# ARC Machine Monitoring System

By
Akhil Kodumuri

Individual Progress Report for ECE 445, Senior Design, Fall 2022
TA: Zhicong Fan

31 October 2022
Project No. 22

# Contents

# 1. Introduction

## 1.1 Team Project Overview

The Activities and Recreations Center (ARC) on campus is a popular location on campus where students can take a break from their busy lives and work their stress away. However, one big problem that students deal with are lines to their favorite exercise machine. For this reason, we are building an ARC machine sensor that contains a button a student can press that updates our website where students can view to see what machines are in use. Once movement is detected or a button is pressed, a signal from our ESP32 microcontroller is sent to a centralized Raspberry Pi, then to a AWS server in which our website is hosted and receives updates on. The system will also contain a motion sensor that will detect movement at the machines. This motion sensor provides a fail safe for the scenario a student does not press the button. The use of a machine is also signaled with an LED that will shine on our system, so students at the ARC can see a machine is in use. It should also be noted that our project will sustain multiple ARC machine sensors in parallel which will send their data to a centralized Raspberry Pi, which, in turn, will communicate this information to the AWS server. The overall project can be divided into the following subsystems: Button/Motion sensor to IoT device, IoT device to AWS Server, AWS Server to website, and charging/recharging (Power).

## 1.2 General Overview of Individual Responsibilities

In general, I have done work amongst all of our subsystems. In the following sections, I will go into more detail on my work in each. Within the power subsystem, I designed how our batteries will be recharged. For the Button/Motion sensor to IoT device, I have done research and started working on creating a server that can accept messages from the microcontroller using the popular MQ Telemetry Transport (MQTT) protocol. I have also done similar research and work into how the IoT device will communicate with our AWS server. Finally, I have done research and preliminary work into how our website will be hosted onto a AWS server.

