

Team/Device Name: MedMate

Team Members:

Stephen Mock

Miguel de los Reyes

Formal Statement

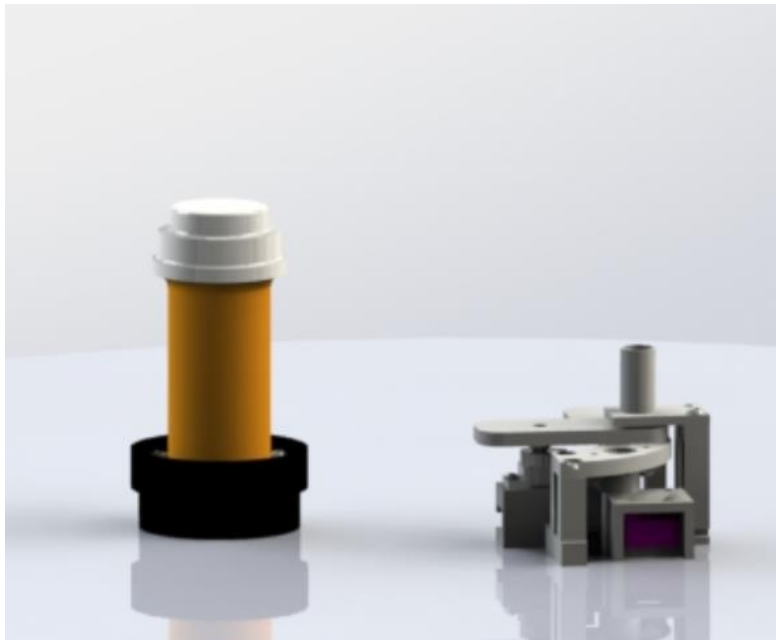
Description of Project:

MedMate is a device that helps patients and caretakers monitor medicine intake. MedMate comes in two forms: a pill bottle monitor and a pill dispenser. MedMate tracks when a user should take their prescription, senses when a user has taken their medicine, and then logs this information in a database that is presented on a webpage.







Webpage Link: <https://medmate.vercel.app/>

YouTube Demonstration: <https://youtu.be/uWqPvaWqItg>



GitHub Link: <https://github.com/migueldlr/medmate>



List of Hardware:

Hardware	Images
ESP32 IoT Microcontroller	
VCNL4010 Proximity Sensor	
DRV5053 Analog-Bipolar Hall Effect Sensor	
Tower Pro SG90	
5V Power Supply	
Breadboard, 220 Ohm Resistors, Multicolor LED	

Tools Used:

Ender 3 Printer	 A black 3D printer with a frame structure, a motorized extruder, and a build platform. It is shown from a side-on perspective.
Solder	 A soldering iron with a blue and black handle and a silver tip, resting on a black soldering station. A spool of solder is also visible in the background.

Software Used:

- Arduino C (C/C++)
- Eclipse Paho MQTT Python client (Python)
- Cloud Firestore (Google Firebase)
- React (JS)

Protocols:

- MQTT (Message Queuing Telemetry Transport)
- I2C (Inter-Integrated Circuit)

Process of Creating Project:

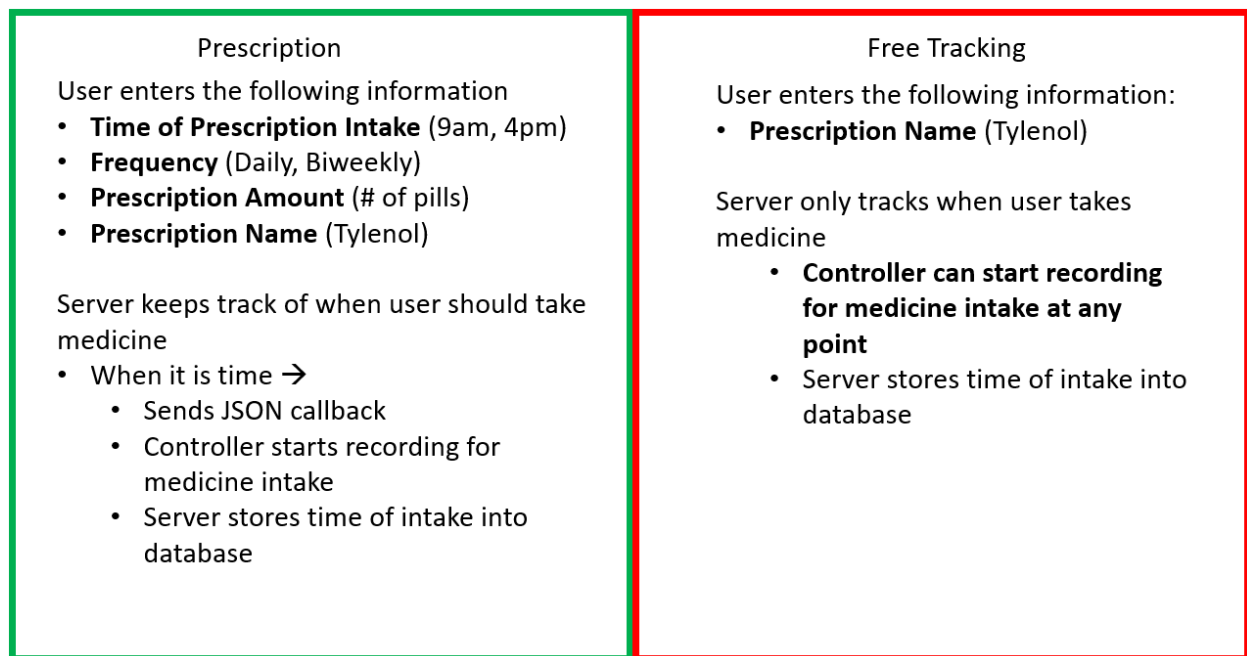
The motivation to create MedMate was to help Stephen's Dad track his medications. We wanted to create a system that would not only remind his father to take his medicine, but also monitor intake history.

