ARC Machine Monitoring System

By
Rohan Inampudi
Akhil Kodumuri
Calvin Lee

Project Proposal for ECE 445, Senior Design, Fall 2022
TA: Zhicong Fan

22 September 2022
Project No. 22

Contents

1. Introduction

1.1 Problem

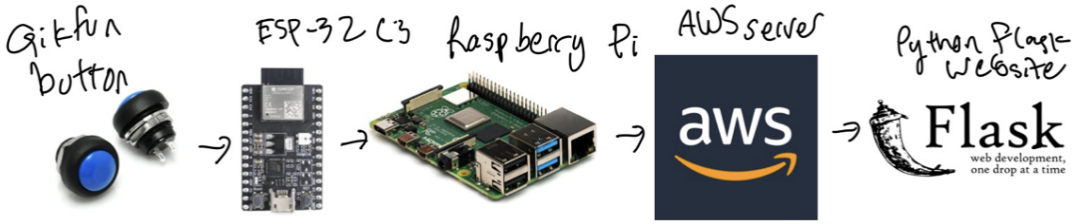
One question that is always on a college student's mind is: "Is the ARC busy?". There have been many times throughout our college career where we have gone to the ARC expecting a quick workout just to see lines for the machines we want to use. We've always wished that we could see what machines were being used and what were not. To combat this, we would like to create an interface where students can use their phone to visually see which equipment at the ARC are being used and which are not.

