ARC Machine Monitoring System

By
Rohan Inampudi

Lab Notebook for ECE 445, Senior Design, Fall 2022
TA: Zhicong Fan

Project No. 22

Table of Contents

Week 1 9/12 - 9/19

This week consisted of a lot of brainstorming to come to a conclusion on which project we would be continuing with. We were debating between the two following projects:

1) Squat Weightlifting Helper: The premise of this idea is the fact that many people struggle to have the correct form while squatting. As a result, individuals who squat with incorrect form are very prone to injury. From personal experience, I have injured my knees due to incorrect form. Thus, it would be helpful if there was some app where one could record their squat form and get immediate feedback on what needs to be fixed. Essentially, we would mount some sensors on the bar itself and track the bar path. We would also have to develop some sort of algorithm that simulates the optimal bar path during the squat for each individual depending on his/her dimensions. Then, we would contrast the optimal bar path with the user's bar path and provide the user with feedback.
2) ARC Machine Monitoring System: One question that always crosses the mind of college students is: "Is the ARC bust?". There have been one too many times throughout my college career when I expect to get a quick workout done at the ARC, but the machine I want to use is taken. Thus, it would be beneficial to know whether one's favorite machine is in use or not at the ARC. Therefore, creating an interface where students can use their phones to visually see which equipment at the ARC is being used or not would be ideal. Essentially, we would have our PCB encased and mounted on a machine at the

arc. It would have a button and a motion sensor. Based on these peripherals, availability would be updated on a website, where arc-goers would be able to check the status of their favorite machine.

After much deliberation, we ended up going with the ARC Machine Monitoring System. This is because the Squat Weightlifting Helper idea had too many question marks around it. The more we thought about this specific idea, the more complex the technologies used were becoming, and it seemed like far too complex of a project. Therefore, we felt that the ARC Machine Monitoring System was a far more attainable prospect. Additionally, we felt that we could add enough complexity to the machine monitoring system to make it a solid project. Feel free to view our final proposal [here](#).

Figure 1: Visual Aid for Initial Idea

The picture above details the vision we currently have for the Arc machine monitor. A button on our PCB will communicate the status of a given machine at the ARC to our microcontroller, the ESP32. Then, the microcontroller will relay this information to the Raspberry Pi through MQTT protocol. In essence, the Pi acts as a middleman to communicate information to our website, which is hosted on an AWS server. The website itself will be constructed using a framework called Python Flask. I have some experience with this framework, thus setting up the website should not be too much of a hassle.

In Figure 2, our first block diagram is pictured. So far we have four subsystems. I will briefly

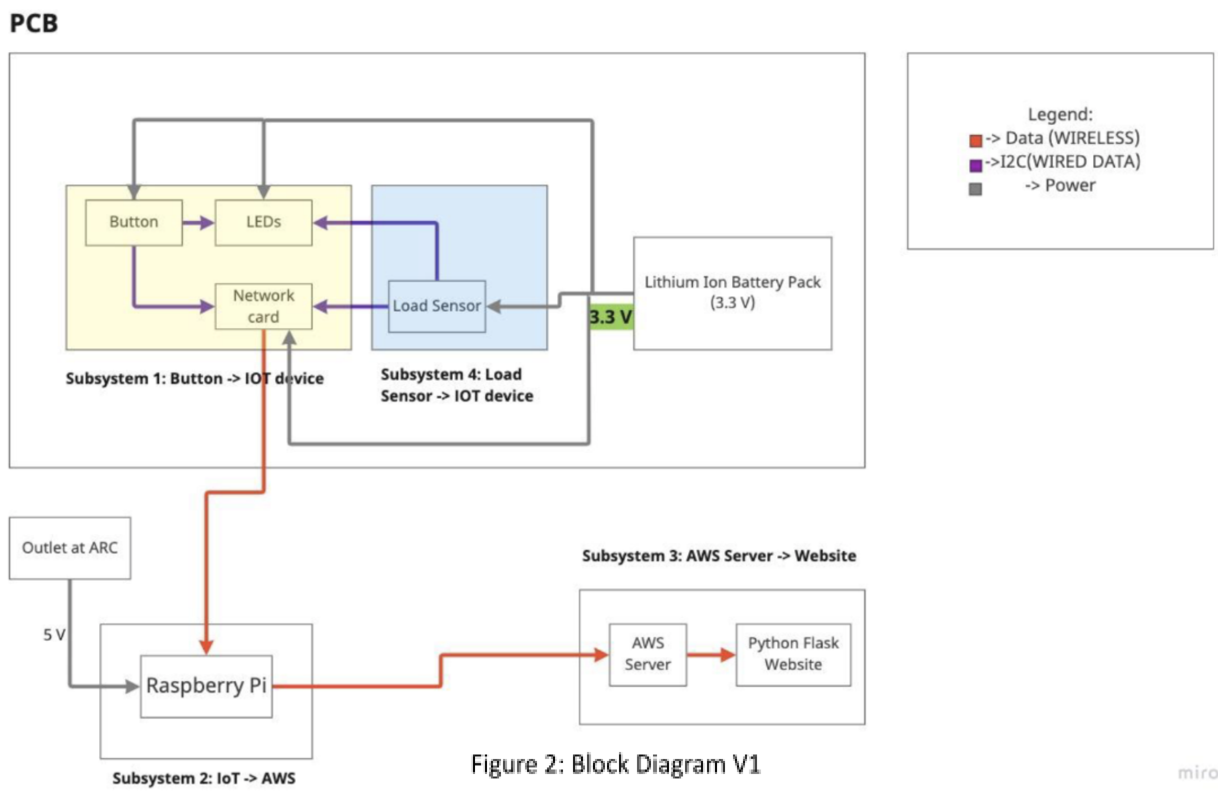


Figure 2: Block Diagram V1

explain them here:

1) Button to IoT device: To detect availability of a machine we will have a PCB with a button, ESP32, and LEDs with different colors. Upon a button press, the light will switch colors to indicate availability. We intend to use rechargeable lithium-ion batteries to power our PCB.
2) IoT to AWS: MQTT network protocol will be used for communication between the ESP32 on board to the Raspberry Pi. Since MQTT protocol sends strings only due to its lightweight nature, we will send the machine availability in this format. Once the Pi

receives this information, it will be sent to the website server. To send data to AWS cloud we will utilize AWS IoT Device SDK on the Raspberry Pi 3 Model B. This way all information will be sent to the AWS cloud.

3) AWS to website: A website will be available for users to visually see if their favorite machine is in use or not at the ARC. We will also integrate email notifications, where users will be able to subscribe to machines they desire to be notified for. Python Flask will be used for the development of the website
4) PIR motion sensor to IoT device: If there is time we will use a motion sensor to detect use of equipment when the button has not been pressed. This feature is meant to act as a failsafe, when one forgets to press the button.

Week 2 9/19 - 9/26

Our project was approved! During this week, we focused on formalizing the ideas we had regarding the ARC Machine Monitoring System for the Design Document. The main area of work was discussing how we would power all on-board PCB components. We ended up deciding to use 3.6V rechargeable lithium-ion batteries. To charge these batteries, we found the TP4056 USBC Charger Module. The plan is to connect the charger module to a BQ2057 IC charge regulator then to the batteries themselves. After some research, we deduced that we could display the charge of the battery on a LED bar graph chip with the help of the LM3914 battery indicator. Other than that, no other major changes were discussed. Our updated block diagram looks as follows:

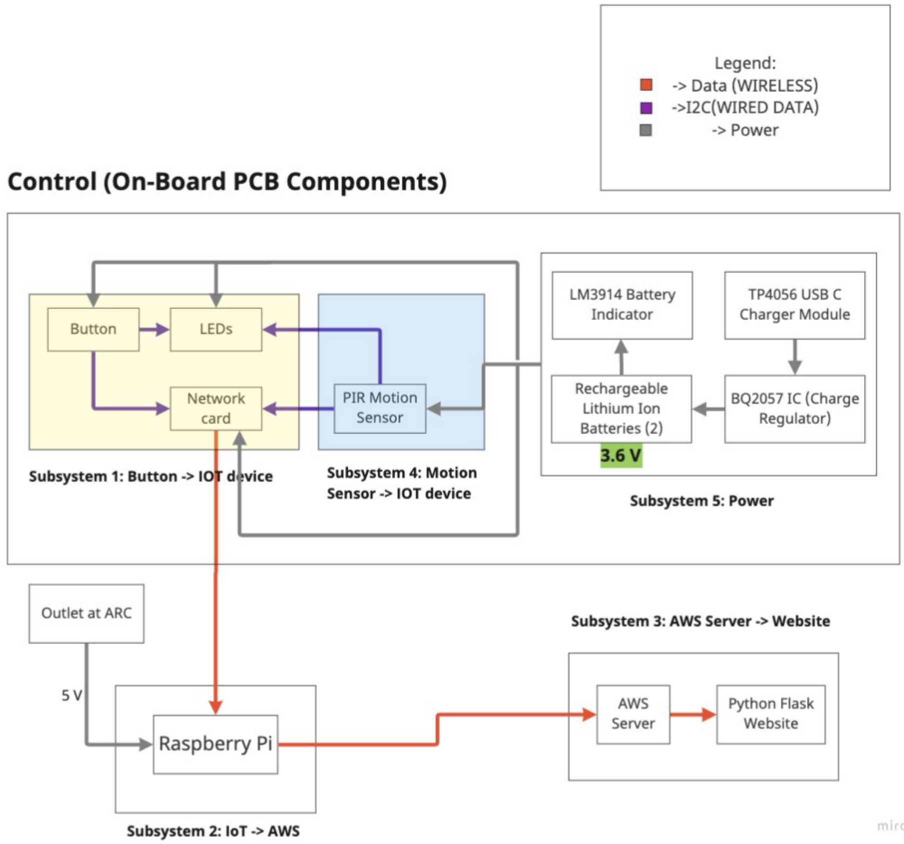


Figure 3: Block Diagram V2

Week 3 9/26 - 10/3

The Design Document is due this Thursday, so we used the majority of this week finalizing our design. We came up with the following high level requirements that determine our system as successful:

1) Battery Life: System should be able to last for multiple hours without a direct power source.

2) Machine Status: Changes in machine availability are correctly and efficiently communicated using multiple peripherals (button and PIR motion sensor).
3) Website Access: Website is accessible to users and correctly displays the most up to date availability of a machine.