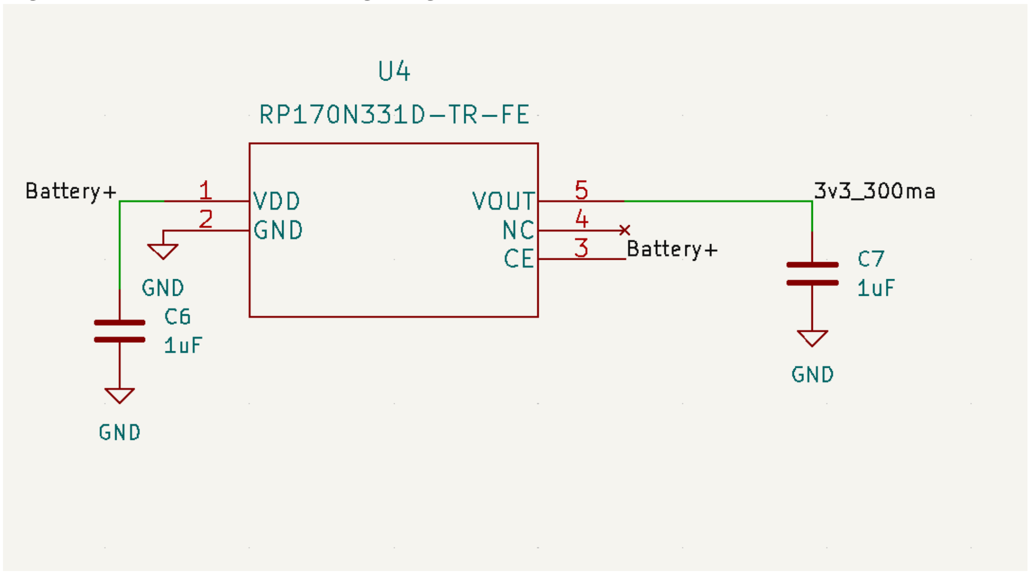
# 2. Design

## 2.1 Considerations

### 2.1.1 Power

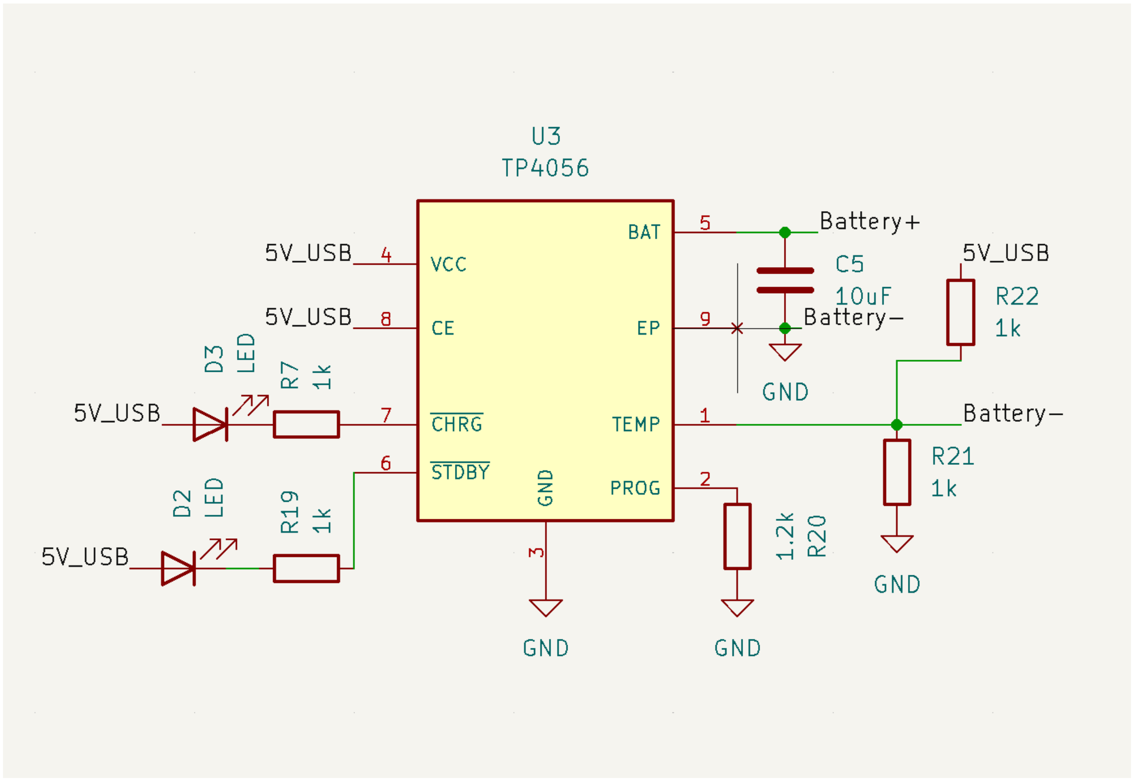
Initially, our group discussed that we should use a TP4056 USB-C charging module system that would provide safe charging to our Lithium Ion batteries. This module would be bought and utilized from Amazon [1]. However, we were advised to redesign this subsystem with our own design. Because of this, I designed how the new charging subsystem would safely integrate with our ESP32. I researched and designed the use of RP170N332D voltage regulator which can be seen in Figure 1. This module allows us to safely supply power to our ESP32 microcontroller. This regulator will also limit the current to 300 mA (which could be altered with the capacitor values), so the optimal amount of current can be fed into our ESP32. In my research, I found that 300 mA is enough to satisfy the ESP32 in active mode [5].

Figure 1RP170N332D voltage regulator



Another point of research was using the TP4056 for charge protection. Since we were advised not to use the TP4056 charging system from Amazon [1], I researched and designed a new charging system that was similar to the module on Amazon. The design uses the TP4056 for charge protection. This new design is 90% of the cost and takes up less space. I read the TP4056 datasheet and designed a circuit that will allow our batteries to be safely charged [7].

