Autonomous Lawnmower

Jonathan Poulose

USER INTERFACE SUBSYSTEM REPORT

REVISION – 3
29 April 2021

Table of Contents

List of Figures

1 Subsystem Introduction

The User Interface subsystem consists of the Java and XML code that was used to construct the android application used to control the autonomous lawnmower. The software used to write and test the code is Android Studio. The android application will write directly to the Firebase Server Database and upload information and commands in realtime. The main function of the application will be to set the specific coordinates of the corners of the user's lawn to draw the path of the lawnmower. To create this application API Keys and OAuth 2.0 Client IDs will be needed for Google Maps, Google Account login, and Firebase integration. Another part of my subsystem was the WiFi component of the ESP32.

2 Android Studio IDE

The entirety of the application will be coded and tested on the integrated development environment, Android Studio version 4.0.1. This software also allows me to test the application on any virtual android device. I have been emulating the apk file on the Pixel 2 and 3a as well as the Nexus 5 & 6 with android 9.0 and 10.0 to test. Android Studio made the integration with google maps very easy especially because there were many built in functions made specifically for google maps.

2.0 Features Added This Semester

- Stop Function
- Scheduling ability
  - Days of the week/time
- Default lawn
- Battery status

2.1 Main Activity Screen

Allows user to login with any google account
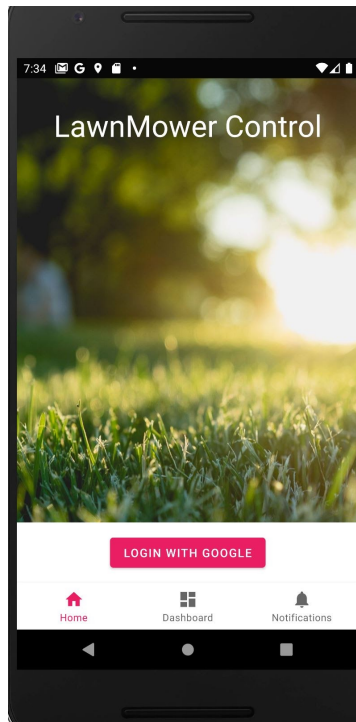
Figure 1: Home Screen Prior to Login
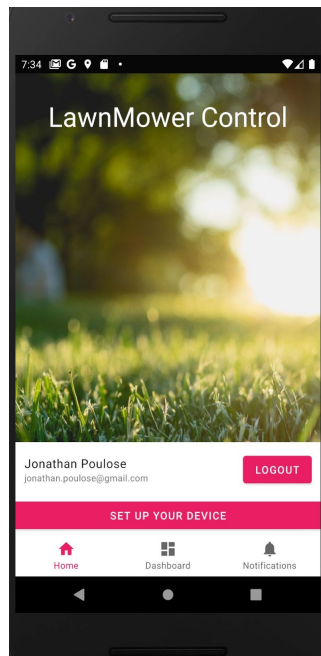
Figure 2: Home Screen When Logged in

2.2 Setup Activity

Allows user to add multiple lawns and shows lawnmower information

Figure 3: Add Multiple Lawns

2.2.1 Default Lawn

Allows users to select default lawn (lawn that will be cutting in real time). This part of the application was crucial because this allowed the ESP32 to get only the coordinates from the selected lawn. The ESP32 code could only get the coordinates from a certain path in firebase so I had to create a "default lawn" section that would change once it was selected in the application.
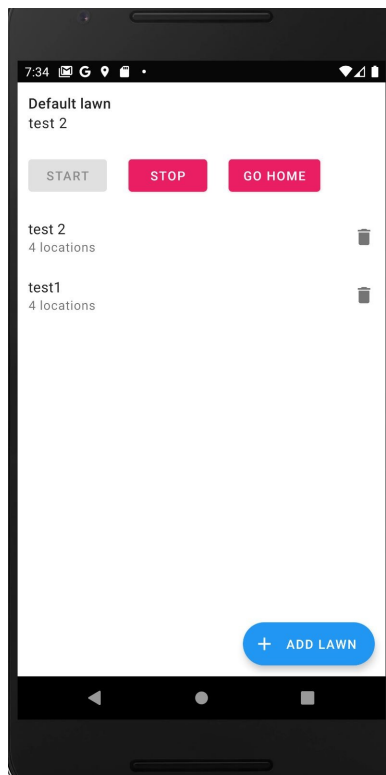
Figure 4: Default Lawn

