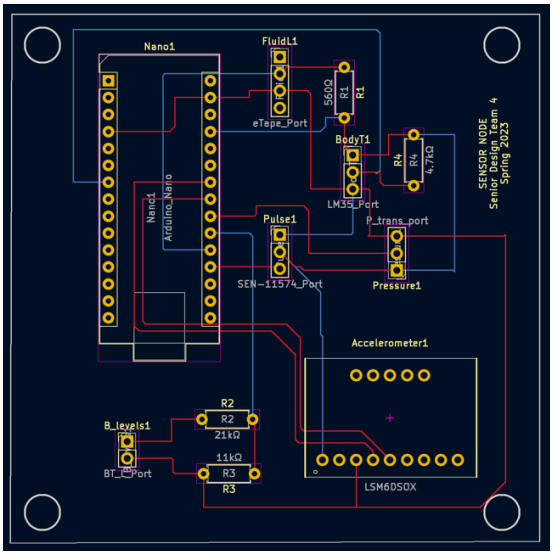


Figure 23: Sensor Node PCB ready for production

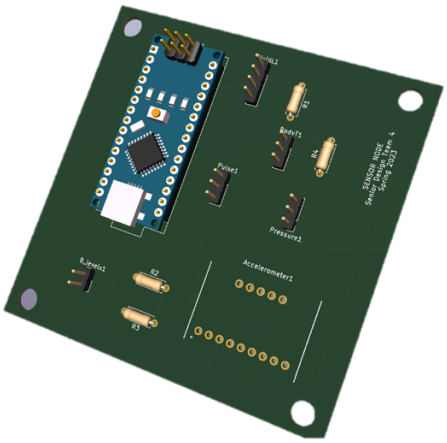


Figure 24: 3D view of Sensor Node PCB

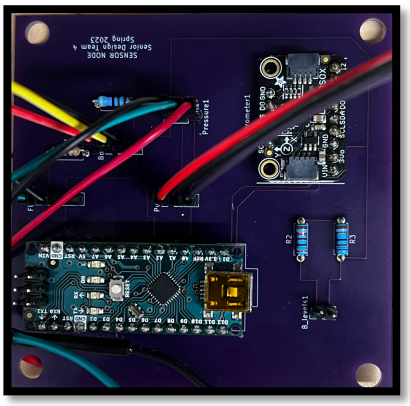


Figure 25: Printed Sensor Node PCB

The reason we created the Sensor node PCB is to create a compact and portable sensor node for measuring the vital signs of astronauts. It is an integration of the various sensors for the project; the Pressure, fluid levels, battery, pulse, and body temperature using female header connector ports as some need to be on the body of the astronaut or on the consumable they need to measure. The accelerometer and Arduino Nano are through-hole components on the board. The Sensor node PCB will measure the pulse, body temperature, acceleration, water levels, oxygen levels, and battery levels of the astronaut. It will perform real-time data collection, processing, and transmission to the raspberry pi for analysis. We will focus on the performance, accuracy, and reliability of the sensors. In addition to optimization through high-quality components, proper assembly, and appropriate software.

XVI. V&V PLAN FOR THE TEAM PCB

Our V&V plan for the Sensor Node PCB is to carefully solder the through-hole components to the printed circuit board (PCB) and make the wire connections for the other components and to the Arduino. We will re-perform all the past sensor validation and verification tests we previously did for the sensors on a breadboard refer to section XIII to confirm the functionality of our PCB.

