Doc’t

Final Group – Group Report

CECS 490B

5-08-2020

Justin Maeder & Hung Le

CONTENTS

1. INTRODUCTION

1.1 MEET THE TEAM

Justin Maeder

Justin.Maeder@student.csulb.edu

Responsible for:

1. Software Design
2. C++ for Arduino
3. Constructed Database
4. Created web application
5. Hardware verification

Hung Le

Hung.Le@student.csulb.edu

Responsible for:

1. Hardware Design
2. Wiring diagram
3. Hardware assembly/debug
4. Software Verification

1.2 DOC’T

Doc’t is an electric transportation rental system with centralized locking and docking charging HUBs and innovative safety equipment.

Our design consists of three main components. Primarily we will be building an Electric Transportation System. Our ETS will include a custom electric skateboard. The second component consists of a docking and locking HUBs. The "HUB" harnesses clean energy from the sun using solar panels and store the energy in a power bank to charge and lock the vehicles.

Lastly, our safety features include a helmet with a Head-Up display to visualize the vehicle’s vitals. The HUD will display the battery level, speed, range, efficiency, and current time.

2. SPECIFICATIONS

2.1 DESIGN SPECIFICATIONS

1. Frontend – Hackathon Boiler Plate with PUG JavaScript with MongoDB
    a. Register User
        i. Name
        ii. Username
        iii. Password
        iv. Email
    b. Login
    c. Book Ride
        i. Show closest available vehicle
        ii. Shows PIN
    d. Forgot Password
    e. Contact Page
    f. Status of the server
    g. My Account (Profile)
2. Backend
    a. Using MySQL with PhpMyAdmin to interface
    b. Virtual private server through Digitalocean:
        i. 1 GB RAM
        ii. 25 GB SSD
        iii. 1 CPU
        iv. 1 TB data transfer
        v. $5.00/ month
            1. ALL user registration information
            2. Vehicle Information
                a. Check in/out time
                b. Range
                c. Battery Usage
                d. GPS
                e. PIN
3. HUB – Charging Station
    a. Charging station consists of a deep cycle SLA battery of 9AH. Enough to charge one vehicle from 0% to 100%.
    b. This battery is recharged by a 25W solar panel.
    c. A solar charge controller is used to safely charge the battery overtime.

   d. Lastly a 500W inverter will convert the DC power from the battery to AC to charge the vehicle.
4. Physical Locker space:
   a. The locking mechanism on the locker space will be controlled by the TTGO TCALL.
   b. Locker space consists of the lockers with electronic locking solenoid and a keypad.
   c. The system will be powered by the central battery and stepdown using the 7805-voltage regulator.
      1. Locking mechanism
         a. GPIO communication w/ each lock
      2. Solar Panel Mounting
         a. Deliver power to controller
      3. Battery storage
         a. Battery capacity to support number of ETS
      4. Keypad for PIN input
5. Transportation Vehicle:
   a. Physical vehicle
      i. Deck
      ii. Trucks
      iii. Wheels
      iv. Bearings
      v. Motor mount
   b. DC Motor
      i. Single brushless belt driven DC motor that can generate 2900 Watts.
   c. Electronic speed control module
      i. Remote Control
      ii. 7-segment display
   d. Battery Bank
      i. Dual battery bank that can provide a range of 7-8 miles range.
         1. 5000mAh
   e. ESP32 TTGO TCALL
      i. GPS module
      ii. Sim Card
6. Safety:
   a. Helmet:
      ii. Starting mechanism:
         1. ETS will not function w/o the helmet strapped.
      iii. Head Up Display:
         1. Arduino Nano driving OLED display:
            a. Battery percentage
            b. Speed
            c. Range

    d. Current Time
2. Bluetooth Communication using HM-10 transceiver
    a. Remote
    b. Vehicle
    c. Helmet

2.2 MAIN COMPONENT DESCRIPTION

2.2.1 HUB

- ESP32 TTGO T-Call SIM800L for its internet connectivity and Bluetooth capabilities
- Power consumption: 2.85 W (~500mA) – 7.6 W (~640 mA) (idle – load)

The HUB will have its operating system that will allow users to reserve a locker space or to borrow an electric transport. It will then display the available lock number to the user. If the user wants to reserve a space, then it will prompt the user to enter a PIN to unlock the locker. The main console will keep track of the time that the locker is being used and the power being consumed if there is any.

An app will be available to the user so their information can be logged into the system and communicates the information to the HUB. This will allow for ease of use where the user can reserve space or transport using the app and arrive at the station with their space ready to use. To conserve power, the HUB will go into sleep mode when it has been idling for a certain amount of time.

HUB is completely self-sufficient in terms of power generation. Arrays of 25w solar panels. HUB will also oversee the power level of the battery and the output levels of the solar panels. Depending on the power of the battery, the power output to the users will be adjusted to prevent the battery from going empty.

Skateboard will have 80-Watt Hour battery (36V at 4.5A) – range 7-8 miles. HUB will consume around 3-8 Watts. Solar Panel outputs around 25 Watts. To support one skateboard a battery will have to be able to output at least 100W per skateboard. This is scalable to the number of electric transport that the station will accommodate.

2.2.2 TRANSPORTATION VEHICLE

- C6374 – 170kV brushless belt driven DC Motor that generates 2900W chosen for reliability and cost.
- Turnigy 3s 5000mAh Battery bank for 80-Watt Hour to give the transportation vehicle 7-8 miles range.
- Using our current longboard for prototyping as we have not decided yet on a chassis.
- ESC (electric speed controller)

What is the appeal to those who cannot ride a skateboard? Doc’t presents an all-inclusive solution by transforming into an easy-to-ride scooter all at the rider’s fingertips. When the rider pulls up to the HUB, they will have the option to check out a scooter handlebar on the reservation app which will click and lock into place on the transportation vehicle deck. The remote will include a throttle potentiometer, a 1.5V battery, and LEDs to display the current setting.

On the electric transportation vehicle side, we will have an ESC (electric speed controller) to drive and control the motor. It will communicate via Bluetooth with the remote to change motor speed and direction. The ESC will also monitor the battery level and an ESP32 will transmit that data to the safety system.

2.2.3 SAFETY SYSTEM

Our Rider’s safety is the number one priority. Here are a few ways to ensure the safety of transportation. First, we created to have a smart helmet. This will also cause the transportation vehicle to active ensuring that the rider is wearing the helmet. The smart helmet will have a

Head-Up display which will display the board’s vitals such as batter level, range, speed, current and time.

3. PARTS REQUIRED

3.1 SAFETY SYSTEM

3.1.1 HEAD UP DISPLAY

Part name	**Quantity**
Arduino Nano	1
HM-10 Bluetooth module	1
Adafruit MicroLipo Battery Charger	1
3.7V 850 mAh lipo battery	1
DIY mall 0.96” OLED module	1
100mm double convex glass lenses	1
0.096” thick acrylic sheet	1
Mirror	1
3d printed casing	5

3.2 VEHICLE

3.2.1 DRIVING MECHANISM

Part name	**Quantity**
Longboard skateboard	1
C6374 brushless motor 170Kv	1
Electric Skateboard flywheel pulley drive kit	1
Single motor electric skateboard controller ESC	1
6S 25.2V Li-ion 18650 Lithium Battery BMS	1
XT60 battery connectors	2
Nano Sim Card	1

3.2.2 GPS SYSTEM

Part name	**Quantity**
ESP-32 TTgo T-call	1
REYAX RY8253F 10Hz GPS module	1
BINZET DC Converter step down regulator 5V	1

3.3 LOCK & DOCKING HUB

3.3.1 POWER SYSTEM

Part name	**Quantity**
Renogy Wanderer 10 Amp 12v/24v PWM Solar charge controller	1
ECO-WORTHY 12v 25w Solar Panel	1
ML9-12 12v 9Ah Rechargeable SLA battery	1
POTEK 500W Inverter DC 12v to AC 110v	1

3.3.2 LOCKING SYSTEM

Part name	**Quantity**
ESP-32 TTgo T-call	1
Adafruit 3x4 matrix keypad	1
GROVE relay module	1
Atoplee Electric Lock Assembly Solenoid DC 12v	1
12V battery source	1

4. CONTRUCTING DESIGN

4.1 SAFETY SYSTEM

4.1.1 HARDWARE

3D PRINTED CASING

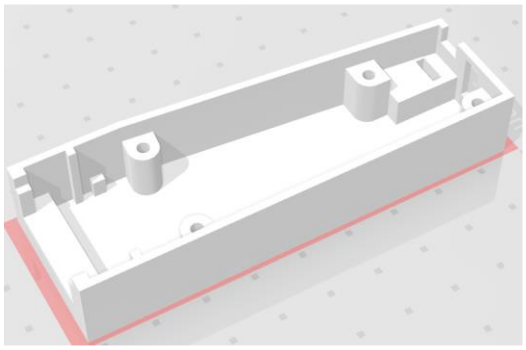


Figure 1 - PART A

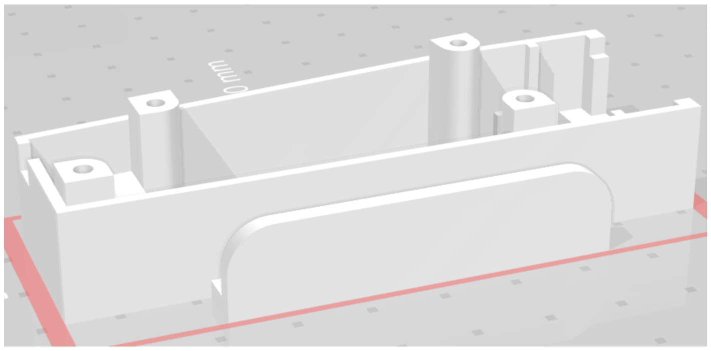


Figure 2 - PART B

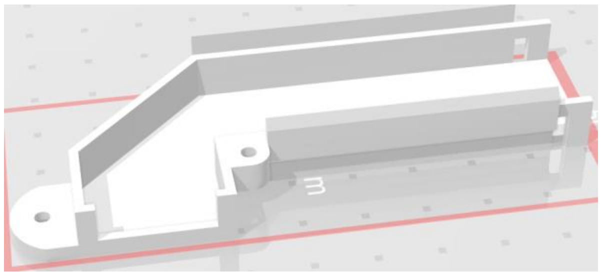


Figure 3 - PART C

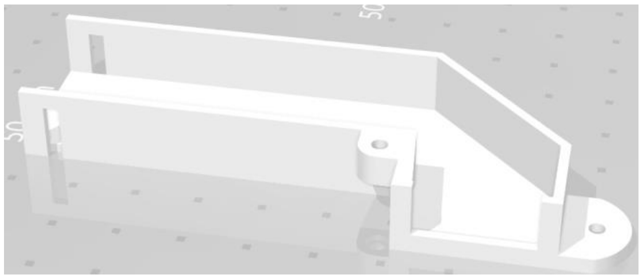


Figure 4 - PART E

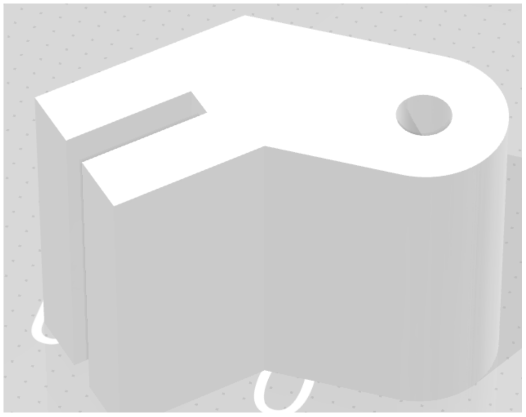


Figure 5 - PART D

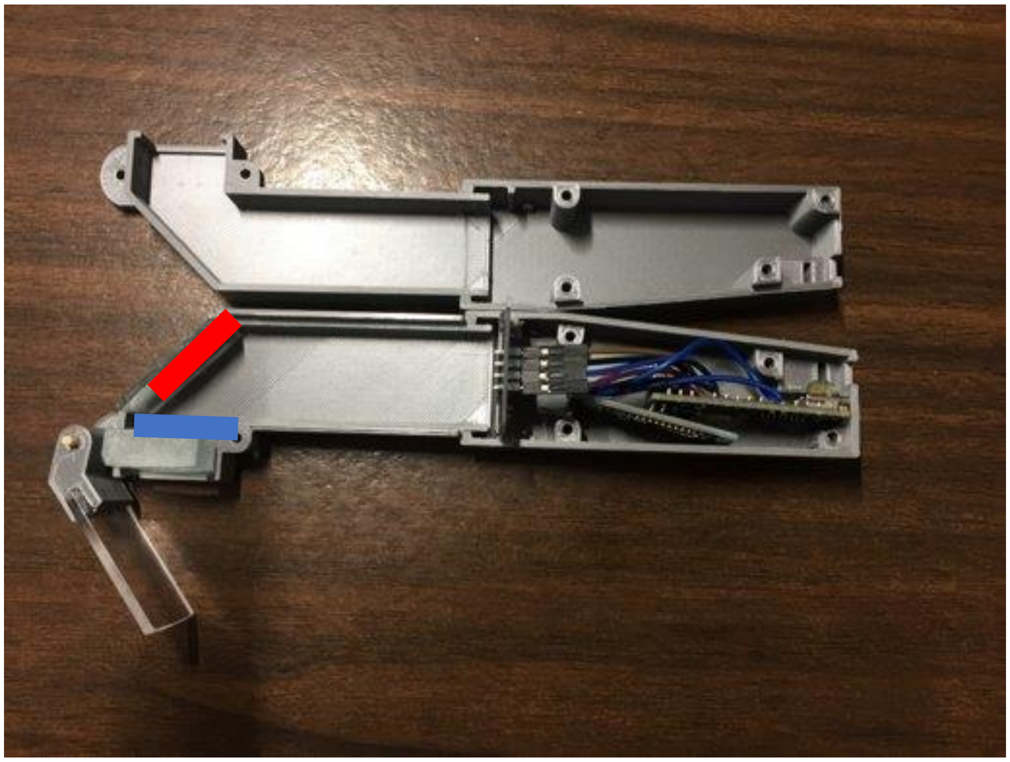


Figure 6 - sample HUD

- The Head Up Display casing can be 3d printed with the 3d model provided below.
- Cut the mirror to fit into the red space above.
- Cut the double convex lens to fit into the blue highlight space
- Cut acrylic sheet into 1”x1.5”in rectangle and secure with glue into part D.

WIRING DIAGRAM

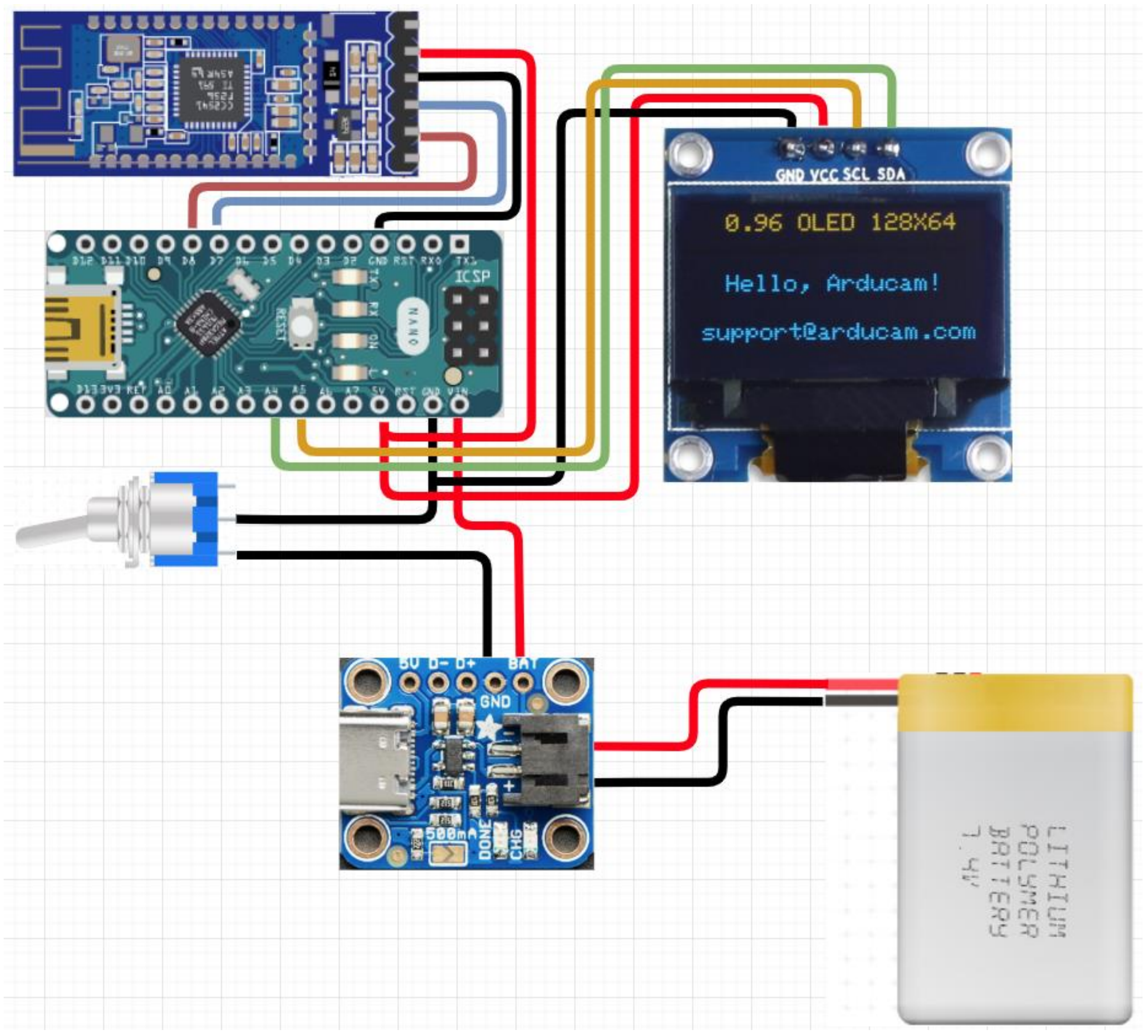

GND VCC SCL SDA
0.96 OLED 128X64
Hello, Arducam!
support@arducam.com
D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
ICSP
RESET
NANO
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN
5V D- D+ BAT
GND
500mA
DONE
CHG
LITHIUM POLYMER BATTERY 7.4V

Figure 7 - HUD wiring

ASSEMBLED DESIGN

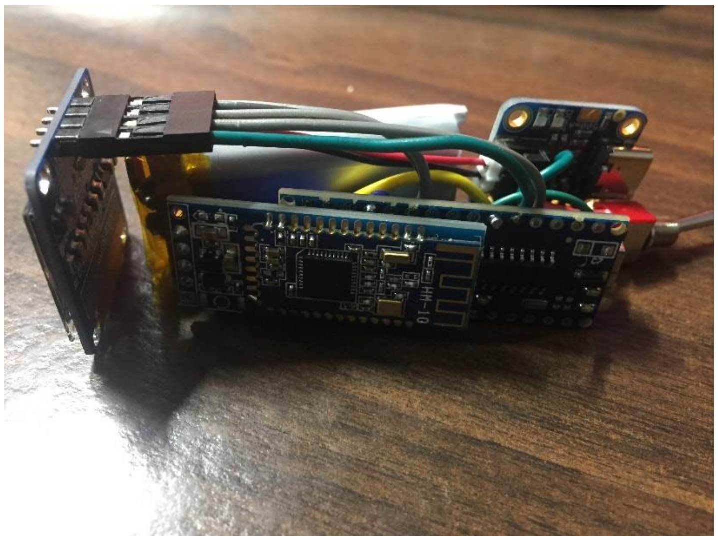


Figure 8 - Assembled HUD 1

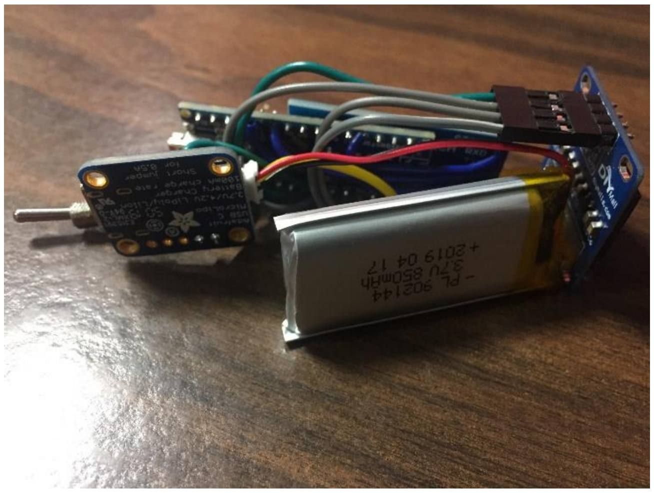


Figure 9 - Assembled HUD 2

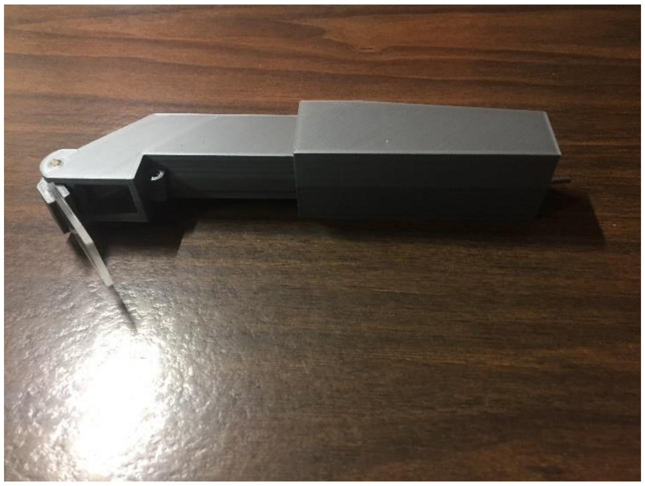


Figure 10 - Assembled HUD 3

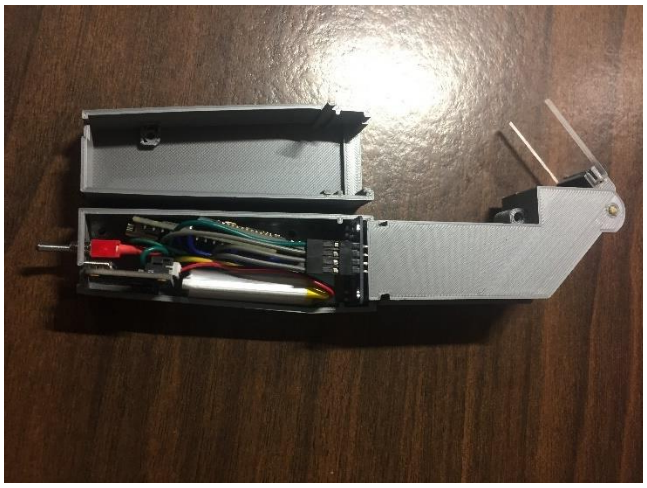


Figure 11 - Assembled HUD 4

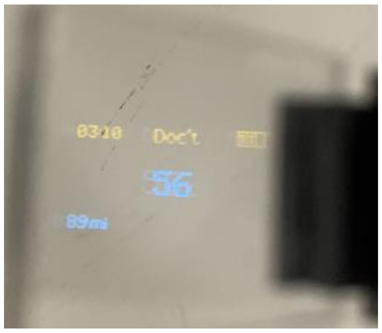
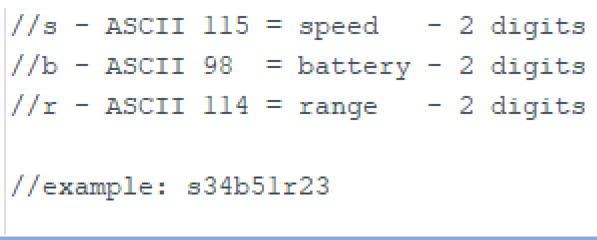


Figure 12 - Head up display view

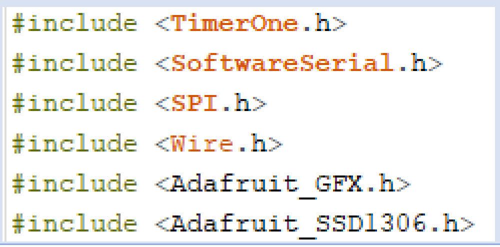
4.1.2 SOFTWARE DESCRIPTION

Full software files provided bellow.



