

GPS TRACKER



Gps Tracking System

Supervised by: Dr.Ashraf Salem
Eng.Tasneem Adel

Team Members

Name	ID
Mohamed Tarek Mohamed El-Sayed	1701220
Mohamed Abdelazeem Ahmedy	1701235
Mohamed Khaled Mohamed Radi	17T0119
Mostafa Samy Fayz	15X0076
Salma Ali Abdelhaleem	1600654
Nourhan Mostafa Habib	1701614

GitHub Link:

<https://github.com/Muhammed-El-Sayed/GPS-Tracker-System.git>

Table of Contents:

- 1. Functional and Non-Functional Requirements.**
- 2. Our Project Features.**
- 3. UML Diagrams**
 - 3.1. State Diagram**
 - 3.2. Sequence Diagram**
 - 3.3. Component Diagram**
 - 3.4. Layered Architecture**
- 4. Drivers**
 - 4.1. Configurable Port Driver**
 - 4.2. Configurable DIO Driver**
 - 4.3. Configurable UART Driver**
 - 4.3.1. UART using polling**
 - 4.3.2. UART using Interrupt**
 - 4.4. SysTick Timer Driver (Polling & Interrupt).**
 - 4.5. EEPROM Driver**
 - 4.6. LED Driver**
 - 4.7. LCD Driver**
 - 4.8. GPS Driver**
 - 4.9. esp8266 Wi-Fi module Programming**
- 5. Calibration**
- 6. Output Screen-Shots**
- 7. Attached Videos**
- 8. Final PCB Product**

1.Functional and Non-Functional Requirements.

Functional Requirements:

- 1- Display Trajectory on IOS App.**
- 2- Display Trajectory on Google maps using python.**
- 3- Measure covered distance.**
- 4- Store GPS readings in Google Sheet**

Non-Functional Requirements:

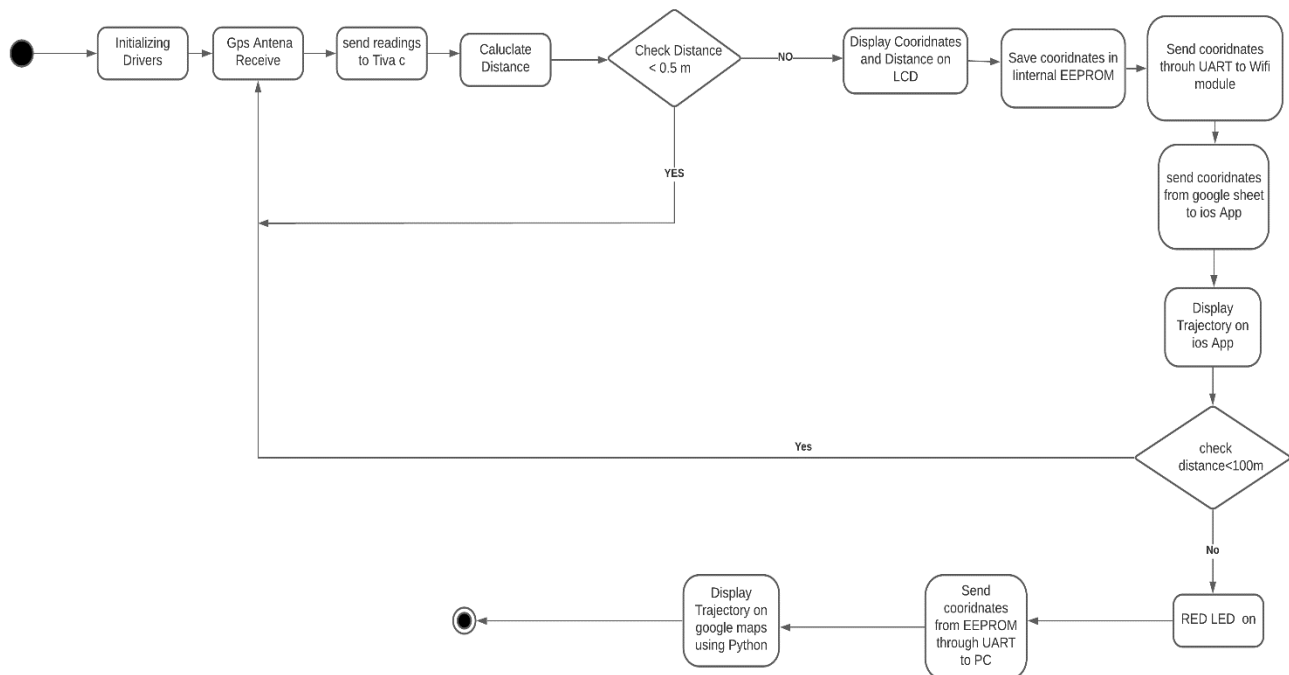
- 1- Programming language embedded C.**
- 2- Deadline 13th June at 11:59 pm.**
- 3- Software should be calibrated, to decrease error in GPS readings.**

2.Our Project Features.