XVII. CAD FILES FOR COMPONENTS CASING

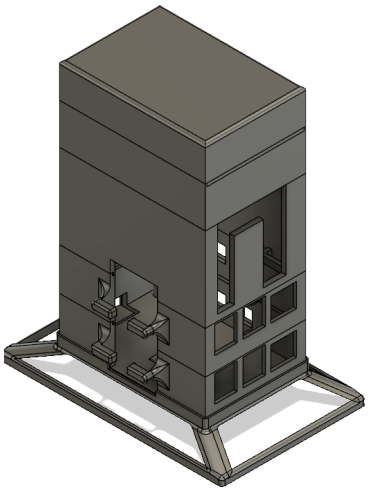


Figure 26: CAD casing image for components

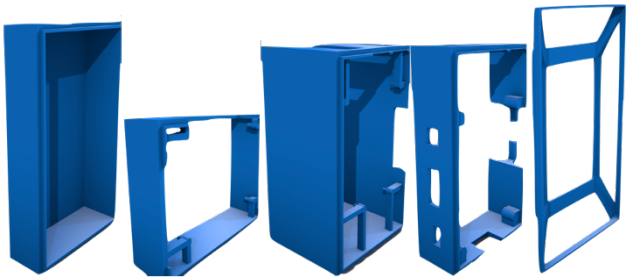


Figure 27: Broken down CAD casing image for components

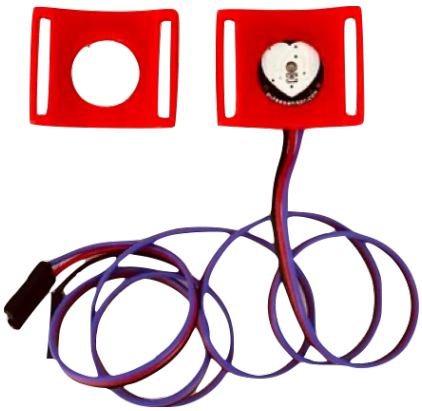


Figure 28: Pulse Sensor Holder

- The CAD file casing provides a protective and secure housing for the sensor node and ensures that the sensors are positioned accurately to provide accurate readings. The casing is designed to be lightweight, durable, and easy to attach the sensors to the astronaut's body or tank.
- The ring provides a way to obtain better readings from the user's finger and ensures accurate monitoring of pulse rates.

XVIII. PROJECT DELIVERABLES

The deliverables that will be presented by the end of the project are a Unity GUI system, a display, a battery management system, communication between the sensor hub and sensor node, communication between the sensor hub to HoloLens, communication between sensor hub to sensor hub, and calculations of consumables and activity level.

- **Sensor node:** An Arduino with a PCB of the connected sensors and an Arduino program that receives all the sensor data simultaneously and organizes the data ready for use by the raspberry pi machine learning program
- **Sensor Hub:** A raspberry pi with the Arduino connected using USB serial communication which allows the Pi to receive the sensor data using a Python program and save the data to a file. We use the data on the file to create an ML math model that can determine if the variation in the level of physical activity and the pulses are normal or abnormal changes. The Hub should be able to give the distance and consumables; how much 02, H2O, and battery are needed to get to features/buildings from the current location. In addition to a server to handle the transfer of information from the sensor hub to HoloLens.
- **XBee modules**: To transfer sensor data from one sensor hub to another using Zigbee protocol.
- **HMD:** A HoloLens capable of receiving and displaying sensor data of the astronaut. A unity GUI of the systems display and a journey logistics program on the pi that is integrated into the GUI to provide the consumables needed to get to different locations.
- **BMS:** A battery management system capable of powering the raspberry pi and HoloLens for 8 hours and a power distribution PCB with ports to connect the Pi and HoloLens cables.

These five items need to all work in conjunction with one another for the system to function properly. The GUI system will be used to control the astronaut's eye and hand movement which will act as an input signal when communicating with the sensor hub, allowing the user to pick which consumables they want to be displayed, or which building they want to be featured. Similarly, having successful communication between the sensor hub and sensor nodes will ensure that the correct data is being displayed, as well as ensuring the right consumable and activity level value. The battery management system will ensure that subsystems are all powered efficiently.