The following are the changes/revelations we have had for each subsystem this week:

1) Button to IoT: We realized that in order to program an ESP32 SoC we will need a USB UART TTL. No further changes have been made to this subsystem.
2) IoT to AWS: The main focus of this subsystem is getting the MQTT protocol up and running. The following diagrams are how we envision to set up the MQTT protocol and AWS Cloud interface:

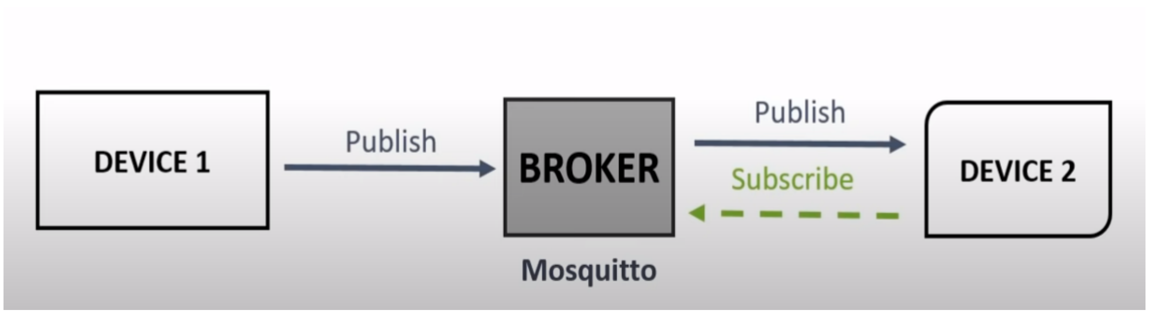


Figure 4: MQTT Protocol Setup

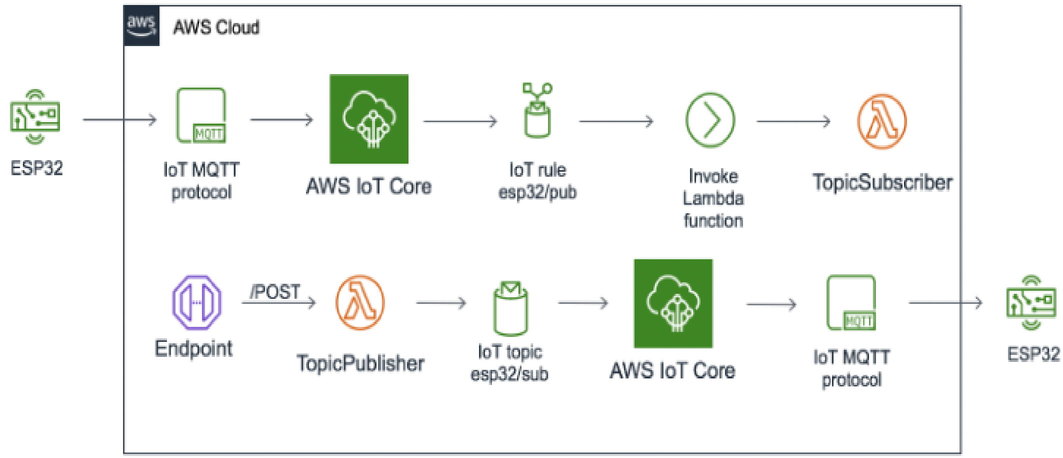


Figure 5: ESP32 to AWS Cloud Workflow

3) AWS to Website: No changes to this subsystem. We still intend to use Python Flask and AWS for our website.

4) PIR Motion Sensor to IoT: We would like to implement scheduled downtimes. We don't want the PIR motion sensor to continuously be turned on when the ARC is closed. Thus, we have looked into implementing the Deep Sleep mode within the ESP32. This would be able to cut power to the circuit during the intended times, thus conserving significant power.

5) Power: We decided to use the Panasonic NCR18650GA 3450 mAH 10A lithium ion rechargeable batteries. The estimated power consumption worst case of our system is around 6553.5 mAH. Therefore, by using two 18650 batteries, we are able to output 6900 mAH. This should last us 5 full days of operational hours at the ARC. A table with our calculations has been provided below. To calculate power consumption we use

$$Power\ Consumption\ (per\ 5\ days)\ =\ Current\ Draw\ *\ 17\ operation\ hrs\ *\ 5\ days$$

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

Figure 6: Power Calculations (mAH)

We discovered that when the ESP32 is in Active Mode (WiFi, bluetooth, processor running), greater than 240 nA is required. In Deep Sleep mode, the power consumption is as low as 0.011 mA. On average, we should expect a power consumption in the range of 39 mA - 55 mA for the ESP32.

Using the TP4056 USB C Lithium-Ion Charger will be imperative for providing a product that requires little maintenance work. If the charging circuit works as intended, an ARC worker will only need to plug in our system for charge every 5 days. The charge of the battery can be seen on the LM3914 bar graph chip.

Week 4 10/3 - 10/10

Design Review and PCB Board Reviews were due this week. Therefore, we spent multiple hours on KiCad creating our circuit schematic and PCB Board. None of us were experienced in PCB design, so we decided to focus our efforts this week into having a good schematic. Then, at the end of this week and into next week, we worked a bit on the PCB design. Our current schematic has been provided below.

