MEC ENG 100 :: Spring 2021 Final Report

PROJECT TITLE: Pckt Refill (read as “Pocket Refill”)

PROJECT MEMBERS: Kai Yun, Aaron Wagner

PROJECT OBJECTIVE: Develop an IoT device that estimates the remaining amount of hand sanitizer in a dispenser and alerts the user when in need for refill.

I. Introduction

(i) COVID Background

Global Pandemics are few in number, and COVID-19 has upended multiple industries. With that, health safety standards have exponentially increased, especially with disinfectant products. As the goal of the project was to create an Internet of Things device for a real-world issue, it first must understand the importance that this device may play in today's world.

When the initial wave of COVID hit prior to cancelling in-person events, a heavy emphasis was placed on ensuring hand sanitizer had been made available in a majority of buildings, and at college campuses, specifically at UC Berkeley, with the use of automatic hand sanitizer dispensers. And as COVID became more prevalent, automatic hand sanitizer dispensers purchases increased, and more and more industries began to employ, and enforce individuals to use these automatic dispensers. With such a large increase, hand-sanitizer companies were producing at maximum capacity, a capacity that was not meeting the standards that COVID started to demand. Accordingly, we saw companies outside of the hand-sanitizer industry circumvent their own product line to meet the demands that were not being satisfied. Eventually, the hand-sanitizer industry adapted and improved their manufacturing process, and as such, the market has exponentially increased, and has become a relied upon factor for this fight against COVID.

(ii) Concept

With the understanding of the demand that hand-sanitizer has and will have in the years to come, we are able to recognize the issue that needed to be addressed. With automatic hand-sanitizer dispensers facing unprecedented usage, it was common, especially within UC Berkeley, to approach, and receive nothing from a dispenser, that would often go unfilled for days at a time.

As this being common amongst students and staff, we concluded there may have been a disconnect between the janitorial staff and the dispensers in that it became too troublesome to check the multitude of dispensers that UC Berkeley has. Thus began the evaluation process of how to best find the solution to this predicament, especially with the importance it has in the fight against COVID.

Consequently, our general solution is to create a device that measures the usage of each individual hand-sanitizer dispenser and returns an email notification to the janitorial staff

when any dispenser needs to be refilled. To be more detailed, we wanted a device that would measure each usage by a count system that would identify when a hand is placed under the dispenser nozzle. Following this, the count respectively subtracts one usage amount of hand-sanitizer fluid from the actual capacity of fluid that is set by the janitorial staff. For example, the janitorial staff may refill a dispenser, and in using an app, they set the fluid amount of the dispenser to 100 milliliters. The aforementioned count would subtract one usage amount of fluid, let us say 2 milliliters, from the manually set total of 100 and will store this value and title it as the 'amount left' until the next usage and so forth. When the amount left reaches a specified value, a notification will be sent to staff alerting them that this specific hand sanitizer, let us call it Etcheverry Hall #1, needs to be refilled. The notification will be sent at an optimal time that allows the janitorial staff to view, prepare, and then refill the dispenser which minimizes the amount of time when the dispenser is empty. When refilled, the staff would set the fluid capacity value again, and the process would be repeated.