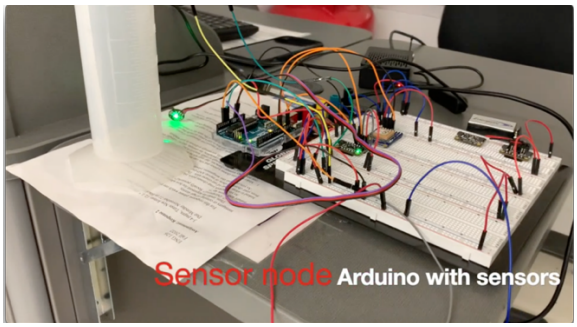


Figure 29: Sensor Node

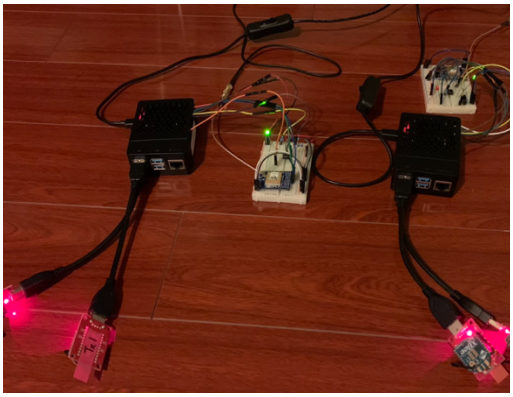


Figure 30: Sensor Hubs with XBee module

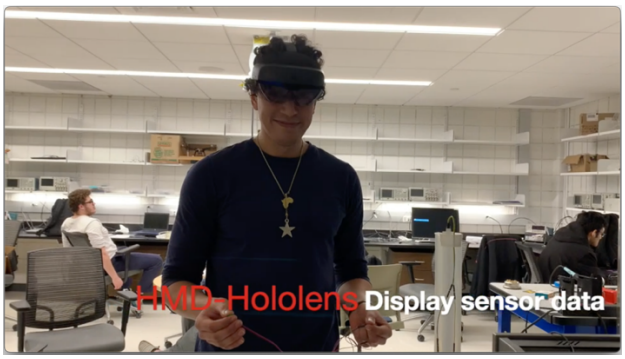


Figure 31: HoloLens HMD

XIX. FINAL EVALUATION OF EACH OF YOUR FIVE PSSC'S

- **Power Delivery**
  The system's power requirements were met by providing a 5V 2.5A power supply for the Raspberry Pi and a 9V 2.0A power supply for the HoloLens. The combined power needed was 14.2V 4.5A (or 31W), and a battery designed for 6 hours of operation was selected, providing 186Wh (or 13.1Ah). To fulfill this, 18650 batteries with a rating of 3.6V 2.5A were used in a 4S (14.2V/3.6V in series) and 6P (13.1A/2.5A in parallel) configuration. Additionally, a Battery Management System was added to ensure the safety and health of the battery cells.

- **Successful continuous collection of sensor data**
  - The pulse was able to output a reading of 60-100bpm depending on who was wearing it, and there was a relationship between the pulse and activity level such that as activity level increases, the output of the pulse also increases.
  - The body temperature sensor provided readings of user temperature of 97-99 which is the average human body temperature.
  - LSM6DSOX outputs x y z values corresponding to the acceleration of the user. Measurements would increase and decrease according to whether the user was running or walking. For example, an output of x = 1.23, y = 0.9, z = 9.8 m/s^2 would be correspondent to walking, while an output of x = 7.74, y = 1.2, z = 9.8 m/s^2 would be correspondent to running because of a drastic increase in the x value that was a result of the user running fast in a straight line.
  - For water level testing, results showed the output varying from 0-2600 ohms or 0-650mL after conversion which concluded its functionality.
  - The pressure test provided readings of 0-130PSI depending on how much was in the 130PSI tank.

- **Sensor node PCB**
  The sensor hub is the center point for all the sensors, and it allows for the continuous collection of accurate sensor data from the sensor node using serial communication. It provides readings of pulse, body temperature, water levels, activity levels, pressure levels, and even battery levels as well. The figure above shows the complete CSV file provided by the sensor node showing all the necessary vitals and suit status parameters.

Time	Sensor1a	Amb_Temp	Sensor1b	Accel_X	Sensor1c	Accel_Y	Sensor1d	Accel_Z	Sensor2a	Resistance	Sensor2b	Volume	Sensor2c	Volume_%
2023-03-25 22:52:49	Amb_Temperature	25.82 C	Accel_X	0.27 m/s^2	Accel_Y	0.29 m/s^2	Accel_Z	9.98 m/s^2	Resistance	56727.69 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:49	Amb_Temperature	25.84 C	Accel_X	0.28 m/s^2	Accel_Y	0.30 m/s^2	Accel_Z	9.99 m/s^2	Resistance	56727.69 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:50	Amb_Temperature	25.86 C	Accel_X	0.28 m/s^2	Accel_Y	0.29 m/s^2	Accel_Z	9.98 m/s^2	Resistance	56727.69 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:51	Amb_Temperature	25.85 C	Accel_X	0.28 m/s^2	Accel_Y	0.29 m/s^2	Accel_Z	9.98 m/s^2	Resistance	56727.69 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:52	Amb_Temperature	25.86 C	Accel_X	0.28 m/s^2	Accel_Y	0.29 m/s^2	Accel_Z	9.99 m/s^2	Resistance	63093.42 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:52	Amb_Temperature	25.80 C	Accel_X	0.28 m/s^2	Accel_Y	0.30 m/s^2	Accel_Z	9.98 m/s^2	Resistance	56727.69 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:53	Amb_Temperature	25.86 C	Accel_X	0.27 m/s^2	Accel_Y	0.27 m/s^2	Accel_Z	9.99 m/s^2	Resistance	56727.69 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:54	Amb_Temperature	25.81 C	Accel_X	0.28 m/s^2	Accel_Y	0.29 m/s^2	Accel_Z	9.99 m/s^2	Resistance	63093.42 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:54	Amb_Temperature	25.83 C	Accel_X	0.27 m/s^2	Accel_Y	0.29 m/s^2	Accel_Z	9.99 m/s^2	Resistance	63093.42 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:55	Amb_Temperature	25.82 C	Accel_X	0.28 m/s^2	Accel_Y	0.29 m/s^2	Accel_Z	9.98 m/s^2	Resistance	56727.69 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:56	Amb_Temperature	25.80 C	Accel_X	0.28 m/s^2	Accel_Y	0.28 m/s^2	Accel_Z	9.99 m/s^2	Resistance	56727.69 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:56	Amb_Temperature	25.84 C	Accel_X	0.28 m/s^2	Accel_Y	0.28 m/s^2	Accel_Z	9.98 m/s^2	Resistance	63093.42 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:57	Amb_Temperature	25.80 C	Accel_X	0.29 m/s^2	Accel_Y	0.30 m/s^2	Accel_Z	9.97 m/s^2	Resistance	63093.42 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:58	Amb_Temperature	25.79 C	Accel_X	0.28 m/s^2	Accel_Y	0.28 m/s^2	Accel_Z	9.99 m/s^2	Resistance	56727.69 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:59	Amb_Temperature	25.85 C	Accel_X	0.28 m/s^2	Accel_Y	0.28 m/s^2	Accel_Z	9.98 m/s^2	Resistance	56727.69 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%
2023-03-25 22:52:59	Amb_Temperature	25.82 C	Accel_X	0.28 m/s^2	Accel_Y	0.28 m/s^2	Accel_Z	9.99 m/s^2	Resistance	63093.42 ohms	Volume	0.00000 mL	W_Capacity_%	0.00%