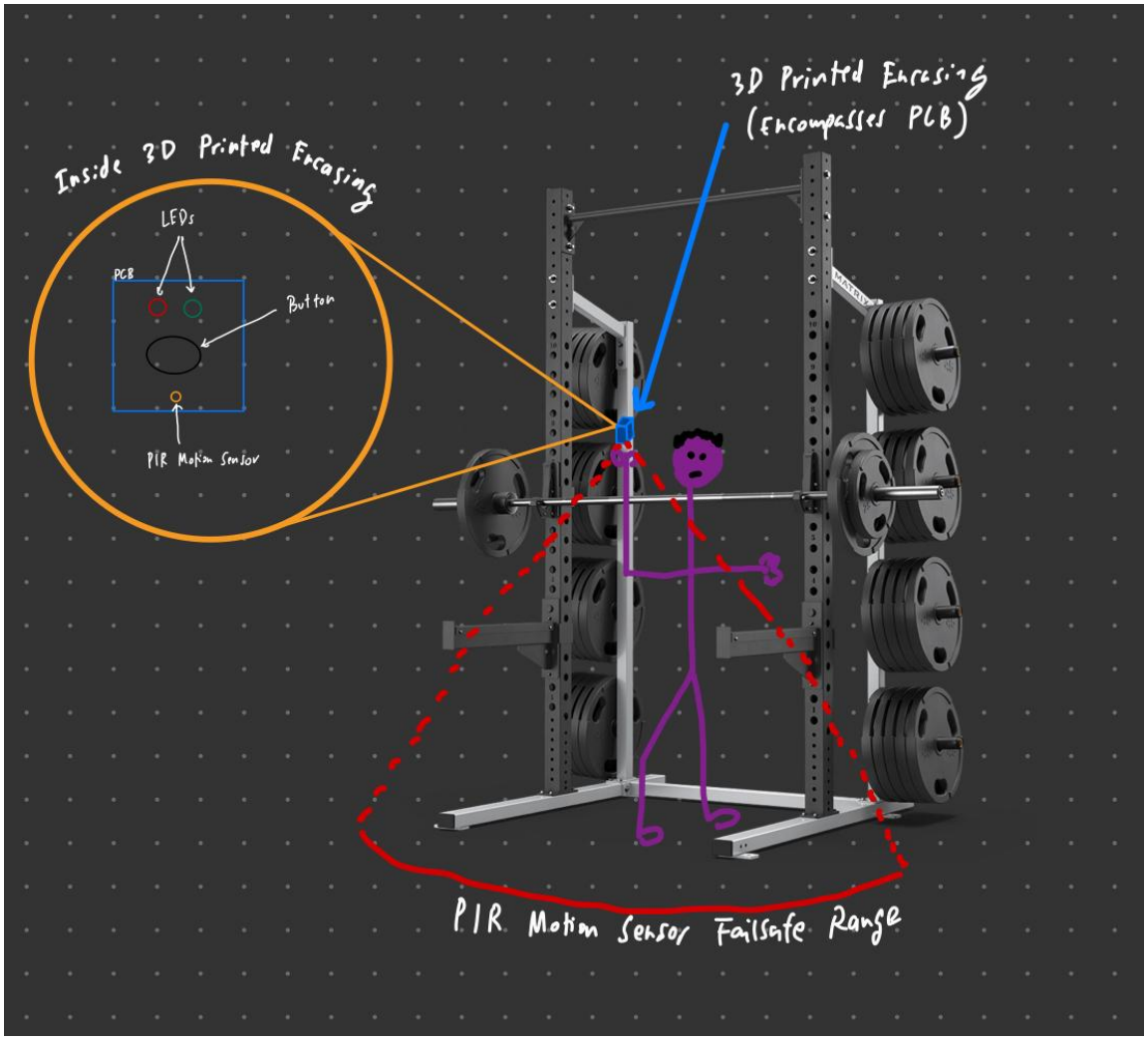
1.2 Solution

We would like to create an interface where students can use their phone to visually see which equipment at the ARC are being used and which are not. This way, students can anticipate whether or not they should go to the ARC. At a high level, there would be a button by the equipment being used. The button, upon being pressed by the user when a machine is being used, would then send a signal to an IoT device which would then send a signal to an AWS server. Our website will then use this server to update a UI which users can utilize to see which machine is being used. It should be noted that this design will be restricted to the Matrix Mega Half Rack which is the most used apparatus at the ARC. This system can be used generally for any machine at the ARC, however, the full capabilities of this system is unlocked for the Matrix Mega Half Rack.

1.3 Visual Aid

The diagram below depicts the full process of how our system works. First, a user will press a button on an ARC machine. Then, the System on Chip (ESP-32) on the PCB will wirelessly communicate the machines used to a Raspberry Pi. This Pi will then send this information to AWS, where our website will get information about whether or not a machine is in use. The UI will then depict to a viewer that a machine is in use.


Gikfun
button
ESP-32 C3
Raspberry Pi
AWS server
aws
Python Flask
website
Flask
web development,
one drop at a time