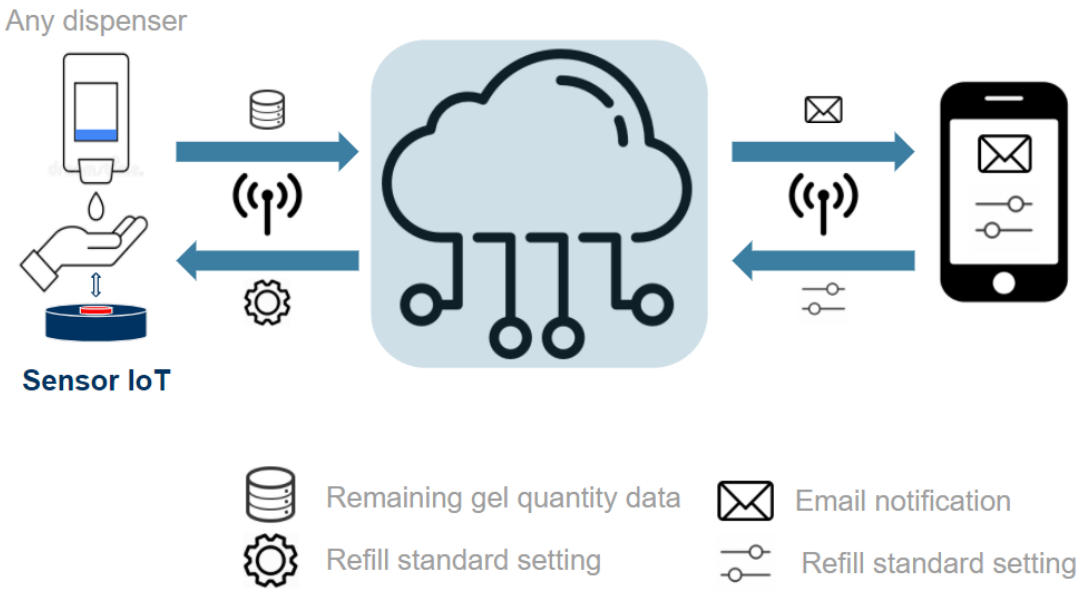


Figure 1. Overall concept/structure of Pckt Refill device and its services.

(iii) How it relates to ME100

At the beginning of this project, there were a few guidelines that had to be followed:

- Application Need
- At least 1 External Sensor
- At least 1 Actuator

- Computing
- Wifi, MQTT, IFTTT

The significance of this is that each parameter requires us to employ what has been taught throughout this past semester in ME100. Starting with the Application Need. In the first few lectures, we discussed the concept Internet of Things and the importance this plays in day-to-day life. Thus with our new understanding, we were able to find a real-world application in the need to rectify the hand-sanitizer dispenser's constant state of being empty, as it would require wireless communication to notify the janitorial staff.

Next, we have the external sensor, our choice being the ultrasonic sensor. As we were taught about circuits, the importance of grounding and resistors, we were able to successfully wire our ultrasonic sensor to our ESP32. And through our labs that required collection of data from these sensors, we had the ability to incorporate our ultrasonic sensor in the computing aspect of our project.

Moving forward, our very first lab dealt with an actuator, and to make it ever so slightly more difficult, we incorporated several actuators, all the same part, just with a different purpose. And again with our knowledge of circuits and the need of resistors, we successfully wired several LED parts in our project. It is important to begin discussing the communication aspect now, as again, in our code, we were able to signify when the yellow LED should turn on, as the ultrasonic sensor triggers it, and the ESP32, having flashed our code onto the part, communicates with the red and green LED by actively turning it on or off.

After that, we have the computational aspect of our project. With our experience by completing the labs and learning python, as previously mentioned, we had the ability to successfully initiate the led lights by communicating with the different ESP32 pins. In addition, we were able to communicate with our Ultrasonic sensor and to internally set, and subtract values for the hand-sanitizer fluid.

Lastly, in our first lab we connected the ESP32 to wifi and remotely communicated with the ESP32 in Lab 5. We chose to go with MQTT communication protocol, AdaFruit IO MQTT Broker, and IFTTT and applied our knowledge from Lab 5 to successfully have our ESP32 connect and communicate with the necessary individuals over the internet.