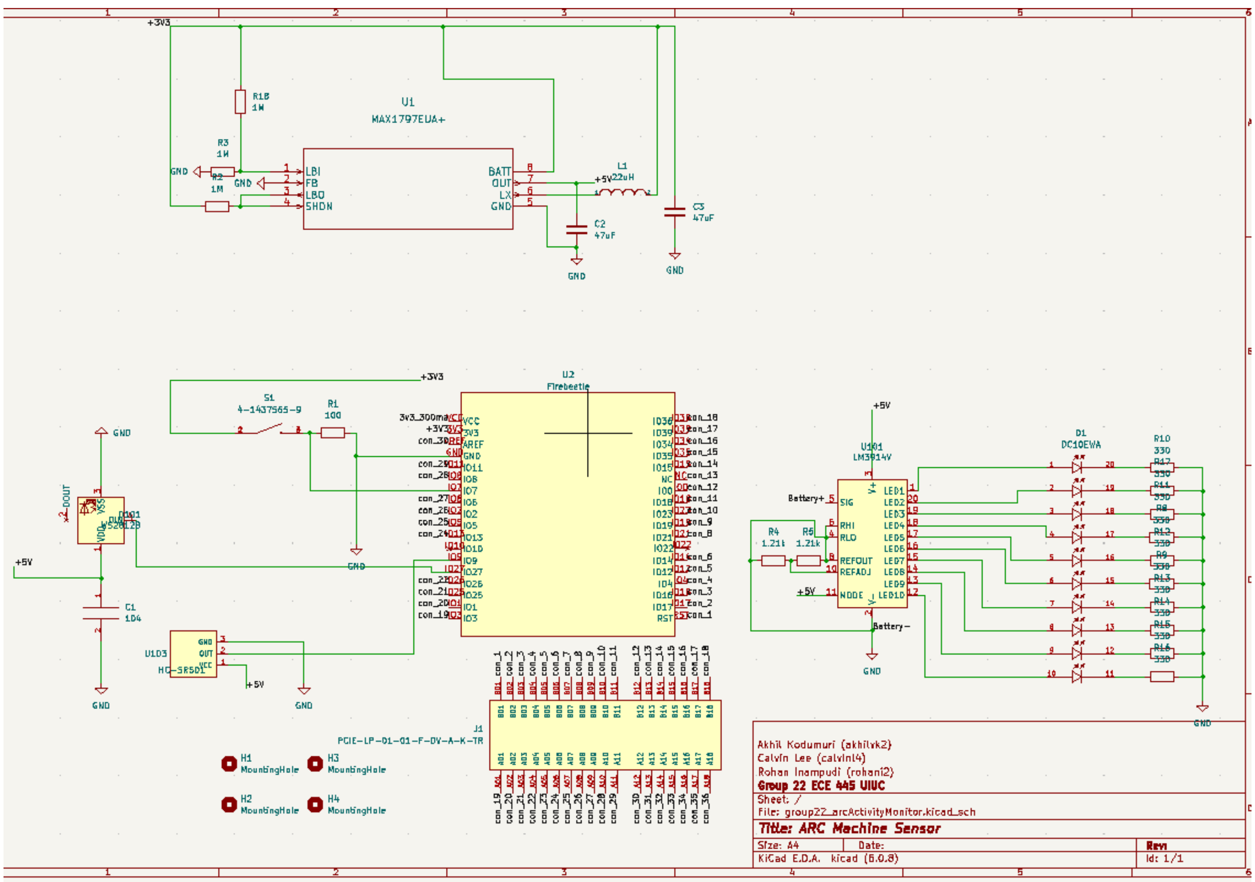


Figure 7: Circuit Schematic v1

We have not placed any part orders yet. We will complete that directly after completing the PCB design to ensure that we have not missed out on any items.

Week 5 10/10 - 10/17

The items due this week include first round PCB orders and the teamwork evaluation. As far as working with my teammates so far, I have had no problems. I have thoroughly enjoyed working on the different aspects of this project thus far with my team! The majority of time spent working this week was on the PCB design. As a group, we had to watch multiple videos and revisit the CAD assignment to learn how to properly construct a PCB on KiCad. We ran into multiple problems at this step of the assignment, largely just because we are inexperienced with the software. But, after learning KiCad, we were able to come up with a serviceable PCB that passed all the checks on PCBway. Some of the problems we faced while constructing the PCB and ensuring it meets the requirements include:

1) Not generating drill file prior to submission
2) Issues with getting some stray wires to ground. Ended up needing to properly update the ground plane.
3) PCB dimensions were initially too big. We had to utilize via's and some surface mount components to reduce the size to 100x100 (mm).
4) Power lines were not thick enough. We needed to revise our power lines to be large enough in width to safely transport power throughout the PCB

The PCB design for the first round PCB order has been provided below:

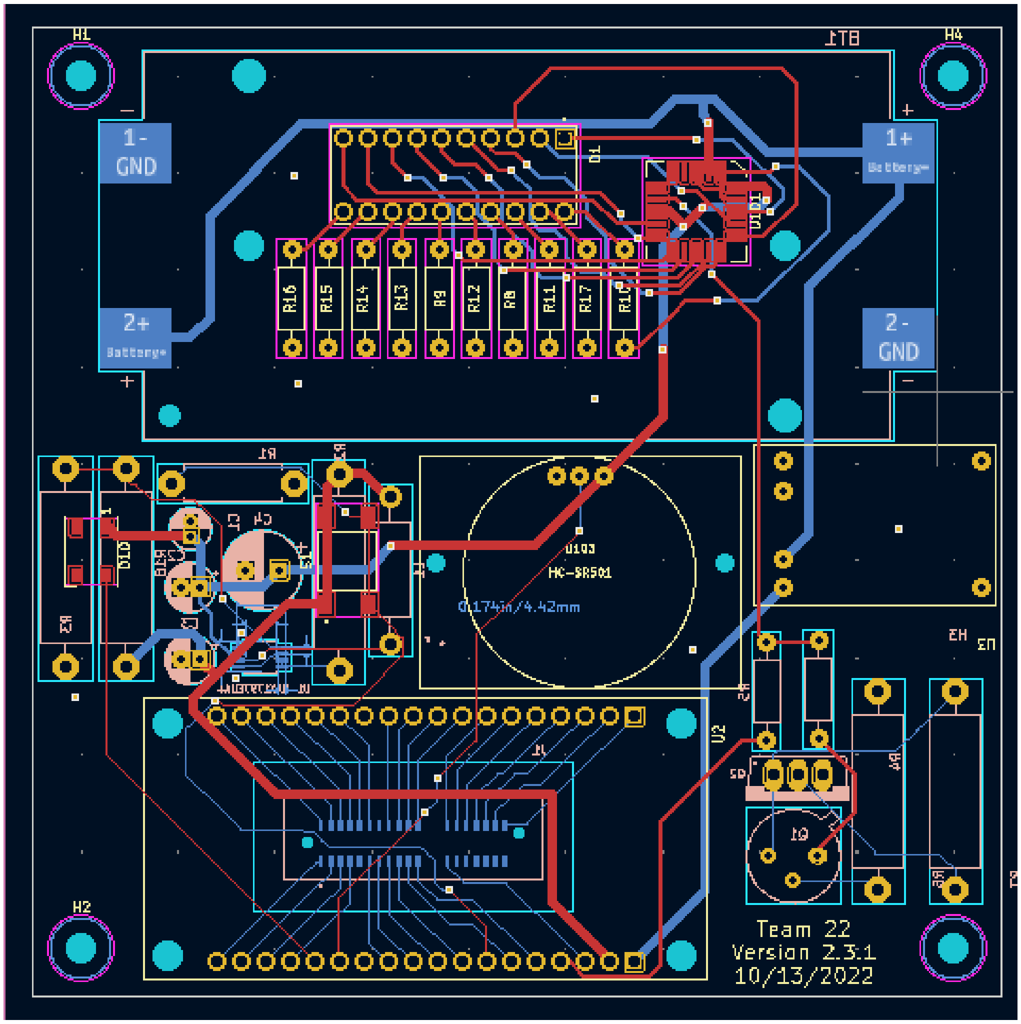


Figure 8: PCB Schematic v1

Week 6 10/17 - 10/24

We spent the majority of this week getting ready for our PCB to arrive. Thus, we began to compile a parts list so that we could place an order. We looked through multiple websites for the best prices. The final list for our first round order consists of the following parts:

Description	Manufacturer	Quantity	Extended	Link

			Price	
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 updated
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
BQ2057CTS Charge Controller	Texas Instruments	4	$1.56	link
TP4056 USB C Li-Ion Charger Module	diymore	1	$9.59	link
8GB Micro SD Card for Raspberry Pi	Verbatim	1	$6.19	link

We estimate all the parts to cost around $150. Due to working hard for the past few weeks, we decided to tone down our workload this week. As a result, we simply put in our parts order and waited for our PCB to come in.

Week 7 10/24 - 10/31

This week we successfully received our PCB board and the supplementary parts. However, we were met with a setback. Professor Gruev notified us that he would like to see a change to our power circuit. Originally, we used the TP4056 USB C Charging Module to help implement the recharging circuit for the lithium-ion batteries. However, after more careful analysis, Professor Gruev recommended that we build this charging circuit ourselves. We agreed to this change as it would help us learn more about how the lithium-ion batteries were being recharged and the circuitry behind it. Instituting this change proved to be a lot of work because we had to look into each chip's datasheet on the TP4056 charging module and learn how to replicate it manually. As a result, we spent a majority of this week on designing the new charging circuit.