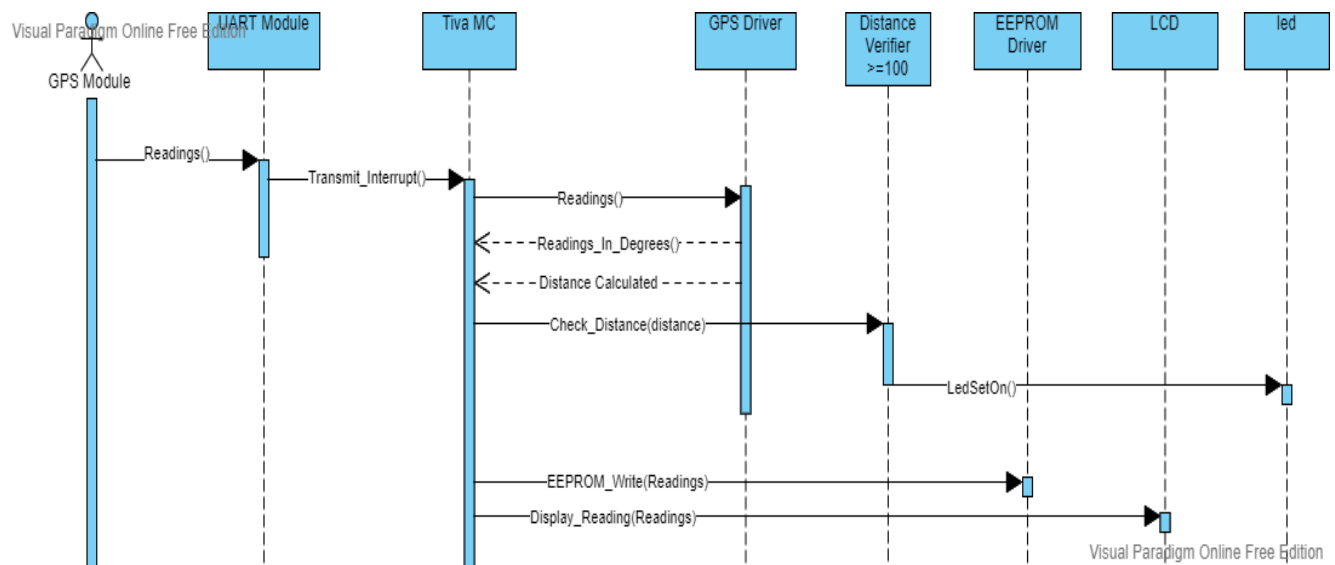
- 1. Esp8266 Wi-Fi Module Programming.**
- 2. Sending GPS coordinates to a server using Wifi module.**
- 3. Display Trajectory on IOS App Using data sent to the server by Wifi.**
- 4. Saving GPS Coordinates in EEPROM**
- 5. Display Trajectory on Google Maps Using Python Script.**
- 6. Layered Architecture is applied (Application layer, HAL, MCAL).**
- 7. EEPROM Driver (eeprom.c & eeprom.h)**
- 8. Configurable Port Driver (port.c & port.h).**
- 9. Configurable DIO Driver (dio.c & dio.h).**
- 10. Configurable UART Driver (uart.c & uart.h).**
- 11. LCD Driver (lcd.c & lcd.h).**
- 12. GPS Driver (gps.c & gps.h)**
- 13. LED Driver (led.c & led.h)**
- 14. SysTick Driver (systick.c & systick.h)**
- 15. LCD displays live Total distance, Latitude, Latitude direction, Longitude & Longitude direction.**
- 16. Any interrupt is implemented using call back pointers to abstract Application Layer**