II. Circuit Design and Components

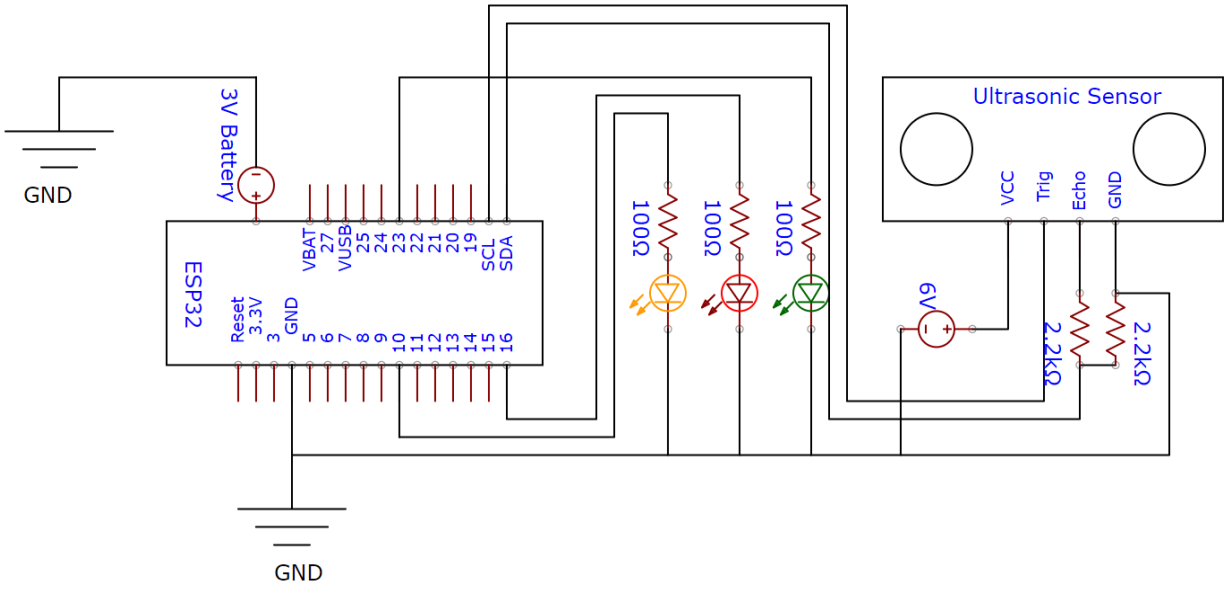


Figure 2. Circuit Diagram of the device.

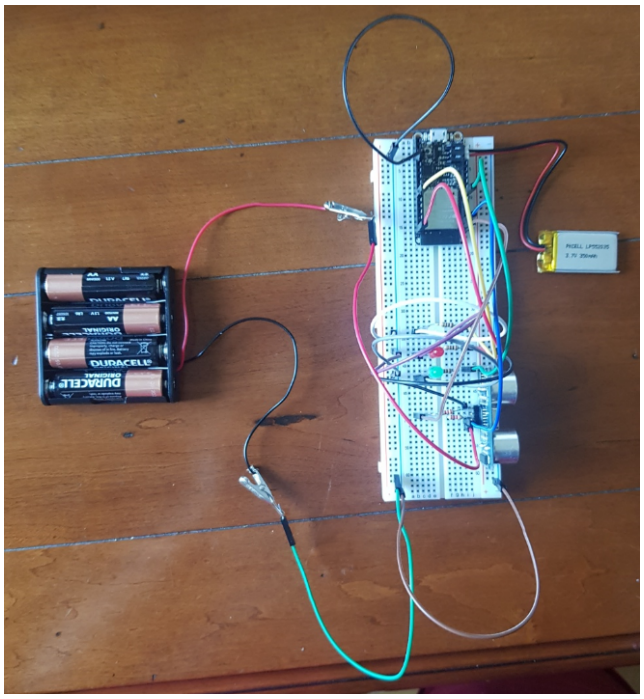


Figure 3. Completed device.

In this section, the basic key components of the circuit are explained, including the reasons these specific components were chosen. Please refer to the circuit diagram in Figure 2. (Some of the following sections include trials and error explanations when using the component)

(i) ESP 32

_____An essential element of this project, and probably the most important was the ESP32. As taught from the course, we had flashed our code onto the ESP32, and then wired the rest of the circuit accordingly.

To look a little further at the wiring between the ESP32 and the rest of the circuit, let us first recognize the power. To make the device remote, we have to power the ESP32 with our 3.7V Lithium Battery supplied by UC Berkeley. Powering for the ultrasonic sensor is explained in the "6V Battery" section.

Next, for the LEDs, we chose GPIO pins 4, 23, and 27 to connect with the ESP32. As previously mentioned, we utilized the ESP32's ground and connected it to the LED's and Ultrasonic sensor. Lastly, for the Ultrasonic Sensor, we connected both the ESP32's SCL and SDA. Overall, the ESP32 is essentially the brain of the project, communicating via MQTT, running the code constantly to measure usage, and actively turning on and off the LED components.

