# Internet of Things System Design Project

# REPORT

Green Corridor for Ambulance using IOT based smart Traffic management system.


**Team Members**

| | |
|---|---|
| Abhinav Srinivas R | 110120001 |
| Rahul Raaghav A | 110120089 |
| Sai Venkatakrishnan | 110120097 |

# Progress

**Github Repository link:**
https://github.com/RahulraaghavA1308/IOT_project.git

## 1. Main script

The main script implements all the subsystem in one single script which performs all the following functions.

1. Updating ambulance coordinate into Google cloud.
2. Getting the coordinate of the ambulance from the database.
3. Waypoint generation from ambulance to Hospital.
4. Monitoring distance from ambulance to closest traffic junction.
5. Overriding traffic signal if distance is less than threshold.
6. Releasing override once ambulance crosses the traffic signal.
7. Repeating the procedure every 5 seconds.

```
import math
import gspread
import time
import os
import webbrowser
from waypoints import waypoints_func


from distance_heading import dist_head_func

def sendData(sheetname,datacell,data):
    gc = gspread.service_account(filename='main\client_secret.json')
    wks = gc.open(sheetname).sheet1
    wks.update(datacell,[[data]])

def recieveData(sheetname,datacell):
    gc = gspread.service_account(filename='main\client_secret.json')
    wks = gc.open(sheetname).sheet1
    output = wks.get(datacell)
    return output[0][0]



def calc_bearing(lat1, long1, lat2, long2):
  # Convert latitude and longitude to radians
  lat1 = math.radians(lat1)
  long1 = math.radians(long1)
  lat2 = math.radians(lat2)
```