3. UML Diagrams:

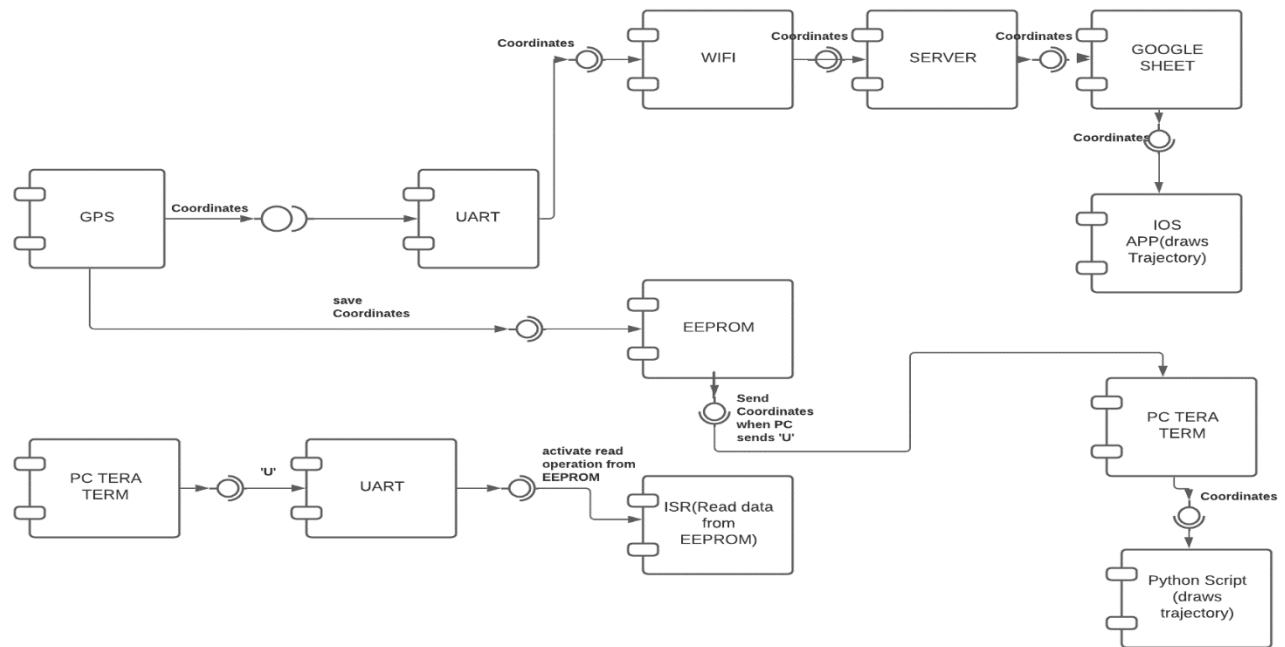
3.1. State Diagram



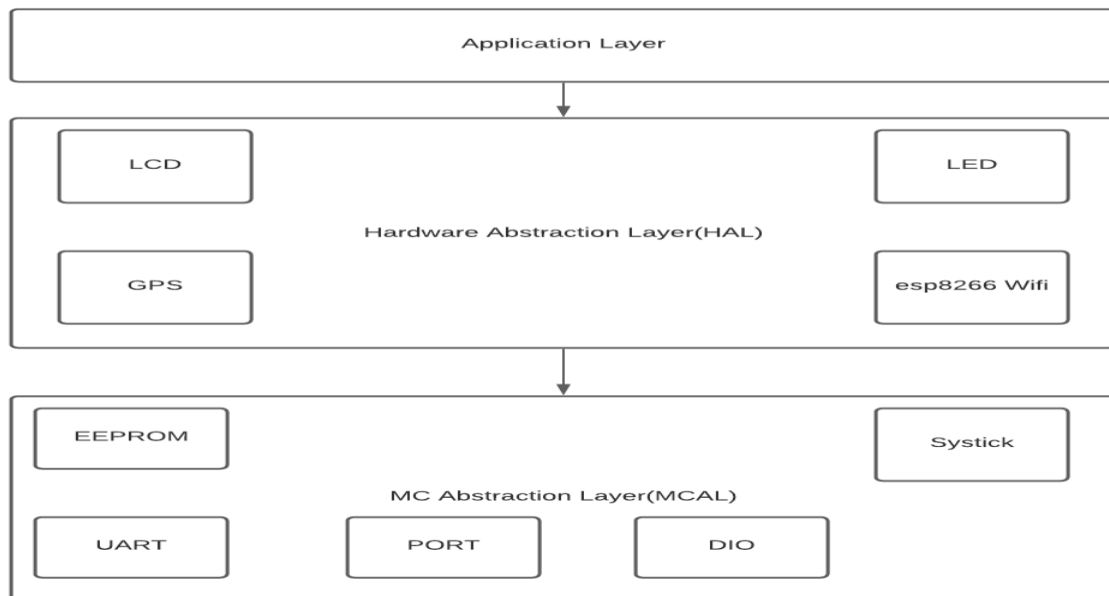
3.2. Sequence Diagram



3.3. Component Diagram



3.4. Layered Architecture



4. Drivers

4.1. Configurable Port Driver

- Initializes all Microcontroller ports and pins
 - Direction (In/Out)
 - Initial Values for output pins (High/Low)
 - Setup the internal pull up/down resistors for input pins
 - Mode

4.2. Configurable DIO Driver

- Read digital Port/Pin
- Write digital Port/Pin

4.3. Configurable UART Driver

//User Configuration Example from Code

```
#define UART_0_RX      0xE0A0 //E:Recieve ,0:UART0, A0: Pin A0
#define UART_0_TX      0xC0A1 //C:Send ,0:UART0, A1: Pin A1
```

```
....
```

```
....
```

```
#define DATA_LENGTH_5_BITS  (0U)
```

```
#define DATA_LENGTH_8_BITS  (3U)
```

```
....
```

```
....
```

```
#define ONE_STOP_BIT      (0U)
```

```
#define TWO_STOP_BITS     (1U)
```

//Using Configurable Uart in main

//Here User chooses UART 4 Pin C5 to be initialized as Tx.