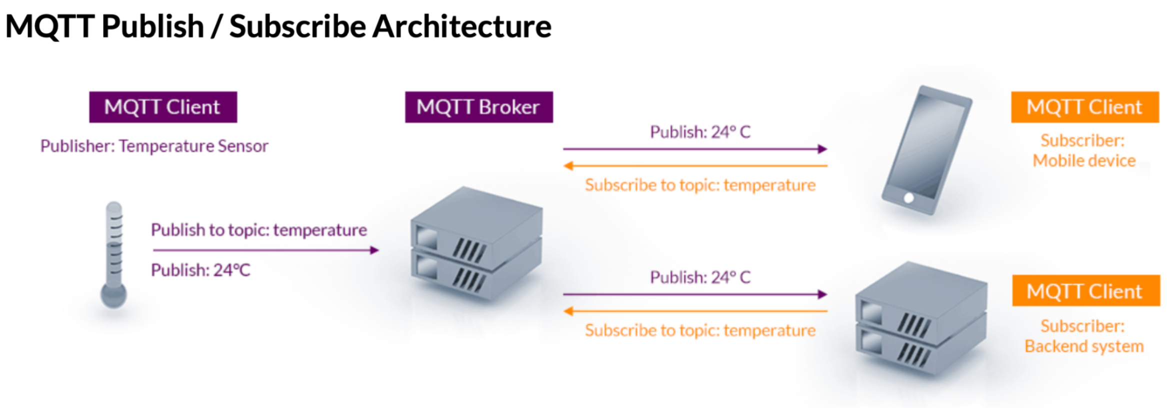
Figure 2 TP4056 Charging Module



## 2.1.2 Network Protocol

One important design consideration that was made is the use of the MQTT network protocol. MQTT is a standard network protocol that is used for IoT messaging and communication [6]. Unlike HTTP and other application level network protocols, MQTT allows two way communication. This allows bidirectional communication between clients and its server. Messages are broadcasted in a manner suited for multiple devices connected to a network. Another inherited feature of using the MQTT network protocol is that it is lightweight [6]. Because MQTT demands little resources from its clients, it is the most optimal for small subsystems. MQTT messages also use headers that are very small in size so they can utilize low network bandwidth [6]. One design consideration our group talked about was creating the smallest PCB design and power footprint possible. This prompted the research into lightweight network protocols. MQTT is the perfect protocol for our use case because our system requirement requires multiple working ARC machine systems. I have also found multiple tutorials and guides on creating MQTT broker (server) [2]. The MQTT network protocol requires the following model in order to be used: A Publisher and Subscriber connected through a Broker. In this model, both Publishers and Subscribers are used as clients, while the broker is used as a conventional server. The most important feature of the MQTT network protocol is the transmission of topics across ports of communication. Communication is specified and organized as topics in order to distinguish and better clarify why the data is being sent the way it is. The theoretical organization of messaging using MQTT can be seen in Figure 2.

Figure 3. MQTT Network Architecture [6]


MQTT Publish / Subscribe Architecture
MQTT Client
Publisher: Temperature Sensor
MQTT Broker
Publish to topic: temperature
Publish: 24°C
Publish: 24° C
Subscribe to topic: temperature
Publish: 24° C
Subscribe to topic: temperature
MQTT Client
Subscriber: Mobile device
MQTT Client
Subscriber: Backend system

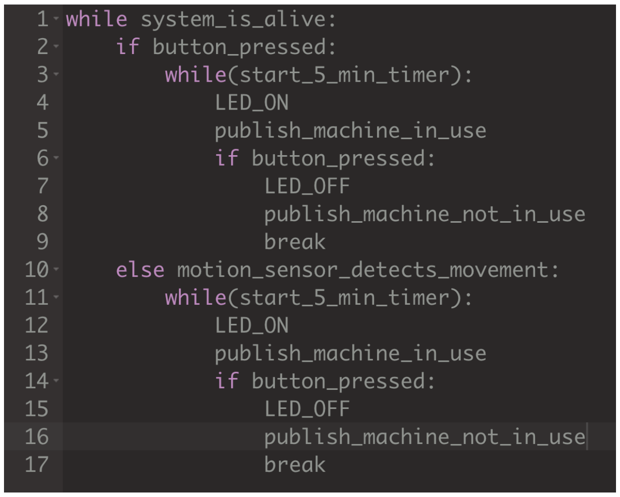
### 2.1.3 Button/Motion Sensor to IoT Device