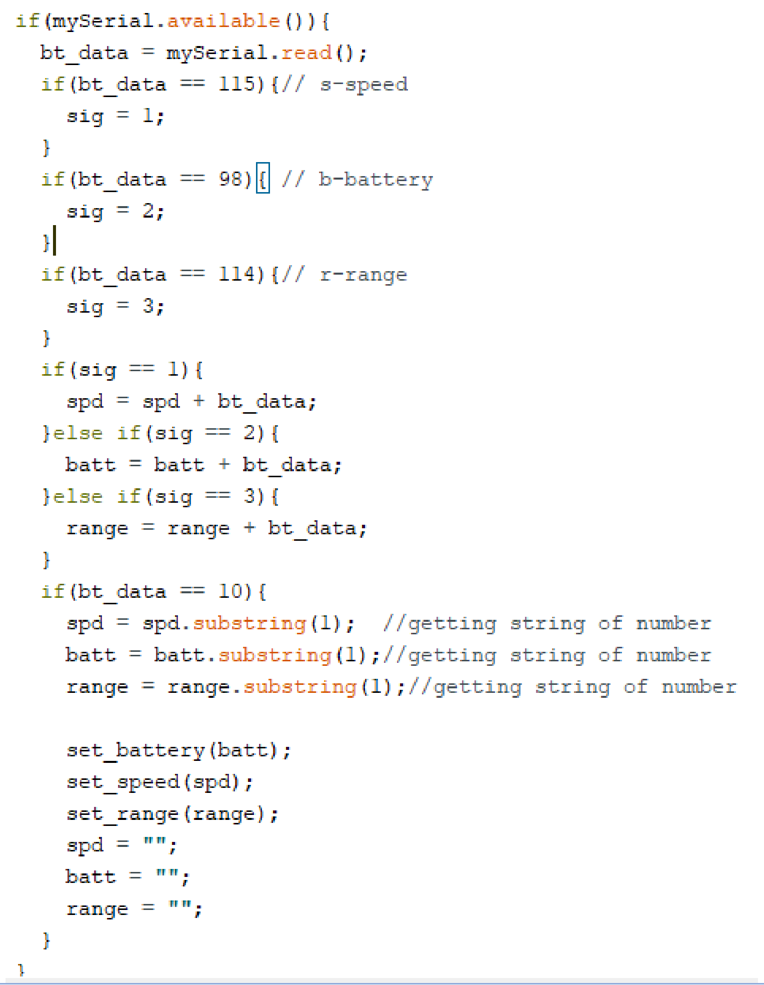
```
//s - ASCII 115 = speed   - 2 digits
//b - ASCII 98  = battery - 2 digits
//r - ASCII 114 = range   - 2 digits

//example: s34b51r23
```

Input string to send to HM-10



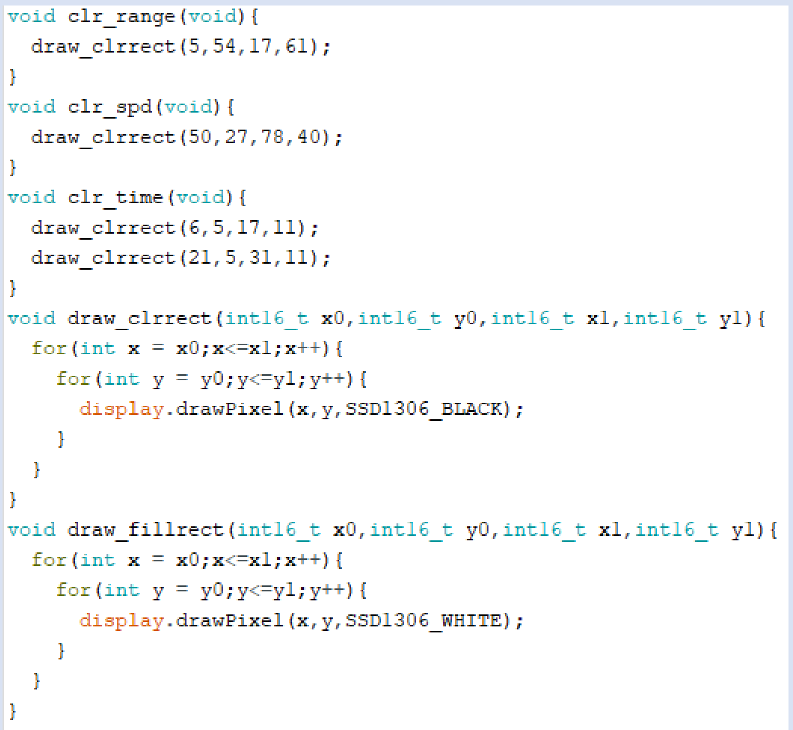
```
#include <TimerOne.h>
#include <SoftwareSerial.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

Libraries used.



```
if(mySerial.available()){
  bt_data = mySerial.read();
  if(bt_data == 115){// s-speed
    sig = 1;
  }
  if(bt_data == 98){ // b-battery
    sig = 2;
  }
  if(bt_data == 114){// r-range
    sig = 3;
  }
  if(sig == 1){
    spd = spd + bt_data;
  }else if(sig == 2){
    batt = batt + bt_data;
  }else if(sig == 3){
    range = range + bt_data;
  }
  if(bt_data == 10){
    spd = spd.substring(1);  //getting string of number
    batt = batt.substring(1);//getting string of number
    range = range.substring(1);//getting string of number

    set_battery(batt);
    set_speed(spd);
    set_range(range);
    spd = "";
    batt = "";
    range = "";
  }
}
```

Decoding method to decode data according to inputs

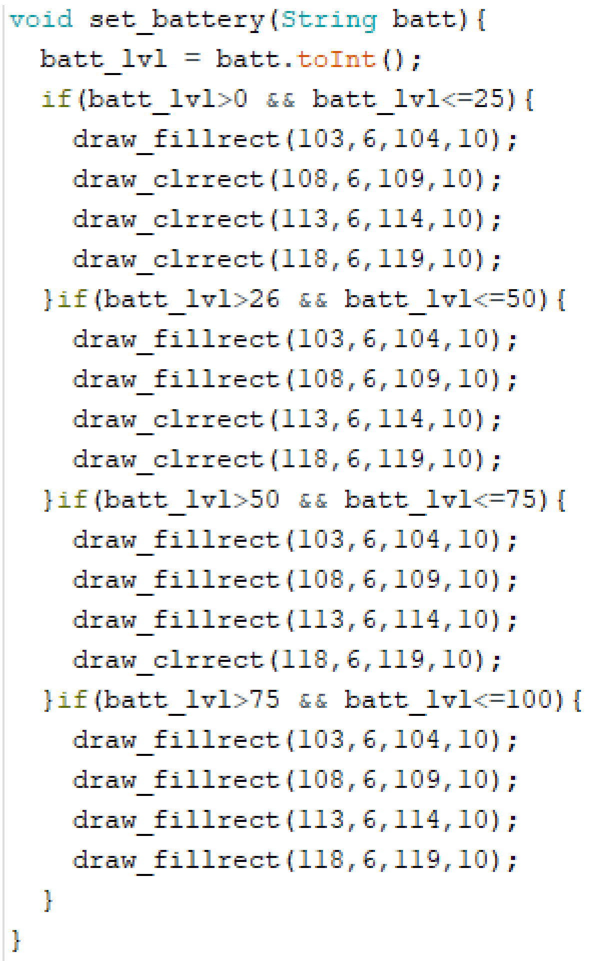


```
void clr_range(void){
  draw_clrrect(5,54,17,61);
}
void clr_spd(void){
  draw_clrrect(50,27,78,40);
}
void clr_time(void){
  draw_clrrect(6,5,17,11);
  draw_clrrect(21,5,31,11);
}
void draw_clrrect(int16_t x0,int16_t y0,int16_t x1,int16_t y1){
  for(int x = x0;x<=x1;x++){
    for(int y = y0;y<=y1;y++){
      display.drawPixel(x,y,SSD1306_BLACK);
    }
  }
}
void draw_fillrect(int16_t x0,int16_t y0,int16_t x1,int16_t y1){
  for(int x = x0;x<=x1;x++){
    for(int y = y0;y<=y1;y++){
      display.drawPixel(x,y,SSD1306_WHITE);
    }
  }
}
```

Supporting Methods:

Clr_range(), clr_spd(), clr_time(): clear the screen at specified bit locations by drawing a rectangle.

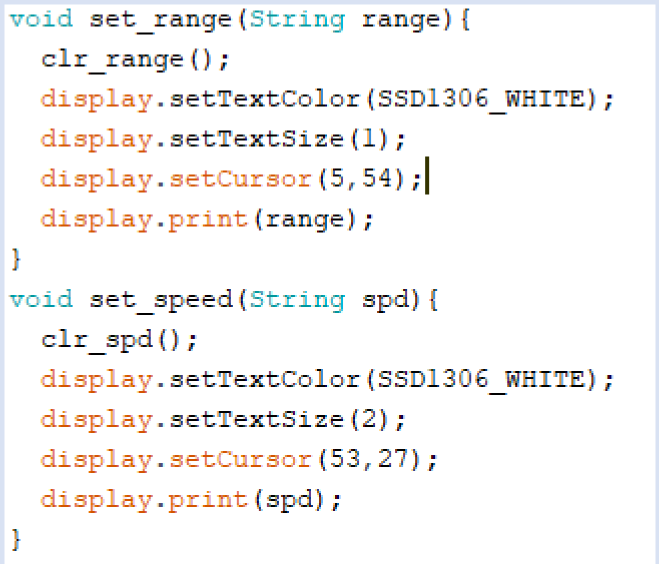
Draw_clrrect(), draw_fillrect(): turning on/off specified pixels.



```
void set_battery(String batt){
  batt_lvl = batt.toInt();
  if(batt_lvl>0 && batt_lvl<=25){
    draw_fillrect(103,6,104,10);
    draw_clrrect(108,6,109,10);
    draw_clrrect(113,6,114,10);
    draw_clrrect(118,6,119,10);
  }if(batt_lvl>26 && batt_lvl<=50){
    draw_fillrect(103,6,104,10);
    draw_fillrect(108,6,109,10);
    draw_clrrect(113,6,114,10);
    draw_clrrect(118,6,119,10);
  }if(batt_lvl>50 && batt_lvl<=75){
    draw_fillrect(103,6,104,10);
    draw_fillrect(108,6,109,10);
    draw_fillrect(113,6,114,10);
    draw_clrrect(118,6,119,10);
  }if(batt_lvl>75 && batt_lvl<=100){
    draw_fillrect(103,6,104,10);
    draw_fillrect(108,6,109,10);
    draw_fillrect(113,6,114,10);
    draw_fillrect(118,6,119,10);
  }
}
```

Displaying Battery:

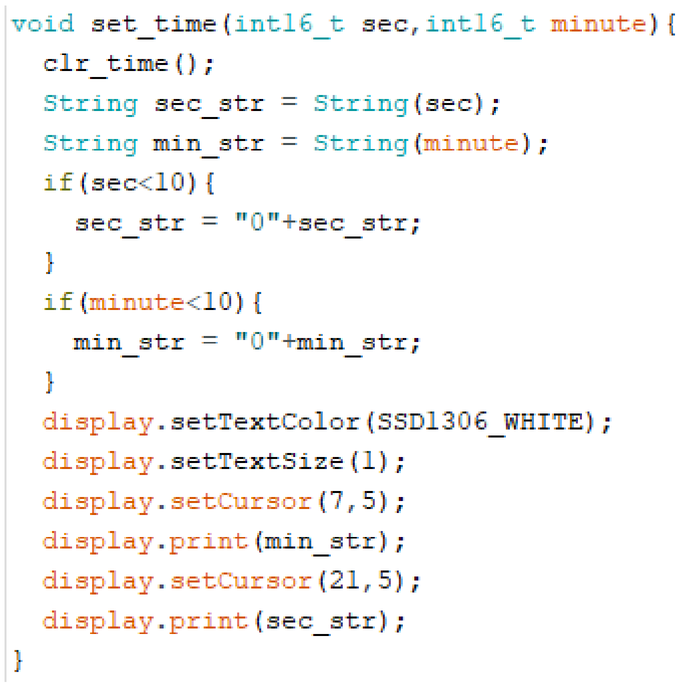
Based on input of battery level, draw in display that corresponds to the percentage.



```
void set_range(String range){
  clr_range();
  display.setTextColor(SSD1306_WHITE);
  display.setTextSize(1);
  display.setCursor(5,54);
  display.print(range);
}
void set_speed(String spd){
  clr_spd();
  display.setTextColor(SSD1306_WHITE);
  display.setTextSize(2);
  display.setCursor(53,27);
  display.print(spd);
}
```

Display range/speed:

Based on input of speed and range. Write the numbers on the display at specified locations.