2.2.2 Start/Stop Changing

Start and Stop gets Greyed out when one of the buttons is clicked
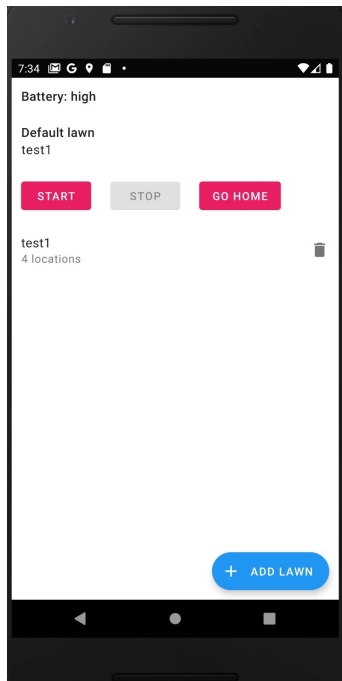
Figure 5: Start/Stop

2.2.3 Battery Level

Battery Level is shown as high or low
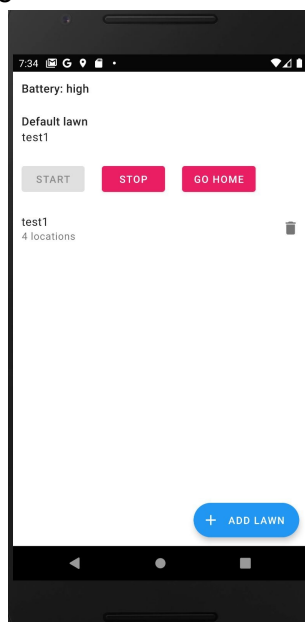
Figure 6: Battery level

2.3 LawnActivity

Allows users to use current location, set the path of the lawn, and name the lawn. Each marker placed starts from 1 to desired amount. Can add unlimited coordinates
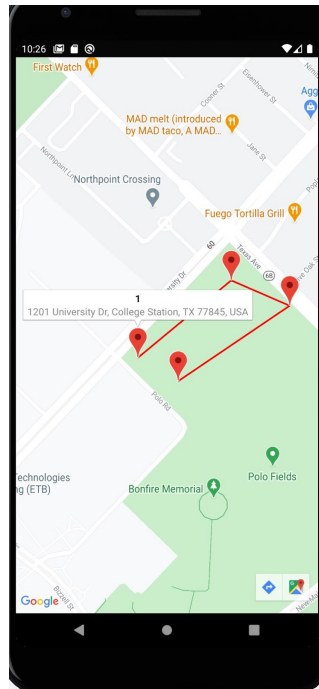
Figure 7: 4 coordinates

2.3.1 Lawn Name

Allows user to add lawn name and unlimited number of coordinates
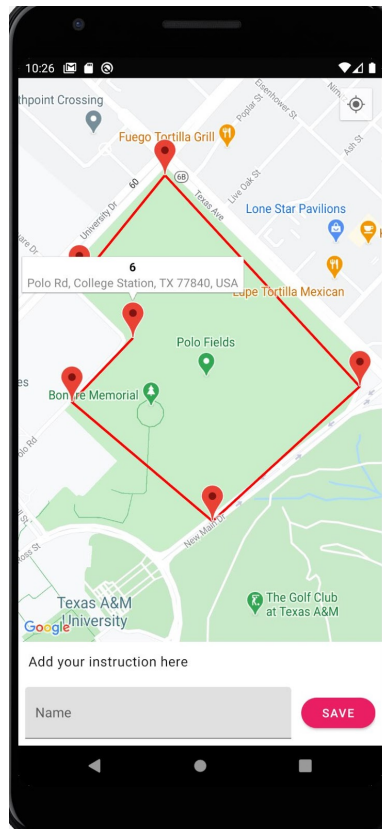
Figure 8: 6 coordinates