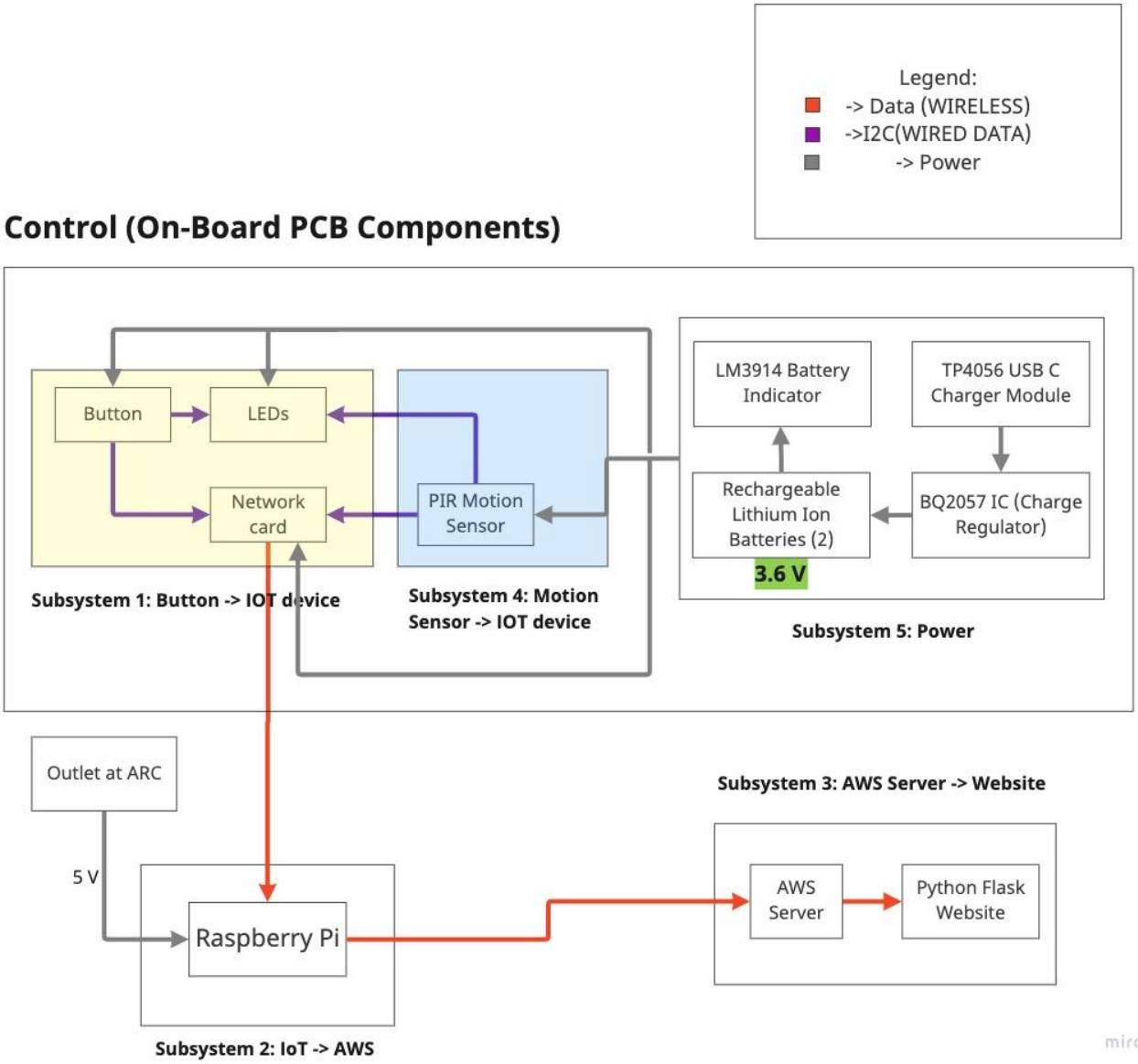

Inside 3D Printed Encasing
LEDs
PCB
Button
PIR Motion Sensor
3D Printed Encasing
(Encompasses PCB)
PIR Motion Sensor Failsafe Range

1.4 High level requirements list

Battery Life: PCB should be able to last for multiple hours without a direct power source.
Multiple Machines Serviced: Our system will be able to support up to 5 machines being used.
Website Access: Multiple people will be able to access the website at one time

2. Design

2.1 Block diagram:



2.2 Subsystem Overview:

2.2.1 Subsystem 1 (Button to IoT device)

In order to detect whether or not a machine is in use, we would like to create a PCB with the following components: a button that will be pressed whenever a machine is being used, an esp32 system on chip, and a led of different colors to indicate that a machine is being used. We will use the MQTT network protocol to communicate between the esp32 on our PCB and the Raspberry Pi. Information on machine status will be shared via the MQTT protocol to the Raspberry Pi. The Raspberry Pi will also receive information from our website that will then be communicated to the PCB. Once the button is pressed, the led on our PCB will also light up to signify the machine is in use. In order to power this subsystem, we will have rechargeable lithium ion batteries, on a battery pack. These batteries can be easily charged. In order for this subsystem to act in accordance with our high level requirements, our IoT device should be able to handle the communication of multiple machine sensor pcbs at once. Also, this subsystem should be able to last multiple hours with a single 3V lithium ion battery.

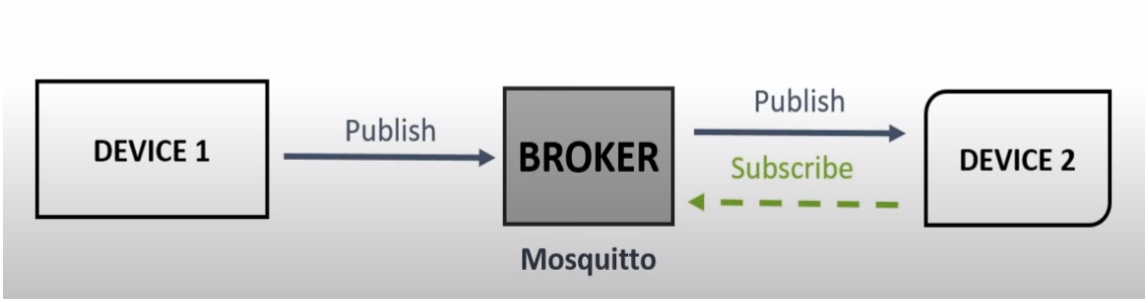
In order to test and program the esp32 system on chip, we will be using a USB UART TTL in order to program and configure the esp32. We will only require a single USB UART TTL to program all the esp32s that we will be using.

Button Requirements	Verification
Requirement 1: A button press should be able to send a MQTT packet to our IoT device.	Verification 1: On our IoT device, which will act as a MQTT server, we can verify what devices are requesting information on it, by a simple command on the IoT device.
Requirement 2: When a user presses the button to use the ARC machine, the red LED should light up to signify the machine is in use.	Verification 2: This can be easily verified by pressing the button on our system and visually checking the activation of the red led.
Requirement 3: When a user presses the button to use the ARC machine, the green LED should light up to signify the machine is not in use.	Verification 3: This can be easily verified by pressing the button on our system and visually checking the activation of the green led.

Requirement 4: Multiple machines must be able to send messages simultaneously	Verification 4: on the IoT device command line can be used to check what devices are trying to send messages to the device
Requirement 5: Control subsystem should last 4 days without being recharged.	Verification 5: We will conduct tests to make sure the Control subsystem can last 4 days.