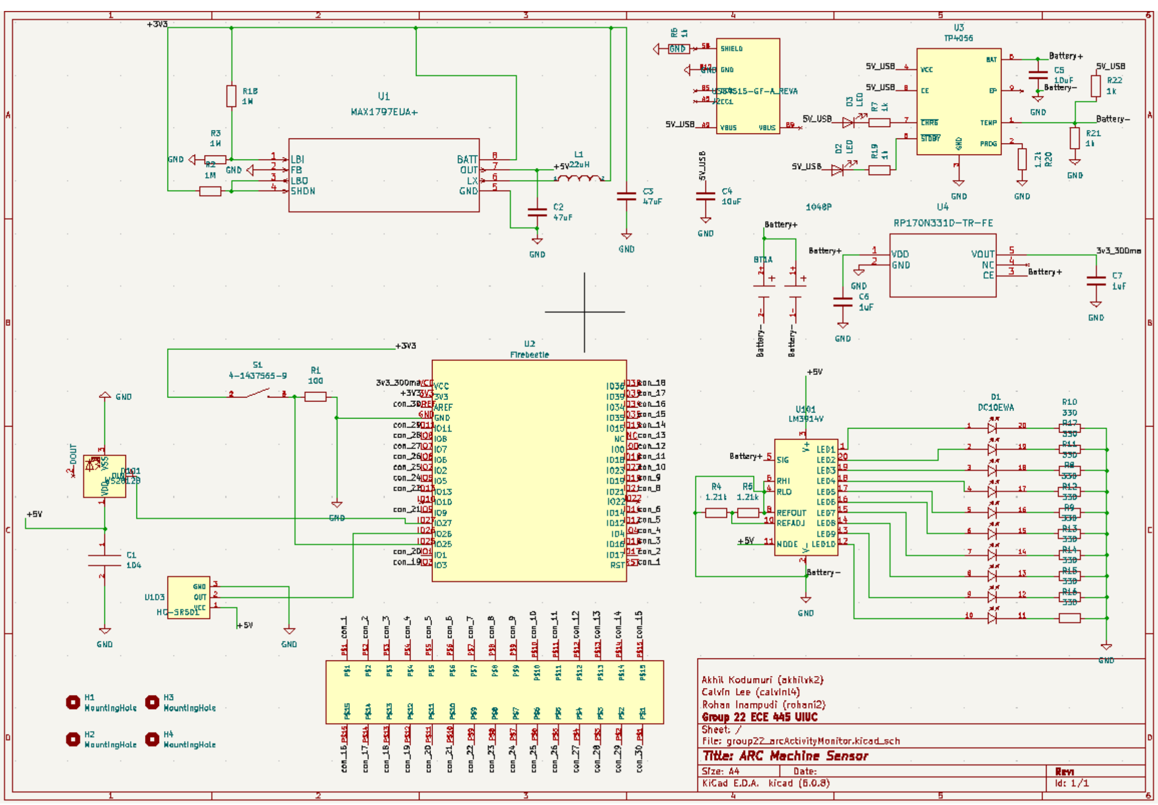


Figure 9: Circuit Schematic v2

Week 8 10/31 - 11/7

Second round PCB orders were due this week along with the individual progress report. I got the individual progress report out of the way early this week. I believe my performance so far has been up to par. I have been involved heavily on creating the circuit schematics for our system. Since we had to create a new PCB in accordance with Professor Gruev's request, we spent all of this week wiring the PCB on KiCad. This process was significantly easier than the first time because we are now familiar with KiCad. Nonetheless, it still took multiple hours to ensure that our PCB turned out the way desired. The new PCB schematic we submitted for the second round orders has been provided below.

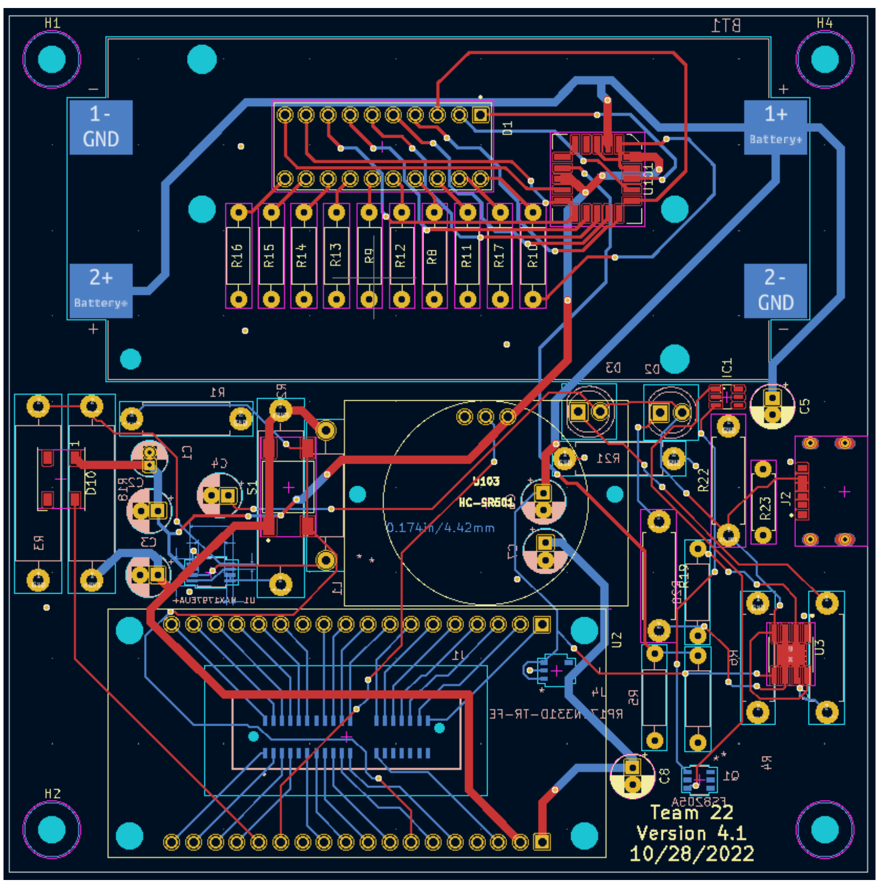


Figure 10: PCB Schematic v2

Week 9 11/7 - 11/14

After further analysis of our circuit schematic and some preliminary breadboarding, we realized that our past two iterations of PCBs would not work. This is because we incorrectly set the GPIO pins for where we receive input. Initially, the pins we fed the button input and PIR output to are restricted pins on the ESP32-Firebeetle that we are using. As a result we spent the entirety of this week redoing the circuitry and PCB schematic. Due to this setback we had to forgo the charging circuit proposed by Professor Gruev. Since mock demos are the following week, we decided to spend as much time as possible this week to get a working PCB. Furthermore, we also started working on the website. I set up a simple locally hosted website utilizing the Flask Python web framework. After many sleepless nights, we ended up with the following circuit schematic and PCB schematic by the end of this week:

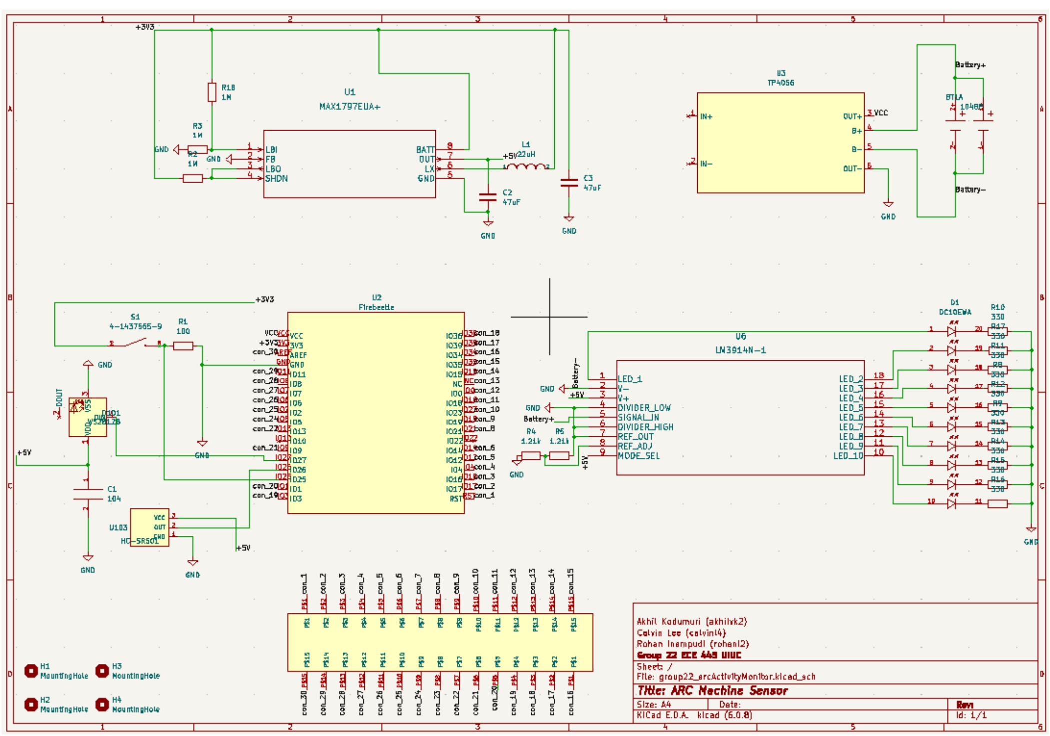


Figure 11: Circuit Schematic Final

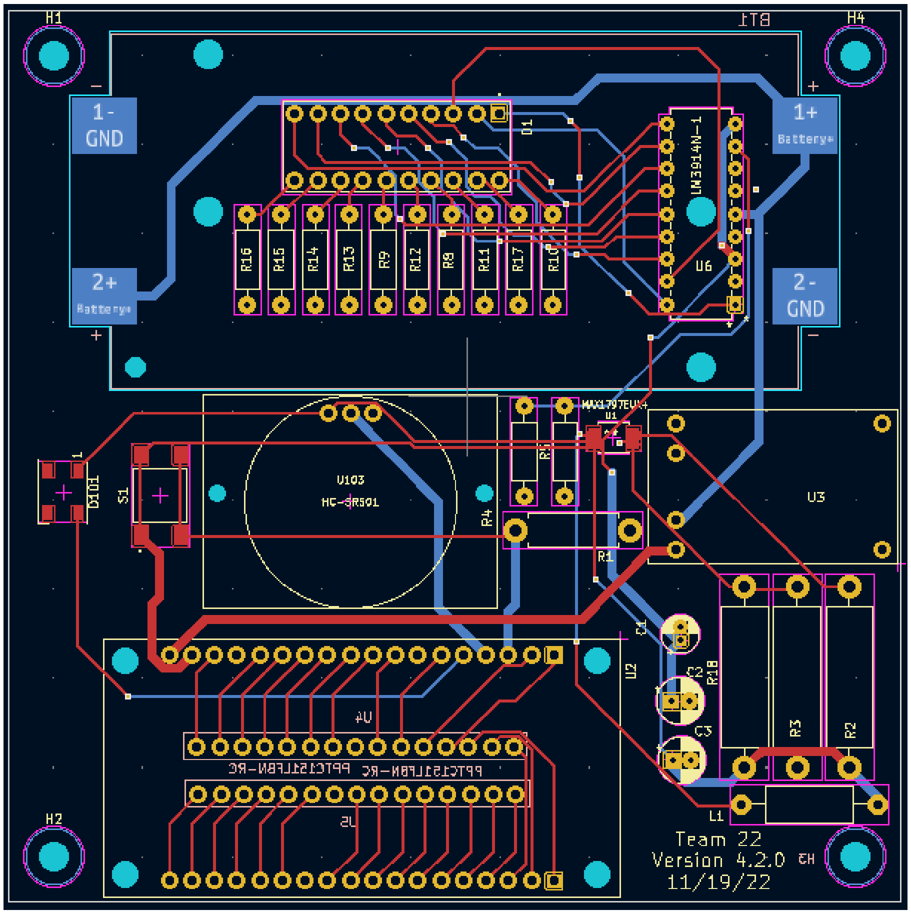


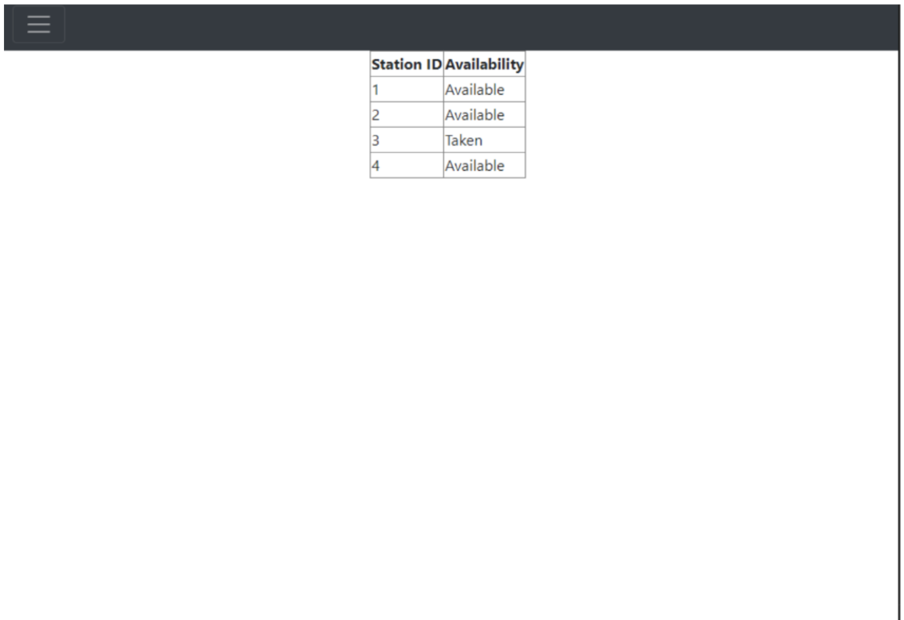
Figure 12: PCB Schematic Final

Week 10 11/14 - 11/21

Much of this week was spent preparing for the Mock demo. Calvin was in charge of soldering the components of the PCB, while Akhil and I spent time on the networking and website side of things. We pretty much had all of the website functionality completed by the end of this week. Unfortunately, after testing our new PCB we ran into even more errors. We realized that our 3.3V to 5V boost converter was stepping up voltage at the cost of current. This resulted in our microcontroller not receiving enough current. This rendered our new PCB as invalid. Although we tried incredibly hard to debug and get this PCB working, there was not enough time to get the PCB working prior to the Final Demo. We submitted one more PCB order to work on during break.

Week 11 11/21 - 11/28

Although this was supposed to be Fall Break, we spent time trying to get our PCB to work. Calvin worked on the PCB from home at his father’s lab. Akhil and I added additional functionality to the website, such as email notifications. The website was able to be fully completed by the end of break. Pictures of the website have been provided.



Station ID	Availability