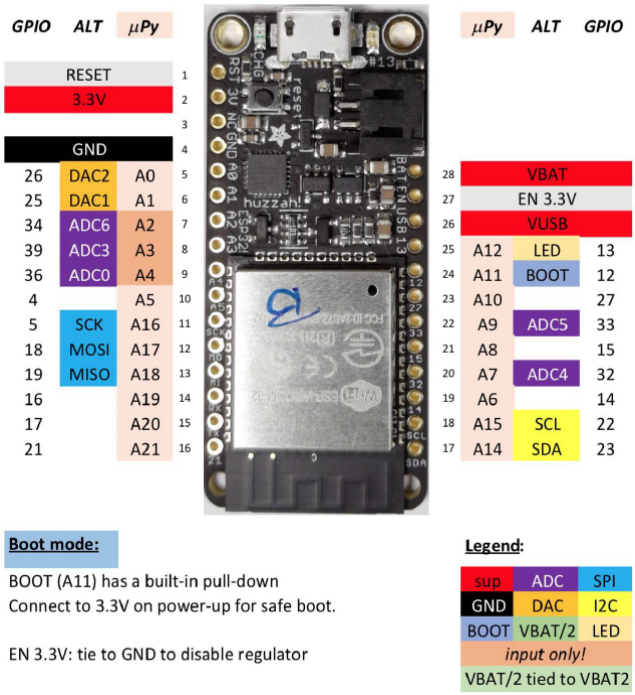


Figure 4. Assembled Adafruit HUZZAH32 – ESP32 Feather Board.

(ii) Light Emitting Diodes

As one of the parameters of the project dealt with actuators, the Light Emitting Diodes best fit our project needs. As the read-world issue that we wanted to solve dealt with approaching, and consequently failing to receive hand-sanitizer for a dispenser, we wanted to remove this all together by displaying to the users of when the dispenser was empty (red), when it can be used (green), and when it is currently in use (yellow).

Figure 5. Light-emitting diodes (red, green, and yellow).

The circuitry that was involved with the LEDs is straightforward, we have to reduce the current going to the LEDs, since they may burn, and thus we appropriately inserted 100 ohm resistors for each LED. Then we ground it by connecting to the ground on the ESP32.

(iii) Ultrasonic Sensor

The key to the success of this project was to figure out a simple method to measure the remaining amount of hand sanitizer in the dispenser. Considering that the device should be easily attachable and cheap, an ultrasonic sonar distance sensor came out as the most appropriate component to measure the approximate amount of hand sanitizer left. If a weighing scale were to be used, we would have had to attach a weighing plate to the bottom of the dispenser which would have really complicated the structure and made the attachment of the device to existing dispensers difficult and expensive.

The ultrasonic sensor constantly measures the distance between the device and the dispenser. When someone reaches out their hand to use the dispenser, the distance between the dispenser and the device decreases. We have set this distance to be 10 cm, but this can be adjusted by changing one line of code. The average pump dose for most hand sanitizer dispensers is approximately 2 milliliter. Thus, 2 milliliter is subtracted from the initial total amount of hand sanitizer in the dispenser everytime the ultrasonic sensor detects a distance decrease to a certain standard. This “standard distance” should be set by the user depending on the

dispenser's structure. This can also be set with one line of change in the code. (Such customizations are some of the things we need to improve to make this real-world applicable.)

Figure 6. HC-SR04 Ultrasonic Sonar Distance Sensor.

HC-SR04 Ultrasonic Sonar Distance Sensor provided in the lab kit is used. Since it requires at least 5V, we used a 6V battery as described below. This power source for the ultrasonic sensor is separate from the one for the ESP32 board itself since the ESP32 board does not support 6V and has a 3.3V voltage regulator. In one of our trials, we actually connected a 6V battery to pin 28 (VBAT) of the board, which fried one of our ESP32's voltage regulator. Another important thing to consider was which resistors to utilize when connecting the Echo pin of the sensor to the SDA pin of the board. The Echo pin sends back voltage to the board to alert when a measurement is made, and without resistors, it would send all of its voltage back, which will most certainly cause the board to break. Thus, we utilized a voltage divider to limit the amount of voltage coming from the ultrasonic sensor down to ~2.5V to 3V.

(iv) 6V Battery

As explained above, the ultrasonic sensor requires a voltage source of at least 5V to power it. Thus, we have chosen the 6V battery case shown in Figure 7. This holder connects 4 double-A batteries in series, each having 1.5V, thus producing 6V in total. The input voltage is connected to the Vcc pin of the ultrasonic sensor.

Keeping in mind that the ultrasonic sensor operates on 5V input, we tried to limit the voltage to 5V with a voltage divider made up of 1 kOhms and 4.7 kOhms. This did not produce good results. The ultrasonic sensor kept measuring distances of 5cm even when there were no objects anywhere near it. The software was not the issue since it produced wanted results when the power source was the laptop via USB cable. After talking with Dr. Anwar, we realized that we made the most elementary error, mainly due to our lack of understanding. Voltage dividers limit the current as well as the voltage and with those two resistors connected in series, the

current would have dropped to approximately 1mA. Thus, the ultrasonic sensor was producing unreliable readings of the distance.