2.3.2 Scheduling

Allows user to schedule the lawn mower to go at a certain time and day of the week
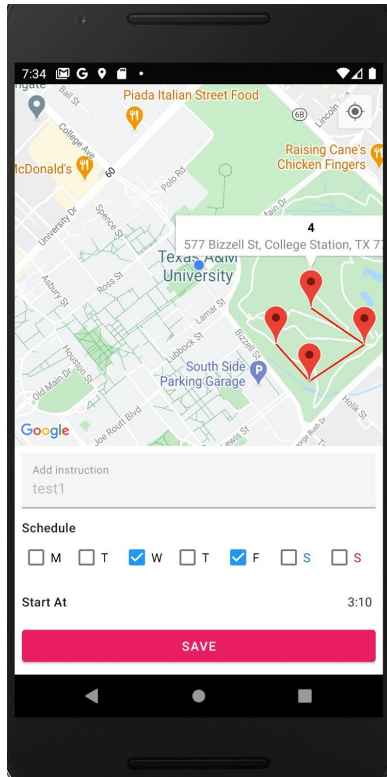
Figure 9: Days of the week
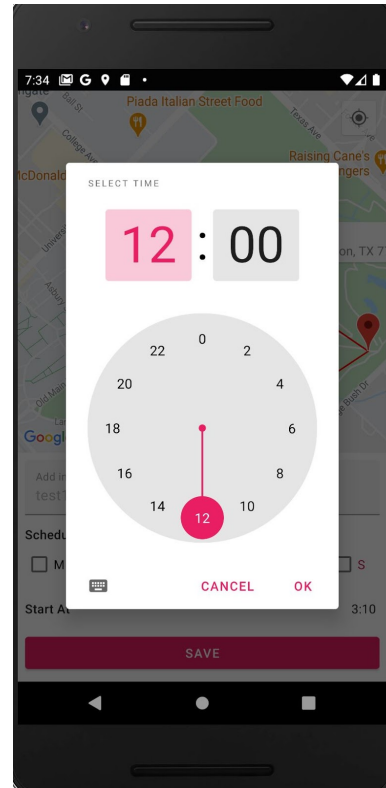
Figure 10: Time

3 Firebase

Using the Realtime Database in Google Firebase platform, the latitude and longitude coordinates are constantly updated as soon as a user signs in and saves a "lawn". The child hierarchy goes in order from Application Name>user ID>lawn name>marker number>latitude and longitude. When connecting the server to the WiFi module, the coordinates have to be in the same format as whatever the MCU can read it in. Once the application is complete all information such as usage statistics and navigational data will be transferred through the wifi module attached to the MCU and the Google Firebase Database.
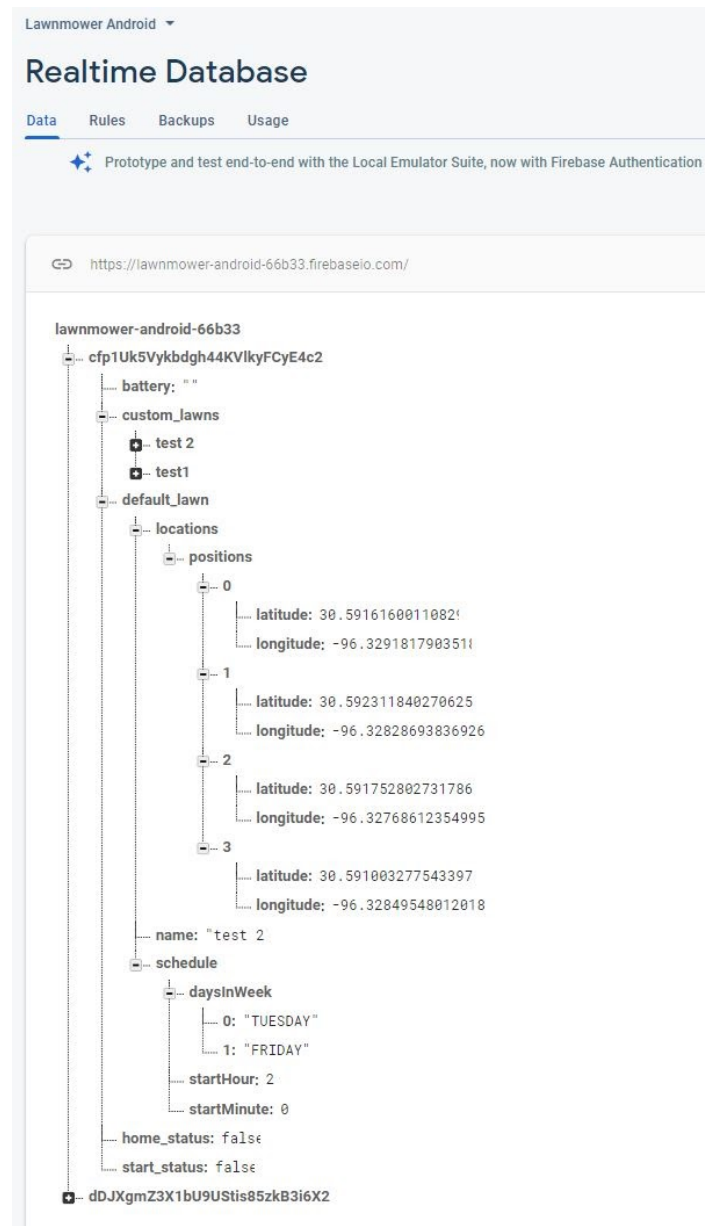
Figure 11: Full Firebase Server for 1 user