2.2.2 Subsystem 2 (IoT device to AWS Server)

For a high level overview of our project, the button is directly connected to the ESP-32, which will communicate with the Raspberry Pi via Message Queue Telemetry Transport, or MQTT. Once the data is on the Raspberry Pi, we are planning on utilizing AWS sdk to publish the data from the sensor onto the cloud.



MQTT publish/subscribe model

MQTT is the standard of communication for IoT designs due to its lightweight publish/subscribe system, and is designed for constrained devices with low bandwidth. Using MQTT, we can have multiple clients sending and receiving data to and from a single “broker”, a hub that receives and filters all messages, and publishes the messages to all clients that are subscribed to that topic. In our case, the ESP-32 will be acting as a client, and the Raspberry Pi will be acting as both a client and a broker. Once the button is pressed, the data for a certain topic (for example, “data/arc_availability”) will be published from the ESP-32 to the Pi, and the Pi, which is subscribed to that topic, will receive that published data.

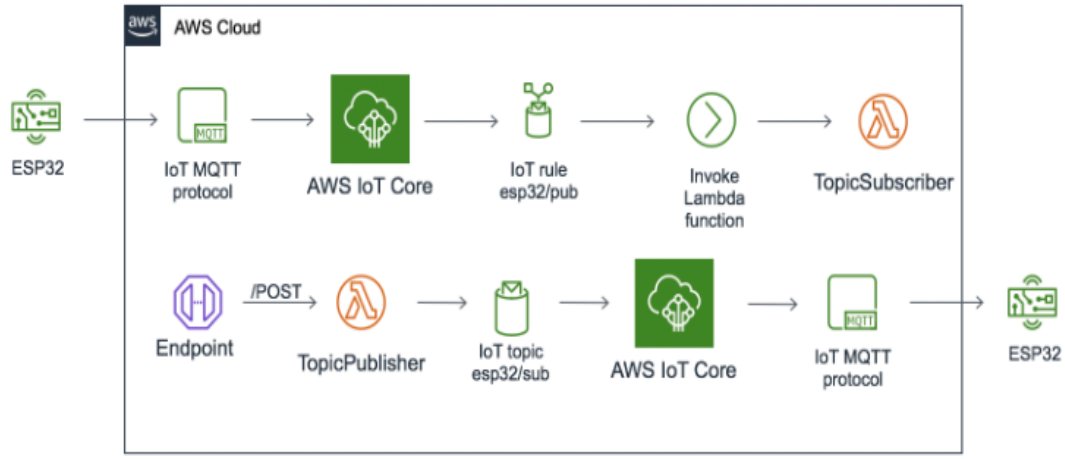


Diagram displaying AWS cycle

In order to send data to the AWS cloud, we will install AWS IoT Device SDK onto our Raspberry Pi 3 Model B. All machine information that is sent to the Pi will be sent to the AWS cloud. Ultimately, the Raspberry Pi will act as a gateway of communication between all ARC machine sensors to the AWS server. Ideally, the ARC would only need one Raspberry in order to send ARC machine sensors information to AWS. Because of this, in order to charge the Raspberry Pi, we will just plug it into a free outlet at the ARC.

IoT Device Requirements	Verification
Requirement 1: Manage multiple MQTT publish messages from a single ESP32 Device	Verification 1: Command line within the IoT device interface can be used to verify whether or not each publish message from a particular ESP32 device is being received
Requirement 2: Manage and differentiate multiple MQTT publish messages from multiple ESP32 Devices.	Verification 2: Command line within the IoT device interface can be used to verify if the IoT device is receiving all packets from multiple ESP32 devices
Requirement 3: Manage different types of MQTT publish messages from AWS server and be able to differentiate when AWS server is referring to a particular ESP32 device.	Verification 3: Command line within the IoT device interface can be used to verify if the IoT device is receiving all packets from our AWS server.

2.2.3 Subsystem 3 (AWS Server to website)

Once the data is on the cloud, we can use AWS API commands to directly access the data from our web server. This server will host all information on which machine is being used. The website we will design will keep track and display what machines at the ARC are being used by using the AWS server configured in Subsystem 2. We will have the feature of push notifications on our website. If a student wants to be alerted that a machine is available, they can provide their email address and will be contacted when a machine is not in use. This website will be configured using the Python framework Flask. In order to act in accordance with our high level requirements, multiple users should be able to use our website without any issues.

AWS Server Requirements	Verification
Requirement 1: Server must be able to receive MQTT messages from Raspberry Pi 3	Verification 1: AWS has a console to interface with its servers. We can use the AWS console to ensure the proper packets are being sent from and to the AWS server.
Requirement 2: Server must send packets to our website with information on which machine is in use.	Verification 2: We can check to see if the web server hosting our website can check whether or not messages from the AWS server are being received.