The design philosophy was the following

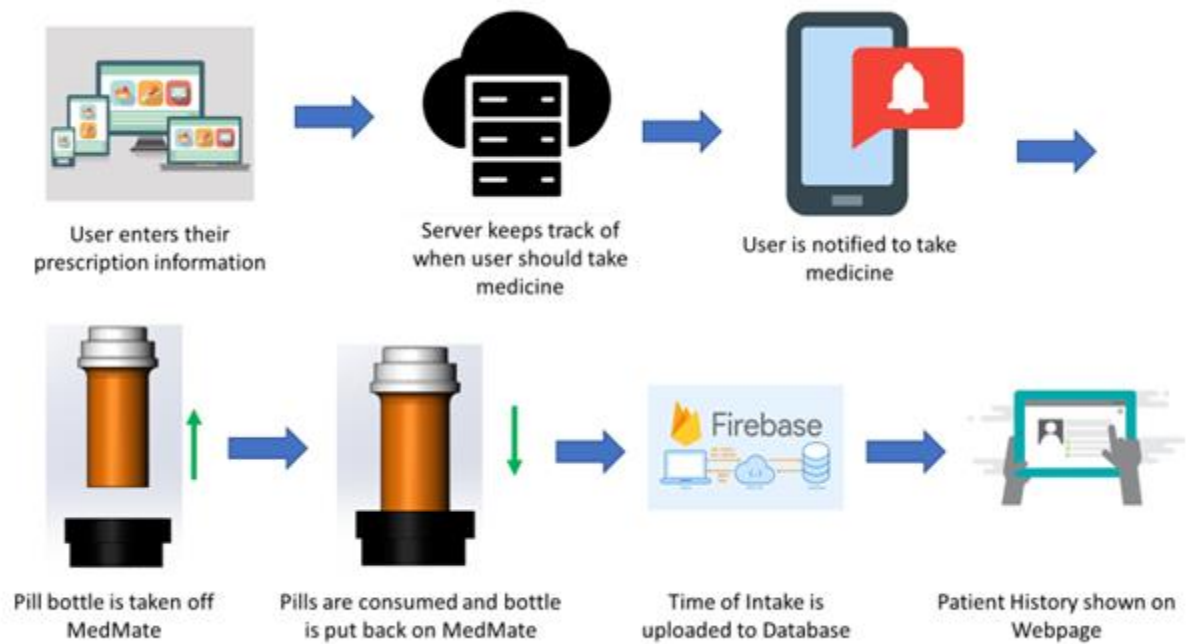
- Low profile
- Cheap
- Simple hardware
- Simple user interface
- Interfaces with any standard pill bottle

ESP32 was our chosen controller because it is configurable with the Arduino client and has full IoT functionality. The chosen network protocol was MQTT, as it is designed for simple communication between a controller and host server.