```
void set_time(int16_t sec,int16_t minute){
  clr_time();
  String sec_str = String(sec);
  String min_str = String(minute);
  if(sec<10){
    sec_str = "0"+sec_str;
  }
  if(minute<10){
    min_str = "0"+min_str;
  }
  display.setTextColor(SSD1306_WHITE);
  display.setTextSize(1);
  display.setCursor(7,5);
  display.print(min_str);
  display.setCursor(21,5);
  display.print(sec_str);
}
```

Timer:

Display the timer since the device has been powered on. Time is in hr-min format.

4.2 VEHICLE

4.2.1 HARDWARE

DRIVING MECHANISM

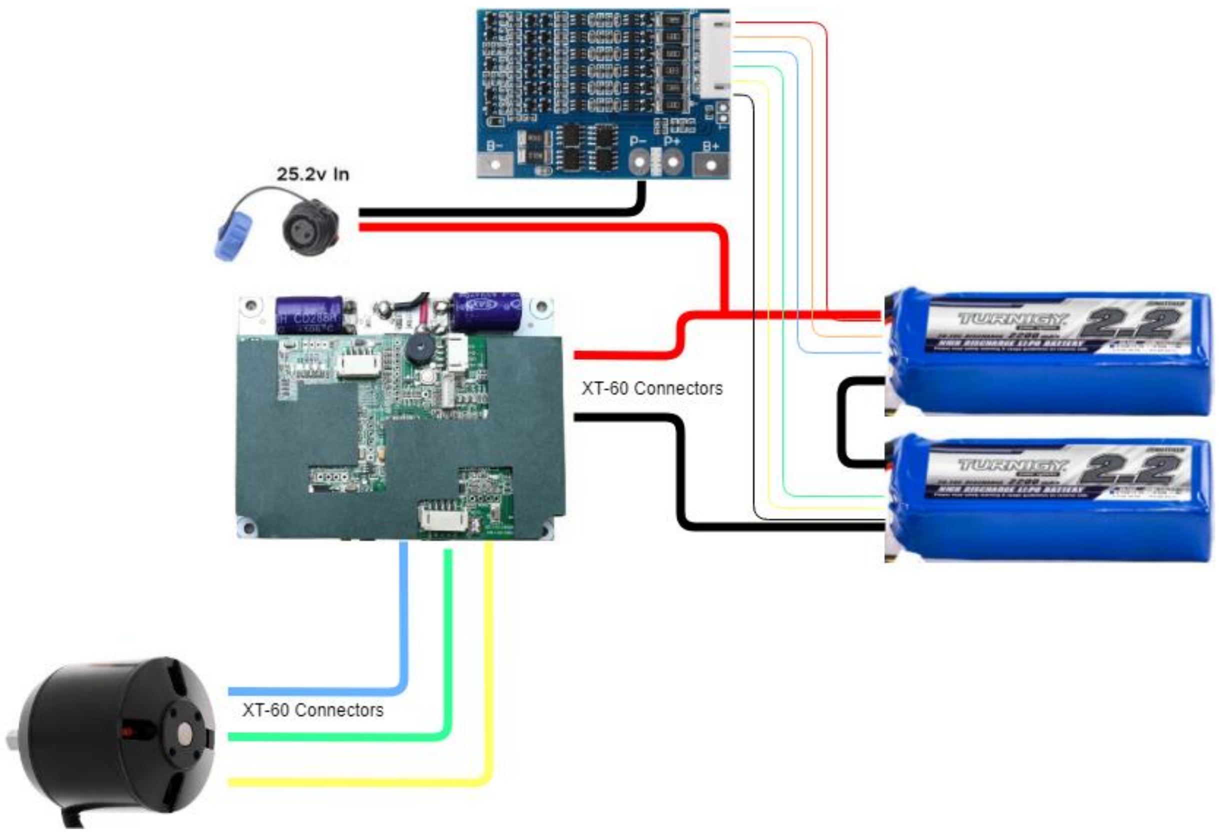


Figure 13 - Driving system connection

- 5000mAh 11.1V batteries, BMS (battery management system), connectors.
- Connect the two batteries in series using connectors.
- Wire BMS in parallel with batteries. BMS will be used to charge the batteries.
- Connect battery to ESC
- ESC and remote
- Connect power input wires to battery.
- Connect the 3 phase wires to motor. Connect motor control wires to ESC.

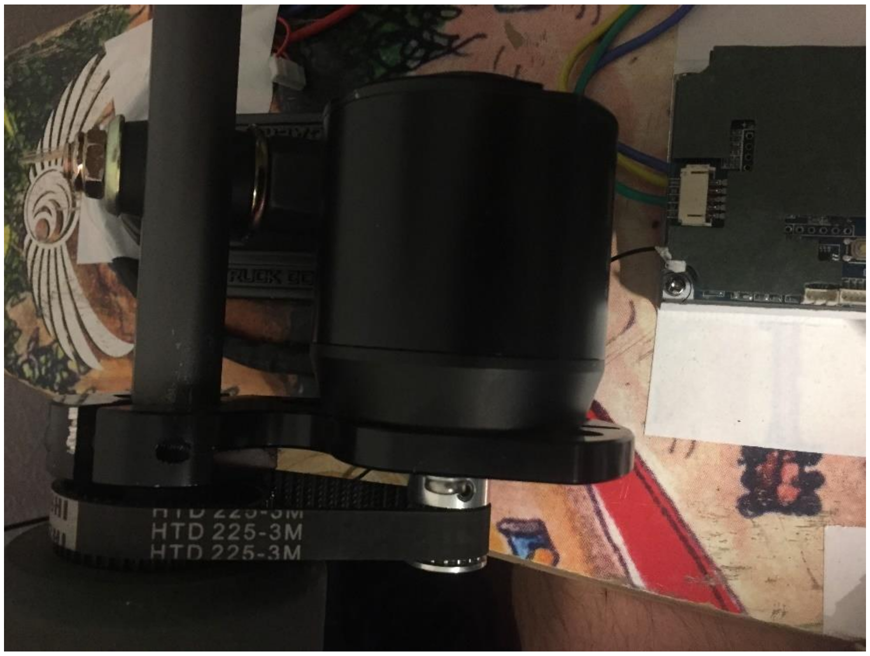


Figure 14 - Motor Mounting

- Trucks, motor mount, pulley system, motor, wheels.
- Bolt front and rear trucks to board.
- Attach motor mount to rear truck. Bolt motor onto motor mount.
- Attach pulley gear to motor. Attach pullet gear to wheel. Put drive belt connecting motor and wheel.

GPS SYSTEM

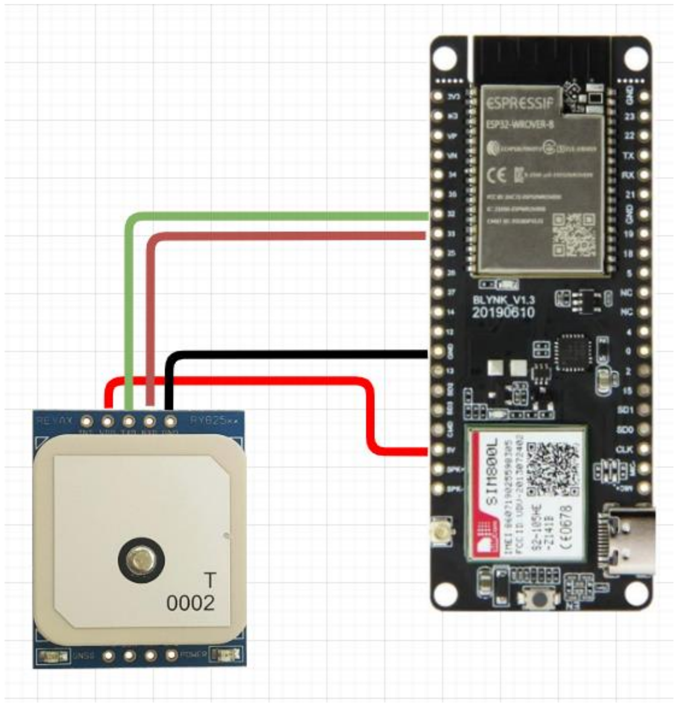


Figure 15 - GPS-ESP32 Wiring

- LILYGO TTGO T-Call V1.3 ESP32, REYAX RY8253F 10Hz GPS, SIM card
- Setup ESP32 to communicate with server.
- Setup ESP32 for Bluetooth communication.
- GPS module connected to ESP32 via UART protocol. ESP32 will receive GPS data and send data of the location back to the server via 2G.

ASSEMBLED DESIGN

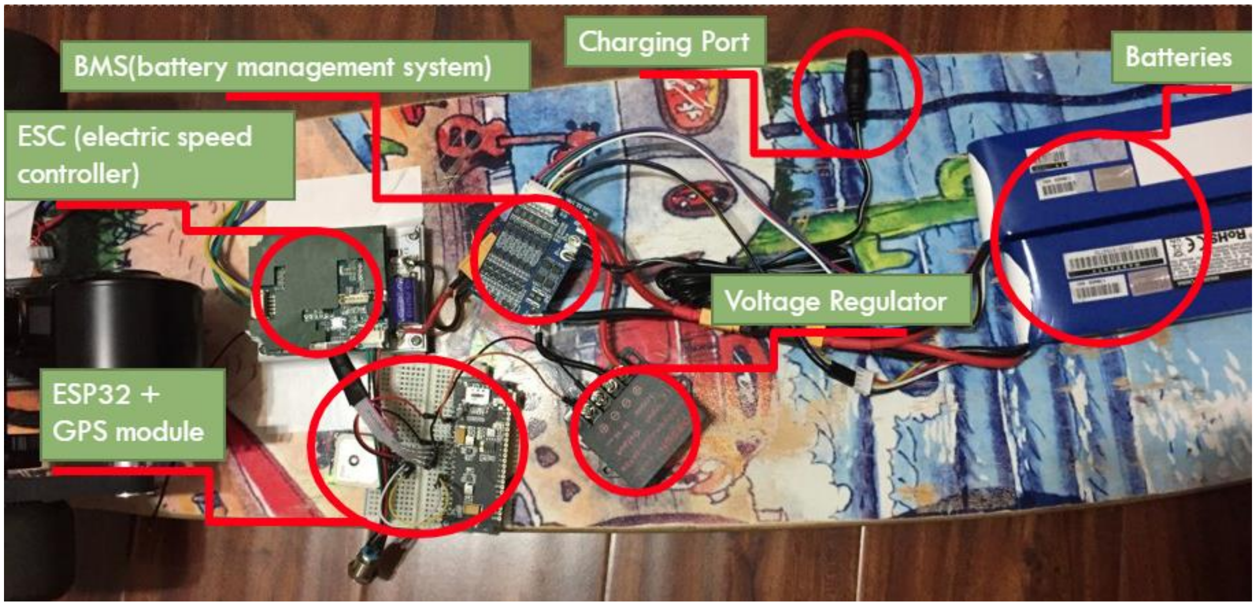
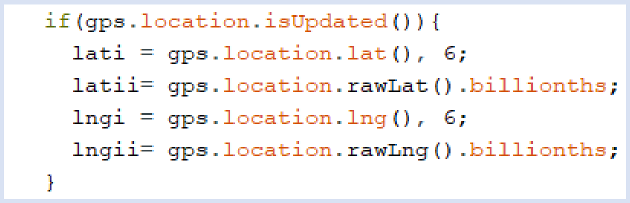

BMS(battery management system)
Charging Port
Batteries
ESC (electric speed controller)
Voltage Regulator
ESP32 + GPS module

Figure 16 - Assembled vehicle

4.2.3 SOFTWARE DESCRIPTION

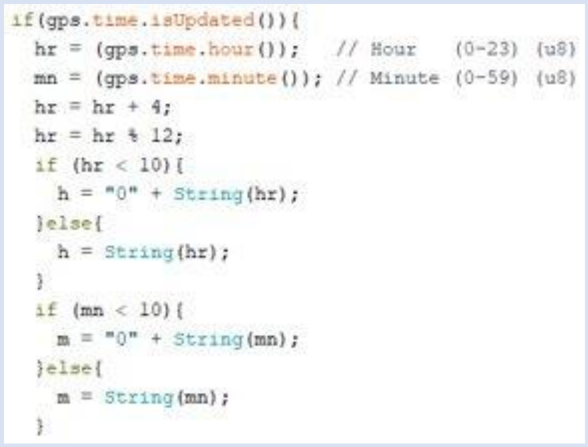
- GPS tracking:
  - ESP32 development will be in Arduino IDE environment using C.
  - Set up UART communication between GPS module and ESP32. Each vehicle will have an identifying number.
  - Process GPS data. Send out GPS data along with identification number to server via data using 2G network.
  - Location will be sent every 15 seconds.

`#include "TinyGPS++.h"` `#include <SoftwareSerial.h>`	**Libraries used.**
`const char apn[]      = "hologram"; // Your APN` `const char server[] = "167.99.109.182";` `TinyGsmClient client(modem);` `const int  port = 8081; //8081`	APN, Server, Port setup
`while(mySerial.available() > 0){` `gps.encode(mySerial.read());` `}`	Feeding data from GPS module to TinyGPS++ library.

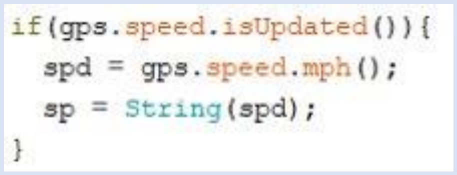


```
   if(gps.location.isUpdated()){
     lati = gps.location.lat(), 6;
     latii= gps.location.rawLat().billionths;
     lngi = gps.location.lng(), 6;
     lngii= gps.location.rawLng().billionths;
   }
```

Get latitude and longitude to the billionths decimal place from GPS module.

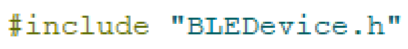


```
if(gps.time.isUpdated()){
  hr = (gps.time.hour());    // Hour   (0-23) (u8)
  mn = (gps.time.minute()); // Minute (0-59) (u8)
  hr = hr + 4;
  hr = hr % 12;
  if (hr < 10){
    h = "0" + String(hr);
  }else{
    h = String(hr);
  }
  if (mn < 10){
    m = "0" + String(mn);
  }else{
    m = String(mn);
  }
```



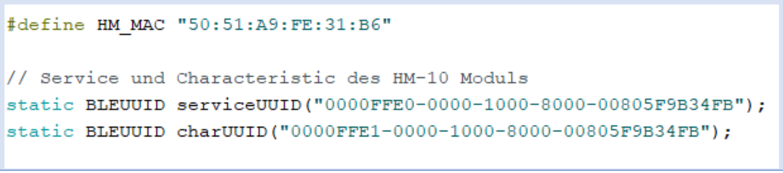
```
if(gps.speed.isUpdated()){
  spd = gps.speed.mph();
  sp = String(spd);
}
```

Get an accurate time Pacific Standard time with Spring Day Light Savings Time taken from GPS module.



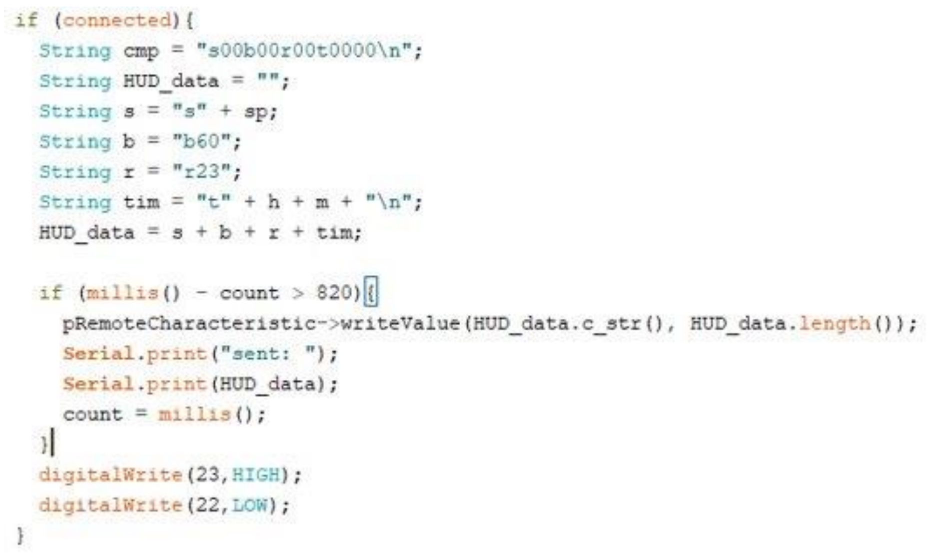
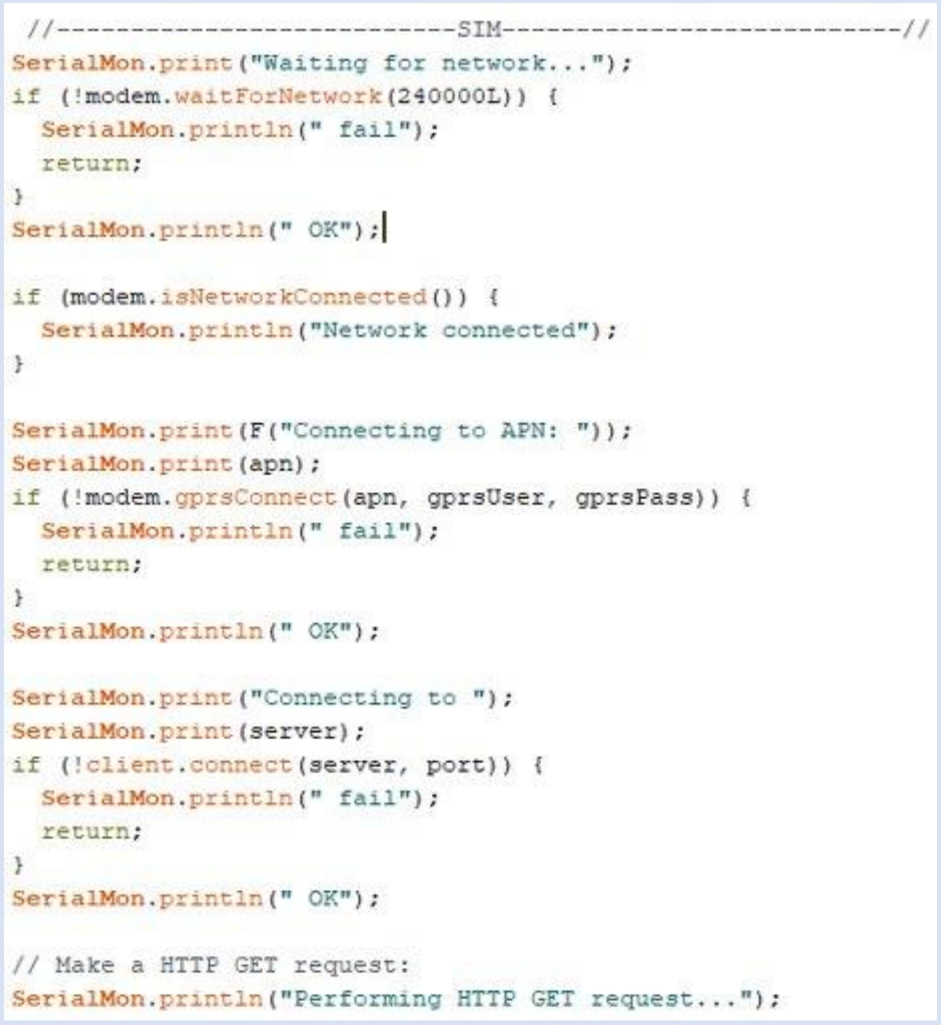
```
#include "BLEDevice.h"
```

Library for Bluetooth connection.



```
#define HM_MAC "50:51:A9:FE:31:B6"

// Service und Characteristic des HM-10 Moduls
static BLEUUID serviceUUID("0000FFE0-0000-1000-8000-00805F9B34FB");
static BLEUUID charUUID("0000FFE1-0000-1000-8000-00805F9B34FB");
```

MAC ID of HM-10 for connection.

 ```if (connected){ String cmp = "s00b00r00t0000\n"; String HUD_data = ""; String s = "s" + sp; String b = "b60"; String r = "r23"; String tim = "t" + h + m + "\n"; HUD_data = s + b + r + tim; if (millis() - count > 820){ pRemoteCharacteristic->writeValue(HUD_data.c_str(), HUD_data.length()); Serial.print("sent: "); Serial.print(HUD_data); count = millis(); } digitalWrite(23,HIGH); digitalWrite(22,LOW); }```	Sending test string from ESP32 to HM-10.
 ```//----------------------------SIM----------------------------// SerialMon.print("Waiting for network..."); if (!modem.waitForNetwork(240000L)) { SerialMon.println(" fail"); return; } SerialMon.println(" OK"); if (modem.isNetworkConnected()) { SerialMon.println("Network connected"); } SerialMon.print(F("Connecting to APN: ")); SerialMon.print(apn); if (!modem.gprsConnect(apn, gprsUser, gprsPass)) { SerialMon.println(" fail"); return; } SerialMon.println(" OK"); SerialMon.print("Connecting to "); SerialMon.print(server); if (!client.connect(server, port)) { SerialMon.println(" fail"); return; } SerialMon.println(" OK"); // Make a HTTP GET request: SerialMon.println("Performing HTTP GET request...");```	On first loop, or if reconnect, SIM sets up modem and connects to APN and Server

```
unsigned long timeout = millis();
while (client.connected() && millis() - timeout < 10000L) {
  // Print available data
  while (client.available()) {
    char c = client.read();
    SerialMon.print(c);
    timeout = millis();
  }
}
```

Sending data to server every 10 seconds.

```
  if((lati == 0) || (lngi == 0) || (latii == newlatii)){
    goto label;
  }
  newlatii = latii;
```

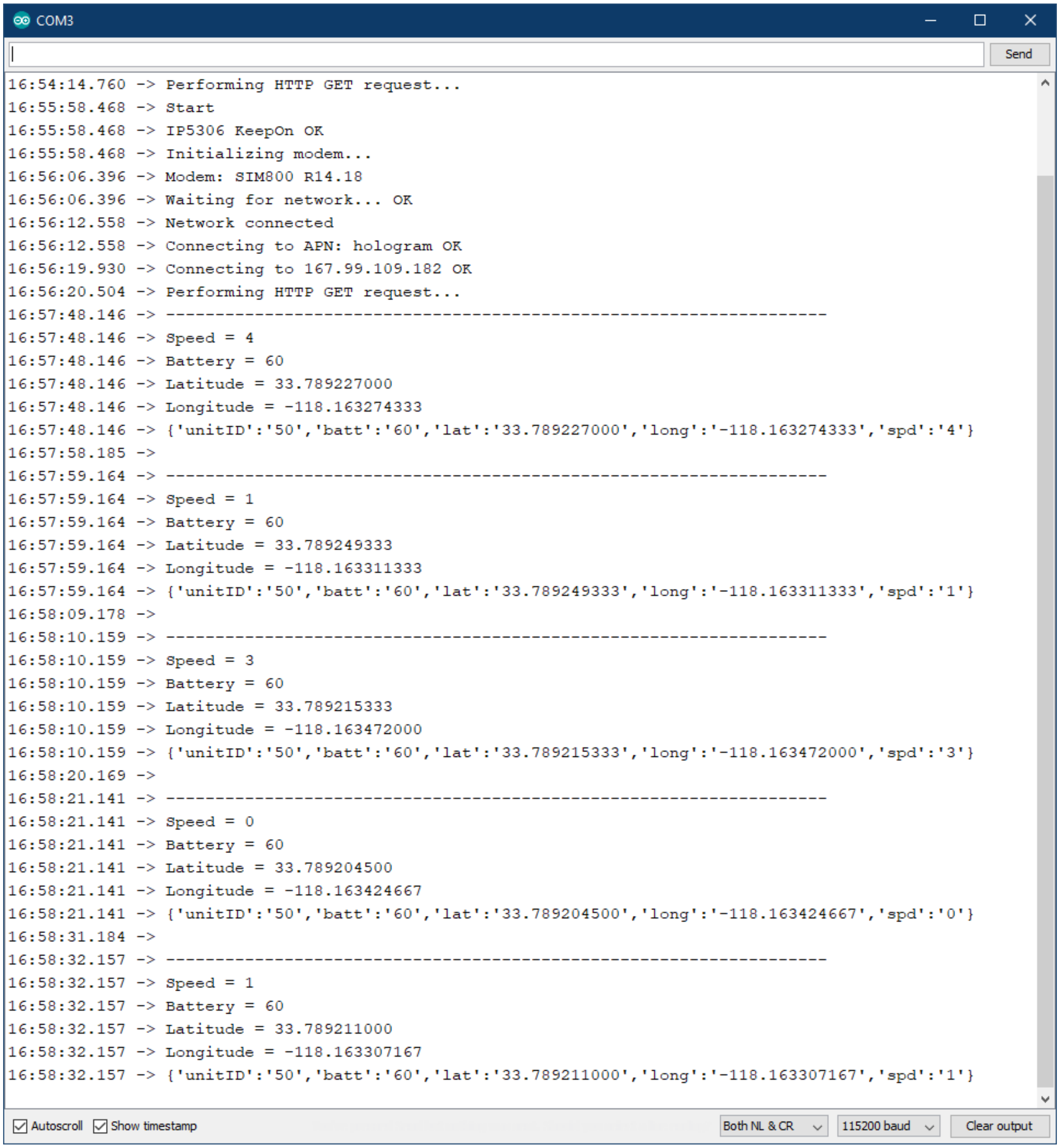
Make sure new data is sent to server.

```
if (!client.connected()) {
  SerialMon.println("Disconnected from server");
  delay(30000); // wait 30 seconds to reconnect
  goto reconnect;
}

// Reconnect after 60 seconds
if(true){   //count < 20
  //delay(49869); // 49.869 seconds + 2G delay (10.131 seconds)
  goto label;
}
else{
  // Shutdown
  client.stop();
  SerialMon.println(F("Server disconnected"));

  modem.gprsDisconnect();
  SerialMon.println(F("GPRS disconnected"));

  esp_deep_sleep_start();
}
```

Infinite loop to send data to server every minute otherwise disconnect and send ESP32 to deep sleep.

```
  Serial.println("--------------------------------------------------------
  Serial.println("Speed = "+String(spd));
  Serial.println("Battery = "+String(batt));
  Serial.println("Latitude = "+String(lati)+"."+String(latii));
  Serial.println("Longitude = "+String(lngi)+"."+String(lngii));
  Serial.println("{'unitID':'"+String(unitID)+"','batt':'"+String(batt)+
  client.println("{'unitID':'"+String(unitID)+"','batt':'"+String(batt)+
  count++;
```

Print acquired data to serial terminal (when plugged in) and to client (server). JSON string is sent for easy decode to server.