Website Requirements	Verification
Requirement 1: Website receives packets from AWS server.	Verification 1: Web Server can be checked to see what packets are being sent to it.
Requirement 2: Website is able to send email notifications to students when a machine is not being used.	Requirement 2: We will have our phones registered to receive email notifications when machines are not in use to ensure the notification system works.

2.2.4 Subsystem 4 (Motion sensor to IoT device)

If there is time in the project, we would like to attach a motion sensor in order to detect use of the equipment that occurred without pressing the button. This would allow our website to be changed if a person forgot to press the button. We would like to include a fail safe in order to provide the most accurate information to students. The motion sensor will be on the PCB, and, thus, will be powered by the rechargeable lithium battery. We will incorporate logic to limit the power consumption of the motion sensor, so it is not being used continuously. An example of this logic is scheduled downtimes for when the ARC is closed. Since we will be using a PIR motion sensor, we will use electronic tape in order to limit the range of view of the motion sensor. This will ensure that we will not take in more information that we don't need and we can control the sensor to take in input from individuals using the machine.

Motion Sensor Requirements:	Verification:
Requirement 1: Motion sensor should only be able to detect movement from when the machine is in use and not due to random movement.	Verification 1: We will test movement manually to make sure random movement isn't tracked. We will also use electronic tape to limit the angle of motion intake.
Requirement 2: Motion sensor should only trigger if a machine is in use if the button is not pressed.	Verification 2: We will test to make sure the machine status on the website isn't changed when the button is pressed.
Requirement 3: Motion sensor should only detect movement up to 3 meters.	Verification 3: We will utilize the potentiometer on the PIR sensor to ensure the range of the motion sensor is up to 3 meters.

2.2.5 Subsystem 5 (Power)

Power System Requirements:	Verification:
Requirement 1: Be able to power all on-board components.	Verification 1: All subsystems within the PCB work as intended.

Requirement 2: Be able to sustain power to all on-board components for 5 days	Verification 2: All subsystems within the PCB work as intended for 5 days worth of operation hours at the ARC.
Requirement 3: Be able to safely charge each rechargeable battery.	Verification 3: The BQ2057 IC will be able to manage the charging of each lithium ion battery. This chip will prevent overcharging and overheating of the Power subsystem.
Requirement 4: Be able to display when a battery needs to be recharged for the convenience of ARC employees.	Verification 4: We are using the LM3914 bar graph LED to provide a visual to when the batteries will need to be recharged.

Component	**Power Consumption**	**Power Consumption (5 days)**
Button	-	-
ESP32	55 mA * 17 hrs * 5 days	4675 mAh
Battery Life Indicator	10 mA * 17 hrs * 5 days	850 mAh
LEDs	12mA * 17 hrs * 5 days	1020 mAh
PIR Sensor	0.1 mA * 17 hrs * 5 days	8.5 mAh
		Total: 6553.5 mAh

The power consumption for the battery life indicator, LEDs, and PIR Sensor are straightforward to calculate, as that information was directly on each component's datasheet. However, we must closely analyze the consumption of the ESP32 for our specific use case of transmitting and receiving data via WiFi:

When the ESP32 is in Active Mode (Wifi module, bluetooth module, and processing core running at all times), greater than 240 mA is needed for operation. Furthermore, it has been documented that large spikes in power of almost 790 mA occur when Wifi and Bluetooth are in operation together. For our use case, we will only be leveraging the WiFi connectivity feature of the ESP32 for data communication purposes with the Raspberry Pi. Furthermore, we intend to let the ESP32 operate in Deep Sleep mode so that it stays at

a low power consumption of 0.011 mA when it is not transmitting or receiving data via WiFi. Therefore, on average, we expect a power consumption of 39 mA - 55 mA for the ESP32.

In order to power all on-board components (listed above), we will be using two Panasonic NCR18650GA 3450 mAH 10A lithium ion rechargeable batteries. According to our calculations, we estimate a total power consumption (worst case) of around 6553.5 mAh. Thus, utilizing two 18650 batteries is essential as we will be able to output 6900 mAh, lasting a total of 5 full days of operational hours at the ARC.

Because we would like to ensure the safety and electrical integrity of our system, we are incorporating the BQ2057 IC for each rechargeable battery being used for the system. This will ensure that each battery is safely being charged to its full capacity and nothing more.

We will be charging the lithium ion batteries using a TP4056 USB C Li-Ion Charger module. Using this module, we will be able to charge the battery pack using a USB-C cable, thus enabling the entire system to charge without removing the physical batteries from the case.

For the convenience of the employees of the ARC, we will visually display the battery life of each power subsystem. The battery life indicator we will use is the LM3914 bar graph chip. The display will allow ARC workers to know when the rechargeable batteries will need to be plugged in for charging.