Originally, we presented that we would use the Raspberry Pi 3 B, however, all major Pi distributions were out of stock. So, I did research into finding another acceptable Pi. I found that with little effort the Raspberry Pi 3 B+ can be used as an adequate substitute. I also conducted research on how the Pi can be made into a server that can accept MQTT requests and provide MQTT responses. I conducted heavy research into how the connections will work with the ESP32 and the Pi. As discussed in the previous subsection, I conducted much research into how to apply the Publisher, Broker, and Subscriber structure to our project. Figure 3, depicts the structure that was created from this architecture. We are using the Firebeetle development board to interface with the ESP32. The ESP32 will be programmed to publish topics based on activity from the button on the ARC Machine module or motion sensor.

The ESP32 on our ARC machine system will contain logic that will determine when and how the centralized Raspberry Pi will get information when a machine is in use. I came up with the following logic to be programmed onto the ESP32 which gives pseudocode into when packets will be sent to our MQTT server. The PIR motion sensor is designed in such a way to always output a high signal, 1,  if it detects any movement. Thus, it is always conducting power. Since, the PIR motion sensor will be included on top of the ARC machine system in order to detect the use of an equipment if a student/user has not pressed the button, I have designed logic to be used to determine the correct course of action the ESP32 must take a publish a topic to the Pi. Figure 3 shows the logic I created that will be in use when publishing the information about the ARC machine system.

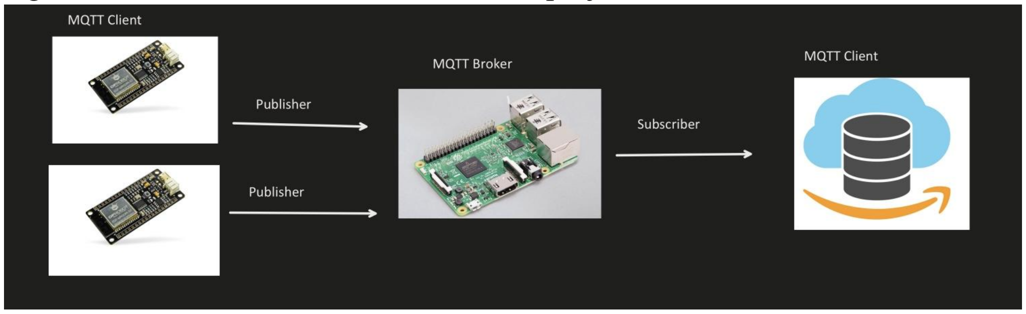
Also discussed in the previous section, each system using the MQTT network protocol requires the need of a broker. This will take the place of the Raspberry Pi 3 B + which can be seen in Figure 5.

Figure 4 Pseudocode for ESP32 decision making



```
while system_is_alive:
    if button_pressed:
        while(start_5_min_timer):
            LED_ON
            publish_machine_in_use
            if button_pressed:
                LED_OFF
                publish_machine_not_in_use
                break
    else motion_sensor_detects_movement:
        while(start_5_min_timer):
            LED_ON
            publish_machine_in_use
            if button_pressed:
                LED_OFF
                publish_machine_not_in_use
                break
```

Figure 5. General Network architecture of project



## 2.1.4 IoT device to AWS Server

In order for our website to be used by a mass amount of students, I have done research into using Amazon Lightsail [8]. Amazon Lightsail is designed for web servers that do not require more than 5 servers. I have already looked into the services it provides and how to make an account. I have also made sure that the Raspberry Pi used in this project is compatible to send subscriber based information to the Amazon Lightsail server [4].

# 3. Testing

## 3.1 Power

Since the PCB orders recently arrived, the only way I verified that the TP4056 chip was being utilized was with the following formula from the datasheet (seen in Figure 6) [7]. This formula provided insight into what resistors needed to be connected to the TP4056. There are also tables within the TP4056 datasheet that provided proof that the connections that were being made were correct. Also, before I start soldering on the PCB, and all the necessary parts are delivered, I plan on routing the appropriate connections on a breadboard first. This ensures that all the parts will be used correctly. Also, if any resistor or capacitor values are incorrect, they can be easily swapped out. If there are any big issues that arise with our ESP32, voltage regulator, or board design, they can be seen faster with the breadboard, so time is not wasted de-soldering anything. In the week of October 31st, I plan on putting the power subsystem on the breadboard, and by the end of the week, I will have soldered it onto the breadboard. A voltmeter and ammeter can also be used to ensure that each component gets the expected power.