```
16:54:14.760 -> Performing HTTP GET request...
16:55:58.468 -> Start
16:55:58.468 -> IP5306 KeepOn OK
16:55:58.468 -> Initializing modem...
16:56:06.396 -> Modem: SIM800 R14.18
16:56:06.396 -> Waiting for network... OK
16:56:12.558 -> Network connected
16:56:12.558 -> Connecting to APN: hologram OK
16:56:19.930 -> Connecting to 167.99.109.182 OK
16:56:20.504 -> Performing HTTP GET request...
16:57:48.146 -> ------------------------------------------------------------------
16:57:48.146 -> Speed = 4
16:57:48.146 -> Battery = 60
16:57:48.146 -> Latitude = 33.789227000
16:57:48.146 -> Longitude = -118.163274333
16:57:48.146 -> {'unitID':'50','batt':'60','lat':'33.789227000','long':'-118.163274333','spd':'4'}
16:57:58.185 ->
16:57:59.164 -> ------------------------------------------------------------------
16:57:59.164 -> Speed = 1
16:57:59.164 -> Battery = 60
16:57:59.164 -> Latitude = 33.789249333
16:57:59.164 -> Longitude = -118.163311333
16:57:59.164 -> {'unitID':'50','batt':'60','lat':'33.789249333','long':'-118.163311333','spd':'1'}
16:58:09.178 ->
16:58:10.159 -> ------------------------------------------------------------------
16:58:10.159 -> Speed = 3
16:58:10.159 -> Battery = 60
16:58:10.159 -> Latitude = 33.789215333
16:58:10.159 -> Longitude = -118.163472000
16:58:10.159 -> {'unitID':'50','batt':'60','lat':'33.789215333','long':'-118.163472000','spd':'3'}
16:58:20.169 ->
16:58:21.141 -> ------------------------------------------------------------------
16:58:21.141 -> Speed = 0
16:58:21.141 -> Battery = 60
16:58:21.141 -> Latitude = 33.789204500
16:58:21.141 -> Longitude = -118.163424667
16:58:21.141 -> {'unitID':'50','batt':'60','lat':'33.789204500','long':'-118.163424667','spd':'0'}
16:58:31.184 ->
16:58:32.157 -> ------------------------------------------------------------------
16:58:32.157 -> Speed = 1
16:58:32.157 -> Battery = 60
16:58:32.157 -> Latitude = 33.789211000
16:58:32.157 -> Longitude = -118.163307167
16:58:32.157 -> {'unitID':'50','batt':'60','lat':'33.789211000','long':'-118.163307167','spd':'1'}
```

Figure 17 Serial Terminal for vehicle showing outputs and sent JSON data from vehicle 2G SIM every 10 seconds.

4.3 LOCKING & DOCKING HUB

4.3.1 HARDWARE

LOCKER

- The locker can be constructed from any desired material (wood, metal). Locker should be large enough to fit the vehicle. This locker is smaller for demonstration purposes.
- Note in figure 17, there is a slit in the door for the wires to connect to the keypad.

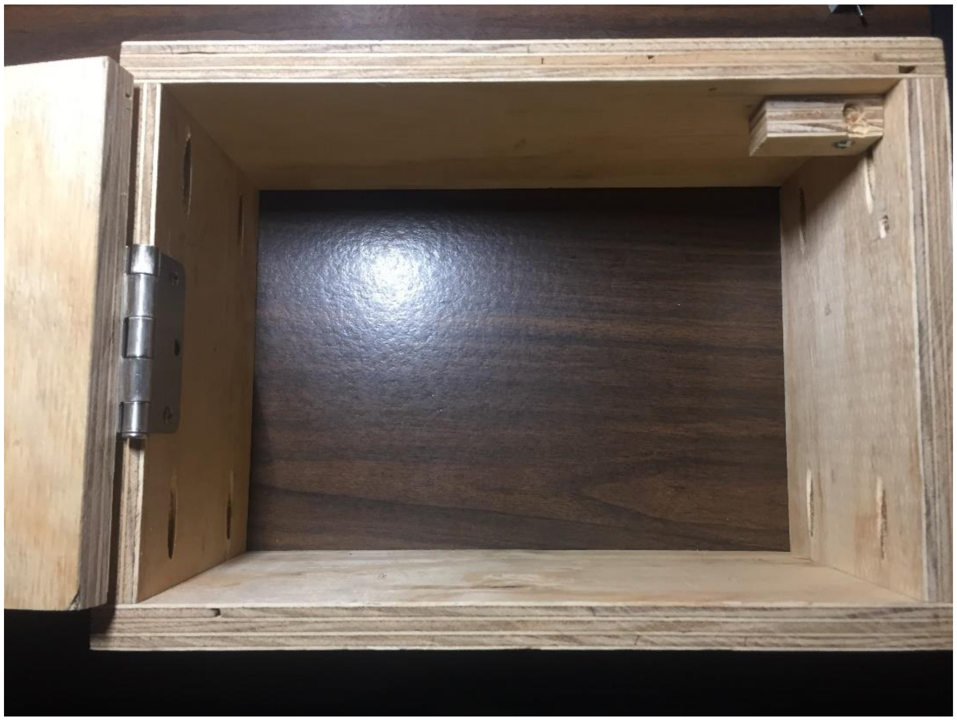


Figure 18 - Locker inside

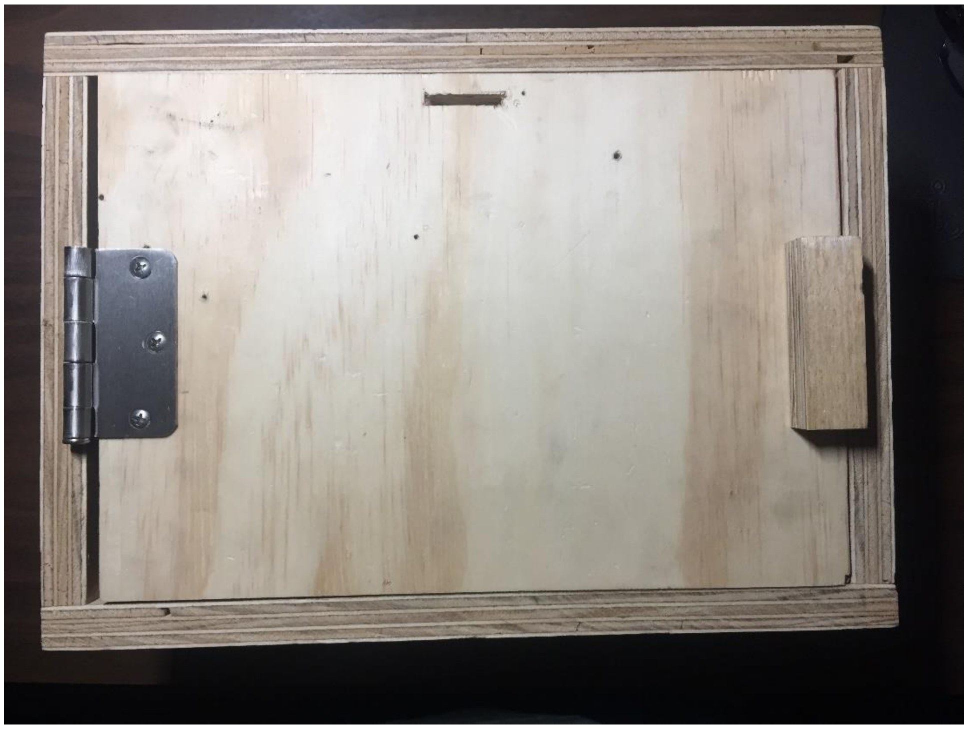


Figure 19 - Locker outside

LOCKING MECHANISM

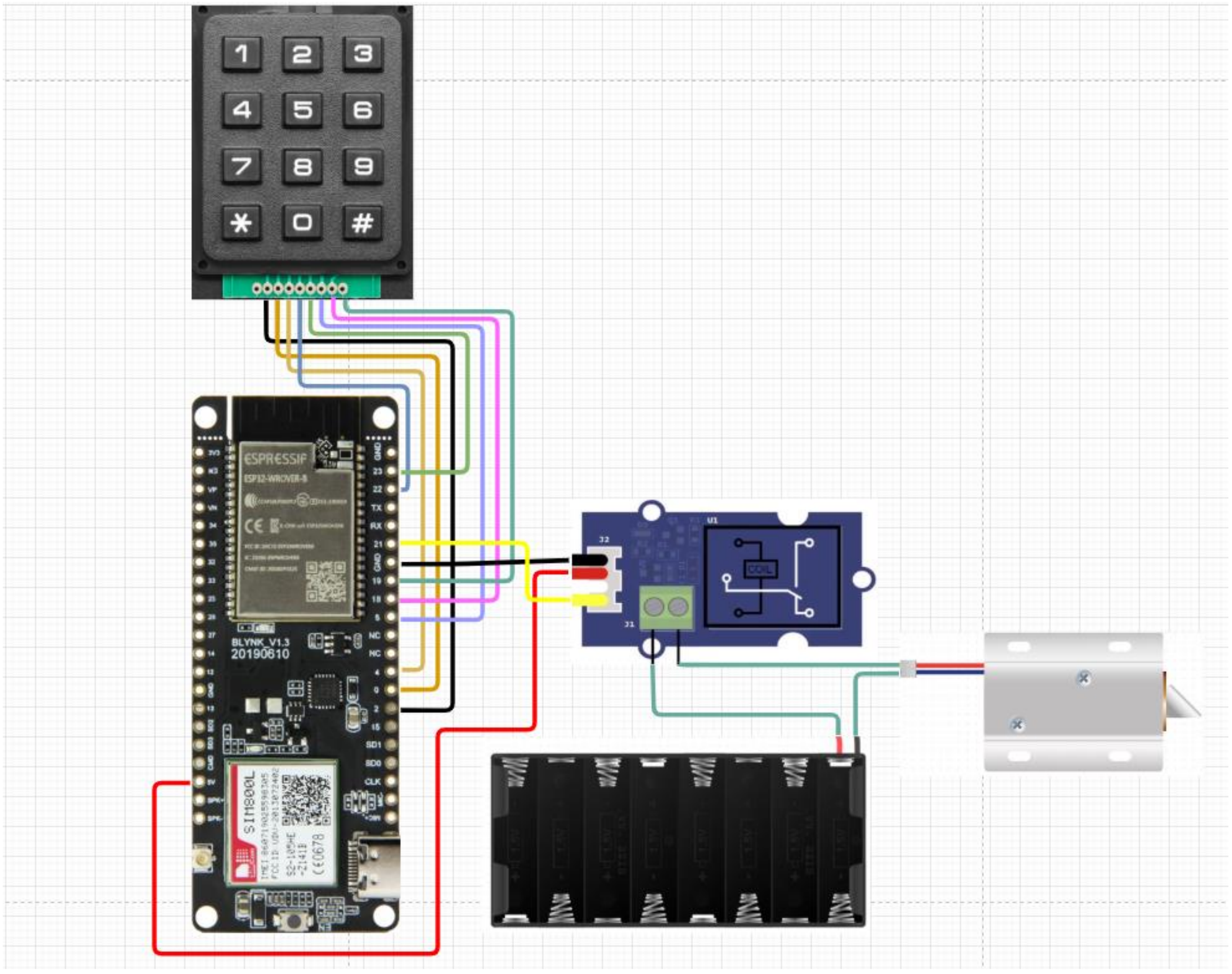


Figure 20 - Locking mechanism wiring diagram

SOLAR POWER SYSTEM

- The solar system described in figure 19 is rated for 100w solar panel, so 20A inline fuses are in place to protect from over current.
- For smaller solar system, as will be demonstrated, inline fuses will not be necessary as the charge controller can modulate the current sufficiently.

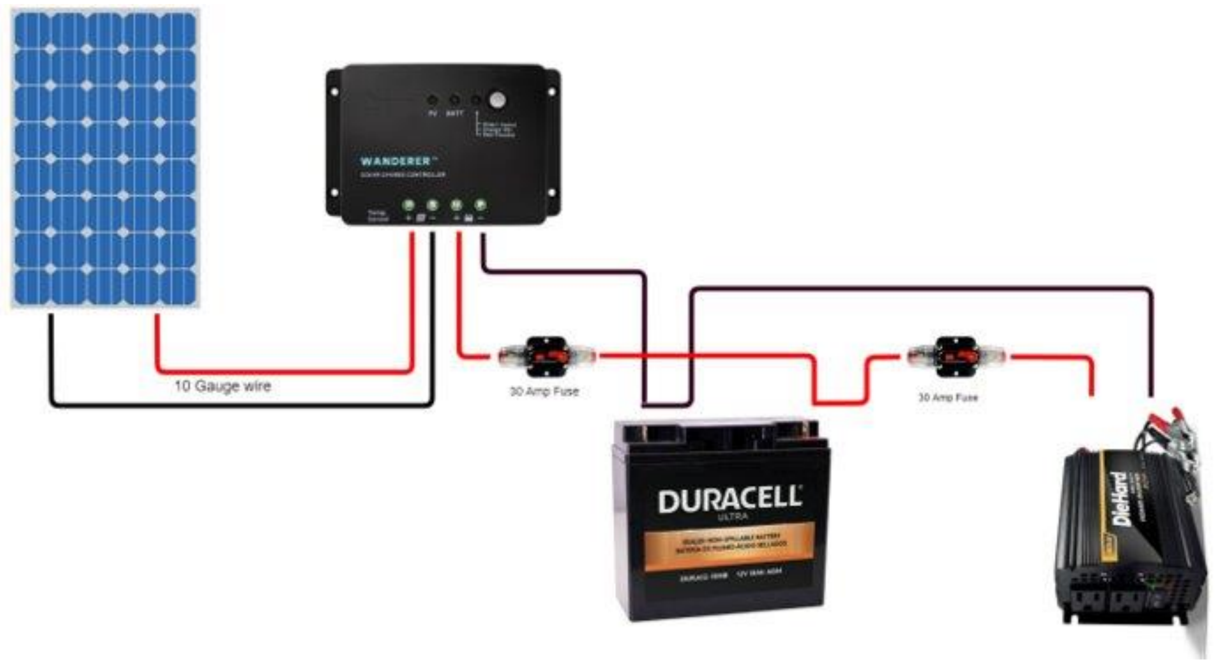


Figure 21 - Solar system diagram

- Solar Panel:
  - Wire solar panel with 10-gauge wire.
  - Connect power wires to input of charge controller.
- Charge controller:
  - Wire charge controller with 10-gauge wire.
  - Connect +/- output of charge controller to +/- terminals of battery.
  - Add 30A inline fuse to the + wire from the charge controller to the battery.
- Inverter:
  - Wire inverter with 10-gauge wire.
  - Connect inverter +/- cables to +/- terminals of battery.
  - Add 30A inline fuse to the + wire from the battery to the inverter.

ASSEMBLED DESIGN

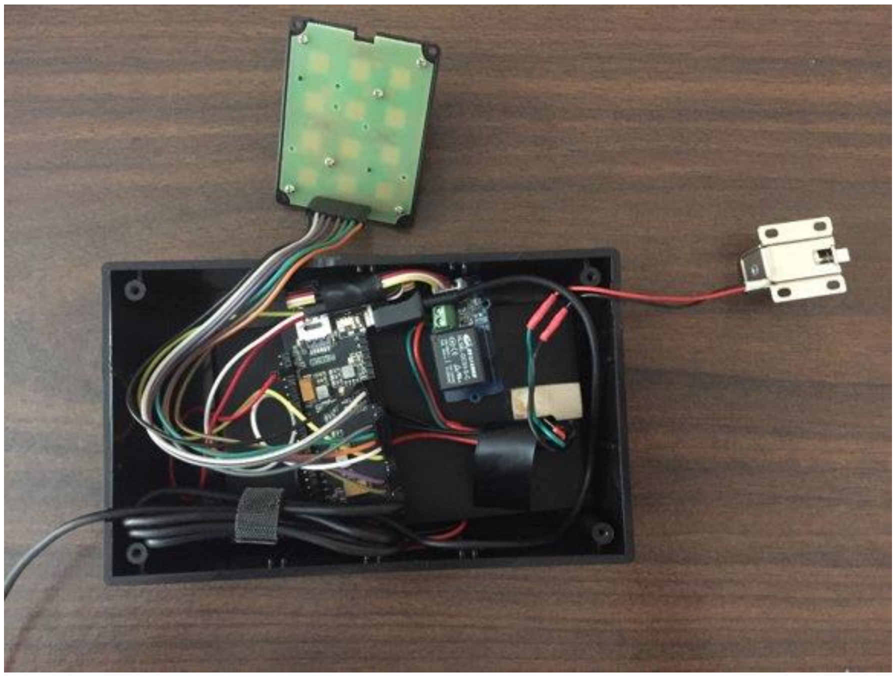


Figure 22 - Locking mechanism

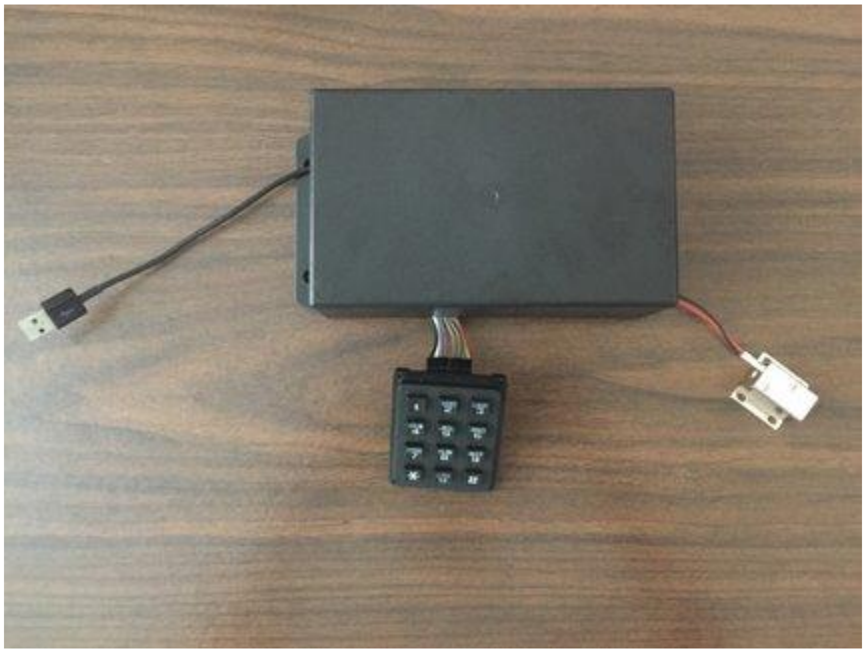


Figure 23- Locking mechanism 2

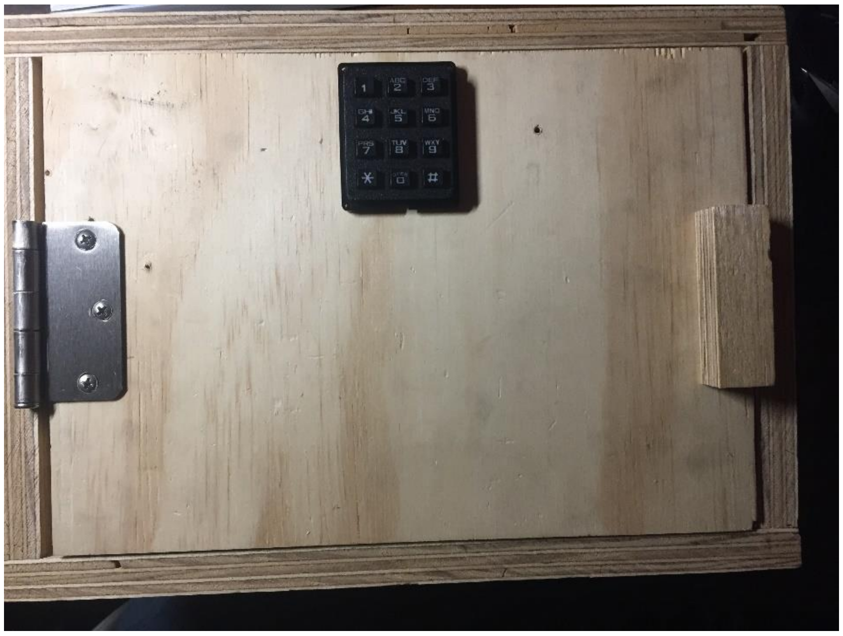


Figure 24 - Locker and locking mechanism

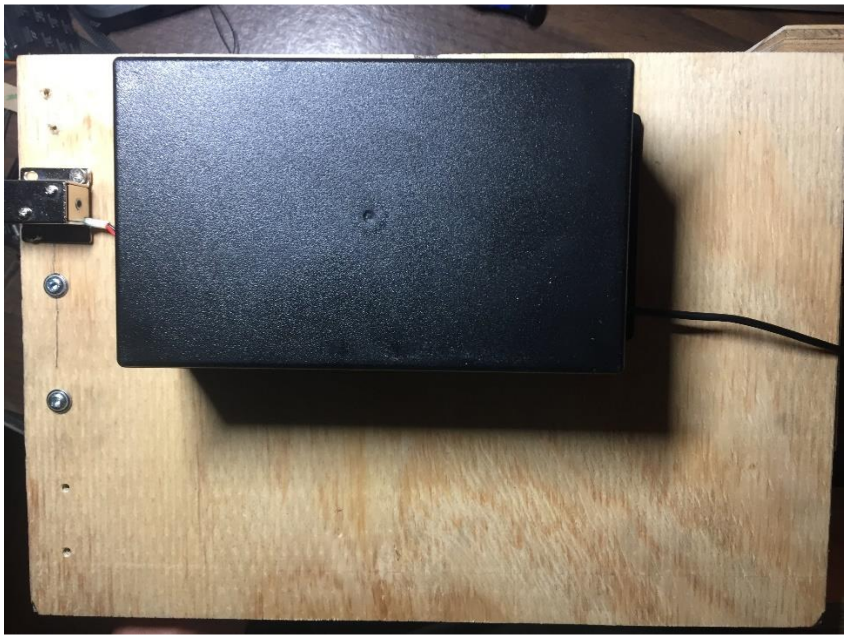


Figure 25 - Locker and locking mechanism 2

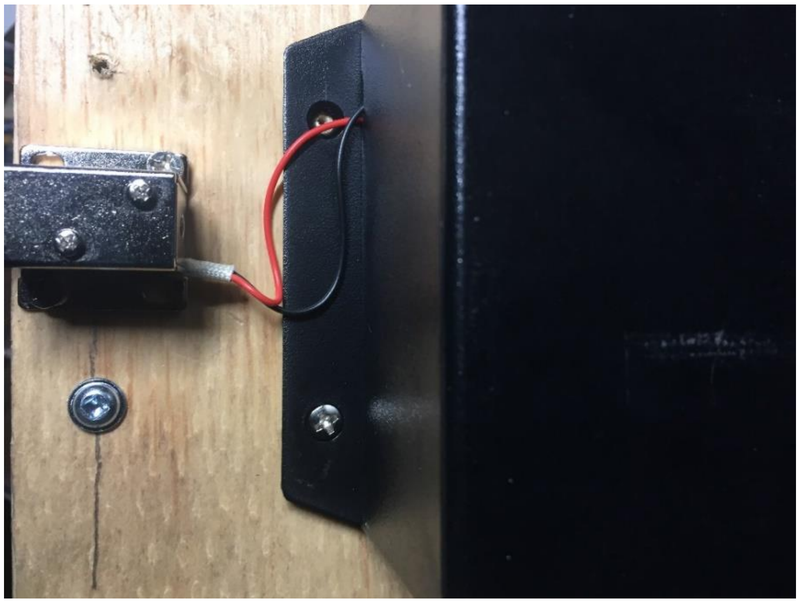


Figure 26 - Locker and locking mechanism 3

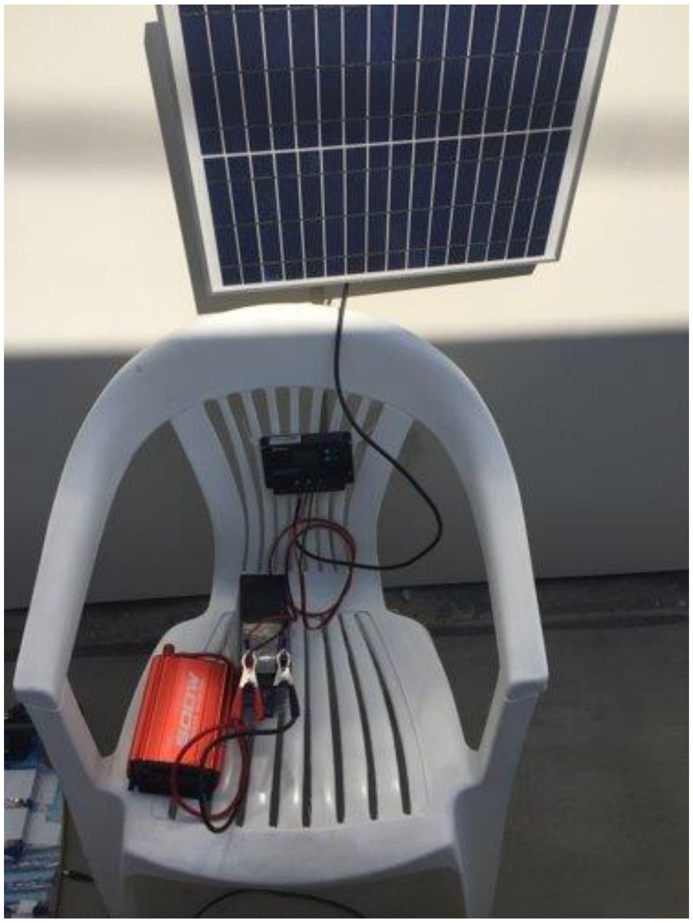
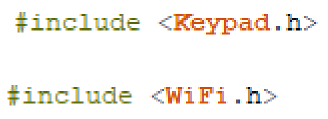


Figure 27 - Solar system

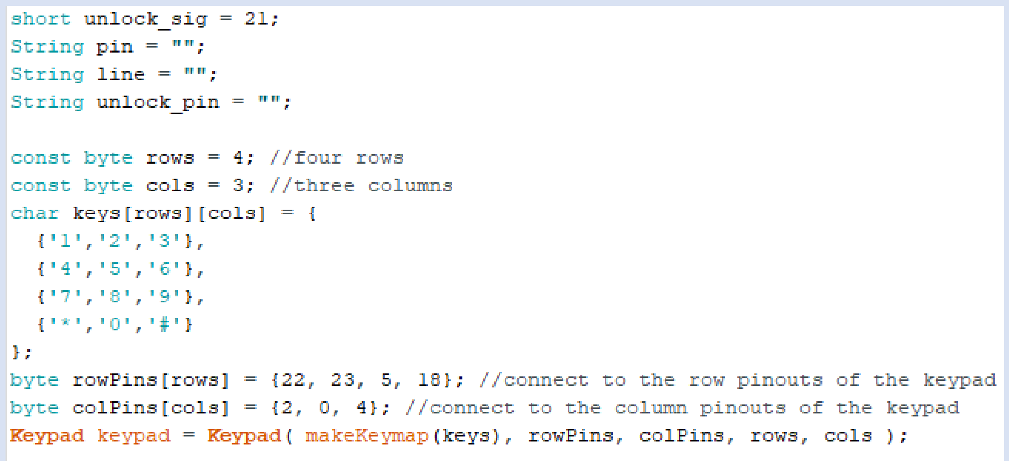
4.3.2 SOFTWARE DESCRIPTION

- ESP32 TTGO T-Call:
    - Receive data from keypad. Decode data from keypad to a PIN number.
    - Match PIN number to data base.
    - Unlock locking solenoid via GPIO if PIN matches.