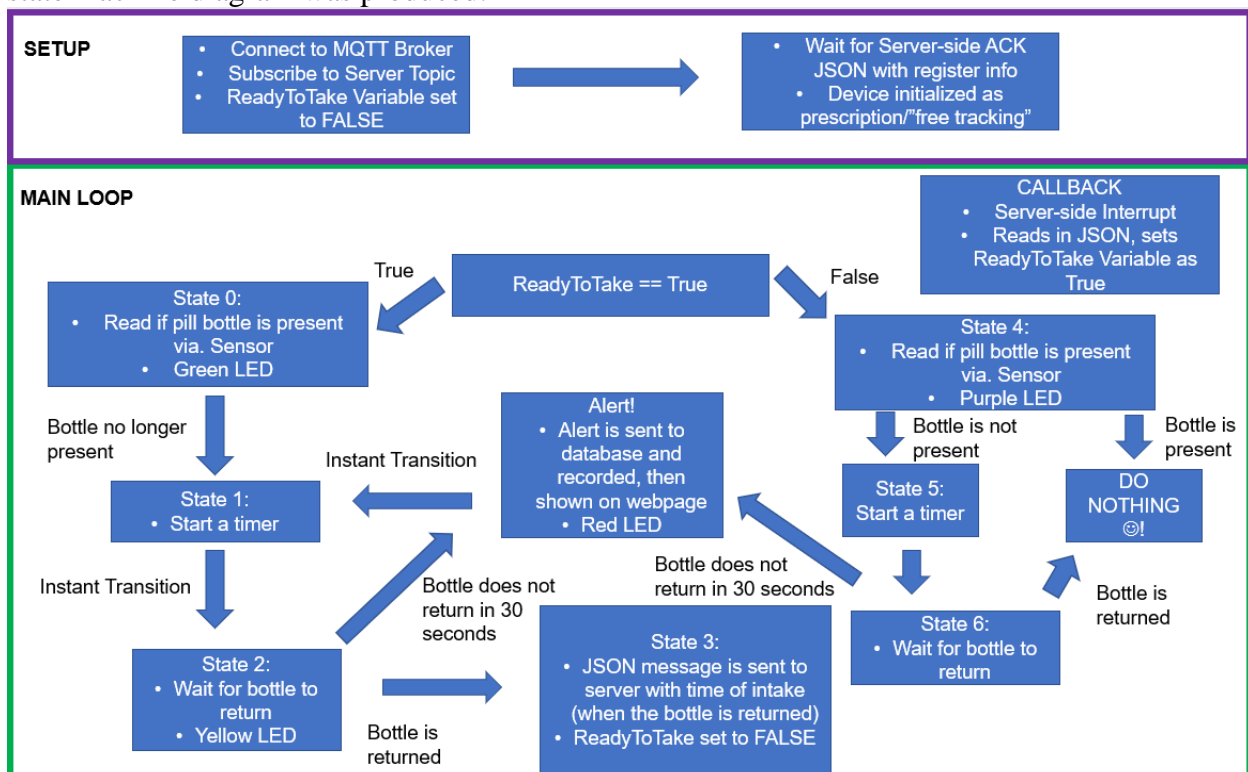
MedMate has two modes: Prescription and "free tracking", as shown in the following figure:



To understand the nominal use case, a storyboard is shown below:



To better develop the connection between hardware, frontend and backend, the following state machine diagram was produced:



To summarize, the server (our computer) uses threads to determine when a user should take medicine. When it is time, a JSON message is sent to the device and read via callback, signaling the device to start recording for when a user takes medicine. This changes the “ReadyToTake” variable. The device waits for the user to pick up and put back down the device, then records when this operation occurs. This is recorded in a Firebase database for the webpage to process. If a user does not return the pill bottle after a set time, an alert will be sent to the database and shown on the webpage. To communicate, JSON formatted messages were sent between the server and device as shown:

FROM DEVICE TO SERVER

```
ALERT                                STARTUP
{                                    {
  type: 'alert',                    'type': 'register',
  'userID': (string),               'deviceId': (string)
  date: (day) (int),
  hour: (int),
  min: (int)
}

EVENT (bottle put down after picking up)
{
  'type': 's_event'
  'userID': (string),
  'prescription': (bool),
  'prescriptionID': (string)
  Date: (day) (int),
  Hour: (int),
  Min: (int)
}
```