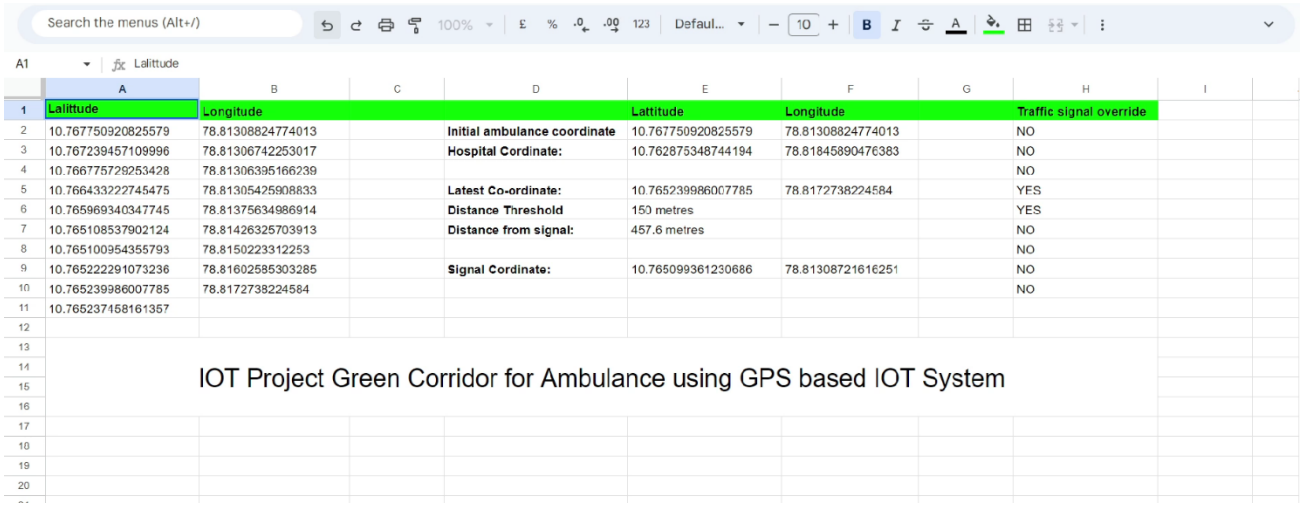
```
  long2 = math.radians(long2)

  # Calculate the bearing
  bearing = math.atan2(
      math.sin(long2 - long1) * math.cos(lat2),
      math.cos(lat1) * math.sin(lat2) - math.sin(lat1) * math.cos(lat2) *
math.cos(long2 - long1)
  )

  # Convert the bearing to degrees
  bearing = math.degrees(bearing)

  # Make sure the bearing is positive
  bearing = (bearing + 360) % 360

  return bearing


distance_threshold = 100

# Starting coordinate will be ambulance coordinate
# Get starting coordinate from google sheet
# ending cordinate will be hardcoded to an hospital

start_lat, start_lon = 10.763379188669989, 78.81447607338754    # Diamond
hostel
end_lat, end_lon = 10.757355527717523, 78.81823453408565        # NITT Library


print(" ")
print(" ")

print("-----------------------------------------------------------------")
print("        GREEN CORRIDOR FOR AMBULANCE USING IOT                   ")
print("-----------------------------------------------------------------")

print(" ")
print(" ")

i = 0

while True:
    i = i + 1
    flag = False
    print(f'*** Iteration : { i + 1 }')

    W = waypoints_func(start_lat, start_lon, end_lat, end_lon)
```

```
    file_path =
"D:\Rahul\College\A_Semester_6\IOT_project\Code\main\maps\index.html"

    chrome_path = 'C:\\Program
Files\\Google\\Chrome\\Application\\chrome.exe'  # Path to Chrome executable
    webbrowser.register('chrome', None,
webbrowser.BackgroundBrowser(chrome_path))
    webbrowser.get('chrome').open(file_path)

    waypoints = []

    for w in W:
        waypoints.append([w["lat"],w["lon"]])

    ##########################################################################
#####
    # Assumming that the traffic junction is hardcoded into this code
    # we will now consider that the second waypoint returned is the junction
    # So we need to find the distance of ambulance from the second waypoint




    print(f'Ambulance coodinate : {start_lat,start_lon}')
    print(" ")
    print(f'Traffic Junction coodinate : {waypoints[1]}')
    print(" ")
    print(f'Next waypoint coodinate : {waypoints[2]}')
    print(" ")

    distance, ret = dist_head_func(start_lat, start_lon, waypoints[1][0],
waypoints[1][1])

    if distance < distance_threshold:
        flag = True

    # We will now assume that the third waypoint returned is the waypoint
    # after the signal. This will be used to get the heading of the ambulance
    # required.

    heading = calc_bearing(start_lat, start_lon, waypoints[2][0],
waypoints[2][1])

    print(f'Distance of Ambulance from Traffic Junction : {distance} metres')
    print(" ")
    print(f'Heading of Ambulance after Traffic Junction : {heading} degrees')

    rel_orientation = ''
```

```
    if(heading > 130 and heading < 170):
        rel_orientation = 'West-South'
    elif(heading > 170 and heading < 190):
        rel_orientation = 'North-South'
    elif(heading > 190 and heading < 230):
        rel_orientation = 'East-South'
    elif(heading > 230 and heading < 260):
        rel_orientation = 'North-West'
    elif(heading > 260 and heading < 280):
        rel_orientation = 'East-West'
    elif(heading > 280 and heading < 310):
        rel_orientation = 'South-West'
    elif(heading > 10 and heading < 40):
        rel_orientation = 'West-North'
    elif(heading > 350 and heading < 10):
        rel_orientation = 'South-North'
    elif(heading > 310 and heading < 350):
        rel_orientation = 'East-North'
    elif(heading > 40 and heading < 80):
        rel_orientation = 'South-East'
    elif(heading > 80 and heading < 100):
        rel_orientation = 'West-East'
    elif(heading > 100 and heading < 130):
        rel_orientation = 'North-East'

    print(" ")
    print(f'Orientation of Ambulance after the signal : {rel_orientation}')
    time.sleep(5)
    print(" ")
    print(" ")


# print(" ")
# print(" ")
# print("------------------------------------------------------------")
# print("                          END                               ")
# print("------------------------------------------------------------")
# print(" ")
```

# Output

```
(OSRM_ENV) D:\Rahul\College\A_Semester_6\IOT_project\Code>d:/Rahul/College/A_Semester_6/IOT_project/Code/OSRM_ENV/Scripts/pyt
hon.exe d:/Rahul/College/A_Semester_6/IOT_project/Code/main/main.py


------------------------------------------------------------
        GREEN CORRIDOR FOR AMBULANCE USING IOT
------------------------------------------------------------


*** Iteration : 1
Ambulance coodinate : (10.763379188669989, 78.81447607338754)

Traffic Junction coodinate : [10.763079, 78.81637]

Next waypoint coodinate : [10.758092, 78.816635]

Distance of Ambulance from Traffic Junction : 209.56550338532026 metres

Heading of Ambulance after Traffic Junction : 158.14154351823368 degrees

Orientation of Ambulance after the signal : West-South
Updating data in google sheet


*** Iteration : 2
Ambulance coodinate : (10.763379188669989, 78.81447607338754)

Traffic Junction coodinate : [10.763079, 78.81637]

Next waypoint coodinate : [10.758092, 78.816635]

Distance of Ambulance from Traffic Junction : 209.56550338532026 metres
```



| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Lalittude | Longitude | | | Lattitude | Longitude | | Traffic signal override | |
| 2 | 10.767750920825579 | 78.81308824774013 | | Initial ambulance coordinate | 10.767750920825579 | 78.81308824774013 | | NO | |
| 3 | 10.767239457109996 | 78.81306742253017 | | Hospital Cordinate: | 10.762875348744194 | 78.81845890476383 | | NO | |
| 4 | 10.766775728253428 | 78.81306395166239 | | | | | | NO | |
| 5 | 10.766433222745475 | 78.81305425908833 | | Latest Co-ordinate: | 10.765239986007785 | 78.8172738224584 | | YES | |
| 6 | 10.765969340347745 | 78.81375634986914 | | Distance Threshold | 150 metres | | | YES | |
| 7 | 10.765108537902124 | 78.81426325703913 | | Distance from signal: | 457.6 metres | | | NO | |
| 8 | 10.765100954355793 | 78.8150223312253 | | | | | | NO | |
| 9 | 10.765222291073236 | 78.81602585303285 | | Signal Cordinate: | 10.765099361230686 | 78.81308721616251 | | NO | |
| 10 | 10.765239986007785 | 78.8172738224584 | | | | | | NO | |
| 11 | 10.765237458161357 | | | | | | | | |
| 12 | | | | | | | | | |
| 13 | | | | | | | | | |
| 14 | | | | | | | | | |
| 15 | | | | | | | | | |
| 16 | | | | | | | | | |
| 17 | | | | | | | | | |
| 18 | | | | | | | | | |
| 19 | | | | | | | | | |
| 20 | | | | | | | | | |

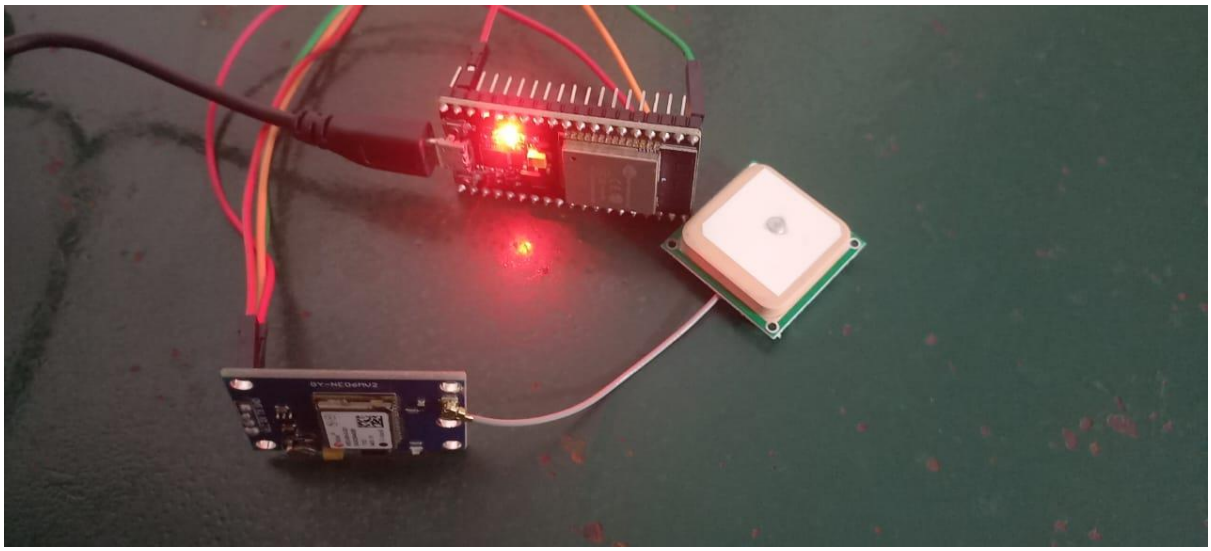
IOT Project Green Corridor for Ambulance using GPS based IOT System