```
#include <Keypad.h>

#include <WiFi.h>
```

Two libraries used for communication with the keypad and WIFI

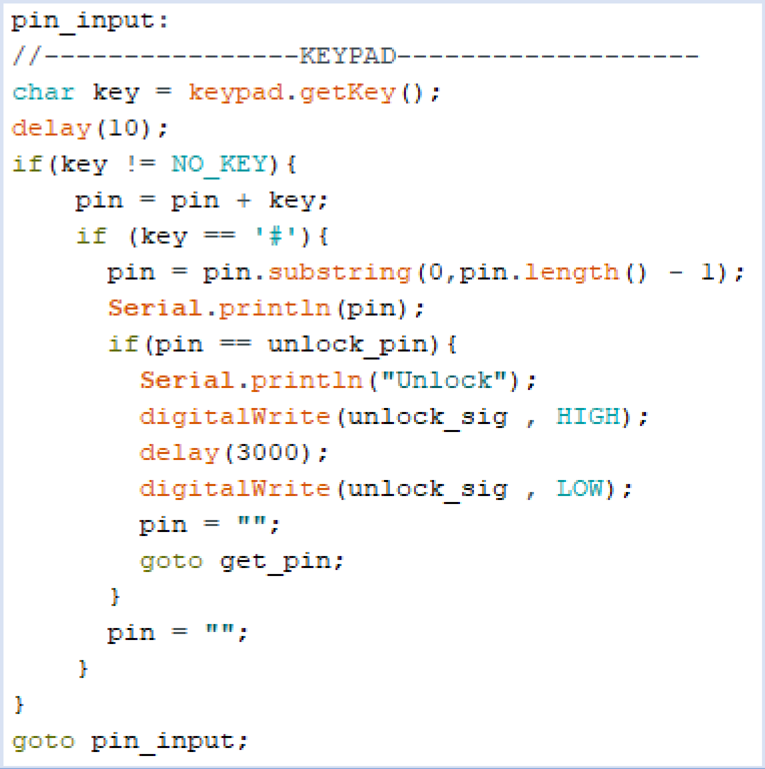


```
short unlock_sig = 21;
String pin = "";
String line = "";
String unlock_pin = "";

const byte rows = 4; //four rows
const byte cols = 3; //three columns
char keys[rows][cols] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
byte rowPins[rows] = {22, 23, 5, 18}; //connect to the row pinouts of the keypad
byte colPins[cols] = {2, 0, 4}; //connect to the column pinouts of the keypad
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, rows, cols );
```

Initializing the 3x4 matrix keypad.

```
//---------------Getting pin from server-------------//
get_pin:
pinMode(unlock_sig, OUTPUT);
delay(100);
WiFiClient client;
Serial.printf("\n[Connecting to %s ... ", host);
if (client.connect(host, 3000)){
  Serial.println("[connected]");
  client.println("GET /pin HTTP/1.0\n");
  Serial.println("[Response:]");
  while (client.connected() || client.available()){
    if (client.available()){
      String line = client.readStringUntil('\n');
      if(line.length() == 4){
         Serial.println("gotthepin");
         unlock_pin = line;
         Serial.println(unlock_pin);
         goto pin_input;
      } //End if
    } // End if
  } // End while

  client.stop();
  Serial.println("\n[Disconnected]");
}
else{
  Serial.println("connection failed!]");
  client.stop();
}
```

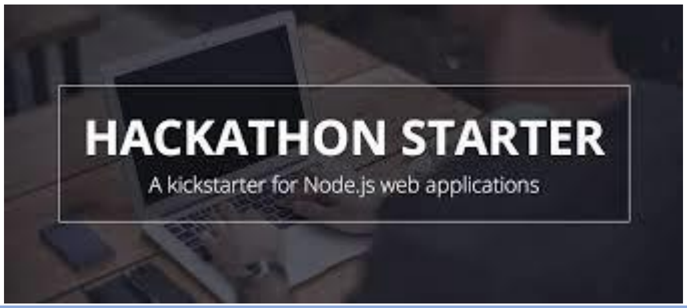
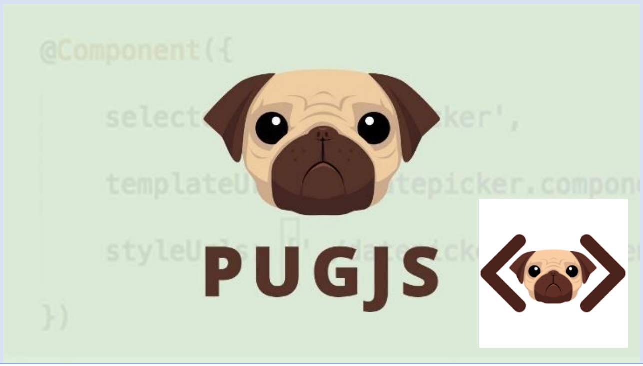
Http get request to gather data from the server. Data in this case is the 4-digit pin.



```
pin_input:
//----------------KEYPAD-------------------
char key = keypad.getKey();
delay(10);
if(key != NO_KEY){
    pin = pin + key;
    if (key == '#'){
      pin = pin.substring(0,pin.length() - 1);
      Serial.println(pin);
      if(pin == unlock_pin){
        Serial.println("Unlock");
        digitalWrite(unlock_sig , HIGH);
        delay(3000);
        digitalWrite(unlock_sig , LOW);
        pin = "";
        goto get_pin;
      }
      pin = "";
    }
}
goto pin_input;
```

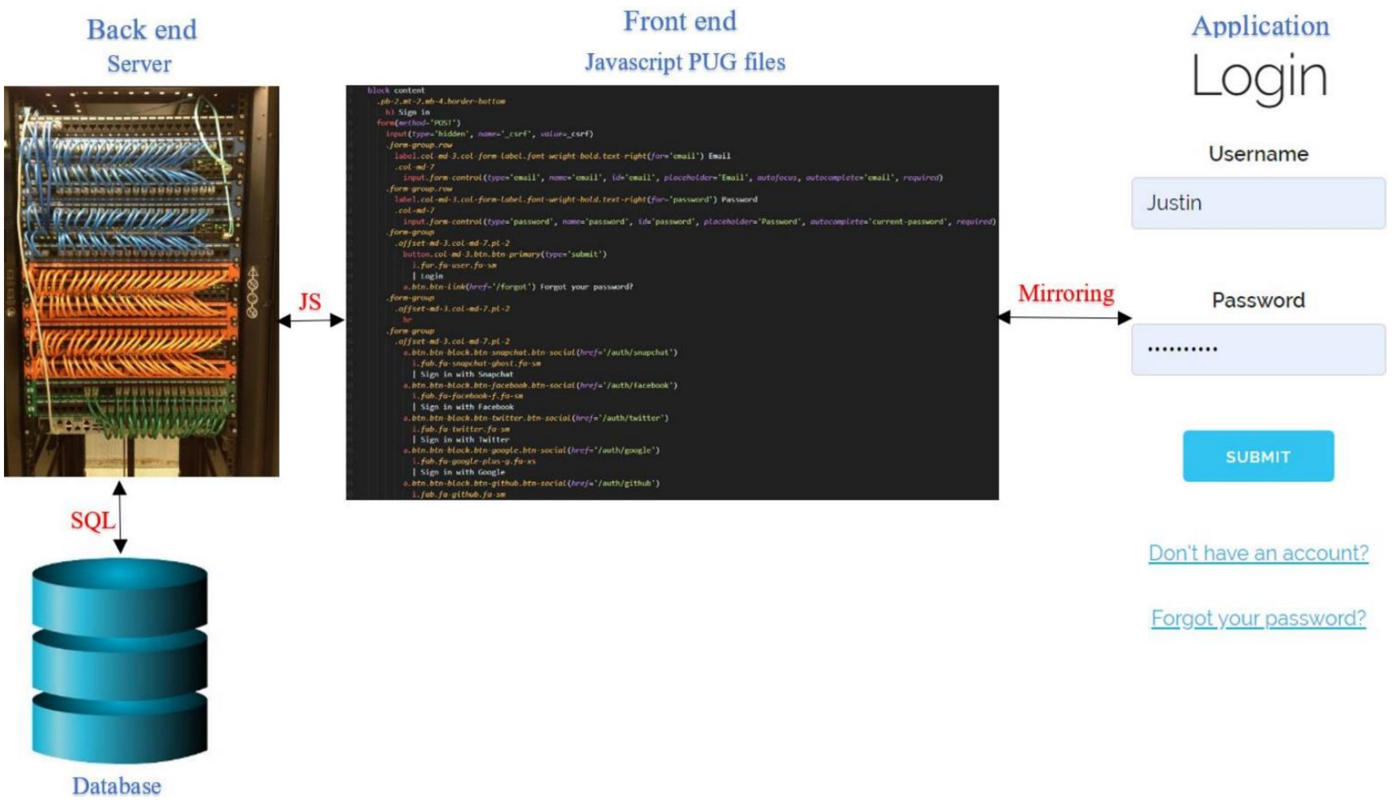
Once pin is received, wait for input from the keypad. Then compare the input pin to the unlock pin received from sever. If the pins match, send a signal which will unlock the solenoid.

4.4 SERVER & DATABASE

 HACKATHON STARTER A kickstarter for Node.js web applications	**First, we downloaded the hackathon boilerplate.**
	The boilerplate comes with mongoDB which is a built-in database to hold the user's private information securely.
	We bought a server on DigitalOcean with one Gigabyte of RAM, 25 Gigabytes storage, one CPU, and one Terabyte data transfer for only $5 per month.
	The application frontend is programmed with pug JavaScript and CSS onto a website domain http://doct.fbi.moe/.
	To interface with the database, we are utilizing MobaXTerm with an SSH protocol.

	We program in C++ on Arduino IDE to program the ESP32 TTGO TCALL Wrover chip to gather information communicate with the server.
	Every 15 seconds we concatinated these values into JSON format and sent the string data from the ESP32 TTGO TCALL through the modem with a sim card over 2G network with T-mobile carrier and Hologram APN.
	Once the server receives this information, the latitude and longitude are inputs to the Google Maps API and give a live update on where the vehicle is in the world.

SERVER COMMUNICATION FLOW