Using zener diodes was recommended by Dr. Anwar. However, before testing the diodes, we wanted to see if just the 6.2V we had is good enough to power the ultrasonic sensor with accurate readings. And to our surprise, it most certainly did and we opted to just use the 6.2V battery connected without any voltage limiter component.

Figure 7. 6V battery case. This connects 4 AA batteries, each 1.5V, in series to generate 6V.

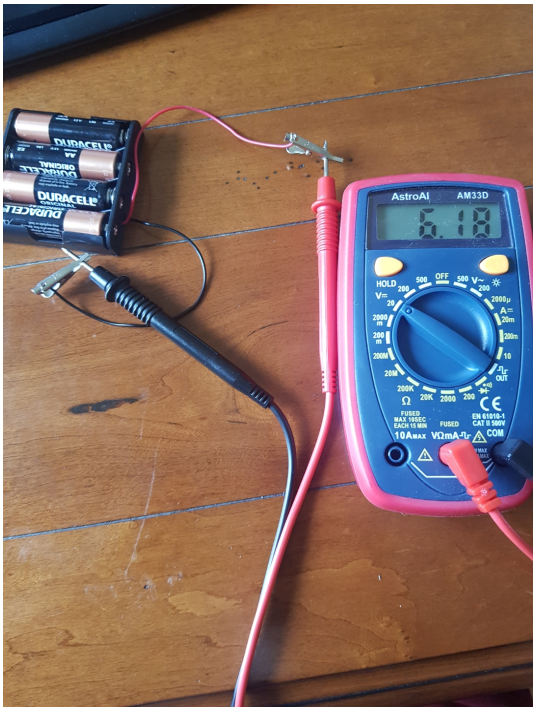


Figure 8. Four 1.5V batteries are connected in series to output ~6.18V.

III. Software: Computation and Connection

The Message Queuing Telemetry Transport (MQTT) is used as the communication protocol between the device and the server. Pckt-Refill has two main services. First one is sending an email to the user when the dispenser is out of hand sanitizer. The second one is allowing the user to set the “initial amount”, i.e. the amount of hand sanitizer in the dispenser right after being filled by the user, of hand sanitizer in the dispenser. In fact, the user should be able to set many different options using their phone, but since this project’s objective is proof-of-concept, we have limited the functionality to just setting the initial amount. All of these can be done by sending simple text messages between the publishers and subscribers which can be achieved with MQTT the easiest for it is optimal at lightweight communication and is free.

We are using the Adafruit IO MQTT Broker since it provides great user-friendly dashboards. Specifically, we are using the toggle bar block in the AdaFruit IO dashboard to allow the user to set how much hand sanitizer initially fills the dispenser. This dashboard interacts with the “pckt-refill” AdaFruit feed to send the initial amount data to the IoT device. Currently, the toggle dashboard can have inputs in the range of [50, 100], i.e. the user can set the initial amount of hand sanitizer in the dispenser to any number between 50 milliliter and 100 milliliter. This is very small compared to a real-life dispenser’s capacity, but we chose this range since having too big a number for demonstration would take forever to showcase the moment hand sanitizer falls short.

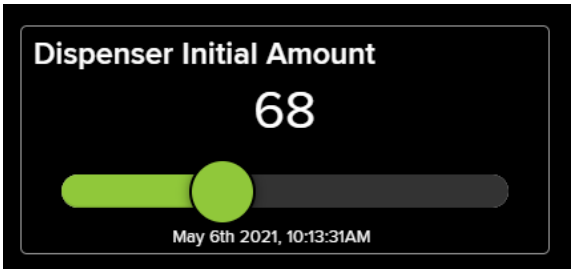


Figure 9. Toggle bar in the AdaFruit IO dashboard. By sliding the dot, the user can set how much hand sanitizer initially fills the dispenser.

Whenever the device counts a usage, it computes how much is remaining in the dispenser. The relevant code is shown below. This is a very simple computation mechanism, but

does the job effectively. (The specifics can be found from the code at https://github.com/kaiyun717/Pckt-Refill/blob/main/pckt_refill.py)

```
def return_amount_left(self):
    """
    Calculate how much sanitizer is left and return the value.
    :return: amount_left
    """
    self.amount_left = self.initial_amount - self.number_used * self.per_usage

    return self.amount_left
```

Figure 10. `return_amount_left` method in the `Dispenser` class. This accounts for the computation part of this project.