Figure 32: Part of the Hub CSV file showing sensor node data

- **Machine Learning**
  The machine learning model was fully capable of training the baseline data and predicting that activity for each line of data. The code is also able to successfully able to receive the real-time sensor being uploaded into it and use the prior math model to detect the activity of the user. In the thresholds section of the code, each sensor data was taken into consideration. Each threshold labeled the status for each sensor data. And the status that is most prominent determined the user's condition.

- **Journey Logistics**
  Journey logistics successfully calculates the distance from the astronaut and all buildings on the campus as well as the time to make the trip. The closet feature to the user will be printed out on the terminal. The consumables necessary to make the walk have also been successfully accomplished. For future improvements on the journey logistics would be fully integrating machine learning into it such as importing the status of the user into code and altering

the equations to where condition and activity is considered for the consumable depletion.

- **RF Hub-to-Hub**
  The RF Hub-to-Hub system is fully operational, enabling seamless communication between astronauts. This allows us to transmit and receive hub data files encompassing vital signs collected from the sensor node, condition, and status ML predictions, as well as the required consumables for various locations.

- **Server Hub-to-HoloLens**
  The Server Hub-to-HoloLens task has been successfully completed, achieving our objectives in transmitting Hub 1 and 2 data CSV files from the Python client to the Server and subsequently to the Unity C# client. The implementation includes generating new files after processing every 200 sets of data, allowing for efficient data organization. The Unity Client has undergone significant improvements, enabling the reception and proper saving of Hub 1 and Hub 2 data files into designated folders. This ensures easy accessibility and display of the data on the interface.

- **HoloLens Display**
  The HoloLens display interface has made notable advancements. We successfully set up Unity with the Microsoft Mixed Reality Toolkit (MRTK), enabling HoloLens hand tracking and interactive capabilities. The interface includes dropdown menus, buttons, and scene switching, allowing for the selection and display of CSV data for different astronauts. While not all subsystems and features could be fully integrated due to time constraints, the progress made showcases a functional and interactive user interface. Further enhancements and the implementation on the HoloLens device can be pursued in the future.

XX. FUTURE WORK

After completing the initial project deliverables, there may be several potential areas of future work to consider. Here are a few possibilities:

1. **Further development of the machine learning model:** While the initial ML model may be able to detect normal and abnormal changes in physical activity and pulses, it may be possible to improve the model's accuracy by adding more data or optimizing the model architecture.
2. **Expansion to other environments:** The project initially targets monitoring astronauts in space, but it has the potential for expansion to include monitoring workers in other hazardous environments. The system's modularity plays a crucial role in achieving this adaptability, as it allows for effortless customization based on specific environment requirements. Removing components like the water level indicator for non-water-carrying workers becomes a simple task, while incorporating additional functionalities such as detecting ambient temperature or gases can be achieved by plugging in compatible Arduino sensors and updating the code accordingly. This flexibility makes the system approachable for diverse monitoring needs across different hazardous work settings.
3. **Integration with other systems:** The current design is strictly monitoring the wearer’s health status and any consumables they might be carrying, and the data can be transmitted to a single partner. In the future, it could be possible to turn this into a mesh network of many wearers sharing information with each other. Also, different missions might want to know the status of the equipment they are working on or using, the fuel levels of their vessel, or the velocity of their target are some parameters that come to mind. Each device that would be desired to be monitored would need a sensor hub and sensors of their own.
4. **Advanced data visualization:** While the current system displays sensor data to the astronaut using a simple Interface, more advanced and organized data visualization techniques could be employed to make it easier for the astronaut to interpret the data. Unity is a very robust tool we were just barely scratching the surface of its capabilities as the project was coming to an end. Future teams might be able to utilize the HoloLens’ object detection and program some specific images into Unity to reference some known values of buildings or objects around the wearer’s environment
5. **Integration with other devices:** The current project involves a sensor hub, sensor nodes, and HoloLens, but there may be other devices that could be integrated with the system to improve functionality. For example, a smartwatch could be used to display

additional information to the astronaut, or a drone could be used to collect environmental data.