```
block content
  .pb-2.mt-2.mb-4.border-bottom
    h1 Sign in
  form(method='POST')
    input(type='hidden', name='_csrf', value=_csrf)
    .form-group.row
      label.col-md-3.col-form-label.font-weight-bold.text-right(for='email') Email
      .col-md-7
        input.form-control(type='email', name='email', id='email', placeholder='Email', autofocus, autocomplete='email', required)
    .form-group.row
      label.col-md-3.col-form-label.font-weight-bold.text-right(for='password') Password
      .col-md-7
        input.form-control(type='password', name='password', id='password', placeholder='Password', autocomplete='current-password', required)
    .form-group
      .offset-md-3.col-md-7.pl-2
        button.col-md-3.btn.btn-primary(type='submit')
          i.far.fa-user.fa-sm
          | Login
        a.btn.btn-link(href='/forgot') Forgot your password?
    .form-group
      .offset-md-3.col-md-7.pl-2
        hr
    .form-group
      .offset-md-3.col-md-7.pl-2
        a.btn.btn-block.btn-snapchat.btn-social(href='/auth/snapchat')
          i.fab.fa-snapchat-ghost.fa-sm
          | Sign in with Snapchat
        a.btn.btn-block.btn-facebook.btn-social(href='/auth/facebook')
          i.fab.fa-facebook-f.fa-sm
          | Sign in with Facebook
        a.btn.btn-block.btn-twitter.btn-social(href='/auth/twitter')
          i.fab.fa-twitter.fa-sm
          | Sign in with Twitter
        a.btn.btn-block.btn-google.btn-social(href='/auth/google')
          i.fab.fa-google-plus-g.fa-xs
          | Sign in with Google
        a.btn.btn-block.btn-github.btn-social(href='/auth/github')
          i.fab.fa-github.fa-sm
```

SOFTWARE

The second part of this project was constructing the application. First, we downloaded the hackathon boilerplate. We then bought a server on Digitalocean with one Gigabyte of RAM, 25 Gigabytes storage, one CPU, and one Terabyte data transfer for only $5 per month. The application frontend is programmed with pug JavaScript and CSS onto a website domain http://doct.fbi.moe/.

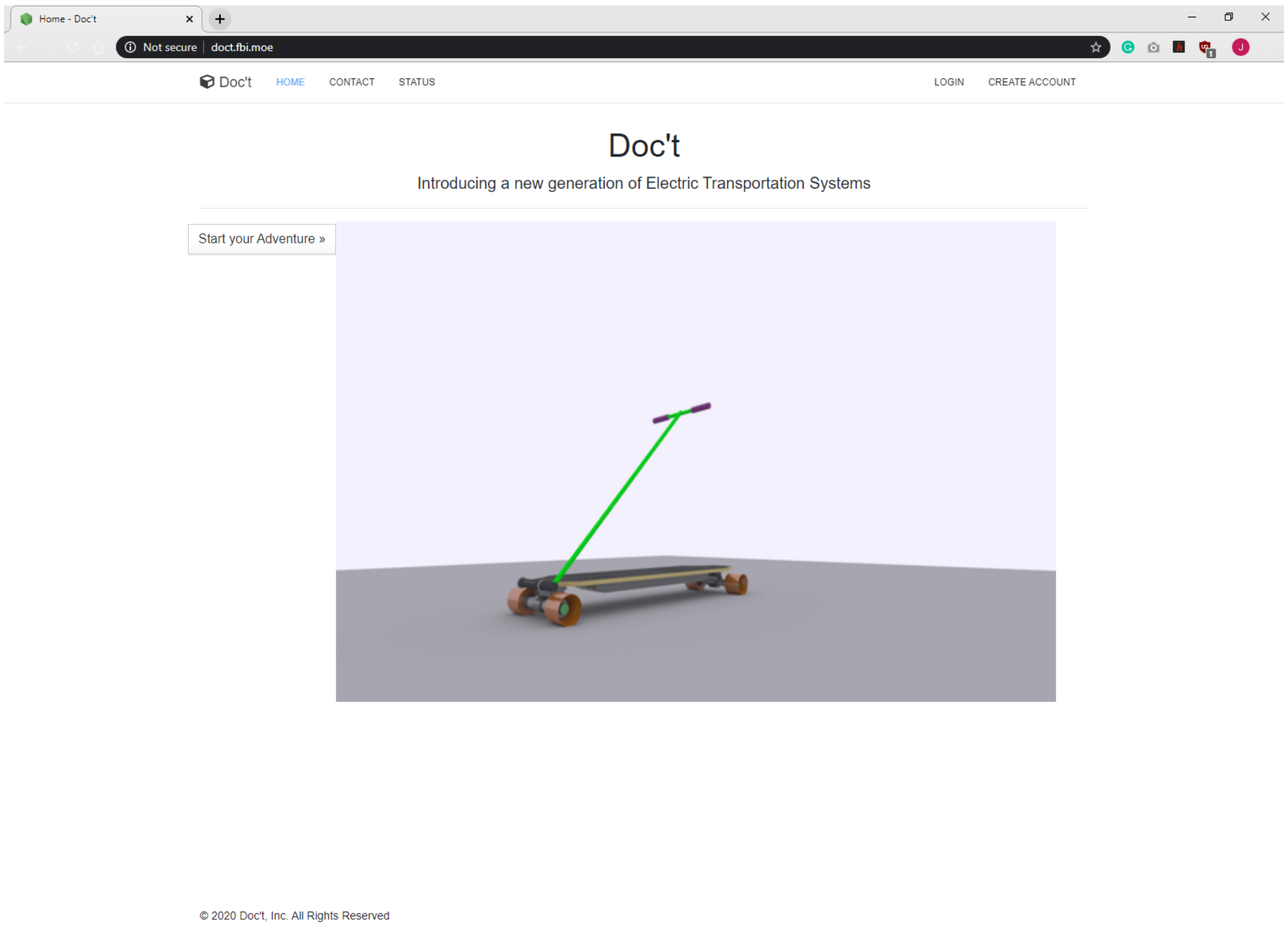
The register-user page will give riders the option to register as a new user with name, username, password, and email. Each user password is encrypted. Next, the user can login and request a vehicle, and the closest available vehicle will display as well as a server 4-digit RNG PIN. If the user forgets their password, then there will be a page to change their password. We have also setup a contact us page if there are any comments or advice for our electric transportation system. The server status is also available for the developers to make sure all of our software and hardware is doing what it is supposed to do.

The backend consists of the server and database. To interface with the database, we are utilizing MobaXTerm with an SSH protocol. In the database we are storing all the private rider's information as well as the vehicle information. The vehicle information includes the time that the vehicle has been checked out and when it will be checked in, the range in miles, the current battery percentage, the current GPS location (longitude and latitude, and the temporary 4 digit PIN given to the user to check out the vehicle from the Doc't HUB. The server will also contain the front end which contains the graphical user interface. The user will be able to go through our easy-to-navigate website to select a ride and get to their final destination with ease. The web application will mirror the front end to prevent the users from accessing confidential interface from our private database as an extra security precaution.

4.4.1 DATABASE

As mentioned, before we purchased a server through Digitalocean to handle our database, website, and application. First, we utilized hackathon boilerplate which uses PUG JavaScript to create a template for the frontend. We then modified it to include a homepage and pages for the user to register, login, book ride, My Account, forgot password, and status page. The boilerplate comes with mongoDB which is a built-in database to hold the user's private information securely. All the pages are currently displayed in the upcoming figures of this section.

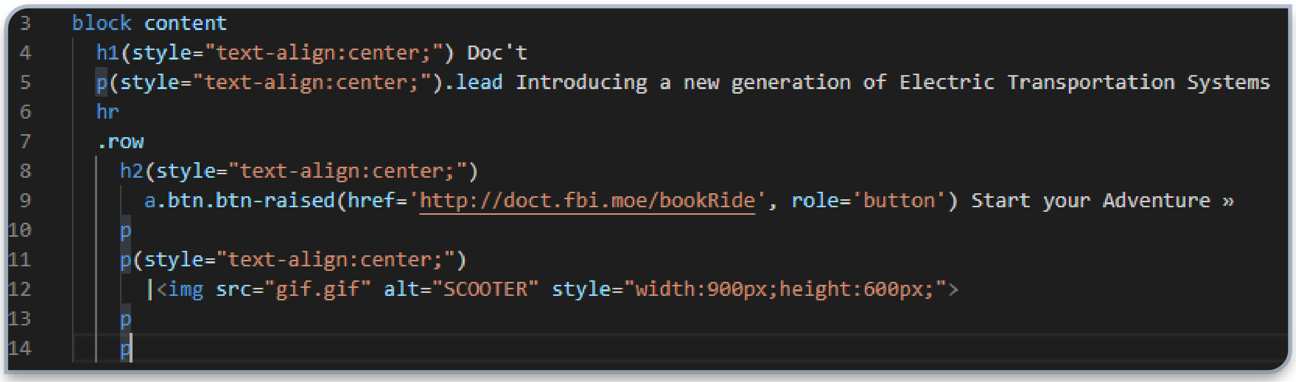
The GPS module uses the TinyGPS++ library to extract and decode GPS information. From this we pulled the latitude, longitude, range, battery, speed, and time. For our purposes we only neeeded this information every one minute. Every 15 seconds we concatinated these values into JSON format and sent the string data from the ESP32 TTGO TCALL through the modem with a sim card over 2G network with T-mobile carrier and Hologram APN. Once the server receives this information, the latitude and longitude are inputs to the Google Maps API and give a live update on where the vehicle is in the world. All of this is shown in extensive detail with pictures in this section.



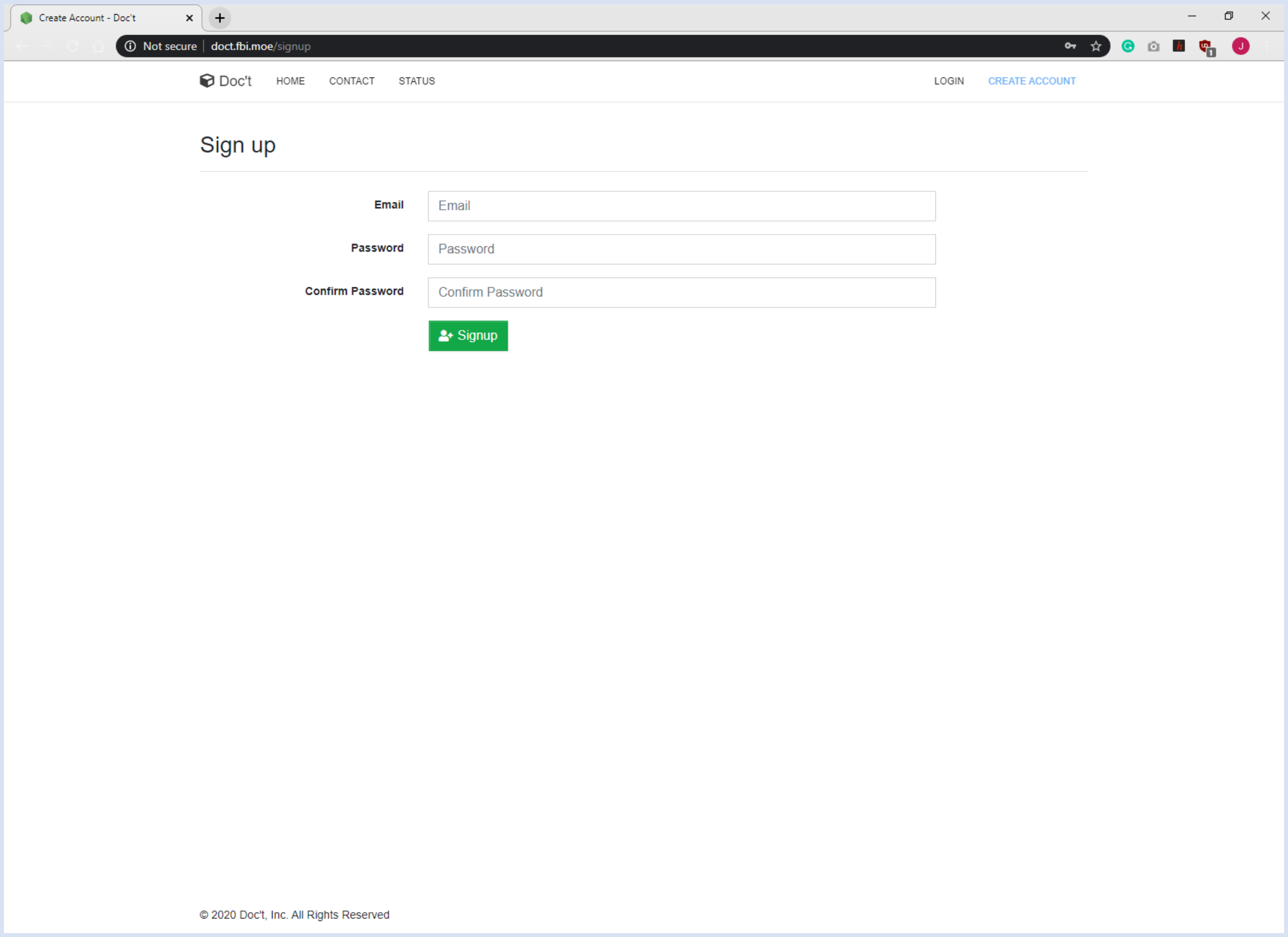
Doc't

Introducing a new generation of Electric Transportation Systems

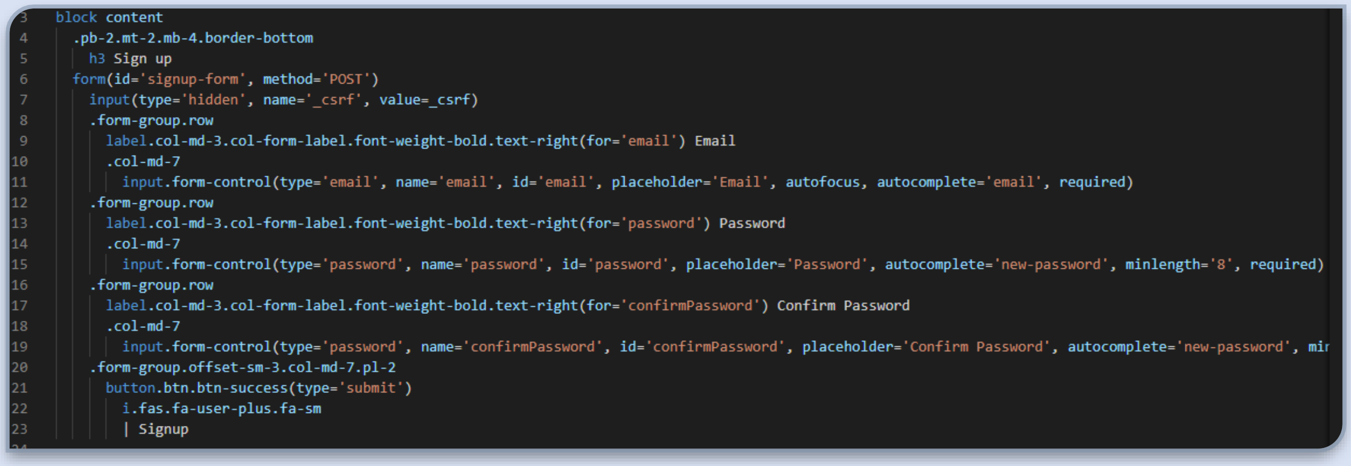
© 2020 Doc't, Inc. All Rights Reserved



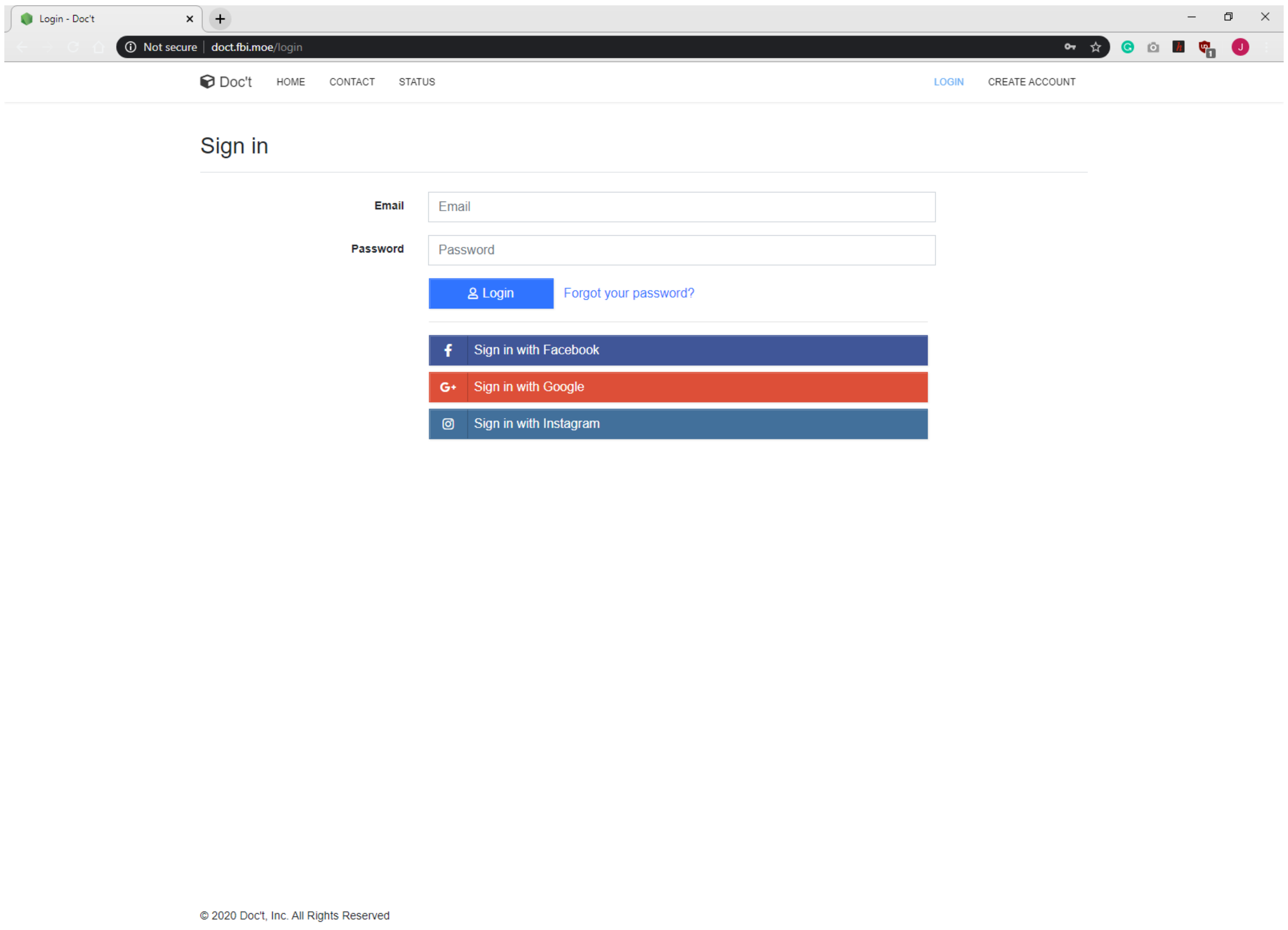
```
block content
  h1(style="text-align:center;") Doc't
  p(style="text-align:center;").lead Introducing a new generation of Electric Transportation Systems
  hr
  .row
    h2(style="text-align:center;")
      a.btn.btn-raised(href='http://doct.fbi.moe/bookRide', role='button') Start your Adventure »
    p
    p(style="text-align:center;")
      |<img src="gif.gif" alt="SCOOTER" style="width:900px;height:600px;">
    p
    p
```

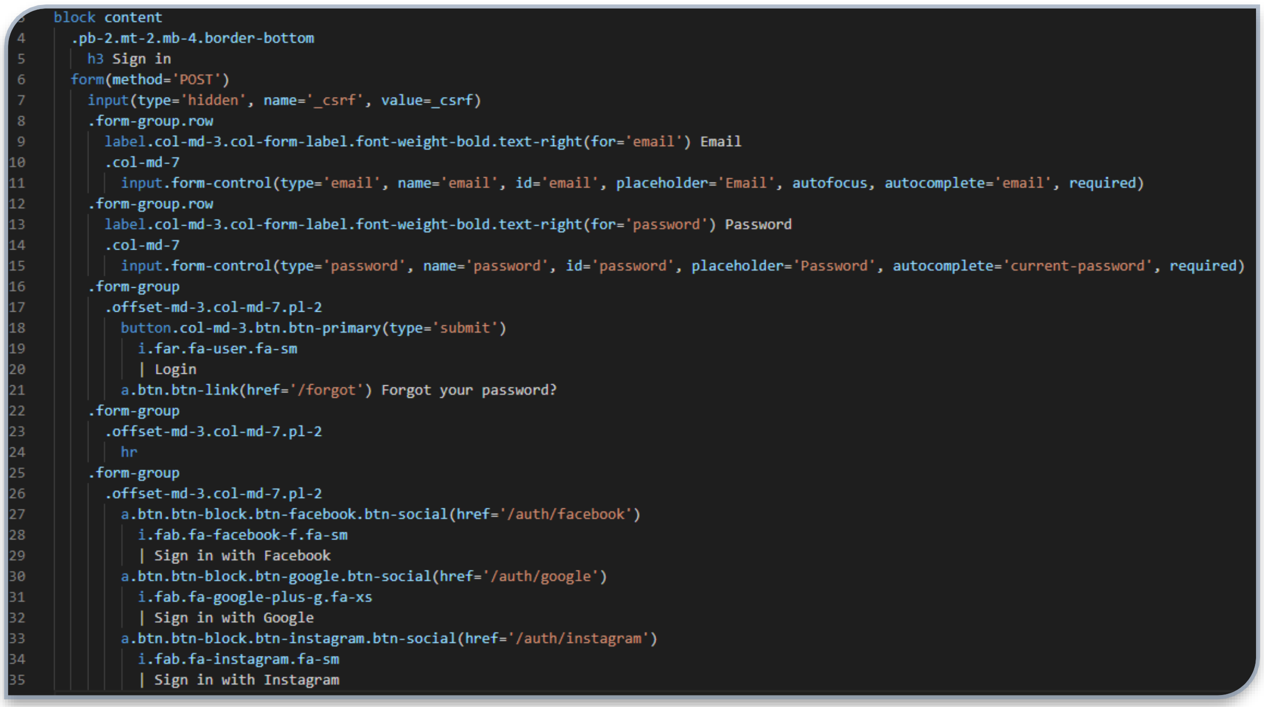


© 2020 Doc't, Inc. All Rights Reserved

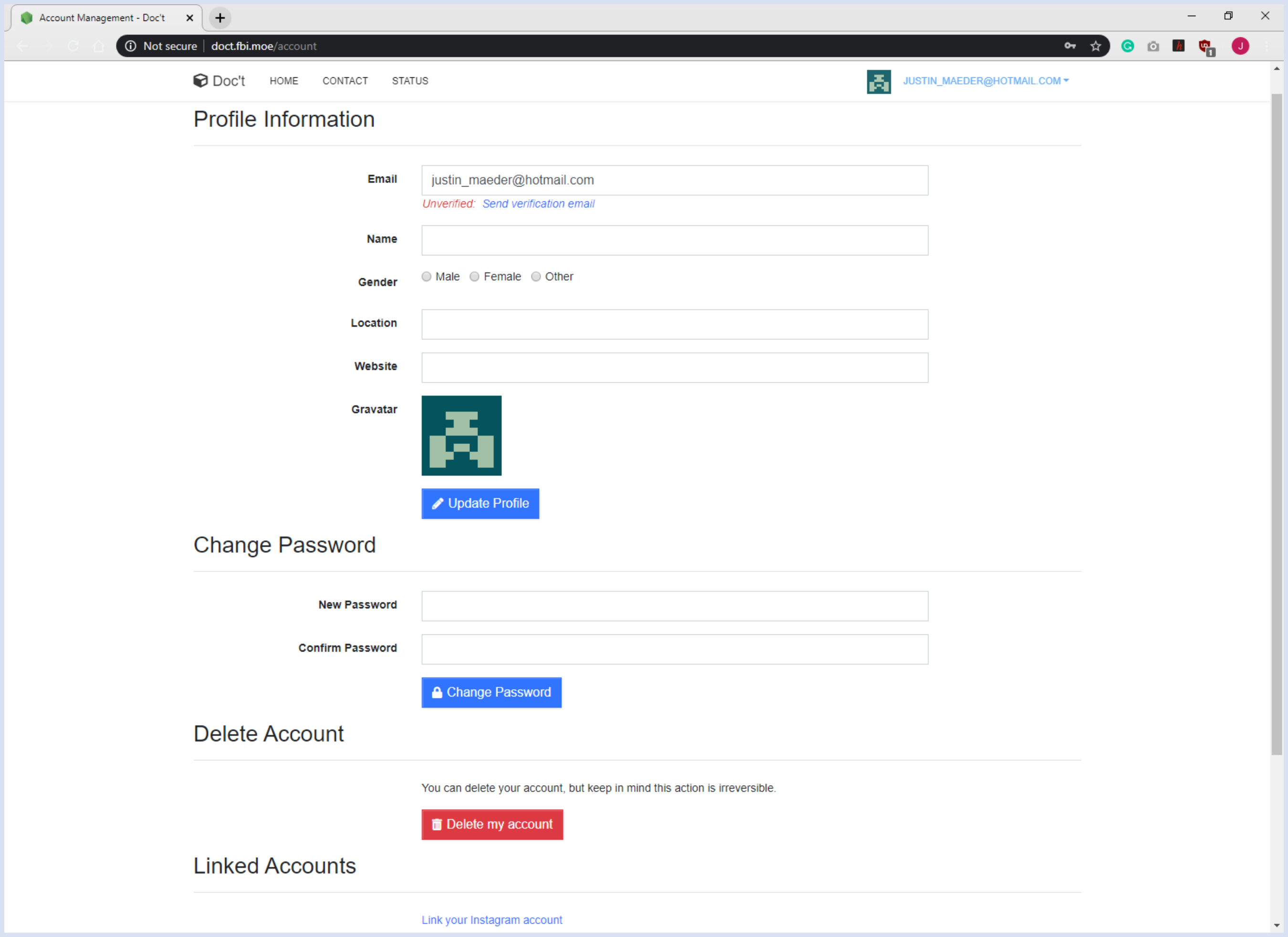


```
block content
  .pb-2.mt-2.mb-4.border-bottom
    h3 Sign up
  form(id='signup-form', method='POST')
    input(type='hidden', name='_csrf', value=_csrf)
    .form-group.row
      label.col-md-3.col-form-label.font-weight-bold.text-right(for='email') Email
      .col-md-7
        input.form-control(type='email', name='email', id='email', placeholder='Email', autofocus, autocomplete='email', required)
    .form-group.row
      label.col-md-3.col-form-label.font-weight-bold.text-right(for='password') Password
      .col-md-7
        input.form-control(type='password', name='password', id='password', placeholder='Password', autocomplete='new-password', minlength='8', required)
    .form-group.row
      label.col-md-3.col-form-label.font-weight-bold.text-right(for='confirmPassword') Confirm Password
      .col-md-7
        input.form-control(type='password', name='confirmPassword', id='confirmPassword', placeholder='Confirm Password', autocomplete='new-password', mi
    .form-group.offset-sm-3.col-md-7.pl-2
      button.btn.btn-success(type='submit')
        i.fas.fa-user-plus.fa-sm
        | Signup
```