Figure 6.

$$I_{BAT} = \frac{V_{PROG}}{R_{PROG}} \times 1200$$

## 3.2 Networking

As described in section 2.1.3, I have done work in setting up the MQTT broker onto the Raspberry Pi 3 B+. In order to test if the broker works as expected, I followed the following steps from this tutorial [2]. I ran the following command on the Raspberry Pi command line to start the Pi as a broker.

```
sudo systemctl enable mosquitto
```

Since our ESP32 hasn't arrived yet, I will set up my Macbook as a MQTT client (publisher and subscriber) to check if it is possible to use the Pi as a broker. In the week of November 6th, I will have finished setting up the network connections necessary for the Pi.

Once the ESP32 arrives, I will follow the following tutorial [7] in order to configure the communication between the ESP32 and Raspberry Pi. By the end of the week of November 6th, this communication I plan on finishing the connection between the two.

# 4. Conclusion

## 4.1. Self Assessment

One aspect that our group excelled in was creating an equal work environment. We ensured that each member was satisfied with the amount of work they were expected to accomplish. Thus far, I did accomplish the work that was expected of me individually. As can be inferred from this document, much of the schematics and electrical work was done together as a group, while much of the external aspects (networking, website work, etc) was split between the us. My tasks included understanding how the network connections between our subsystems would work. I was successful in this task, and I already started configuring them. I also was able to design a new power system when we received criticism from the professors about our previous one. Currently, we are on schedule to finish a mock of our system before "Mock Demo".

## 4.1. Plans

In order to satisfy the high level objectives of our project, we need to be able to support multiple ARC machine sensors working in parallel, multiple people should be able to utilize our website, and there should be real time updates to our website and locally (people should be able to know whether or not a machine is in use in a quick and timely manner). So realistically, I would like to make sure that the ARC Machine sensor is able to be used simultaneously at 2 different machines. What that means for my work individually, is the Pi should be able to handle publish packets from 2 different ARC Machine sensors, and communicate them to the AWS server. My goal is to have this all done before demos in the week of November 14th. With regards to the Power subsystem, we have already placed an order with the TP4056 charging system bought from Amazon. So, by November 14th demos, I would like to have this system working with no errors. However, as previously mentioned, I designed a new charging system using the TP4056. This design was submitted to be ordered in the week of November 1st. My ambitious objective is to have this PCB be the final design for the November 28th final demo.

## 5. References

[1] “Amazon.com: Hiletgo 3pcs TP4056 Type-C USB 5V 1A 18650 lithium battery ...” [Online]. Available: https://www.amazon.com/HiLetgo-Lithium-Charging-Protection-Functions/dp/B07PKND8KG. [Accessed: 02-Nov-2022].

[2] Author Lee, L. A. veteran programmer, Lee, A. veteran programmer, E. F. says: B. says: L. says: M. 26, and L. says: “Introduction to IOT: Build an MQTT server using Raspberry Pi,” *App Code Labs*, 10-May-2019. [Online]. Available: https://appcodelabs.com/introduction-to-iot-build-an-mqtt-server-using-raspberry-pi. [Accessed: 01-Nov-2022].

[3] 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: 01-Nov-2022].

[4] F. Németi, 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. [Accessed: 01-Nov-2022].

[5] 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/. [Accessed: 01-Nov-2022].

[6] “The standard for IOT messaging,” *MQTT*. [Online]. Available: https://mqtt.org/. [Accessed: 01-Nov-2022].

[7] “TP4056 1a standalone linear Li-Lon Battery Charger with thermal ...” [Online]. Available: https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf. [Accessed: 02-Nov-2022].

[8] “Websites,” *Amazon*, 2005. [Online]. Available: https://aws.amazon.com/websites/. [Accessed: 01-Nov-2022].