XXI. INTERNATIONAL STANDARDS AND ETHICAL CONSIDERATIONS

To address the ethical considerations in our design and safety we will address the calibration of our sensors to correlate with the expected levels, i.e., the average resting heart rate (between 60 and 100 beats a minute) and body temperature of a human (98.6 F (37 C)), how much oxygen and water is needed for a duration of time. In addition to that, we need to ensure the machine learning program has the correct threshold to determine if the variation of the level of physical activity and the pulses are normal or abnormal changes. As we are monitoring life support systems, we need to give importance to our calibration and validation tests of the design.

- Engineers shall hold paramount the safety, health, and welfare of the public. Taking into consideration the production of the design and the choice of materials.
- Giving importance to calibration and validation tests as the design will be monitoring life support systems.
- Conformity with applicable standards when using HoloLens and communication protocols.
- IEC 62133-2:2017: Specifies the requirements and tests for the safe operation of lithium-ion cells and batteries used in portable applications. This standard includes tests for electrical, mechanical, and environmental safety, as well as performance and abuse testing.
- IEEE 1296-1998: A method for evaluating the usability of a product by systematically walking through a task from a user's perspective. To ensure the Unity UI design is user-friendly and efficient, we will implement the Cognitive Walkthrough method.

XXII. COLLABORATIVE EFFORTS WITH OTHER TEAMS

To execute the project to the best design we look to our peers that have experience in the area or are working on something similar for their Senior Project. When working on the machine learning portion of our project Team 8 was valuable and gave us advice on how to go about analyzing sensor data as they had run into the same issue. They told us to use neural networks for sensor data machine learning due to the data being nonlinear. In addition to them, we collaborated with team 1 as they are working with XBee modules for their project like us. They sought our input on how to fix the XTCU software as they ran into an issue but as we hadn't reached that part of the project, we gave them helpful debugging techniques to help fix it, such as redownloading the software, trying another team members computer to see if the issue persists and so on. They also gave us useful information for when we start working with the XBee modules. They said to set up all the modules we plan to communicate with one another at the same time as they had previously run into an issue where the modules wouldn't recognize each other and fixed it by doing that. Team 6 was very helpful as we were building our battery and shared resources to help us understand and gave us tips on what to expect and look out for like when we had the wrong BMS; a 3 series as opposed to a 4 series which is what we were working with they were helpful in notifying us as we had missed that.

XXIII. ACKNOWLEDGEMENT

We would like to acknowledge Professor Tomas Materdey our CM/TM for helping us understand our project requirements and asking the difficult questions to help us move the project forward. We would like to acknowledge Andrew Davis; the director of Technical Operations for getting us our parts and answering any questions we have. Last but not least we acknowledge our fellow peers and faculty for all the help they have provided for us as we work on the project.

REFERENCES

[1] Wild, F. (ed.) (2020) *What is a spacewalk?*, *NASA*. NASA. Available at: https://www.nasa.gov/audience/forstudents/k-4/stories/nasa-knows/what-is-a-spacewalk-k4.html

[2] Canadian Space Agency (2021) Bio-monitor: Keeping an eye on astronauts' vital signs, Canadian Space Agency. / Gouvernement du Canada. Available at: https://www.asc-csa.gc.ca/eng/sciences/bio- monitor.asp.

[3] Garcia, M. (ed.) (2022) Innovative 3D telemedicine to help keep astronauts healthy, NASA. NASA. Available at: https://www.nasa.gov/feature/innovative-3d-telemedicine-to-help-keep-astronauts-healthy.

[4] Ed. Raspberry Pi Arduino Serial Communication - Everything You Need to Know. The Robotics Back-

End. Available at: https://roboticsbackend.com/raspberry-pi- arduino-serial-communication/.

[5] "WPA2-Enterprise Configuration & Certificate Issuance on Raspberry Pi." SecureW2. Available at: https://www.securew2.com/solutions/wpa2-enterprise-certificates-raspberry-pi.

[6] Jain, Rishabh. "How to Interface XBee Module with Raspberry Pi." Circuit Digest. Available at: https://circuitdigest.com/microcontroller-projects/raspberry-pi-xbee-module-interfacing.

[7] Dayani, Peru. "Connecting Hololens: Introduction." Medium, Medium. Available at: https://medium.com/@perudayani/connecting-hololens-introduction-618d66c60d12.

[8] "Unity Development for Hololens - Mixed Reality." Mixed Reality | Microsoft Learn. Available at: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/unity-development-overview?tabs=arr%2CD365%2Chl2.