Once the `amount_left` is equal to or smaller than the `refill_standard`, the device publishes a message to the AdaFruit feed stating “Dispenser empty. Need REFILL!” (Note that the `refill_standard` is set as percentage, i.e. if `refill_standard` = 10, it means that when the `amount_left` is below 10% of the `initial_amount`, the device recognizes that as a need for refill.)

We then use IFTTT (If This Then That) applet that monitors the “pckt_refill” feed on Adafruit IO and sends an email to the user when it detects “Dispenser empty. Need REFILL!” In other words, an email is sent to the user whenever the dispenser has less than the `refill_standard` percentage of hand sanitizer left. Thus, this completes the full Pckt Refill service.

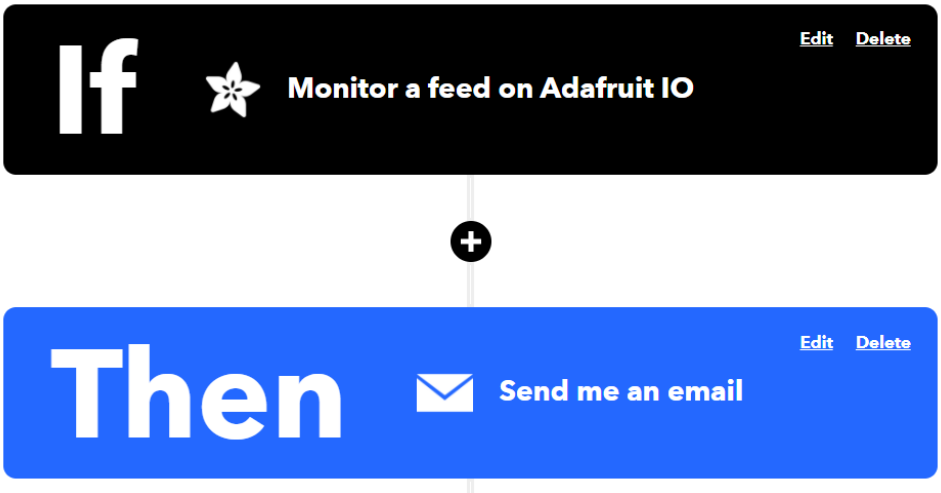


Figure 11. IFTTT applet that monitors “pckt_refill” feed on AdaFruit IO and sends an email to the user when the dispenser is out of hand sanitizer.

IV. Test Results

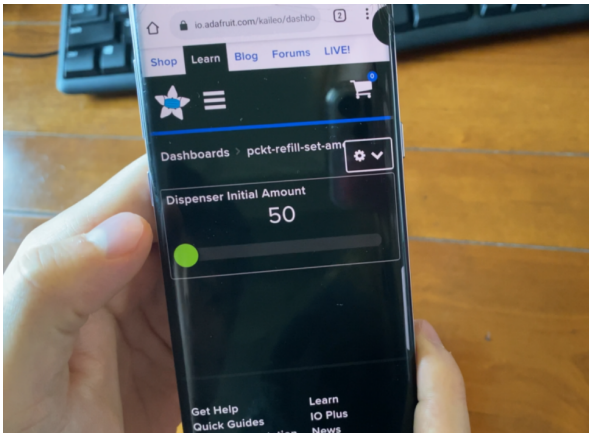
Device Demonstration:
https://drive.google.com/file/d/15Vhqhs-s5xEQFrBj75mmauE7lBEffv5K/view?usp=sharing

Command Prompt (Putty) Screen:
https://drive.google.com/file/d/1TRhy4lNu1gL9eA4ickBT8dYKS5bRi3pK/view?usp=sharing

We successfully made the circuit to run remotely via telnet. The final demonstration videos can be found from our submission. There are two videos: one shows the screen and the other shows the circuit.

Due to our limited budget, we were not able to purchase an actual dispenser. Thus, we showcased our device by simply putting a hand in front of the sensor, which counts the decrease in distance to a certain amount as a “usage.” We believe this clearly shows that our device works as intended and will work accordingly when used with a real dispenser. Moreover, when the hand is in front of the sensor, the sensor continuously counts multiple usages. This was implemented since there are many people who utilize the dispenser multiple times in one setting. Having the sensor continuously count usages as long as the hand is in front of it accounts for such real-life scenarios.