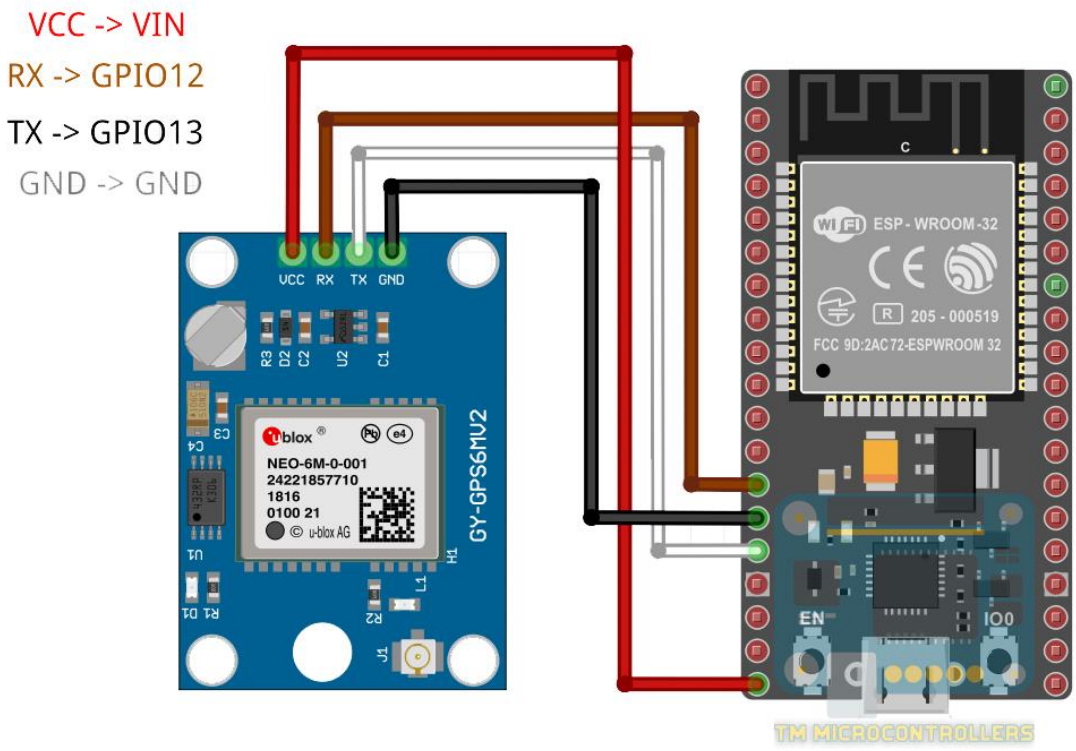
## 2. Project Demo

Data from the ESP32 is updated into the **cloud storage** (google sheets) and live execution is done to verify the logic.

## 3. Electronic Circuit design

The ESP32 is connected to the NEO6M GPS module to enable transmission of the GPS coordinates to the cloud. This is placed on the ambulance with a battery to track the ambulance coordinates.





## 4. Waypoint generation

The Starting and ending coordinates are specified as latitude and longitude and using this an **HTTP** request URL is built. This request uses the **OSRM python API** to get the waypoints between the given coordinates.

The OSRM supports API calls for different options like driving in **cars** & **bike** or by **foot**.

**Code**:

```
import requests
import folium
import json

# Define the start and end coordinates
start_lat, start_lon = 10.768326787577756, 78.813116724416   # Amber hostel
end_lat, end_lon = 10.762364823965529, 78.81867565936166       # NITT Hospital

# start_lat, start_lon = 10.759278170362593, 78.81326494926076
# end_lat, end_lon = 10.777865835062338, 78.79859918760036



# Define the OSRM API endpoint and parameters
osrm_url = "http://router.project-
osrm.org/route/v1/driving/{},{};{},{}".format(start_lon, start_lat, end_lon,
end_lat)
params = {
    "steps": "true",
    "geometries": "geojson",
    "overview":"full"
}

# Make the API request and get the response
response = requests.get(osrm_url, params=params)
response_data = response.json()


# Extract the route geometry from the response
geometry = response_data["routes"][0]["geometry"]

# Extract the coordinates of each waypoint from the response
waypoints = []
for feature in response_data['routes'][0]['geometry']['coordinates']:
    waypoint = {
        "lat": feature[1],
        "lon": feature[0]
    }
    waypoints.append(waypoint)
```

```python
# Print the list of waypoints
print("Way points generated : ")

#Create a map
map = folium.Map(location=[waypoints[0]["lat"], waypoints[0]["lon"]],
zoom_start=10)

# Add the route to the map
folium.GeoJson(geometry).add_to(map)

# Add markers for the waypoints
for wp in waypoints:
    print(wp)
    folium.Marker([wp["lat"], wp["lon"]]).add_to(map)

# Save the map
map.save("index.html")
```