2.3 Tolerance Analysis

2.3.1 Determining Battery Life

In order for the ARC Machine Sensor to operate correctly, it needs to be connected to a large enough rechargeable battery in order to function properly. Thus, to calculate the amount of energy needed to power a single ARC Machine Sensor for 5 days (this is in accordance with our high level requirements), we are using the following equation, E_5_days = E_SumOfAllComponentsFor5Days. We postulate that the total amount of energy that the rechargeable batteries need to hold should be equal to the total amount of energy to fully power the ARC Machine Sensor system for 5 straight days of use without a recharge session. To calculate this we took the sum of the power needed to use each component of the ARC Machine Sensor system, which can be seen in the table below. The total power consumption is also shown. As can be seen, the total power consumption is 6553.5 mAh. With this number, we now know that a battery that provides enough total

power to 6553.5 mAh is needed. For this reason, we are using 2 18650 rechargeable batteries to power the ARC Machine Sensor system which will satisfy the power requirement of 5 days.

Component	**Power Consumption**	**Power Consumption (5 days)**
Button	-	-
ESP32	55 mA * 17 hrs * 5 days	4675 mAh
Battery Life Indicator	10 mA * 17 hrs * 5 days	850 mAh
LEDs	12mA * 17 hrs * 5 days	1020 mAh
PIR Sensor	0.1 mA * 17 hrs * 5 days	8.5 mAh
		Total: 6553.5 mAh

2.3.1 Subsystem 1 (Button to IoT device)

One facet of this subsystem we will need to account for is placement of our PCB. Though we plan on implementing a weight sensor in order to detect usage of ARC equipment without the button being pressed, ideally, the button will be primarily used to indicate that a machine is in use. For this reason, we will be testing out different locations to place our PCB, so a person at the ARC can easily locate and press the button.
Another facet of this subsystem we will need to account for is battery usage and consumption. Since datasheets may not contain accurate information about the product in use, we will need to test each component separately to ensure that we know the appropriate operating conditions to work within.

The biggest facet of this subsystem we will need to account for is how long the entire subsystem will last with an external, rechargeable lithium battery. Because of this, we will complete the appropriate calculations in order to maximize battery life.

Another facet of this subsystem we will need to account for is handling traffic of multiple sensors on the network. Since, multiple sensors will be communicating to the IoT device, the device must be able to differentiate between each sensor and adapt accordingly.

The final consideration that needs to be made for this subsystem is connecting ESP32 devices onto the Universities network. We will need to get into contact with Engineering IT in order to safely connect our ESP32 devices onto the university network. This step is necessary because we would like to ensure that there are no network vulnerabilities with connecting ESP32 devices onto the university network.

2.3.2 Subsystem 2 (IoT device to AWS Server)

Similarly, to Subsystem 1, the biggest facet of difficulty within this subsystem is managing the packets sent between the AWS Server and IoT Device. We need to ensure that machine information details remain intact during the communication of IoT devices and AWS servers.

As described in section 2.3.1, we will need to get into contact with Engineering IT in order to safely connect our Raspberry Pi 3 (our IoT device) to the university's network. If this cannot be done, we have already researched multiple ways to get around connecting to the university's network. One method is using our Raspberry Pi 3 as its own router and utilizing a private network of our own creation, independent of the university's network.

2.3.3 Subsystem 3 (AWS Server to website)

The goal of the website is to provide a visually appealing interface that students can look upon to view whether or not their favorite machines are being used. We will spend ample time going through multiple designs for the front end of our website and asking our fellow students whether or not they find the website visually appealing. Initially, we would like to alert students that their machine of interest is free to be used via email notification. But, we will also survey students to see how they would prefer being notified.

Similar to Subsystem 1 and 2, we will consult with Engineering IT in order to ensure we are setting up a safe network connection to AWS. Since we would like to send email notifications to students, we will make sure that the AWS server is secure to perform tasks on top of the University's network.

2.3.4 Subsystem 4 (Motion sensor to IoT device)

The primary function of the motion sensor is to detect use of the equipment without the button being pressed. There are a few design considerations that we will need to consider when placing and designing this subsystem. The motion sensor will need to be in a location safe from the weights being used during an exercise. Also, since it is a part of the

overall PCB design it will also need to be in a place where a student at the ARC can easily locate and press the button.

2.3.5 Subsystem 5 (Power)

In order for our power subsystem to work as intended, we expect our rechargeable lithium ion batteries to supply sufficient power to all on-board components on the PCB both safely and efficiently. Especially with delivering power, one must be aware of overheating and overcharging.

Overcharging leads to inefficient power draw, and has the potential to damage and overheat the electrical components of the system itself. Since all of the components on the PCB will be in close proximity, we will also need to ensure that the heating of any component is not interfering with other components. We will also need to make sure the overall apparatus that will encase the entire “ARC Machine Sensor” is ventilated to to release heat generated by the electrical components.