[9] [9] "How to Build an 18650 Lithium Battery Pack with BMS." YouTube, Wes That Tech Guy. Available at: https://www.youtube.com/watch?v=NGp7cGBYOLc.

GitHub project page:
https://github.com/mamukhtar/Project-ARIS.git

Appendix

Table 1: Operation manual with troubleshooting table

System	Manual	Code to run	Troubleshooting
Power Delivery	The Battery connects to two buck converters which should be set to 5V for the one to connect to the raspberry Pi and 9V for the one to connect to the HoloLens. For charging disconnect the buck converters and connect the barrel switch to the charger we provided or a charger that outputs 18V.	No code is needed to run. There is some code in the Sensor node to read battery life	If Raspberry Pi fails to power on ensure the buck converter is set to 5.0V-5.2V. If HoloLens fails to charge, ensure its buck converter is set to 9.0V-9.2V. If the buck converters do not seem to be powered check connections and discontinue use until loose connections are fixed
Sensor node	The pulse sensor is placed on the finger of the user, the body temperature is placed on the armpit of the user, the water sensor can be placed in any tank, but remember to calibrate the code based on the tank volume, and the same can be said for the pressure transducer that was attached to our pressure tank. The acceleration sensor will be placed on the sensor node itself which will show movement according to the user. The sensor node utilizes an Arduino nano which collects and exports data out to our raspberry pie, the data will be collected continuously so no need for any input trigger.	Pulse_Sensor.ino BodyTemp_Sensor.ino Acceleration_Sensor.ino Water_Sensor.ino Pressure.ino Battery_gauge.ino ALL_sensors_wthBattery.ino	If the sensor node is not providing the correct output, ensure that the code has no errors, and the syntax is correct especially when printing a value. This should be done for each sensor to verify where the error lies. If there are no issues with the code, ensure the wiring for each sensor is correct
Sensor node PCB	The sensor node PCB is a more secure and modular design than the "sensor node". It acts as the center point for all the sensors used for the design. It has designated slots for each sensor so ensure that wiring is correct, and place the sensors similar to what was described on "sensor node"	Same code as "Sensor node"	Ensure Each sensor is placed in its dedicated slot and ensure that there are no syntax errors in the code.
Machine Learning	The machine learning will take place in the raspberry pi. The baseline data must be collected using the heart rate, accelerometer, and body temp sensors. The baseline data must be stored within a folder to be called upon in the program.	Python3 AR_Assistant_ML.py	If any errors were to appear while running the code, check if there are no NaN values in the baseline data and real-time data. Drop any unnecessary columns for the neural networks.
Journey Logistics	The GPS modules must be connected to sensor hubs 1 and 2. The push button and GPS module must be wired correctly to the raspberry pi.	Python3 gps2.py	Be in an area where you can have a good signal as the GPS module retrieves the satellite's signal to get a fix to determine the coordinates.
RF Hub-to-Hub	Connect XBee Tx1 and Rx2 on Hub 1 and Tx2 and Rx1 on Hub 2. Check the ports and run	Python3 hub1tohub2.py or	To ensure you are using the right USB ports for the XBees on the pi; run "ls /dev | grep ttyUSB" with one

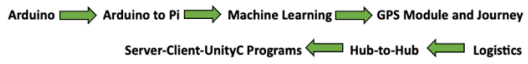
	hub1tohub2 on Hub1 and hub2tohub1 on Hub2 to start transmitting and receiving	Python3 hub2tohub1.py	XBee connected then connect the second and run it again and update the port names.
Server Hub-to-HoloLens	Ensure you are connected to the internet first. Run server.py to start the server and start listening for clients. In the unity client, ensure you are using the server's current IP address. Run the unity client i.e. be on play mode to connect to the server and start listening for broadcasts from the server. Then run the python client to start sending Hub1 and Hub 2 CSV data files to the server, the server then broadcasts it to the listening clients, and the unity client receives it and saves it to the designated folders	Python3 Server.py Python3 Client.py	Ensure you are using the server's current IP address in the unity client
HoloLens Display	Run the Unity project in play mode. It starts out in scene 1, you may need to move backwards as the canvas isn't centered with the buttons. You can click on the drop-down menu using the cursor on the different options and it is set to display the assigned data from the csv file on the vital stats text box you can. Then we have the 5 buttons, and it also just displays the assigned data when the button is pushed in the appropriate text box. The check on Hal button takes us to the second scene to check on astronaut 2 which works just like the first scene, and we can use the Check on Mer to get back to the original scene.		