## Output

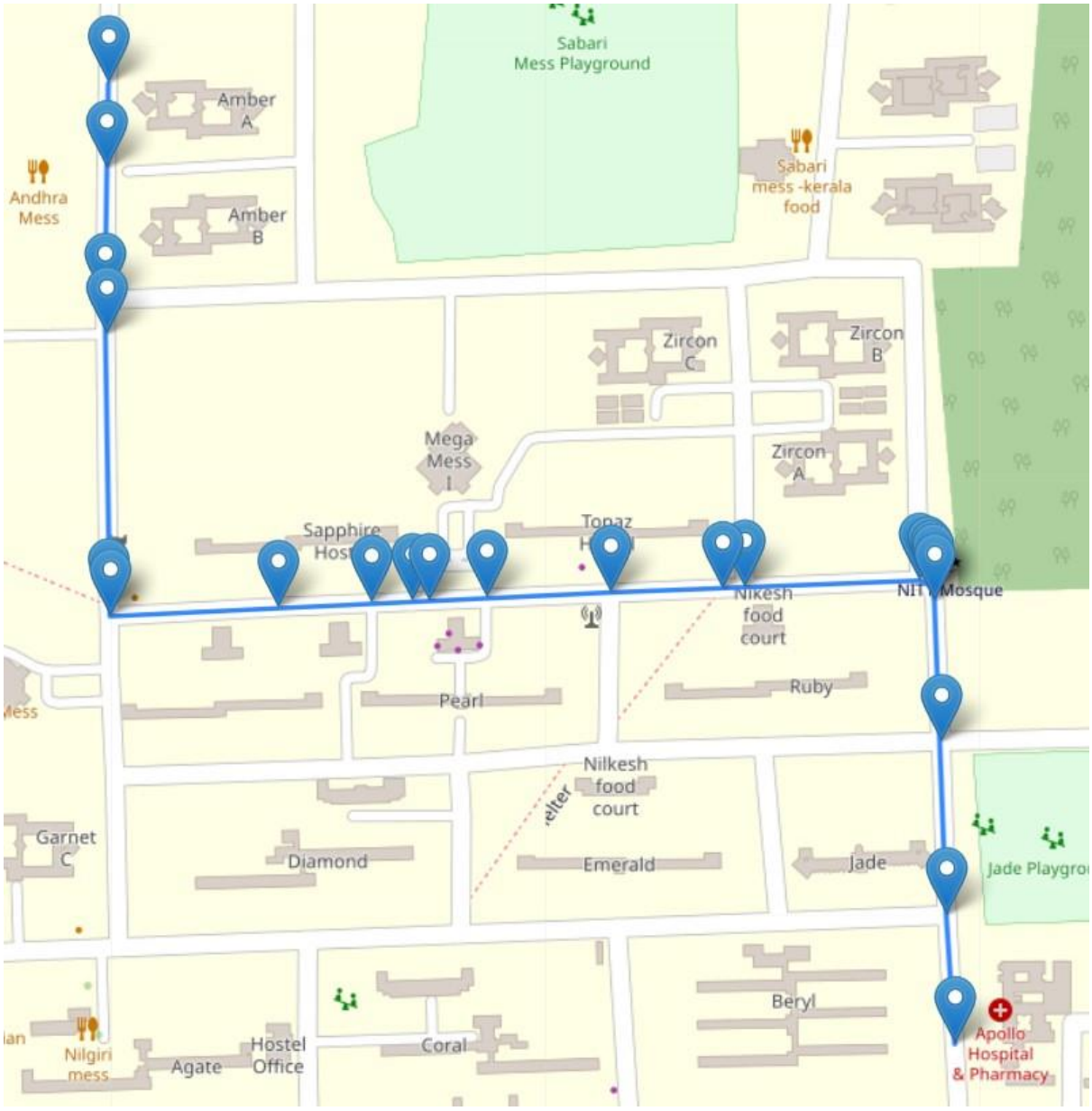
```
(OSRM_ENV) D:\Rahul\College\A_Semester_6\IOT_project\Code>d:/Rahul/College/A_Semester_6/IOT_project/Code/OSRM_ENV/Scrip
ts/python.exe d:/Rahul/College/A_Semester_6/IOT_project/Code/waypoints.py
Way points generated :
{'lat': 10.768326, 'lon': 78.81315}
{'lat': 10.767797, 'lon': 78.81314}
{'lat': 10.766979, 'lon': 78.813132}
{'lat': 10.766766, 'lon': 78.813134}
{'lat': 10.765079, 'lon': 78.813155}
{'lat': 10.765014, 'lon': 78.813159}
{'lat': 10.765071, 'lon': 78.81422}
{'lat': 10.765096, 'lon': 78.814818}
{'lat': 10.765106, 'lon': 78.815073}
{'lat': 10.765111, 'lon': 78.815175}
{'lat': 10.765126, 'lon': 78.81554}
{'lat': 10.765158, 'lon': 78.816325}
{'lat': 10.765187, 'lon': 78.817032}
{'lat': 10.765193, 'lon': 78.817178}
{'lat': 10.765238, 'lon': 78.81828}
{'lat': 10.765216, 'lon': 78.81833}
{'lat': 10.765198, 'lon': 78.818352}
{'lat': 10.765175, 'lon': 78.818368}
{'lat': 10.765109, 'lon': 78.818378}
{'lat': 10.764237, 'lon': 78.818417}
{'lat': 10.763159, 'lon': 78.818455}
{'lat': 10.762355, 'lon': 78.818503}
```