3.1 Google Cloud Platform

The Application credentials types that will be used are API keys & OAuth 2.0 client credentials. The only API key used in the code is the Google Map SDK API key to allow users to use google maps. All the other credentials were autocreated. Google Firebase auto created an android and browser key to allow the information to get updated in the realtime database. Google Services auto created an android client and web client to allow users to sign in to the app using a google account.

API Keys

|	Name	Creation date ↓	Restrictions	Key			
☐	⚠ Google Map SDK	Nov 21, 2020	None	AIzaSyBsW1...cNXQLJsSV8	⎙	✎	🗑
☐	⚠ Android key (auto created by Firebase)	Nov 21, 2020	None	AIzaSyAHc5...HtLqJpq91c	⎙	✎	🗑
☐	⚠ Browser key (auto created by Firebase)	Nov 21, 2020	None	AIzaSyAnLF...QMmaLOfTTE	⎙	✎	🗑

OAuth 2.0 Client IDs
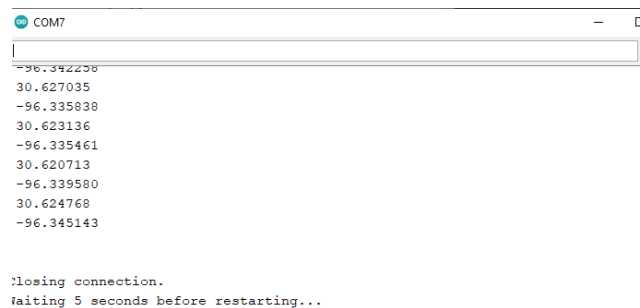
	Name	Creation date ↓	Type	Client ID				
☐	Web client 2	Nov 21, 2020	Web application	287950388755-ap9s...	⎙	✎	🗑	⬇
☐	Android client for lawnmower.app (auto created by Google Service)	Nov 21, 2020	Android	287950388755-ts34...	⎙	✎	🗑	⬇
☐	Web client (auto created by Google Service)	Nov 21, 2020	Web application	287950388755-p4lk...	⎙	✎	🗑	⬇

Figure 12: Screenshot of API credentials created

4 ESP32

My first versions of the code were for two seperate ESP32s, WiFi and Navigation. In the beginning of the semester, we decided that I would work on the WiFi ESP32 and Josh would work on the Navigation ESP32. I ended up writing code for both so that they could communicate with each other and send variables back and forth.

I tested both of the old files/ESPs without the navigation code to see if they would execute all functions and communicate effectively and there seemed to be no issue. You can see that coordinates successfully are being sent from one ESP to the other in the figure below

Figure 13: Successful communication between the two ESP32s (Serial Monitor from Nav ESP)

4.1 WiFi ESP32

For the WiFi ESP, the functions were:

-Connect to Wifi for Firebase Server & Connects to Nav ESP

-Receive coordinates & start_status from Firebase Server

-Recieve battery information/statistics from Nav ESP

-Send battery information/statistics to firebase server

-Send coordinates & start_status to NAV ESP

In the figure below you can see the variables being sent and received as well as connecting to my routers network as well as the the other ESPs network

```
WiFi_MCU

1  //#ifdef ESP32
2  #include <WiFi.h>
3  #include <HTTPClient.h>
4  #include <FirebaseESP32.h>
5  //#include <FirebaseJson.h>
6
7  //Txt files
8  #include <SPIFFS.h>
9  #include "FS.h"
10
11 #include <Arduino.h>
12
13 // Set web server port number to 80 - SoftAP
14 WiFiServer server(80);
15
16 // ESP32 wifi network connection data
17 #define ssidEspServer "ESP32Server"
18 #define PassEspServer "87654321"
19
20 // Connection data to the user's local wifi network
21 #define ssidWiFiLocalSet "Poulose"
22 #define passwordSenhaWiFiLocal "2147707702"
23
24 //Data for connection to the "Firebase" database
25 #define FIREBASE_HOST "https://lawnmower-android-66b33.firebaseio.com"
26 #define FIREBASE_AUTH "4YOEWZHxqEOpZokJi5bhdcMpuIy15Qwgh45MKTO2"
27
28
29 //Define FirebaseESP8266 data object for data sending and receiving
30 FirebaseData fbdo;
31
32 //Variables
33 String receiveCar; // Car Received Data
34 String sendReceive; // Selection of active Tasks
35 String firebaseReceive; // Data Received from Firebase
36 String strJsonFirebase; // Data Json Received from Firebase
37 String strJsonClient = "{\"battery\":\" \",\"start_status\":\" \",\"latitude\":\" \",\"longitude\":\" \",}"; // Data Json Received from client
38
```

Figure 14: Snippet of WiFi MCU Code

4.2 Navigation ESP32

For the Nav_MCU, the functions were:
   -Create WiFi Network so WiFi MCU can connect to it
   -Send battery information/statistics to WiFi ESP
   -Recieve coordinates & start_status from WiFi ESP

Below you can see the variables being sent and received to the Navigation ESP32

```
NavMCUwithoutNav

1  #include <WiFi.h>
2  #include <WiFiMulti.h>
3
4  //Txt files
5  #include <SPIFFS.h>
6  #include "FS.h"
7
8  WiFiMulti WiFiMulti;
9  String receiveData;
10 String battery ; // Percentage value of battery
11 String start_status; // Car status
12 String latitude; // Current car latitude
13 String longitude; // Current car longitude
14 String _0_latitude; // Orientation coordinates
15 String _0_longitude; // Orientation coordinates
16 String _1_latitude; // Orientation coordinates
17 String _1_longitude; // Orientation coordinates
18 String _2_latitude; // Orientation coordinates
19 String _2_longitude; // Orientation coordinates
20 String _3_latitude; // Orientation coordinates
21 String _3_longitude; // Orientation coordinates
22
23 SemaphoreHandle_t myMutex; // Semaphore to allow access to variable by different tasks at different times
24
25 // Method that allows to use "Serial.print ()" in different tasks
26 String str_global = "";
27 void printGlobal(String str) {
28   xSemaphoreTake(myMutex, portMAX_DELAY);
29   str_global = str;
30   Serial.println(str_global);
31   xSemaphoreGive(myMutex);
32 }
33
34 // ---------------------------------------- //----------------------------------------//
35
36 // Method of creating and reading files;
37 // Start: Create filess
38 void createFiles(String _file, String content) {
39   //
40   bool createFile = SPIFFS.exists("/" + _file + ".txt");
41   if (createFile) {
42     File file = SPIFFS.open("/" + _file + ".txt", "r");
43     if (!file) {
44       exit(0);
45     }
46     int s = file.size();
```

Figure 15: Snippet of Nav MCU Code

4.3 Problems

Once I added Josh's old navigation code, the variables would send to each other but the navigation was not working. There were many errors and bugs that led to a rebooting issue with the Navigation ESP.

```
receiveData :[{"start_status":"    ","_0_latitude":" 30.621404 ","_0_longitude":" -96.334650 ","_1_latitude":" 30.622638 ","_1_longitude":" -96.335838 ","
receiveData :[{"start_status":"    ","_0_latitude":" 30.621404 ","_0_longitude":" -96.334650 ","_1_latitude":" 30.622638 ","_1_longitude":" -96.335838 ","

Waiting for WiFi...
WiFi connected
IP address:
192.168.4.2
30.6214
 30.621404
 -96.334650
/home/runner/work/esp32-arduino-lib-builder/esp32-arduino-lib-builder/esp-idf/components/freertos/queue.c:1442 (xQueueGenericReceive)- assert failed!
abort() was called at PC 0x40088869 on core 0

Backtrace: 0x4008c434:0x3ffcfa30 0x4008c665:0x3ffcfa50 0x40088869:0x3ffcfa70 0x400d6023:0x3ffcfab0 0x400d1a32:0x3ffcfad0 0x40088b7d:0x3ffcfb60

Rebooting...
ets Jun  8 2016 00:22:57

rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:8896
load:0x40080400,len:5816
entry 0x400806ac
receiveData :[{"start_status":"    ","_0_latitude":" 30.621404 ","_0_longitude":" -96.334650 ","_1_latitude":" 30.622638 ","_1_longitude":" -96.335838 ","
receiveData :[{"start_status":"    ","_0_latitude":" 30.621404 ","_0_longitude":" -96.334650 ","_1_latitude":" 30.622638 ","_1_longitude":" -96.335838 ","

Waiting for WiFi...
WiFi connected
IP address:
192.168.4.2
30.6214
 30.621404
 -96.334650
/home/runner/work/esp32-arduino-lib-builder/esp32-arduino-lib-builder/esp-idf/components/freertos/queue.c:1442 (xQueueGenericReceive)- assert failed!
abort() was called at PC 0x40088869 on core 0

Backtrace: 0x4008c434:0x3ffcf160 0x4008c665:0x3ffcf180 0x40088869:0x3ffcf1a0 0x400d6023:0x3ffcf1e0 0x400d1a32:0x3ffcf200 0x40088b7d:0x3ffcf290

Rebooting...
ets Jun  8 2016 00:22:57
```