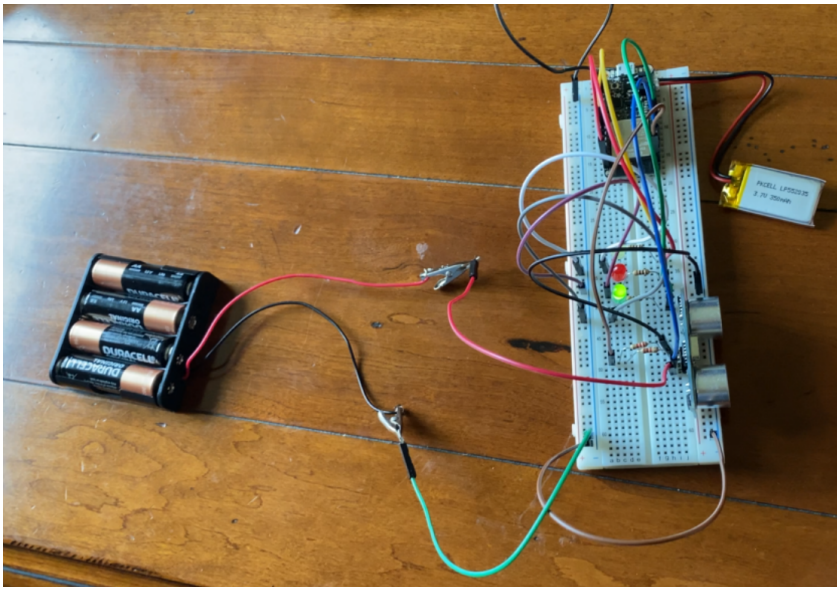
After making the connection with AdaFruit IO, we can set the `initial_amount` with our phone as shown below. In the test case, we set it to 72 milliliter.



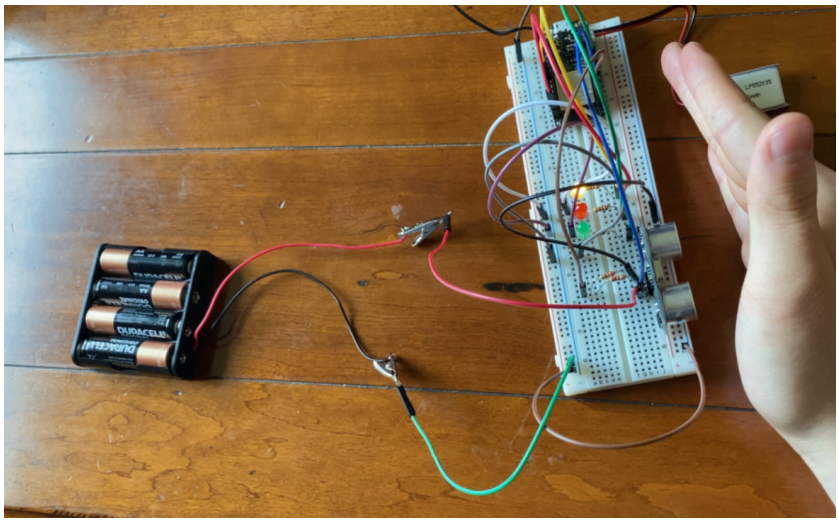
```
Connected to WiFi at IP 192.168.35.243
Connecting to Adafruit...
AdaFruit connection made at: kaileo
Connection made with Pckt-Refill interface.

Waiting for the user to set initial amount...
Initial Amount: 50 milliliter.
Setting finished...
Initial Amount:  50 milliliter
Refill Standard:  10 %
Refill Standard:  5.0 milliliter
Usage Standard:  10 cm
```

Figures 12, 13. Setting the initial amount on AdaFruit IO dashboard toggle (upper figure) sets the starting value for the device (lower figure).

After all connection is set and the initial amount is set, the green LED turns on to signal there is hand sanitizer left in the dispenser.



Whenever a hand gets close to the sensor, which is equivalent to the event that someone uses the dispenser in real-life application of this device, the green LED turns off and the yellow LED turns on.