## 5. Waypoint Visualization

The waypoints generated between the starting and ending coordinates are visualized using **folium map** to verify the generation.

**Folium** makes it easy to visualize data that's been manipulated in Python on an interactive leaflet map.

In the below example waypoint are generated between **Amber hostel** and **NITT Hospital**.



## 6. Algorithm for Controlling the Traffic junction

**Data stored in database -** Co-ordinates of hospitals in an area.
**-** Co-ordinates of signals in the area
**Input to the program** **-** Starting location of ambulance.

**Step 1:** Starting co-ordinate is sent to the cloud from ESP32.

**Step 2:** Starting co-ordinate is compared with the hospital co-ordinates to determine closest hospital.

**Step 3:** Waypoints are from starting co-ordinate to closest hospital co-ordinate using OSRM APIs

**Step 4:** Updated co-ordinates of the ambulance are sent to the cloud every 10 seconds.

**Step 5:** The traffic signals which are present in the generated waypoints' path are extracted by the python script from the database.

**Step 6:** The ambulance's real time co-ordinate is checked if it is at threshold distance (200m) from any of the traffic signals.

**Step 7:** The algorithm considers the successive waypoint to determine the heading of the ambulance at that traffic junction.

**Step 8:** Appropriate labels such as "NS" (North-South), "NW" are assigned based on the calculated heading and is updated to the cloud.

**Step 9:** The ESP32 placed in the traffic junction obtains this data from the cloud and actuates the signal.

**Step 10:** Based on the data received, the corresponding traffic signal is alone turned to **green**, and all others are switched to **red**.

**Step 11:** Once the ambulance crossed the signal, normal time-based switching of signals is resumed.

**Step 12:** The same process is repeated for the next traffic signal.

## 7. Sending data from the Ambulance to the Cloud

Each ambulance has an ESP32 which is connected to the internet through mobile hotspot. The ESP32 uses the **NEO 6M GPS** module to get the GPS coordinates and updates it into the cloud every 10 secs.

Code

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <SoftwareSerial.h>
SoftwareSerial ss(4, 3);

const char* ssid = "No Internet";
const char* password = "abcdefgh";
String GOOGLE_SCRIPT_ID =
"AKfycbyY0rYxiLoYppMUElMJ6OxrkASb0NORlsqELCQNiRcWg7FDSGrNKFfpCpo0JQGMrMQv";

HTTPClient http;
//Configuring the http client

void setup() {
  Serial.begin(9600);
  ss.begin(9600);
  //Connecting to Wifi
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
  Serial.println("Connected to Wifi");



}

void loop(){
 {
  byte gpsData = ss.read();

  String urlFinal =
"https://script.google.com/macros/s/"+GOOGLE_SCRIPT_ID+"/exec?id=Sensor_1&valu
e="+ String(gpsData);
  Serial.println("POSTing data to spreadsheet:");
  http.begin(urlFinal.c_str());
```