One aspect we must monitor in regard to power is how easily ARC employees will be able to recharge the batteries. The batteries being charged is critical to our project’s success due to the fact that without power, all components on our PCB would not be able to function. According to our power consumption calculations in section 2.2.5, we deduce that batteries must be recharged every 5 days. Instead of replacing the batteries whenever they die, ARC employees will be able to easily plug in any USB-C cable to charge the batteries. Whenever the batteries are done charging, this cable can be removed, thus no unnecessary mess will be created. As a result, this improves the safety of our design as a whole. Moreover, from a consumer POV, our design is minimalistic and reduces user interaction while still being able to function effectively.

3. Cost and Schedule

3.1 Cost Analysis

The subtotal for all the parts necessary is $149.03. Assuming the following estimates, the salary per team member would come out to: $35/hr * 2.5 * 15 hrs/wk * 10 wks = $13,125 * 3 = $39,375 total labor cost. Our project is expected to require five hours of labor from the machine shop. $38.17/hr * 5 hours = $190.85. Machine Shop Cost . Total cost including parts comes out to $39,715.88.

Description	**Manufacturer**	**Quantity**	**Extended Price**	**Link**
ESP32-S2R2	Espressif Systems	2	$1.50	link
FireBeetle 2 ESP32-E MCU	FireBeetle	2	$8.90	link
Raspberry Pi 3 - Model B	Adafruit	1	$35.00	link
12mm Button	Gikfun	1	$8.78	link
5x Stemedu HC-SR501 PIR Sensor	Stemedu	1	$9.99	link
RGB LEDs	Adafruit	1	$3.95	link
5x AM312 PIR Sensor	Aideepen	1	$9.59	link
Rechargeable 18650 Battery	Panasonic	4	$4.99	link
Raspberry Pi 3 Power Adapter	Canakit	1	$9.95	link
2x18650 Battery Holder	E-outstanding	1	$8.99	link
BQ2057 Charge Controller	Texas Instruments	4	$1.56	link
TP4056 USB C Li-Ion	diymore	1	$9.59	link

Charger Module				
8GB Micro SD Card for Raspberry Pi	Verbatim	1	$6.19	link

3.2 Project Schedule and Task Allocation

We each are software oriented individuals, so we hope to do most PCB and EE related work together, while splitting up the software work:

Week	Team Deliverables	Rohan	Akhil	Calvin
9/26 - 10/3	Design Doc	Design Doc	Design Doc	Design Doc
10/3 - 10/10	Design Review PCB Board Review	Finalize PCB Design	Finalize PCB Design	Finalize PCB Design
10/10 - 10/17	PCB Order 1 Teamwork Evaluation	Complete Teamwork Evaluation	Complete Teamwork Evaluation	Complete Teamwork Evaluation
10/17 - 10/24	Assemble Full PCB Design	Assemble PCB Start webdev	Assemble PCB Start webdev	Assemble PCB Start webdev
10/24 - 10/31	Ensure PCB Design Works	Test PCB	Test PCB	Test PCB
10/31 - 11/7	PCB Order 2 Individual Progress Reports	Complete Progress Reports	Complete Progress Reports	Complete Progress Reports
11/7 - 11/14	Debug and Refine Software	Debug Full System	Debug Full System	Debug Full System
11/14 - 11/21	Mock Demo Final Testing	Final Testing + Demo Prep	Final Testing + Demo Prep	Final Testing + Demo Prep
11/21 - 11/28	Fall Break	Break	Break	Break

11/28 - 12/5	Final Presentation Final Paper	Work on Final Paper and Presentation	Work on Final Paper and Presentation	Work on Final Paper and Presentation

4. Ethics and Safety

There are a couple ethics and safety issues that our project is facing. We will make sure that nothing in the code of ethics is breached, and that the safety of the user is the utmost priority of the project.

The first ethical issue is the data collection that is performed whenever someone presses the button to modify the machine availability. In order to make sure there are no breaches in privacy, we will not be implementing an account system such that no personal information will be tracked, nor will patterns be able to be perceived and logged. The only use we currently have in mind for our project is for people to be able to see whether or not the gym is busy regardless of who is utilizing the machines.

The other ethical issue is regarding the email notification function of our website. In order to protect our user's data, we will only be using it for notification purposes, and will make sure the user knows the implications of submitting their email information.

Furthermore, we also have a safety issue due to utilizing a battery. In order to prevent any possible hazards, we will make sure to keep the PCB casing organized, and safely use the batteries within the safe operating range.