```
block content
  .pb-2.mt-2.mb-4.border-bottom
    h3 Sign in
  form(method='POST')
    input(type='hidden', name='_csrf', value=_csrf)
    .form-group.row
      label.col-md-3.col-form-label.font-weight-bold.text-right(for='email') Email
      .col-md-7
        input.form-control(type='email', name='email', id='email', placeholder='Email', autofocus, autocomplete='email', required)
    .form-group.row
      label.col-md-3.col-form-label.font-weight-bold.text-right(for='password') Password
      .col-md-7
        input.form-control(type='password', name='password', id='password', placeholder='Password', autocomplete='current-password', required)
    .form-group
      .offset-md-3.col-md-7.pl-2
        button.col-md-3.btn.btn-primary(type='submit')
          i.far.fa-user.fa-sm
          | Login
        a.btn.btn-link(href='/forgot') Forgot your password?
    .form-group
      .offset-md-3.col-md-7.pl-2
        hr
    .form-group
      .offset-md-3.col-md-7.pl-2
        a.btn.btn-block.btn-facebook.btn-social(href='/auth/facebook')
          i.fab.fa-facebook-f.fa-sm
          | Sign in with Facebook
        a.btn.btn-block.btn-google.btn-social(href='/auth/google')
          i.fab.fa-google-plus-g.fa-xs
          | Sign in with Google
        a.btn.btn-block.btn-instagram.btn-social(href='/auth/instagram')
          i.fab.fa-instagram.fa-sm
          | Sign in with Instagram
```



Account Management - Doc't

Not secure | doct.fbi.moe/account

Doc't HOME CONTACT STATUS JUSTIN_MAEDER@HOTMAIL.COM

Profile Information

Email: justin_maeder@hotmail.com

Unverified: Send verification email

Name

Gender: Male Female Other

Location

Website

Gravatar

Update Profile

Change Password

New Password

Confirm Password

Change Password

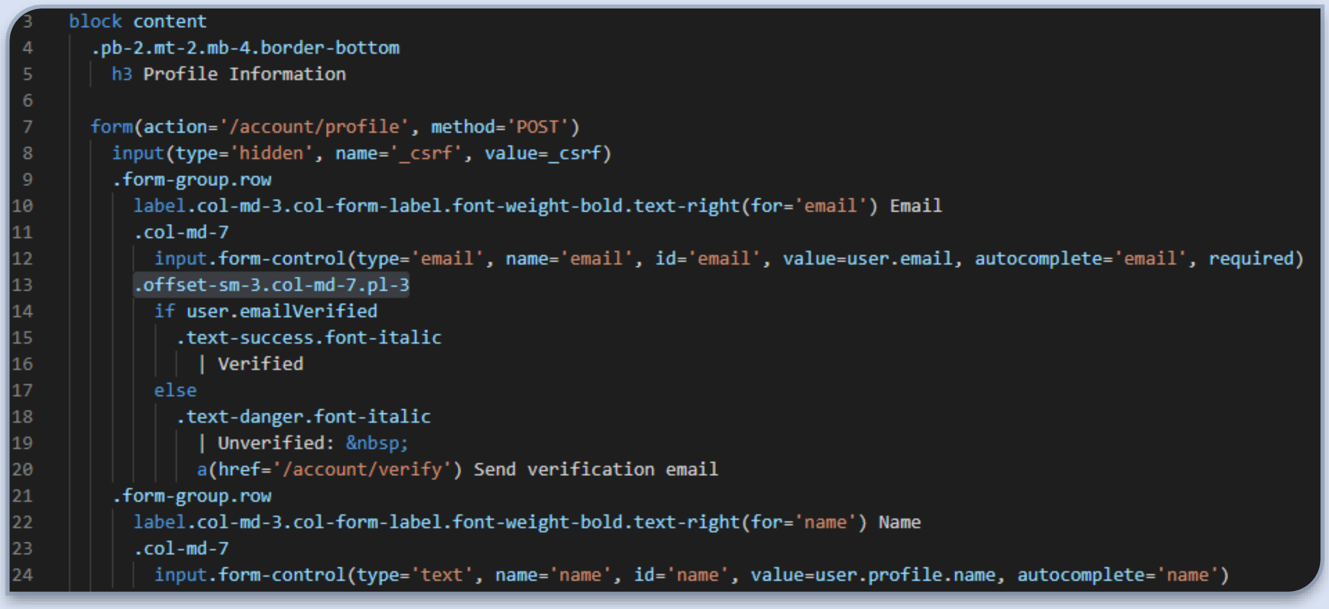
Delete Account

You can delete your account, but keep in mind this action is irreversible.

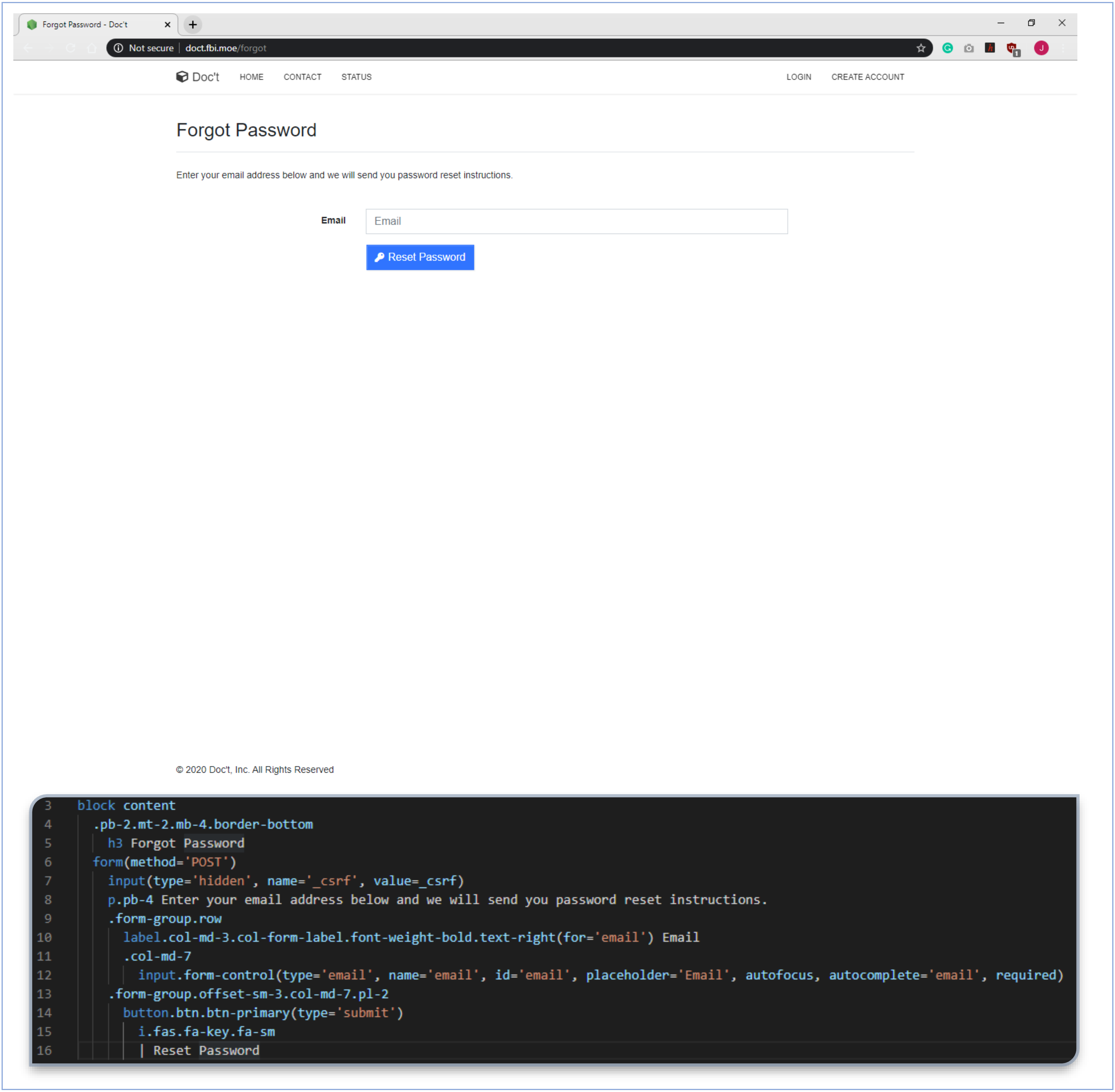
Delete my account

Linked Accounts

Link your Instagram account



```
block content
  .pb-2.mt-2.mb-4.border-bottom
    h3 Profile Information

  form(action='/account/profile', method='POST')
    input(type='hidden', name='_csrf', value=_csrf)
    .form-group.row
      label.col-md-3.col-form-label.font-weight-bold.text-right(for='email') Email
      .col-md-7
        input.form-control(type='email', name='email', id='email', value=user.email, autocomplete='email', required)
      .offset-sm-3.col-md-7.pl-3
        if user.emailVerified
          .text-success.font-italic
            | Verified
        else
          .text-danger.font-italic
            | Unverified: &nbsp;
            a(href='/account/verify') Send verification email
    .form-group.row
      label.col-md-3.col-form-label.font-weight-bold.text-right(for='name') Name
      .col-md-7
        input.form-control(type='text', name='name', id='name', value=user.profile.name, autocomplete='name')
```



Forgot Password

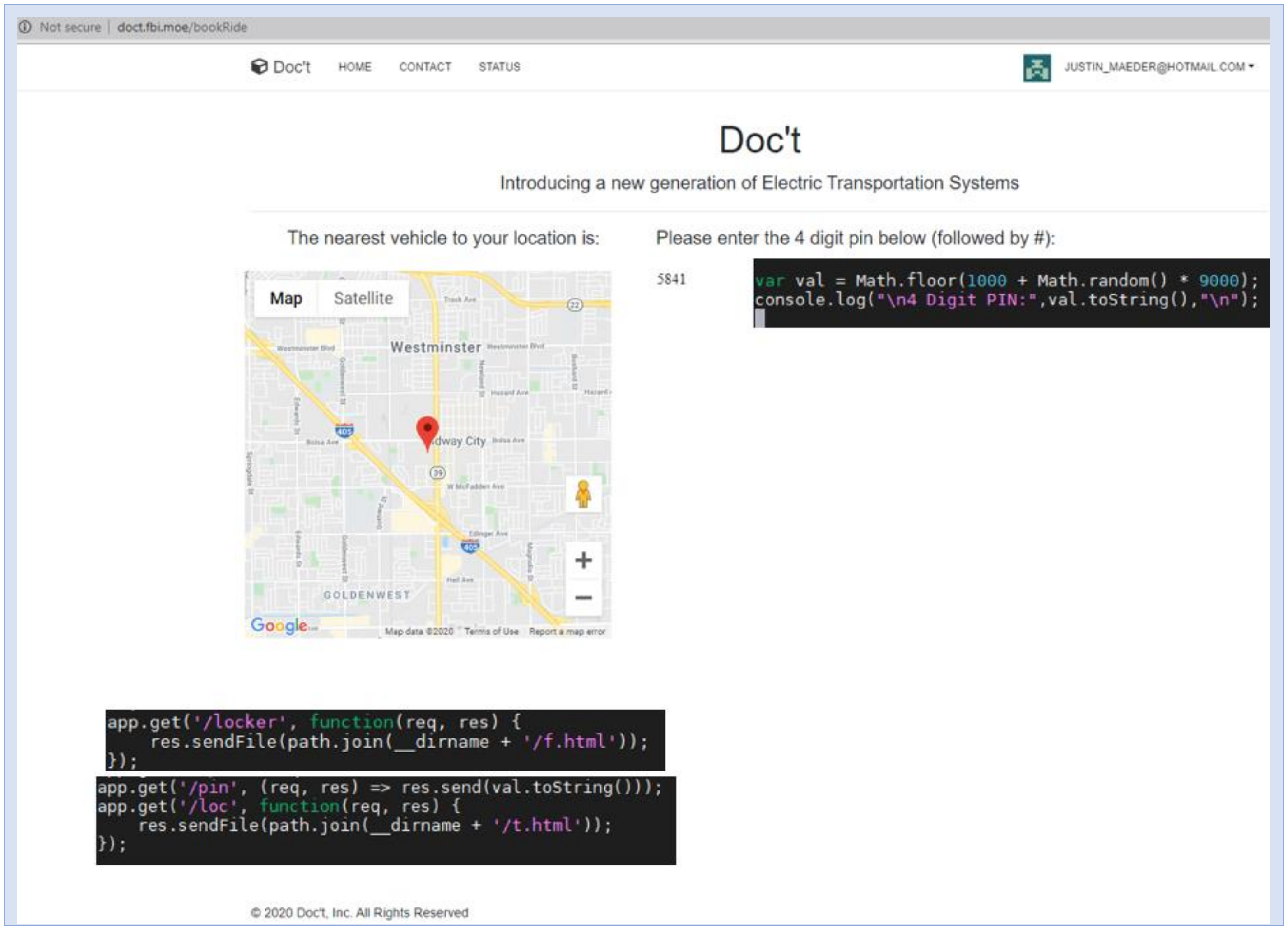
Enter your email address below and we will send you password reset instructions.

Email: Email

Reset Password

© 2020 Doc't, Inc. All Rights Reserved

```
block content
  .pb-2.mt-2.mb-4.border-bottom
    h3 Forgot Password
  form(method='POST')
    input(type='hidden', name='_csrf', value=_csrf)
    p.pb-4 Enter your email address below and we will send you password reset instructions.
    .form-group.row
      label.col-md-3.col-form-label.font-weight-bold.text-right(for='email') Email
      .col-md-7
        input.form-control(type='email', name='email', id='email', placeholder='Email', autofocus, autocomplete='email', required)
    .form-group.offset-sm-3.col-md-7.pl-2
      button.btn.btn-primary(type='submit')
        i.fas.fa-key.fa-sm
        | Reset Password
```


Not secure | doct.fbi.moe/bookRide
Doc't HOME CONTACT STATUS
JUSTIN_MAEDER@HOTMAIL.COM

Doc't

Introducing a new generation of Electric Transportation Systems

The nearest vehicle to your location is:

Map
Satellite
Westminster
Midway City
GOLDENWEST
Google
Map data ©2020 Terms of Use Report a map error

Please enter the 4 digit pin below (followed by #):

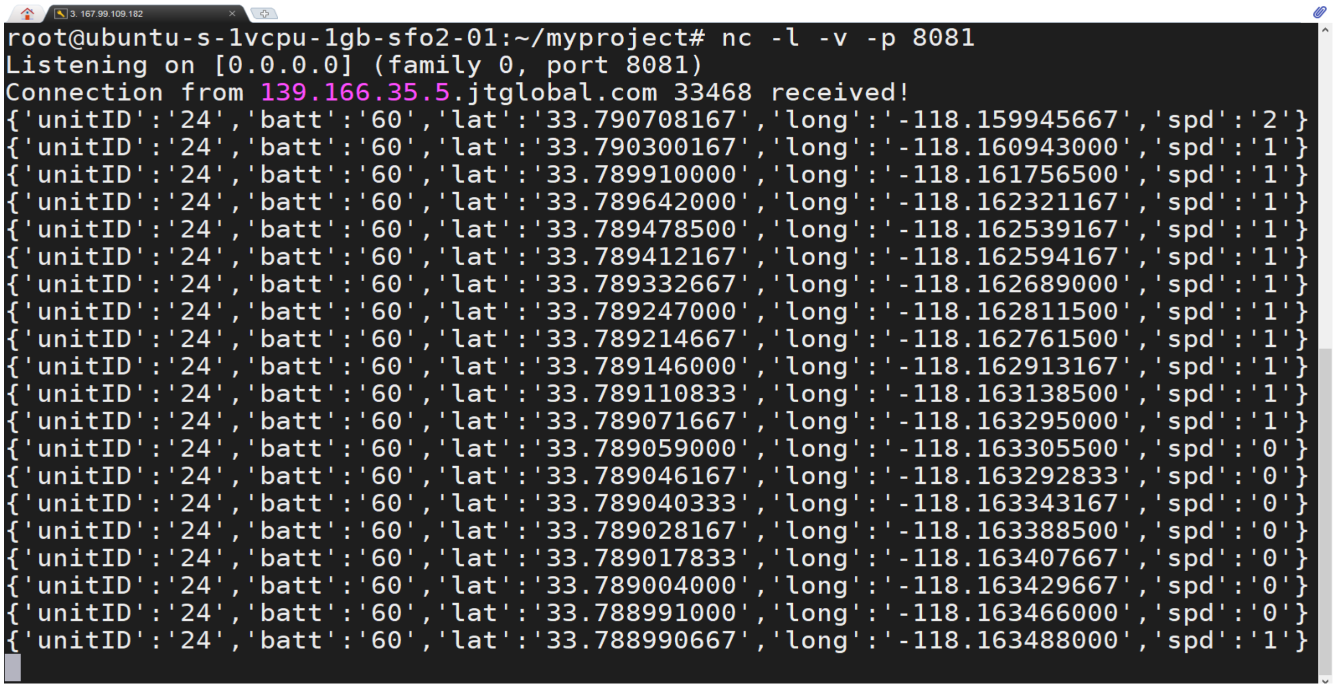
5841

```
var val = Math.floor(1000 + Math.random() * 9000);
console.log("\n4 Digit PIN:",val.toString(),"\n");
```