```
  http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
  int httpCode = http.GET();
  Serial.print("HTTP Status Code: ");
  Serial.println(httpCode);
  delay(10000);
 }
}
```

## 8. Communication between Cloud and Traffic signal

The ESP32 which is installed in the Traffic junction sends an HTTP request to the cloud every 10 secs to receive the data. Once an ambulance is detected to be near a signal the control is overridden, and the direction of ambulance is used to open only the corresponding lane while all other lanes are closed. Once the ambulance crosses the Traffic junction the override is stopped, and routine control continues.

Code

```
#include<WiFi.h>
#include<HTTPClient.h>

const char ss_id = "Network";
const char pwd= "Password";
String Gscpt = " ";

WiFiClientSecure client;

void setup() {Serial.begin(9800);
WiFi.mode(WIFI_STA);
WiFi.begin(ss_id,pwd);


  pinMode(1,OUTPUT);
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
  pinMode(12,OUTPUT);
```

```
}

void loop() {
  String W;
  W=spreadsheet();
  if(W[0]=='N')
   {if(W[1]=='E')
     digitalWrite(1,HIGH);
   else if(W[1]=='S')
     digitalWrite(2,HIGH);
   else if(W[1]=='W')
     digitalWrite(3,HIGH);}
  else if(W[0]=='E')
   {if(W[1]=='N')
     digitalWrite(4,HIGH);
    else if(W[1]=='S')
     digitalWrite(5,HIGH);
    else if(W[1]=='W')
     digitalWrite(6,HIGH);}
  else if(W[0]=='S')
   {if(W[1]=='E')
    digitalWrite(7,HIGH);
    else if(W[1]=='N')
     digitalWrite(8,HIGH);
    else if(W[1]=='W')
     digitalWrite(9,HIGH);}
  else if(W[0]=='W')
  {if(W[1]=='E')
    digitalWrite(10,HIGH);
  else if(W[1]=='S')
    digitalWrite(11,HIGH);
  else if(W[1]=='N')
    digitalWrite(12,HIGH);}
}

char* spreadsheet(){
  HTTPClient http;
  String url=" /"+Gscpt+"/ ";
  http.begin(url.c_str());
  http.setFollowRedirects(HTTP_STRICT_FOLLOW_REDIRECTS);
  int httpCode= http.GET();
  String X;
   if(httpCode>0){
    X= http.getString();
    return X;}
  http.end();
   }}
```

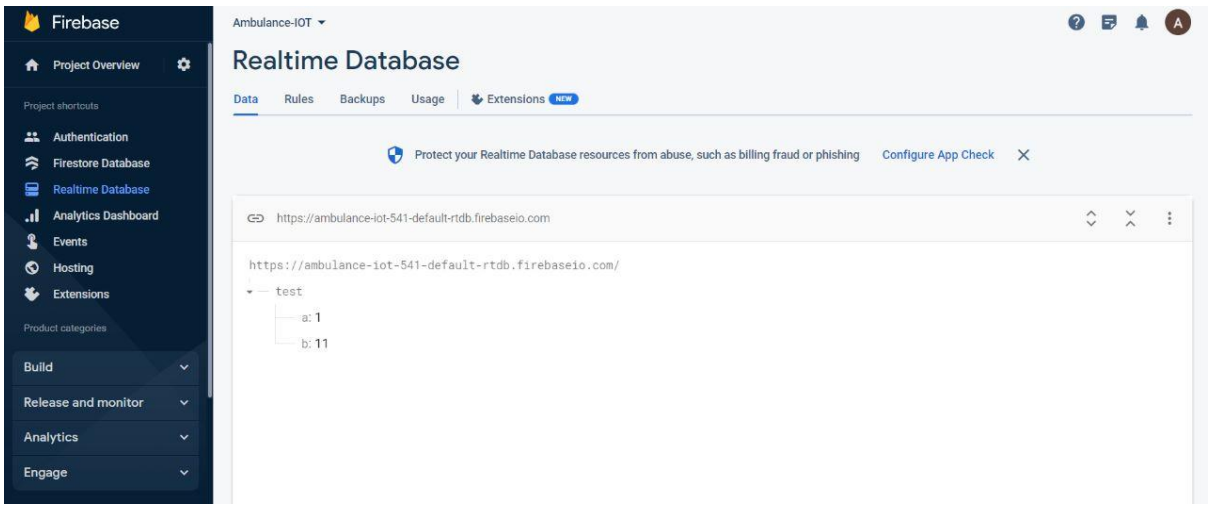
## 9. Receiving data from cloud to Traffic junction