List of files

- **Arduino Program:** Integrated sensors code gets uploaded to the Nano.
- **Arduino to Pi Program:** Reads the sensor data from the Arduino and saves the data to a new CSV file every 200 sets of data.
- **Machine Learning Program:** Takes the read sensor data every 180 seconds and finds the Status and Condition based on the training set and that gets appended to the original sensor data.
- **GPS Module and Journey Logistics Program:** It takes the current location and finds the distance in km and miles and how long it will take to get there using the GPS speed and using the assumption of a walking speed of 3 miles per hour. It does this for the given latitude and longitude of these buildings Wheatly, McCormick, Campus Center, University Hall, etc., and finds how much consumables you need to get there, Oxygen, Water, and battery. This is gets appended to the full Hub data file
- **Hub-to-Hub Program:** Sends the full Hub data file to the other hub and receives the hub's full data file with all the astronaut's information. It sends the hub data every 180 seconds in chunks of 200 bytes. And receives the full appended data and saves it to a file.
- **Server-Client-UnityC# Programs:** The python Client sends hub 1 and hub 2 CSV data files every 180s to the server and the Server saves them to two different folders; Hub 1 and Hub2 and the Unity Client receives the data as it is sent and saves it to two folders in its Assets to use to display.

- **Hub1/2 bash script:** A Bash script that executes a sequence of all the Python scripts in the background, with specified time intervals between them. As some of the background processes are an infinite loop, the script will not terminate until all the programs are manually stopped or the computer is rebooted, therefore the off button still doesn't terminate the script.
- **On/Off Program:** This Python script is designed to run on a Raspberry Pi computer to start the Hub1/2 bash script with a push button to toggle the state of the system on/off and turns on/off LEDs to indicate the state of the system. It listens for button press events on a specific GPIO pin, and when the button is pressed, it toggles the state of the system on/off and turns on/off LEDs to indicate the state of the system. The script also starts and stops a bash script in a subprocess, depending on whether the system is on or off. The script continuously checks for button press events using a while loop and uses the RPi.GPIO library to interact with the GPIO pins on the Raspberry Pi.

Program flow



Imports

- **Arduino Program:** install arduino software, Arduino_LSM6DS3 library, Arduino_LSM6DSOX library, Arduino BusiO lirary, Adafruint unified Sensor library, OneWire library, Pulse sensor playground Library.
- Arduino to Pi Program**:** To import pytz; pip install pytz
- **Machine Learning Program:** To import pandas; pip install pandas, tensorflow; pip install tensorflow, sklearn; pip install scikit-learn
- **GPS Module and Journey Logistics Program:** To import the libraries follow the Adafruit Ultimate GPS RasPi Python Setup
- **Hub-to-Hub Program:** To import the xbee library; pip install digi-xbee

Table 2: Software Reference Guide

Software	Purpose	How to use
MS Project	Project management	Online sources
MS Teams	Project management	Online sources
Fusion 360	Printed casing design	Online sources
KiCad	PCB design	Online sources
Raspberry Pi OS	Raspberry Pi OS	Raspberry Pi set up instructions
VNC viewer	Raspberry Pi remote access	Raspberry Pi remote access instructions
Pitunnel	Raspberry Pi remote access	Raspberry Pi remote access instructions
VsCode	Code editor	Online sources
Arduino	To program all our sensors and read sensor data	Online sources
XTCU	To configure the XBee modules	XBee configuration instructions
Unity 2020, MRTK package is 2.7.2	To create the interface that will be implemented on the HoloLens	Unity development for HoloLens

Parts

H/W	Parts with links	Vendors	Price
H	HoloLens	Provided to us by the school	
H	Raspberry Pi model 4 (includes wires, batteries, fans)	Amazon	$280
H	Pulse Sensor	Vetco Electronics	$10
H	Temperature and Pressure Sensor	adafruit	$20
H	Water Tank Gauge	adafruit	$40
H	Li Batter Fuel Gauge	adafruit	$10
H	Accelerometer sensor	adafruit	$12
H	XBee S2C (Zigbee Modules - 802.15.4)	Mouser Electronics	$33
H	Adafruit Ultimate GPS Breakout	adafruit	$30
H	GPS Antenna - External Active Antenna	adafruit	$20
H	RP-SMA to uFL/u.FL/IPX/IPEX RF Adapter Cable	adafruit	$5
H	Arduino Nano	Arduino	$25
H	Pressure Transducer Sensor (150 Psi)	Amazon	$15
H	LM35 Analogue Precision Centigrade Temperature Sensor	Amazon	$13
H	SparkFun XBee Explorer Dongle	SparkFun	$28
H	SparkFun XBee Explorer Regulated	SparkFun	$12
H	USB 2.0 Male to Female Extension Plug	Amazon	$8
H	Battery PCB Protection Board	Amazon	$11
H	Buck converters	Amazon	$11
H	18650 Lithium-Ion Battery	18650batterystore	$133
H	Parallel Nickel-Plated Belt for Batteries	Amazon	$11
H	Nickel strip for Batteries	Amazon	$6
H	Lithium-Battery-Plastic-Bracket	Amazon	$13
H	Resistant-Masking-Soldering-Protecting-Cellphone	Amazon	$10
H	Boost Charging Module Dual USB 18650 Battery Fast Charger	Amazon	$10
H	TMP36 Temperature Sensor	Amazon	$15
W	Sensor Node PCB (Pack of 3)	OSHPARK	$50