Shown below is the print statement of the device. As you can see, when the distance reading drops below 10cm, the “Amount Left” decreases by 2 milliliters. When the distance is above 10 cm, the “Amount Left” does not change.

```
Distance: 12.4914089347079
Amount Left: 24 milliliter
Distance: 36.426116838488
Amount Left: 24 milliliter
Distance: 6.8213058419244
Amount Left: 22 milliliter
Distance: 34.8453608247423
Amount Left: 22 milliliter
Distance: 11.7353951890034
Amount Left: 22 milliliter
Distance: 7.56013745704467
Amount Left: 20 milliliter
Distance: 5.29209621993127
Amount Left: 18 milliliter
Distance: 4.19243986254296
Amount Left: 16 milliliter
Distance: 22.9037800687285
Amount Left: 16 milliliter
```

When the “Amount Left” is less than the refill standard amount set initially, the device publishes “Dispenser empty. Need REFILL!” to the AdaFruit Feed (the REPL simply prints out “NEED REFILL!!!” instead) and this triggers the IFTTT action to send the user an email. Simultaneously, the red LED on the device turns on to alert any possible user that there is no more hand sanitizer available.

```
Distance: 7.21649484536082
Amount Left: 6 milliliter
Distance: 35.6185567010309
Amount Left: 6 milliliter
Distance: 5.65292096219931
NEED REFILL!!!
>>>
```

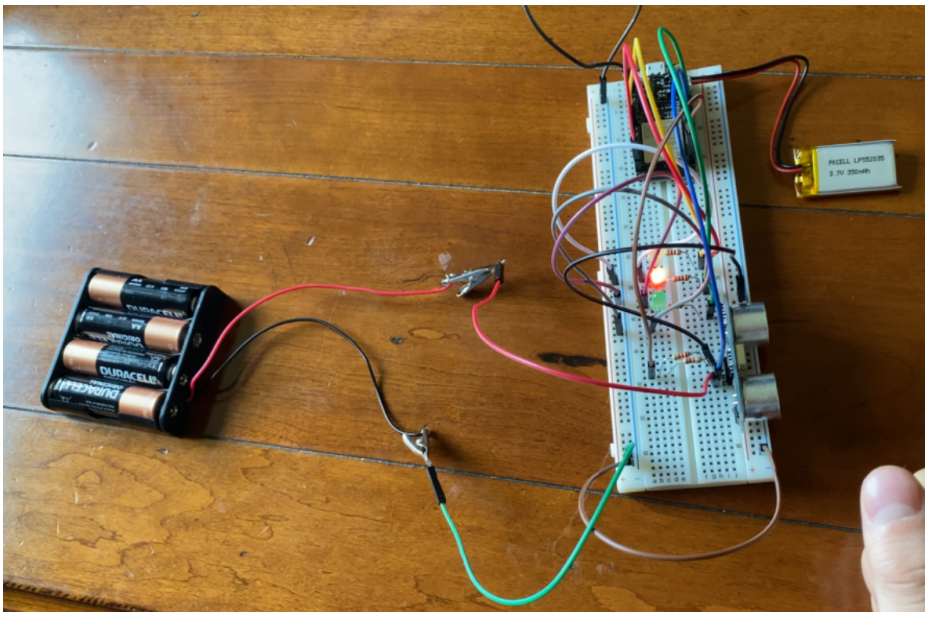
[ALERT] Pckt Refill :: Dispenser needs refill Inbox ×

Adafruit via IFTTT <action@ifttt.com> Unsubscribe
to me

Hello,
Your hand sanitizer dispenser in Etcheverry Hall is now empty. Please refill the dispenser.
Thank you,
Pckt Refill

Manage ›

Unsubscribe from these notifications or sign in to manage your Email service.



V. Discussion

We think that there is a real need for an IoT device that we built in today's world. Concerns over public health have drastically increased within the last 18 months. In this context, incorporating technology with already implemented tools is the key to an efficient and successful management of public health. Pckt-Refill is a simple add-on device that is compatible with any dispenser that is already in-use. There is no need to buy an additional dispenser to utilize our service. With more functionalities added, we are most confident that this can actually grow into a prospective business model.

Such functionalities include, but are not limited to, the following:

- Making a phone application that allows the user to set the `initial_amount`, `refill_standard`, `usage standard`, etc. that are essential for customization.
- Display the amount left in real time, not only when it is empty.

VI. Acknowledgement

We thank Dr. George Anwar and our graduate student instructor Kevin Widjaja for their help and advice over the course of this project and semester itself. MEC ENG 100 (Internet Of Things) course has provided us with new knowledge and experience with electronics that we, as mechanical engineering students, were not exposed to before. We are thankful for this class and thoroughly enjoyed the entirety of the content.