```
app.get('/locker', function(req, res) {
    res.sendFile(path.join(__dirname + '/f.html'));
});
```

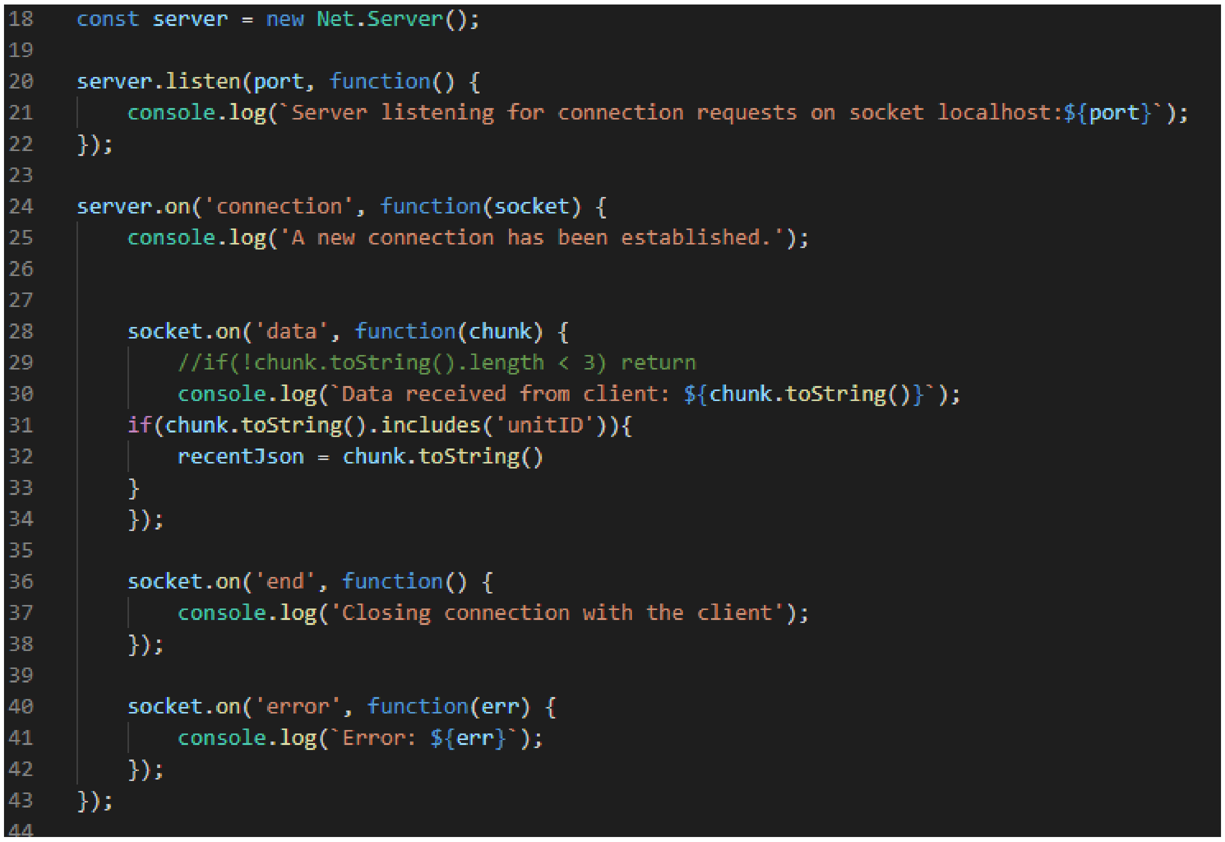
```
app.get('/pin', (req, res) => res.send(val.toString()));
app.get('/loc', function(req, res) {
    res.sendFile(path.join(__dirname + '/t.html'));
});
```

© 2020 Doc't, Inc. All Rights Reserved



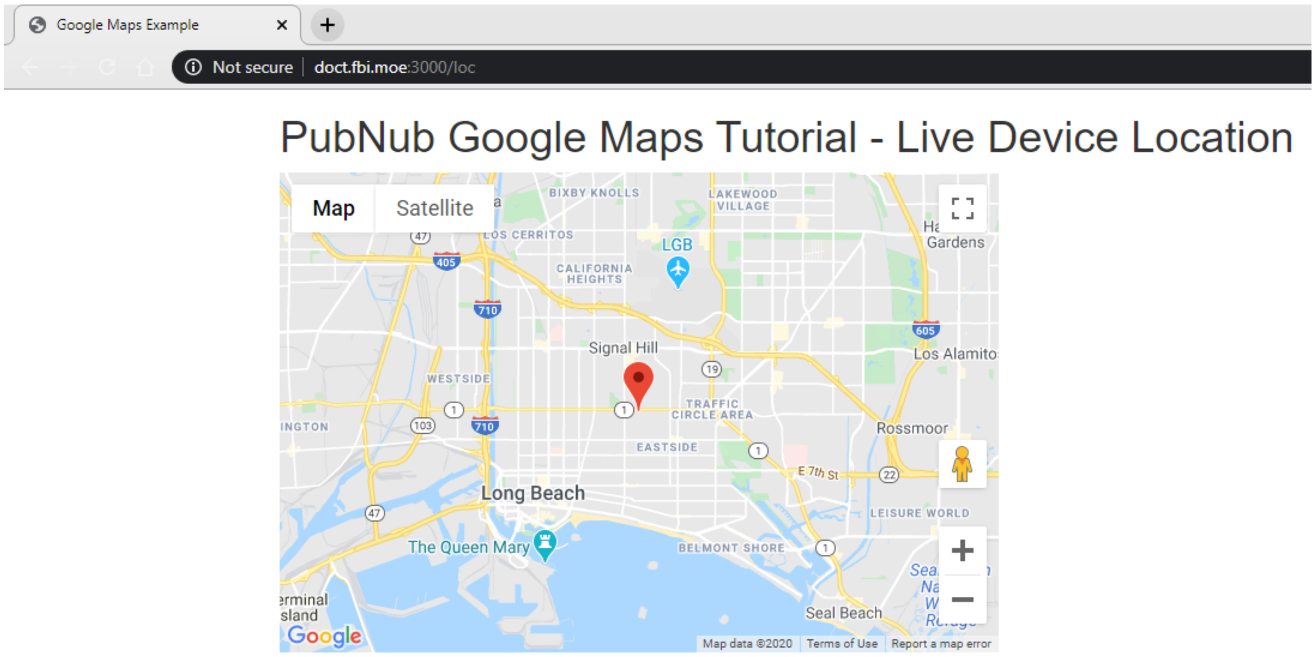
```
root@ubuntu-s-1vcpu-1gb-sfo2-01:~/myproject# nc -l -v -p 8081
Listening on [0.0.0.0] (family 0, port 8081)
Connection from 139.166.35.5.jtglobal.com 33468 received!
{'unitID':'24','batt':'60','lat':'33.790708167','long':'-118.159945667','spd':'2'}
{'unitID':'24','batt':'60','lat':'33.790300167','long':'-118.160943000','spd':'1'}
{'unitID':'24','batt':'60','lat':'33.789910000','long':'-118.161756500','spd':'1'}
{'unitID':'24','batt':'60','lat':'33.789642000','long':'-118.162321167','spd':'1'}
{'unitID':'24','batt':'60','lat':'33.789478500','long':'-118.162539167','spd':'1'}
{'unitID':'24','batt':'60','lat':'33.789412167','long':'-118.162594167','spd':'1'}
{'unitID':'24','batt':'60','lat':'33.789332667','long':'-118.162689000','spd':'1'}
{'unitID':'24','batt':'60','lat':'33.789247000','long':'-118.162811500','spd':'1'}
{'unitID':'24','batt':'60','lat':'33.789214667','long':'-118.162761500','spd':'1'}
{'unitID':'24','batt':'60','lat':'33.789146000','long':'-118.162913167','spd':'1'}
{'unitID':'24','batt':'60','lat':'33.789110833','long':'-118.163138500','spd':'1'}
{'unitID':'24','batt':'60','lat':'33.789071667','long':'-118.163295000','spd':'1'}
{'unitID':'24','batt':'60','lat':'33.789059000','long':'-118.163305500','spd':'0'}
{'unitID':'24','batt':'60','lat':'33.789046167','long':'-118.163292833','spd':'0'}
{'unitID':'24','batt':'60','lat':'33.789040333','long':'-118.163343167','spd':'0'}
{'unitID':'24','batt':'60','lat':'33.789028167','long':'-118.163388500','spd':'0'}
{'unitID':'24','batt':'60','lat':'33.789017833','long':'-118.163407667','spd':'0'}
{'unitID':'24','batt':'60','lat':'33.789004000','long':'-118.163429667','spd':'0'}
{'unitID':'24','batt':'60','lat':'33.788991000','long':'-118.163466000','spd':'0'}
{'unitID':'24','batt':'60','lat':'33.788990667','long':'-118.163488000','spd':'1'}
```

Figure 28 Client Terminal (MobaXTerm) showing server received JSON data from vehicle 2G SIM.



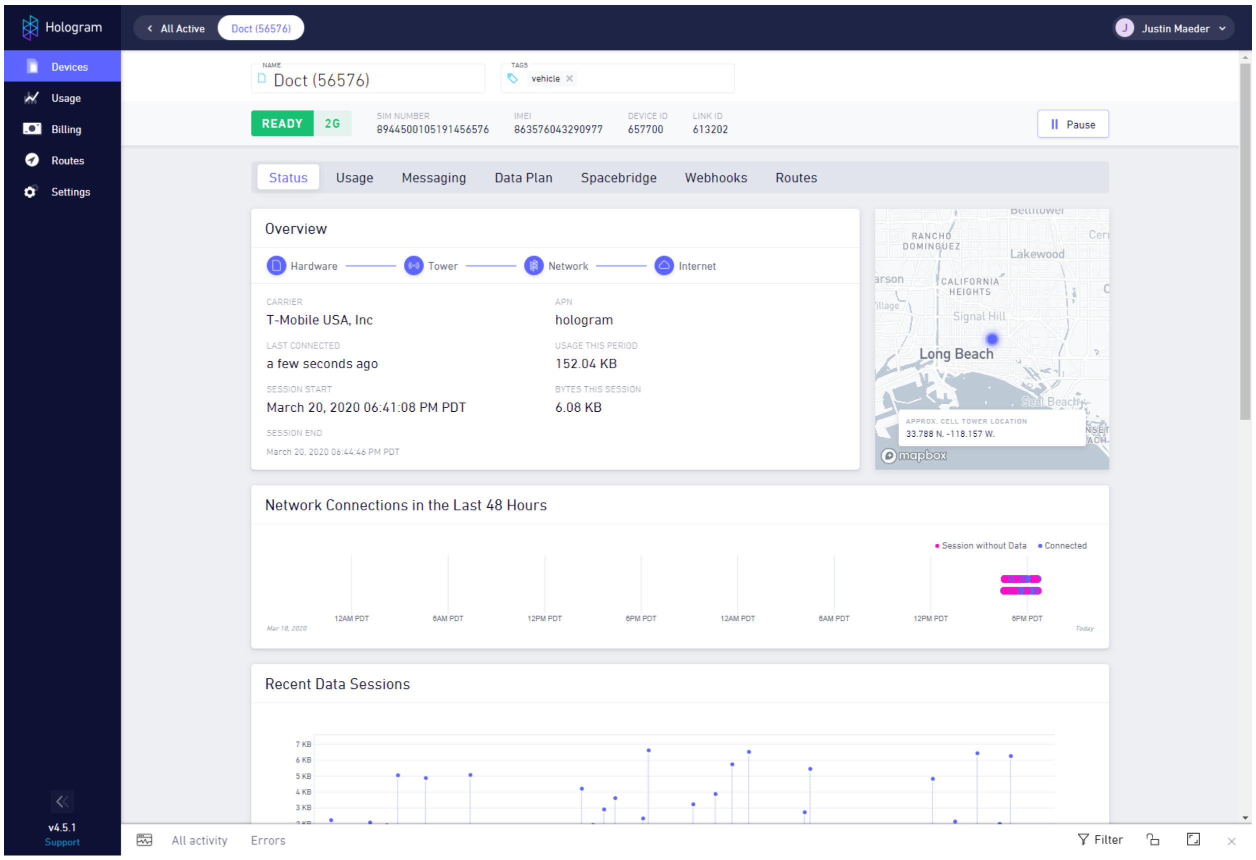
```
const server = new Net.Server();

server.listen(port, function() {
    console.log(`Server listening for connection requests on socket localhost:${port}`);
});

server.on('connection', function(socket) {
    console.log('A new connection has been established.');


    socket.on('data', function(chunk) {
        //if(!chunk.toString().length < 3) return
        console.log(`Data received from client: ${chunk.toString()}`);
    if(chunk.toString().includes('unitID')){
        recentJson = chunk.toString()
    }
    });

    socket.on('end', function() {
        console.log('Closing connection with the client');
    });

    socket.on('error', function(err) {
        console.log(`Error: ${err}`);
    });
});
```

Figure 29 Server.js to listen on selected port for incoming connections.



PubNub Google Maps Tutorial - Live Device Location

Figure 30 Google API and PubNub API logging latitude and longitude on website as data updates every minute.



NAME Doct (56576)
TAGS vehicle

READY 2G

SIM NUMBER 8944500105191456576
IMEI 863576043290977
DEVICE ID 657700
LINK ID 613202

Overview

CARRIER
T-Mobile USA, Inc

APN
hologram

LAST CONNECTED
a few seconds ago

USAGE THIS PERIOD
152.04 KB

SESSION START
March 20, 2020 06:41:08 PM PDT

BYTES THIS SESSION
6.08 KB

SESSION END
March 20, 2020 06:44:46 PM PDT

APPROX. CELL TOWER LOCATION
33.788 N. -118.157 W.

Network Connections in the Last 48 Hours

Recent Data Sessions

Figure 31 Hologram APN showing 2G data successfully sending from ESP32 to server. Each JSON string sent to client is about 500 bytes

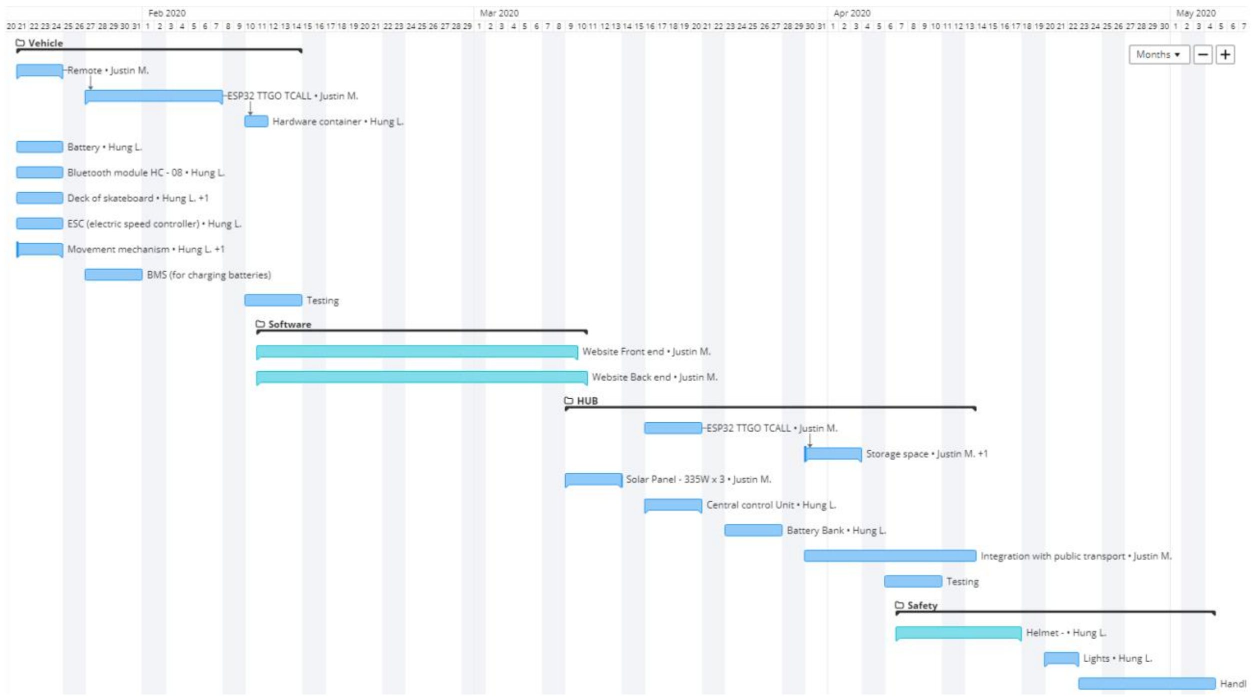
5. SOURCE CODE REPOSITORY

All source code in this project could be found at this repository.

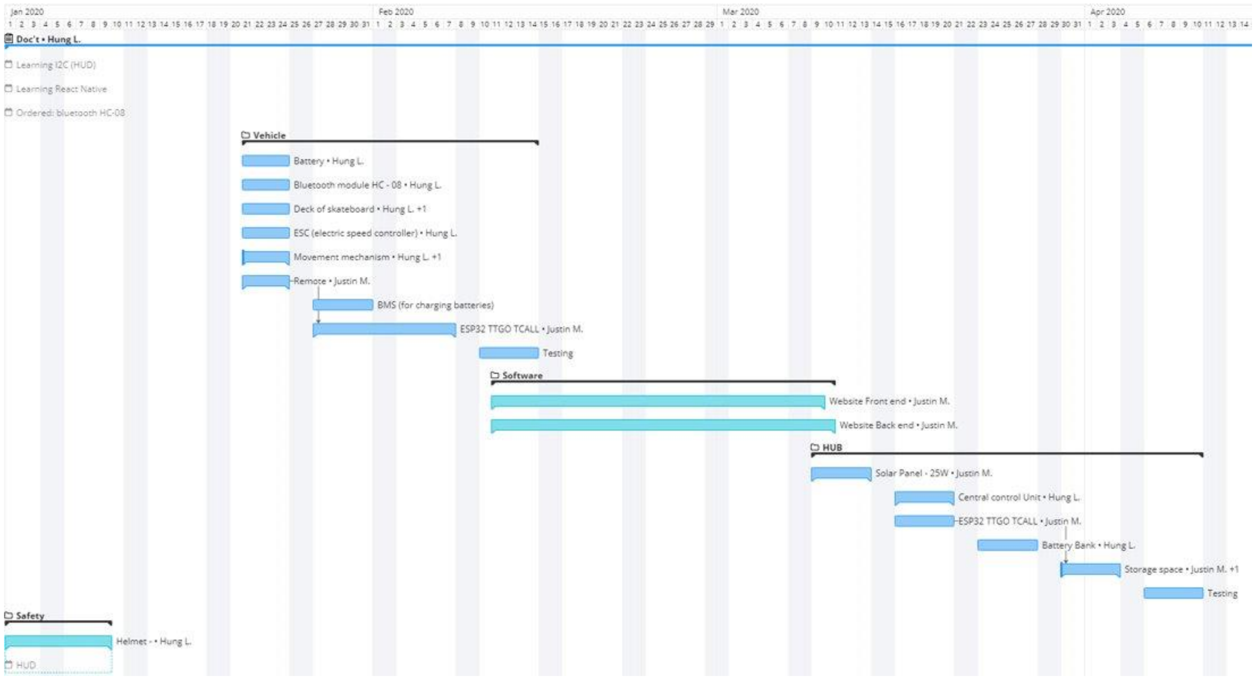
https://github.com/JustinMaeder/Doct-senior-design-project

6. TIMELINE

6.1 PROJECTED TIMLINE



6.2 ACTUAL TIMELINE



6.3 PROGRESS AGAINST TIMELINE

Since last semester our timeline has changed. The reason for this was due to a timing constraint where we only had two weeks before the first demonstration. For our first demo, we completed the Head Up Display as one of the main safety features. The first demo consisted of creating the HUD casing by modifying a 3D printed model. Inside the HUD casing was an Arduino nano, a HM-10 Bluetooth transceiver, connected to an OLED display. The Arduino nano is programmed in the IDE to accept a string to change the speed, battery, and range. This information is displayed on the HUD acrylic display which is attached to the helmet. The current time was also displayed in the first demo. The HUD was changed to the first task as it was a standalone system that has no dependencies on other components.

Starting from the third demo, we must work separately. This will slow down our progress but not by a significant amount since our project is divided in such a way that all parts are manageable by one person. The server components are managed by one person and can be easily collaborated online. But the next part of assembling the HUB, will have to be done differently. The major electronics of the HUB can be done by one person. But the physical locker space and charging assembly will need to be down sized so that it is more manageable by one person.

7. DATA SHEET

7.1 HEAD UP DISPLAY

Arduino Nano:

https://store.arduino.cc/usa/arduino-nano

HM-10:

https://people.ece.cornell.edu/land/courses/ece4760/PIC32/uart/HM10/DSD%20TECH%20HM-10%20datasheet.pdf

128x64 SSD1306 OLED display:

https://www.vishay.com/docs/37902/oled128o064dbpp3n00000.pdf

Adafruit MicroLipo battery charger:

https://learn.adafruit.com/adafruit-microlipo-and-minilipo-battery-chargers/downloads

Arduino Libraries:

https://www.arduino.cc/en/reference/libraries

7.2 VEHICLE

TinyGPS++ library:

http://arduiniana.org/libraries/tinygpsplus/

ESP32 Bluetooth connection:

https://randomnerdtutorials.com/esp32-sim800l-publish-data-to-cloud/

GPS:

https://www.instructables.com/id/ESP32-GPS-Tracker-With-an-OLED-Display/

7.3 SERVER

Hackathon Boiler Plate GitHub:

https://github.com/sahat/hackathon-starter

MongoDB

https://docs.mongodb.com/guides/

TinyGPS++ library:

http://arduiniana.org/libraries/tinygpsplus/

8. OUTSTANDING CONCERNS AND EXPECTED PATH TO RESOLUTION

Our first major concern is finishing the vehicle platform. This means creating the mechanics of the scooter/ skateboard hybrid. This goes out of our realm of Computer Engineering, so we are reaching out to a mechanical engineer for assistance. Our second concern is cost. We might scale down our production for example we might obtain a smaller solar panel and save about $80.

One setback that we have encountered is that one of our ESP32 TTGO T-Call is no longer functional due to misuse. A new ESP32 is on order, but the delivery date is unknown due to the new situations.

With the new situation that prevents us from working together in the same location, we must adapt our original plan. This means that the integration of the scooter will be put on hold because it will be difficult for us to meet with our contributor Taylor to research and develop the transformable handlebar. So, we have decided to suspend this idea.

9. DESCRIPTION OF TOOLS USED, METHODOLOGIES

For the programming of the Arduino Nano in the Head Up Display, we utilized Arduino IDE to display the updated graphics for the user to view the vitals of the board. We are also using Arduino IDE to program the ESP32 TTGO TCALL. For the 3D printed HUD casing we used Fusion 360 and modified a previous design to make it larger to fit our components.

In order to create our timeline of tasks, we used Wrike to make a task list which was easily converted into a Gannt chart. We communicated through Discord and kept track of our reference on there. Lastly, we shared files through Microsoft OneDrive because it was the easiest platform provided by the school to keep track of updated files.

10. CONTRIBUTORS

We can't do everything on our own, so we have recruited some contributors. First, Taylor Smyth at Avalon Sheet Metal helped us with the Database. Next, Long Le supported us with the 3D CAD throughout the project.

Taylor Smyth at Avalon Sheet Metal

Long Le at ANDesign Lab

11. REFERENCES

Electric skateboard:

- Motor drive system: https://www.youtube.com/watch?v=opg3h62p0gc
- Electric Skateboard: http://www.brokking.net/longboard_main.html
- Battery selection: http://mustcalculate.com/electronics/batterycapacity.php
- Motor selection: https://www.youtube.com/watch?v=rD7O4DV-D5k
    - https://www.quora.com/What-kind-of-motor-is-need-to-build-an-electric-skateboard
- ESC module: https://www.youtube.com/watch?v=LJqyWRSqOy4&t=4s
- HUD design: https://hackaday.io/project/12211-arduino-glasses-a-hmd-for-multimeter
    - https://www.youtube.com/watch?v=IpJqzwXWg-k
- Solar panel efficiency: https://magesolar.com/solar-angle-calculator/
- Solar panel: https://www.santansolar.com/sunpower-p17-335w/
- Bluetooth module setup: http://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/#Bluetooth_4_BLE
- ESP32 cellular: https://hackaday.com/2019/07/09/new-part-day-the-15-esp32-with-cellular/
- OLED display: https://www.vishay.com/docs/37902/oled128o064dbpp3n00000.pdf

Database:

- ESP32 Arduino: Timer Interrupts: https://techtutorialsx.com/2017/10/07/esp32-arduino-timer-interrupts/
- Installing ESP32 in Arduino IDE: https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/
- Connecting ESP32 to Wi-Fi Network: https://circuits4you.com/2019/01/05/connecting-esp32-to-wifi-network/

- ESP32 Publish Data to Cloud without Wi-Fi: https://www.youtube.com/watch?time_continue=183&v=aCeVsySh-v4&feature=emb_logo
- Server.js simple TCP server: https://riptutorial.com/node-js/example/22405/a-simple-tcp-server