//The User also chooses the communication protocol (8 data bits & 1 stop bit)

```
UART_Config Configuration3 ={ UART_4_TX,DATA_LENGTH_8_BITS,ONE_STOP_BIT};
UART_Init(&Configuration3);
```

//Also The User can choose the Baud rate, UART clock & the clock divisor.

```
#define UART_CLOCK (16000000.0)
```

```
#define UART_CLOCK_DIVISOR (16.0)
```

```
#define UART_BAUD_RATE (9600.0)
```

```
#define BDR(UART_CLOCK,UART_CLOCK_DIVISOR,UART_BAUD_RATE)
((UART_CLOCK)/((UART_CLOCK_DIVISOR)*(UART_BAUD_RATE)))
```

```

#define UARTIBRD_VALUE
((uint16)BDR(UART_CLOCK,UART_CLOCK_DIVISOR,UART_BAUD_RATE))
#define UARTFBRD_VALUE
((uint16)(((BDR(UART_CLOCK,UART_CLOCK_DIVISOR,UART_BAUD_RATE))-(UARTIBRD_VALUE)
)*(64))+(0.5)))

```

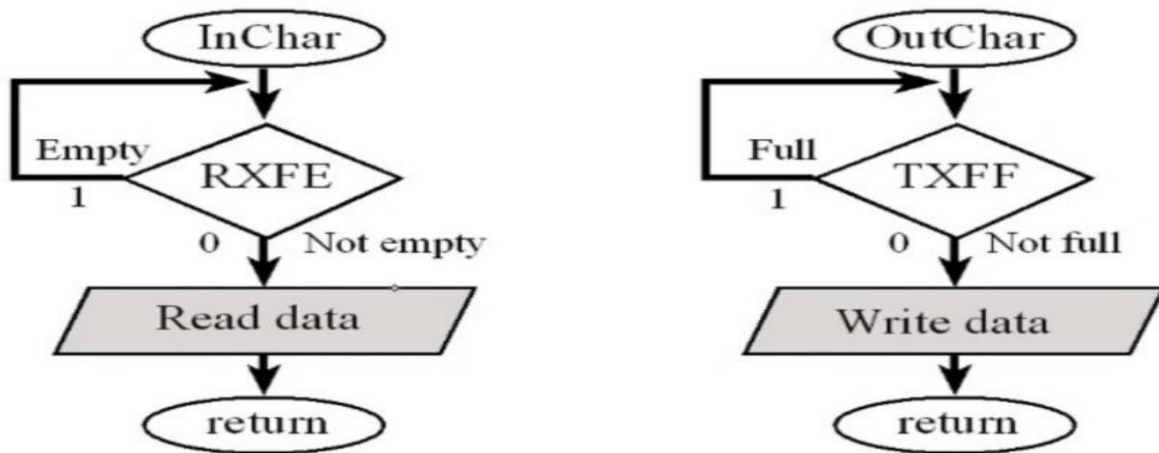
Functions implemented:

```

void UART_Init( const UART_Config * ConfigPtr);
void UART_Send_Byte(uint8 data , uint16 UART_Mode);
uint8 UART_Recieve_Byte(uint16 UART_Mode);
void UART_sendString(const uint8 *Str, uint16 UART_Mode);
void UART_receiveString(uint8 *Str , uint16 UART_Mode);
void UART_2_RX_setCallBack(void(*a_ptr)(void));
void Enable_UART_2_RX_INTERRUPT(void);
void UART_0_RX_setCallBack(void(*a_ptr)(void));
void Enable_UART_0_RX_INTERRUPT(void);
void UART_3_RX_setCallBack(void(*a_ptr)(void));
void Enable_UART_3_RX_INTERRUPT(void);

```

4.3.1. UART using polling



4.3.2. UART using Interrupt

We also supported UART(3 & 2) to receive using Interrupt

- **UART 2 to receive from GPS**
- **UART 3 to receive 'U' from PC on sending data from EEPROM to PC**

4.4. SysTick Timer Driver (Polling & Interrupt)

.c and .h are implemented for systick timer.