TO DEVICE FROM SERVER

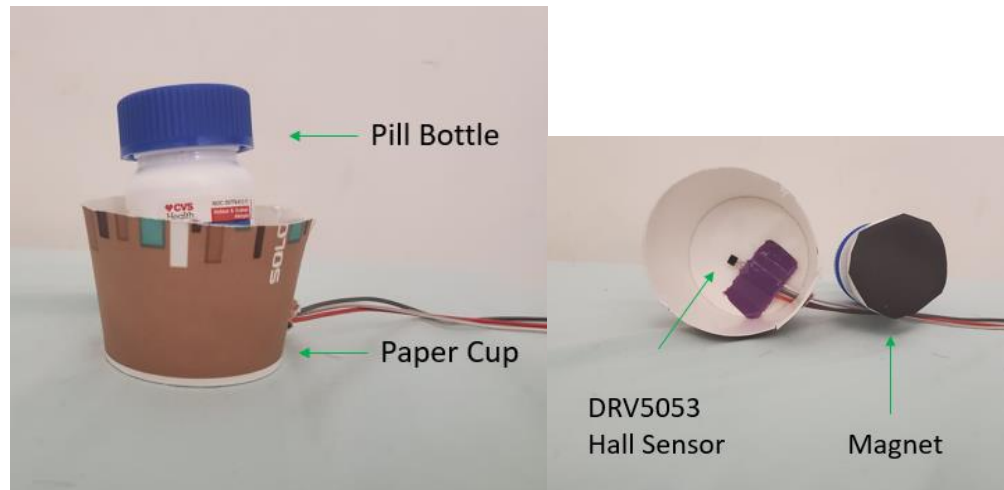
```
STARTUP (Callback)
{
  'type': 'register_ack',
  'userID' = (string),
  'prescription' : (bool),
  'prescriptionID': (string),
  'readyToTake': (bool)
}

ReadyToTake (callback) → this only happens when it's prescription
{
  'prescription' : (bool),
  'prescriptionID': (string),
  'type': 'readyToTake',
  'readyToTake': (boolean)
}
```

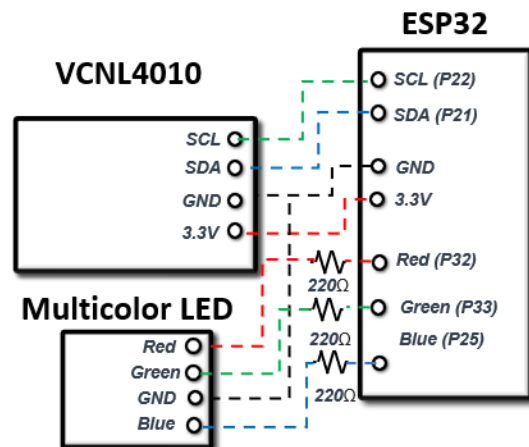
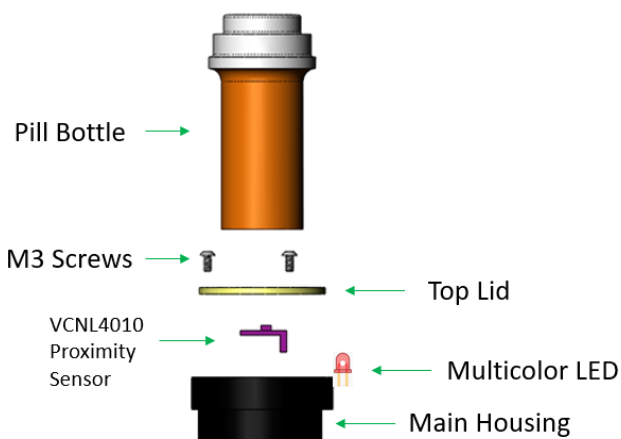
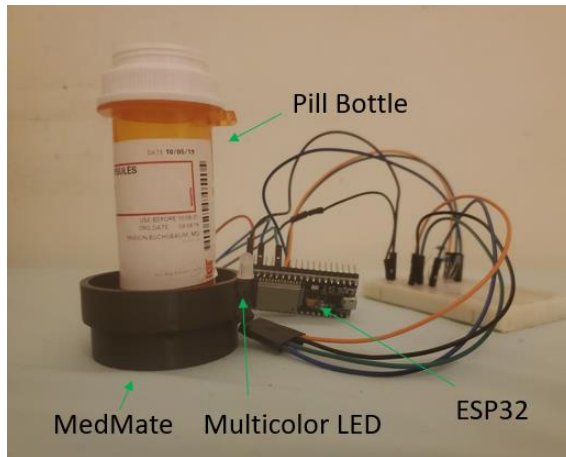
Additionally, the server-side operations and control flow are shown in this figure:



The first prototype used a hall effect sensor to monitor if the pill bottle is present. The sensor was glued to a paper cup, and a magnet was attached to the bottle. When the bottle was in the cup, the hall sensor sensed that it was there via. the magnet.



A second prototype was developed, and 3D printed, but instead used a VCNL4010 proximity sensor. This sensor was more robust and used I2C. Saturation and averaging was implemented for more reliability. A magnet was not required on the pill bottle and the system was easy to assemble, fitting our design philosophy.



Lastly, a third prototype was developed. This was a pill dispenser, which automatically dispenses at a set time. It implements a servo, but still uses the VCNL4010 for sensing if medicine is present after dispensing. It follows the similar server to device logic for communicating when to dispense. This device demonstrated a higher level of fidelity by having control over how much medicine the user can take.

Pill Dispenser State Machine