1	Available
2	Available
3	Taken
4	Available

Figure 13: Availability Page - Website

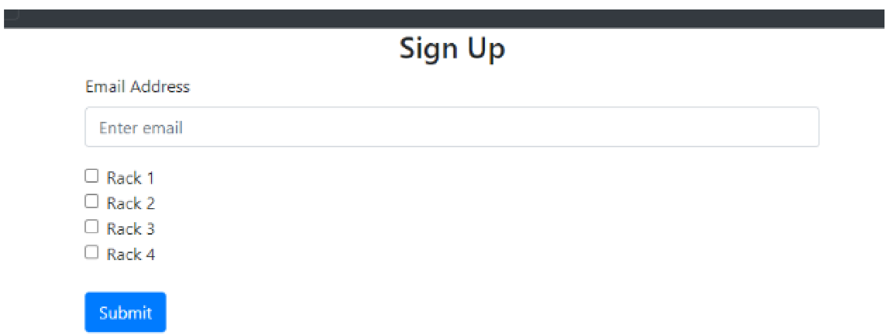


Figure 14: Email Sign Up Page - Website

Week 12 11/28 - 12/5

All of this week, we were preparing for our final demo. Since our final PCB suffered many setbacks and was not working, we decided to breadboard all parts of our design out. We were able to get all subsystems of our design to work together while on the breadboard. This setup is what we ended up presenting during Final Demo. Our most notable setback was not accounting for the fact that our boost converter was a current sink.

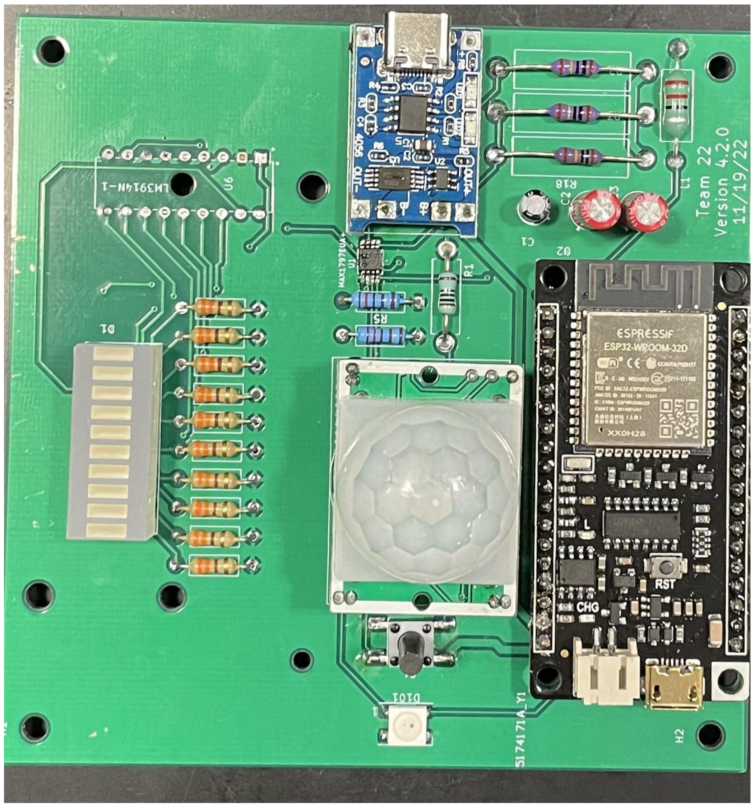


Figure 15: Final PCB

Week 13 12/5 - 12/12

This was the final week of the class. We completed our final presentation on Tuesday and submitted our final paper on Wednesday. This week was strictly preparing for the final presentation and producing the supporting documentation needed to close out our project. Overall, I learned a lot during this class and we were able to get all subsystem functionality working. I am very proud of this group!

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/589828306654039066?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_10260653902913479911&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=1157

24&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].