Components

- **Sensor Node**

Sensor	Sensor pin	Arduino pin
GPS module		
	Vin	5 V
	GND	GND
	RX	7
	TX	8
Accelerometer		
I2C	Vin	5 V
	GND	GND
	SCL	SCL/A5
	SDA	SDA/A4
Pulse Sensor		
ADC	Vin (P2)	5 V
	GND (P1)	GND
	Analog pin (p3)	A0
Body temperature		
ADC	Vin (P1)	5 V
	GND (P3)	GND
	Digital pin (P2)	D4
Water level		
ADC	GND-3RD PIN	GND
	ANALOG-2ND pin to resistor (560 ohm)	A1
	VIN to resistor	5 V
Battery gauge		
	GND	GND

- **RF XBee modules**

Network setting
ID Pan ID: same for all the devices to communicate in the same network.
CE Co-ordinator enable (1) or not (0): Master of the network

Addressing setting
1) **DH** Destination Address High:
It is known XBee address number written in SH format.
2) **DL** Destination Address Low:
it is known XBee address number written in SL format.
3) **NI** Node identifier
It could be any name to identify XBee, in my case I mentioned as coordinator.

Serial interfacing:
1) **BD** Baud rate:
Make sure you XBee talks in same speed by keeping baud rate same for all.
2) **AP** API Enable:
By changing this column, we can set XBee for API mode and AT mode, for serial communication you keep this in AT mode. (transparent mode)

Table 3: XBee set up in 802.15.4 TH firmware

Sr no	Details	XBee 1 (Hub1) – Transmitter **Co-ordinator**	XBee 2 (Hub2) – Receiver **End device 1**	XBee 3 (Hub2) – Transmitter **End device 2**	XBee 4 (Hub1) – Receiver **Co-ordinator**	Remarks
	MAC add	0013A20041F2E02	00013A20041F21DC7	0013A20041F21DAA	00013A20041F21DCD	
Networking						
1	ID PAN ID	3332	3332	3332	3332	same for all xbees
2	CE co-ordinator	Enabled	Disabled	Disabled	Enabled	for coordinator
Addressing						
1	DH Destination Address High	13A200	13A200	13A200	13A200	Interchange address number
2	DL Destination Address Low	2018	2023	5105	2468	Interchange address number
3	MY Source Address	2023	2018	2468	5105	XBee Source Address
4	NI Node identifier	Tx1_xbee1	Rx1_xbee2	Tx2_xbee3	Rx2_xbee4	Any name for identification
Serial Interfacing						
1	BD Baud rate	9600	9600	9600	9600	same for all bee
2	AP API Enable	API 1	API 1	API 1	API 1	for Serial communication.

Table 4: XBee set up in 802.15.4 TH firmware

Sr no	Details	XBee Module 1 (Hub1) – Transmitter **Co-ordinator**	XBee Module 2 (Hub2) – Receiver **Router 1**	XBee Module 3 (Hub2) – Transmitter **Router 2**	XBee Module 4 (Hub1) – Receiver **Router 3**	Remarks
	MAC add	0013A20041F2E02	00013A20041F21DC7	0013A20041F21DAA	00013A20041F21DCD	
Networking						
1	ID PAN ID	3332	3332	3332	3332	same for all xbees
2	JV Channel verification	Disabled	Enabled	Enabled	Enabled	for router
3	CE co-ordinator	Enabled	Disabled	Disabled	Disabled	for co-ordinator
Addressing						
1	SH	13A200	13A200	13A200	13A200	
2	SL	41F20E02	41F21DC7	41F21DAA	41F21DCD	
1	DH Destination Address High	Router 1 SH: 13A200	Coordinator SH: 13A200	Router 4 SH: 13A200	Router 3 SH: 13A200	Interchange address number
2	DL Destination Address Low	Router 1 SL: 41F21DC7	Coordinator SL: 41F20E02	Router 4 SL: 41F21DCD	Router 3 SL: 41F21DAA	Interchange address number
3	NI Node identifier	Tx1_xbee1	Rx1_xbee2	Tx2_xbee3	Rx2_xbee4	Any name for identification
Serial Interfacing						
1	BD Baud rate	9600	9600	9600	9600	same for all bee
2	AP API Enable	API 1	API 1	API 1	API 1	for Serial communcation.