It supports Interrupt & Polling SysTick Timer.

Functions of systick timer:

- 1- **Systick_start()**
- 2- **Systick_stop()**
- 3- **Systick_setcallBack()**
- 4- **Systick_Delay_ms().**

4.5. EEPROM Driver

.c and .h are implemented for Internal EEPROM of TIVA-C.

You can read and write word.

Functions of EEPROM:

- 1- **EEPROM_Init()**
- 2- **EEPROM read()**
- 3- **EEPROM_getsize()**
- 4- **EEPROM write()**
- 5- **EEPROM_writeBytes()**

4.6. LED Driver

Port and DIO drivers are used in LED driver implementation to follow layered abstraction concept (to abstract external hardware from the implementation of the microcontroller)

Functions covered in LED:

LED_setOn()

LED_setOff()

4.7. LCD Driver

Functions covered in LCD:

- LCD_sendCommand()
- LCD_displayString()
- LCD_goToRowColoumn()
- LCD_displayStringRowColoumn()
- LCD_clearScreen()
- LCD_integerToString()
- LCD_doubleToString()
- LCD_displayCharacter()
- LCD_init()

Port and DIO drivers are used in LCD driver implementation to follow layered abstraction concept (to abstract external hardware from the implementation of the microcontroller)

4.8. GPS Driver

.c and .h files are implemented for GPS.

Its functions are used for parsing data comes from GPS antenna.

Functions of GPS:

- 1- Return_Latitude_or_Longitude_In_Degrees()**
- 2- Return_Latitude_Direction().**
- 3- Return_Longitude_Direction().**
- 4- toRadians()**
- 5- Calculate_Distance_between_2_Cooridnates().**
- 6- Updata_Latitude_In_String().**
- 7- Update_Logitude_In_String().**

4.9. esp8266 Wi-Fi module Programming

IOT is applied in our project using Wi-Fi module (esp8266 01).where Wi-Fi is the client and we use Google sheets as a server, then data is sent to IOS APP build using Swift to draw Trajectory on Google maps. The scenario that happens:

- 1. Wi-Fi receives data serially using UART from Tiva-C.**
- 2. Wi-Fi sends request through URL to send data to Google sheets.**
- 3. IOS APP receives these coordinates in Jason format.**
- 4. IOS App draws Trajectory on Google maps.**

5. Calibration

We send coordinates to EEPROM & calculate total distance each 0.5m.

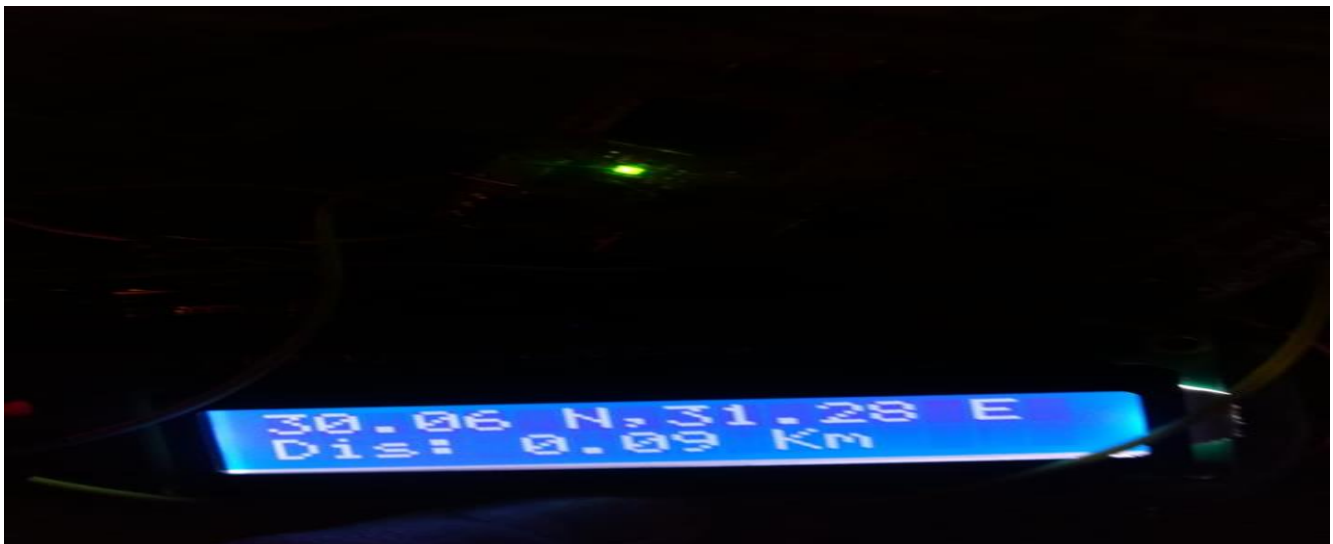
We send coordinates to WIFI module each 5m.