```
/*  Project : Read Google Spread Sheet Data from ESP32  */
/*Refer following video for complete project : https://youtu.be/0LoeaewIAdY*/
/***************************/
//Things to change


#include <WiFi.h>
#include <HTTPClient.h>

#include <Wire.h>


const char * ssid = "Rahul";
const char * password = "rahulnote10s";
String GOOGLE_SCRIPT_ID =
"AKfycbyQwuel6RUsje7t793ao2yIxfXHRz8fW1jV4523gelsSJBgH9O7m69rzYzUJGbuOmMelQ";

const int sendInterval = 5000;
/***************************/

WiFiClientSecure client;

void spreadsheet_comm(void) {
   HTTPClient http;
   String
url="https://script.google.com/macros/s/"+GOOGLE_SCRIPT_ID+"/exec?read";
//   Serial.print(url);
  Serial.print("Making a request");
  http.begin(url.c_str()); //Specify the URL and certificate
  http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
  int httpCode = http.GET();
  String payload;
    if (httpCode > 0) { //Check for the returning code
        payload = http.getString();

        Serial.println(httpCode);
        Serial.println(payload);
      }
    else {
      Serial.println("Error on HTTP request");
    }
  http.end();
}

void setup() {
  Serial.begin(9600);
  delay(10);
```

```
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  Serial.println("Started");
  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

Serial.println("Ready to go");
}

void loop() {
  spreadsheet_comm();
  delay(sendInterval);
}
```

## 10. Firebase as alternative cloud platform

We also tested **Google Firebase** as an alternative cloud platform. Firebase is a set of backend cloud computing services and application development platforms provided by Google. It hosts **databases**, **services**, **authentication**, and integration for a variety of applications, including Android, iOS, JavaScript, Node.js, Java, Unity, PHP, and C++.




**Code**

```
#include <WiFi.h>
#include <FirebaseESP32.h>

//Provide the token generation process info.
#include <addons/TokenHelper.h>

//Provide the RTDB payload printing info and other helper functions.
#include <addons/RTDBHelper.h>

/* 1. Define the WiFi credentials */
#define WIFI_SSID "No Internet"
#define WIFI_PASSWORD "gpassword"

//For the following credentials, see
examples/Authentications/SignInAsUser/EmailPassword/EmailPassword.ino

/* 2. Define the API Key */
#define API_KEY "ZMtBTIyf0LdYXjhpNKBrxNQMKxfmXreb1GSjvz0d"

/* 3. Define the RTDB URL */
#define DATABASE_URL "ambulance-iot-541-default-rtdb.firebaseio.com"
//<databaseName>.firebaseio.com or
<databaseName>.<region>.firebasedatabase.app
//Define Firebase Data object
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
String main="";
int a, b, x, y;
void setup()
{

  Serial.begin(115200);
  delay(2000);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);
```

```
  /* Assign the api key (required) */
  config.api_key = API_KEY;

  config.database_url = DATABASE_URL;
  ////////////////////////////////////////////////////////////////////////////
//////////////////
  //Please make sure the device free Heap is not lower than 80 k for ESP32 and
10 k for ESP8266,
  //otherwise the SSL connection will fail.
  ////////////////////////////////////////////////////////////////////////////
//////////////////

  Firebase.begin(DATABASE_URL, API_KEY);

  //Comment or pass false value when WiFi reconnection will control by your
code or third party library
 // Firebase.reconnectWiFi(true);

  Firebase.setDoubleDigits(5);

}

void loop()
{
    x=random(0,9);
    y=random(10,19);

  if (Firebase.ready())
  {

    //Firebase.setInt(fbdo, main, 5);
    Firebase.setInt(fbdo, "/test/a", x);
    Firebase.setInt(fbdo, "/test/b", y);
    delay(200);

    Serial.printf("Get int a--  %s\n", Firebase.getInt(fbdo, "/test/a") ?
String(fbdo.to<int>()).c_str() : fbdo.errorReason().c_str());
     a=fbdo.to<int>();
    Serial.printf("Get int b--  %s\n", Firebase.getInt(fbdo, "/test/b") ?
String(fbdo.to<int>()).c_str() : fbdo.errorReason().c_str());
     b=fbdo.to<int>();

  Serial.println();
  Serial.print("a:");
  Serial.print(a);
  Serial.print("  b: ");
  Serial.print(b);
```

```
  Serial.println();
  Serial.println("------------------");
  Serial.println();

  delay(2500);
  }
}
```