5. References

[1] *"Amazon.com: Gikfun 12mm waterproof push button momentary on off switch ..." [Online]. https://www.amazon.com/Gikfun-Waterproof-Button-Momentary-Arduino/dp/B07W5TGKQ3. [Accessed: 15-Sep-2022].*

[2] "PS1440P02BT: Digi-key electronics," *Digi*. [Online]. Available: https://www.digikey.com/en/products/detail/tdk-corporation/PS1440P02BT/2236832?utm_adgroup=Alarms%2C%2BBuzzers%2C%2Band%2BSirens&utm_source=google&utm_medium=cpc&utm_campaign=Shopping_Product_Audio%2BProducts_NEW&utm_term=&utm_content=Alarms%2C%2BBuzzers%2C%2Band%2BSirens&gclid=Cj0KCQjw39uYBhCLARIsAD_SzMRocrg56djQZdtSr1banc2WuquRuRWNwZ3Xb1x-w5BqNdJqJw-9-HQaAjB5EALw_wcB. [Accessed: 15-Sep-2022].

[3] "Chipsets: Espressif Systems," *Chipsets | Espressif Systems*. [Online]. Available: https://www.espressif.com/en/products/socs. [Accessed: 15-Sep-2022].

[4] Németi Florian, G. Pauletto, and D. Duay, "IOT: L'émancipation des objets," *Amazon*, 2017. [Online]. Available: https://docs.aws.amazon.com/iot/latest/developerguide/connecting-to-existing-device.html#gs-device-view-msg. [Accessed: 15-Sep-2022].

[5] Cdaviddav, "Send data from ESP8266 or ESP32 to Raspberry Pi via MQTT," *DIYI0T*, 07-May-2021. [Online]. Available: https://diyi0t.com/microcontroller-to-raspberry-pi-wifi-mqtt-communication/. [Accessed: 15-Sep-2022]

[6] "External+Circular+Battery+for+PCB," *Google Shopping*. [Online]. Available: https://www.google.com/shopping/product/5898283066540390 66?q=external%2Bcircular%2Bbattery%2Bfor%2Bpcb&client=safari&rls=en&sxsrf=ALiCzsZaA5BHehX59EQpajrvZOfjehFTwg%3A1663275125169&biw=1440&bih=735&dpr=2&prds=eto%3A2725741664819496371_0%2Clocal%3A1%2Cpid%3A11856754887431340 41%2Cprmr%3A2%2Crsk%3APC_1026065390 2913479911&sa=X&ved=0ahUKEwiX2P6R15f6AhXpjIkEHVlACSgQ8wIIuBE. [Accessed: 15-Sep-2022].

[7] "2-1775485-1 : Battery holders," *TE Connectivity*, 03-Jan-2018. [Online]. Available: https://www.te.com/usa-en/product-2-1775485-1.html?te_bu=Cor&te_type=srch&te_campaign=ggl_usa_cor-ggl-usa-srch-smbmktg-fy22-googlefeed_sma_sma-2210_2&elqCampaignId=11572

4&mkwid=VDybDFVR%7Cpcrid%7C386964346943%7Cpkw%7C%7Cpmt%7C%7Cpdv%7Cc%7Cslid%7C%7Cproductid%7C2-1775485-1%7Cpgrid%7C78782457763%7Cptaid%7Cpla-298884436745%7C&utm_content=VDybDFVR%7Cpcrid%7C386964346943%7Cpkw%7C%7Cpmt%7C%7Cpdv%7Cc%7Cslid%7C%7Cproductid%7C2-1775485-1%7Cpgrid%7C78782457763%7Cptaid%7Cpla-298884436745&gclid=Cj0KCQjwmouZBhDSARIsALYcourXPqfM1fucoo39DQsdyYPBcSBieWtgXxXjzO0ox1Y1Peppnti2pE0aAkw-EALw_wcB. [Accessed: 15-Sep-2022].

[8] Last Minute Engineers, "Insight into ESP32 sleep modes & their power consumption," *Last Minute Engineers*, 04-Jul-2022. [Online]. Available: https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/#:~:text=The%20chip%20consumes%20around%200.15,coprocessor%20is%20on)%20to%2010%C2%B5A. [Accessed: 28-Sep-2022].

[9] Cdaviddav, "Guide to reduce the ESP32 power consumption by 95%," *DIYI0T*, 07-May-2021. [Online]. Available: https://diyi0t.com/reduce-the-esp32-power-consumption/. [Accessed: 28-Sep-2022].

[10] Smt, "18650 battery how to charge - 5 simple builds," *SM Tech*, 14-Aug-2022. [Online]. Available: https://somanytech.com/18650-battery-how-to-charge-18650-battery-with-charger-and-without-charger/#:~:text=USB%20powered%2018650%20battery%20chargers,you%20can%20say%20slow%20charging. [Accessed: 28-Sep-2022].