6. Output Screen-Shots

LCD displays live:

-Latitude LatitudeDirection,Longitude LongitudeDirection

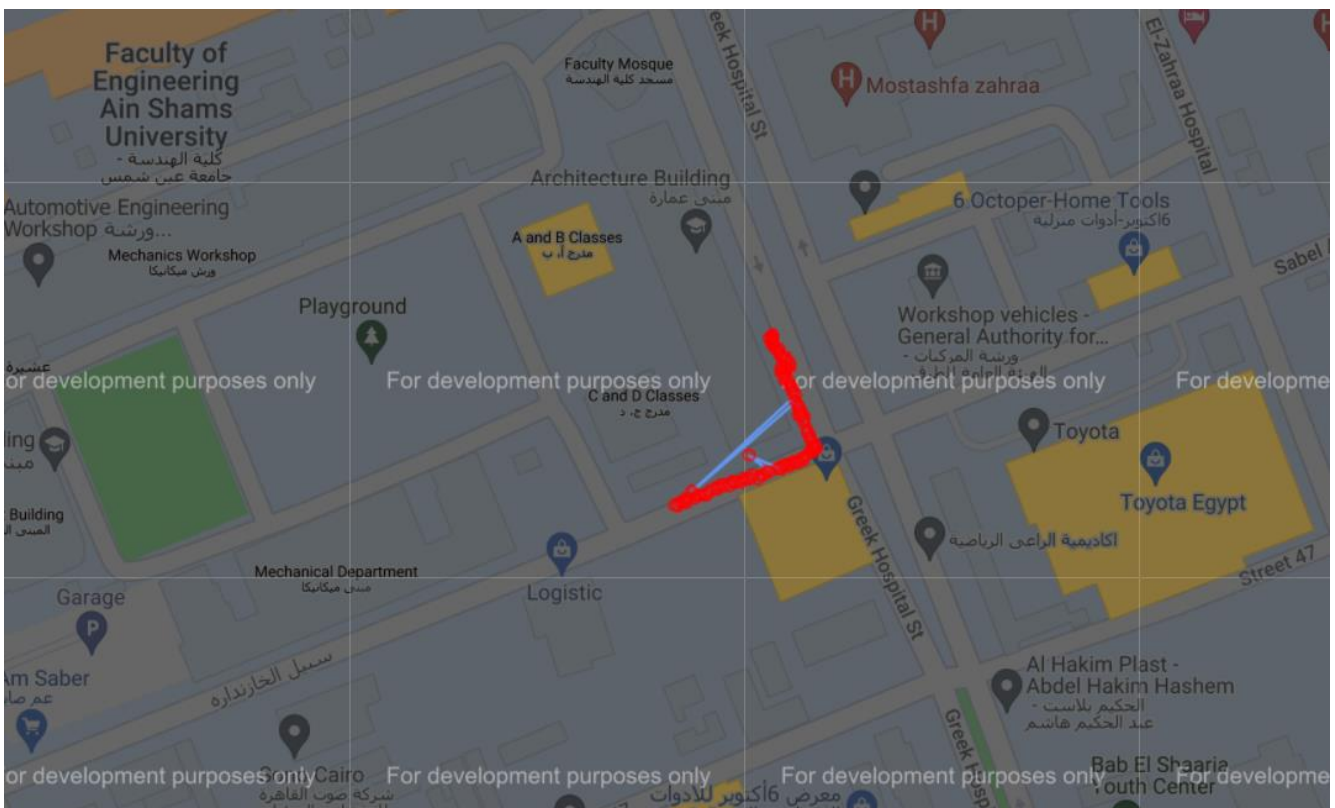
-Distance in Km



Points Stored in EEPROM

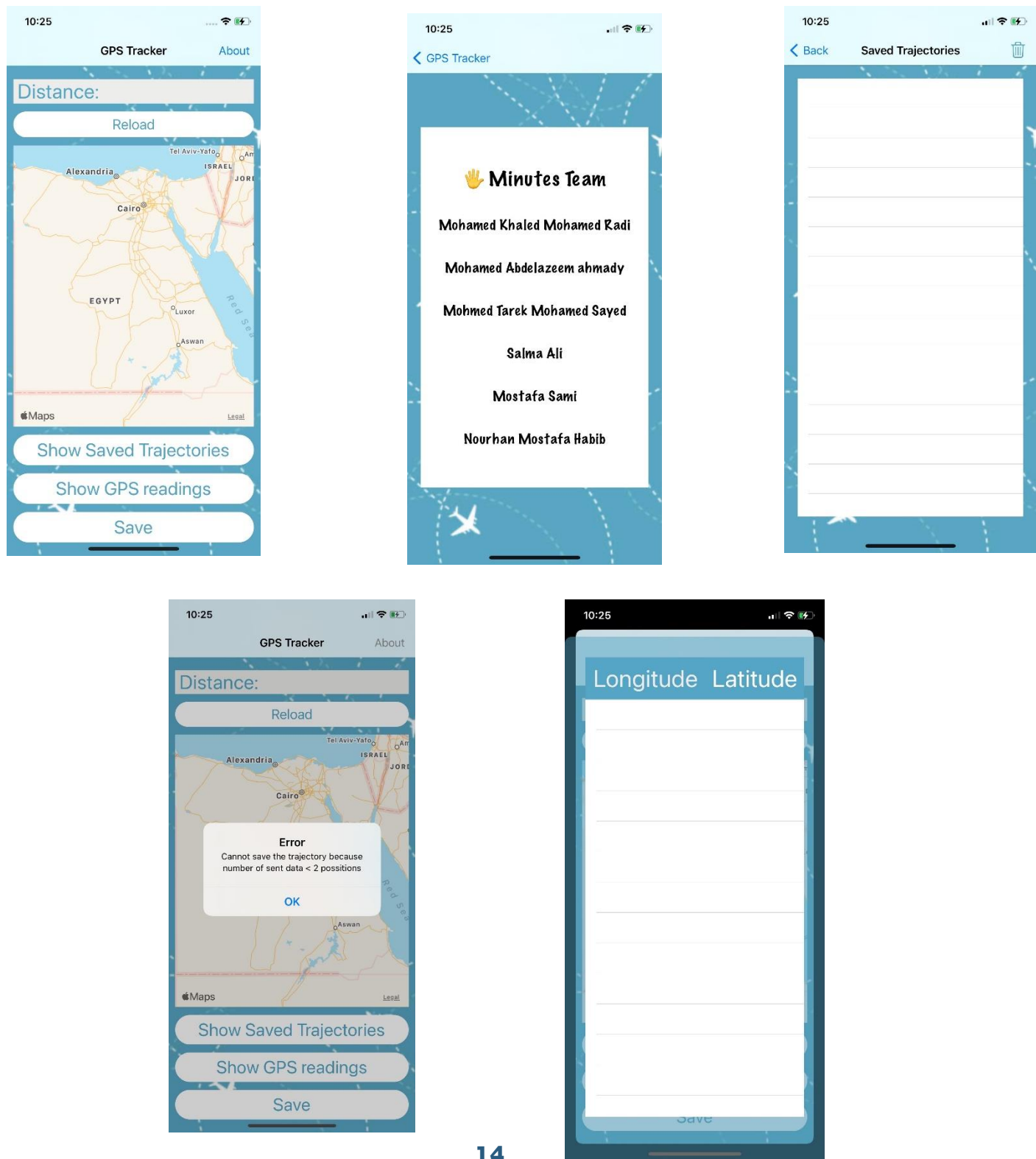
```
COM8 - Tera Term VT
File Edit Setup Control Window Help
U30.06596,31.28027 ##30.06596,31.28026 ##30.06597,31.28026 ##30.06596,31.28023 ##30.06598,31.28023 ##30.06599,31.28023 ##30.06599,31.28021 ##30.06600,31.28021 ##30.06601,31.28020 ##30.06602
,31.28019 ##30.06604,31.28019 ##30.06605,31.28018 ##30.06606,31.28018 ##30.06608,31.28016 ##30.06609,31.28015 ##30.06612,31.28013 ##30.06614,31.28012 ##30.06617,31.28012 ##30.06619,31.28012
##30.06621,31.28011 ##30.06623,31.28011 ##30.06625,31.28010 ##30.06626,31.28010 ##30.06628,31.28009 ##30.06630,31.28009 ##30.06632,28008 ##30.06633,31.28008 ##30.06635,31.28007 ##30.06637,
31.28007 ##30.06638,31.28006 ##30.06639,31.28006 ##30.06641,31.28005 ##30.06642,31.28005 ##30.06644,31.28005 ##30.06644,31.28005 ##30.06645,31.28006 ##30.06646,31.28007 ##30.06646,31.28007
##30.06647,31.28010 ##30.06648,31.28011 ##30.06649,31.28012 ##30.06649,31.28015 ##30.06650,31.28016 ##30.06651,31.28018 ##30.06652,31.28019 ##30.06652,31.28021 ##30.06653,31.28022 ##30.0665
3,31.28023 ##30.06653,31.28025 ##30.06654,31.28026 ##30.06654,31.28027 ##30.06654,31.28028 ##30.06654,31.28030 ##30.06654,31.28032 ##30.06655,31.28033 ##30.06656,31.28033 ##30.06656,31.2803
4 ##30.06657,31.28036 ##30.06657,31.28038 ##
```