Figure 16: Rebooting Issue with Josh's code

4.4 Fixes/Improvements

   To fix this problem, Max ended up rewriting the Navigation Code and I decided to combine all the code into one ESP to have one less step of the line of communication with the variables. This fixed the problem.

   The functions for the final code:
   -Connects to Wifi for Firebase Server
   -Receives coordinates,start_status from Firebase Server in realtime
   -Sends battery information/statistics to Firebase Server in realtime
   -Navigates the Mower (explained in detail above in navigation section)

   Added to new combined code
   -Converts 4 coordinates into Distances which you can see in the figure below

```
Set string data success
Local-Control
start_status:
0
_0_latitude:
30.591616
_0_longitude:
-96.329182
Local-Control
_1_latitude:
30.592312
_1_longitude:
-96.328287
_2_latitude:
30.591753
_2_longitude:
-96.327686
Local-Control
_3_latitude:
30.591003
_3_longitude:
-96.328495
{"battery":" ","start_status":"0","_0_latitude":"30.591616","_0_longitude":"-96.329182","_1_latitude":"30.592312","_1_longitude":"-96.328287","_2_l
Local-Control

Distance to destination(M): 0.000115


Distance to destination(M): 0.000085


Distance to destination(M): 0.000114


Distance to destination(M): 0.000095
```

Figure 17: Shown Distances Calculated from the 4 coordinates (Serial Monitor)

4.5 Summary for ESP32 Code

The ESP32 was set up in STA Mode to connect to the WiFi network. In the final code/demo, we ended up using my iphone as a hotspot to connect to. In the code you can see the network
name as "iPhone" and the password as "jonathan". By connecting to WiFi, I were able to to receive the start_status (Start & Stop) boolean and the longitude and latitude
from the coordinates selected in the App in realtime as well as send battery status/statistics to the server in realtime. The ESP checked the the server for these variables every
110 ms. The hardest part was finding the correct format from the Firebase Server & JSN file to display all of these variables correctly which is a lot of my code.

I added a while loop to the navigation code so that when start_status is 0, the mower will stop and when it is not, it will start.

I also added code to convert the coordinates (8 total variables) longitude and latitude into distances for the outside legs of the navigation using the haversine formula.

The WiFi part of the code worked fine without any issues.

5 Conclusion for User Interface/Server/WiFi

I am very satisfied with all the work I have put into this project. I think I did all I could and to the best of my ability. Although the lawnmower did not have hardware to check for battery level, I still added code to both ESP32s and UI that would allow this information to be passed through to the server and to the mobile application. I successfully created the user interface which had all of the following functions:

- Allows user to login through Google Account
  - as many google accounts as you want
  - saves information for each account
- Allows user to add multiple lawns using Google Maps API
  - Select Default Lawn
  - Select current location
  - Unlimited markers/corners for each lawn
  - records coordinates for each marker and sends to server
  - Numbered each corner from one to desired amount
  - Add lawn name
- Scheduling
  - Add time
  - Days of the week
- Control of Lawnmower
  - Start
  - Stop
  - Go Home
- Lawnmower status
  - Battery level
  - Other statistics
- Uploads all information to Firebase Server

6 Moving on

I was honestly scared to code my first mobile application because I had no idea how. After doing this project, I know that I will be coding more applications in the future which I am very excited about. I am glad I went through this so it will be easier for me in the future. Thanks and Gig'em.