Trajectory on Google maps using Python Script After retrieving data from EEPROM



IOS Application Screen-Shots

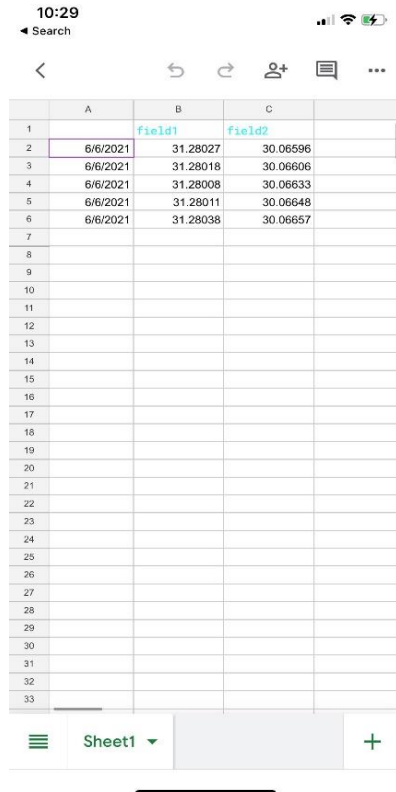
App screens:



Readings set 1:

Google sheet screen shoot, readings from app screen shoot and trajectory version 1 screen shoot

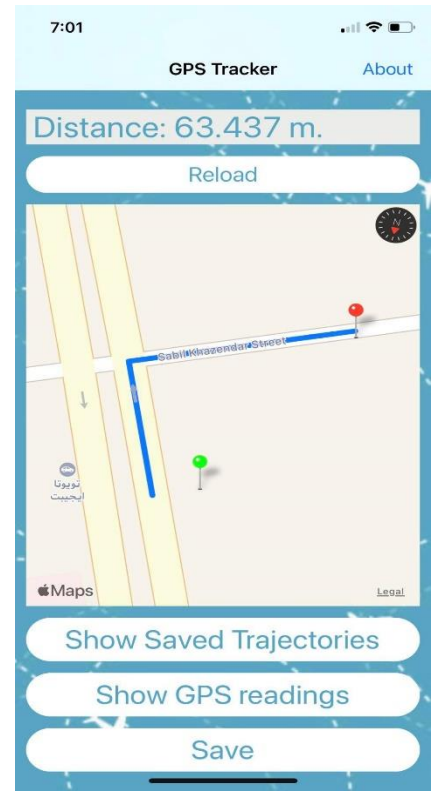
Note: the application calculated distance depending on starting and ending points, not depending on the trajectory.



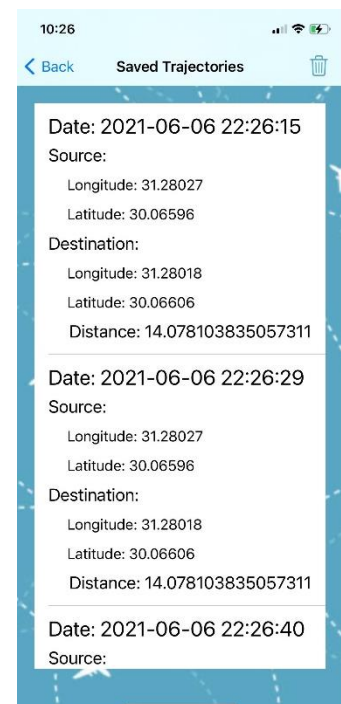
	A	B	C
1		field1	field2
2	6/6/2021	31.28027	30.06596
3	6/6/2021	31.28018	30.06606
4	6/6/2021	31.28008	30.06633
5	6/6/2021	31.28011	30.06648
6	6/6/2021	31.28038	30.06657
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			
32			
33			



Longitude	Latitude
31.28027	30.06596
31.28018	30.06606
31.28008	30.06633
31.28011	30.06648
31.28038	30.06657



Saving trajectory data on phone:



7.Attached Videos

1- Application Testing (GPS,LCD,LED)

<https://photos.app.goo.gl/i9znpxrl7derhC6HA>

2- IOS App testing

<https://drive.google.com/file/d/1AL7dyILrn7GqFBq2TTDfzkOnXB7nyxd8/view?usp=sharing>

3- Final Application Testing (One Shot video-2nd deliverable)

<https://drive.google.com/file/d/1Fs0RwcmR041a4ataKznyjLVPmmrfVm8f/view?usp=sharing>

8. Final PCB Product

