



Department of Computer Engineering and Information Technology  
AMIRKABIR UNIVERSITY OF TECHNOLOGY

B.Sc. THESIS

# DESIGN AND IMPLEMENTATION OF A REAL-TIME OBJECT TRACKING SYSTEM

*Submitted in partial fulfillment of  
the degree*

BACHELOR OF SCIENCE  
IN  
COMPUTER ENGINEERING

Submitted by  
SAREH SOLTANI

Conducted in  
AOLAB (THE ACADEMY FOR AMIRKABIR IoT LABORATORY)

Under the guidance of  
PROF. BAHADOR BAKHSI

June 2019

## **Abstract**

In recent years, IoT technology has grown significantly and has been able to meet diverse and complex needs in various fields. One of the applications of the Internet of Things is real-time tracking. The positioning and tracking system has provided reliable solutions to ensure the safety of people and vehicles, and also has a significant impact on optimizing the quality of monitoring. This system can be used for a broad range of applications such as traffic management and vehicle tracking/anti-theft system, and finally, traffic routing and navigation. It can be applied in many business cases, like public transportation, so passengers can track their buses and trains by following the vehicle account on social networks.

In this project, a system was developed to accurately determine the location, direction of movement, and speed of moving objects at any given time using SIM808 Module. In this system, each object is equipped with a GPS (Global Positioning System) module that receives its location every two minutes from the satellite and sends it to the software servers via the GSM modem. Software servers analyze the information after receiving it. In this part of the project, a web application was developed to process the transmitted data and then store it in a database and finally convert the stored information to visible to users. This allows you to see the speed, direction of movement, and the object's locations on the map. Furthermore, a heatmap was implemented to show the frequently visited places by users.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Tracking system components</b>	<b>4</b>
2.1	System design and architecture . . . . .	4
2.2	System components . . . . .	6
2.2.1	Hardware components . . . . .	6
2.2.2	Software components . . . . .	8
<b>3</b>	<b>Implementation and Performance Analysis</b>	<b>9</b>
3.1	Evaluate Tracking system performance . . . . .	10
3.1.1	Circuit performance evaluation . . . . .	10
3.2	Circuit Architecture evaluation . . . . .	15
3.3	Application . . . . .	15
3.3.1	Display Information . . . . .	16
3.4	Information analysis . . . . .	20
<b>4</b>	<b>Future Work</b>	<b>21</b>
<b>5</b>	<b>Conclusion</b>	<b>22</b>
	<b>References</b>	<b>23</b>
<b>6</b>	<b>Appendix</b>	<b>24</b>

# List of Figures

2.1	Block Diagram of Tracking System [6]	5
2.2	Architecture of Tracking System[3]	6
2.3	Arduino Uno board	7
2.4	SIM 808 module	7
2.5	Arduino IDE	8
3.1	Designed Tracking System	10
3.2	Performance of implemented code on Arduino [3]	11
3.3	Read Information diagram [8]	12
3.4	GSM [8]	13
3.5	GSM [8]	14
3.6	SIM 808 - Arduino	15
3.7	Show the path of the object in the application	16
3.8	Display the speed, time and date of movement of the object in each location on the map	17
3.9	Receive the message sent by the user to receive the location	18
3.10	The recieved message	19
3.11	See the position of the object on the map	19
3.12	high-traffic locations on the heatmap	20

# Chapter 1

## Introduction

The concept of the Internet of Things refers to objects with a unique identity and the ability to transmit data over the network, without the need for human interaction and intervention. Its primary purpose is to make objects intelligent and provide a platform through which objects can send and receive information. The Internet of Things (IoT) refers extensively to the development of computing capabilities and network communications of objects, devices, sensors, or any other item that is not typically considered a computer. These smart objects can collect, analyze and manage data remotely [1].

The Internet of Things is a wide range of sensors and actuators that measure and process various environmental conditions. In recent years, IoT technology has grown significantly and has met many complex needs in various fields. Due to the expansion of new technologies, the production of intelligent sensors, the growth of communication technologies, and the complexity of requirements, the Internet of Things has gained a lot of power. It has led to the expansion of intelligent systems in the environment [2].

These systems must interact with each other to have a positive impact on the environment. IoT-based technologies have different requirements compared to other technologies. Typically, these systems have less memory, power consumption, and bandwidth than other systems. Most smart systems are battery-powered and located in a remote location where they cannot be charged continuously. As a result, the power consumption and coverage range for these systems, especially those implemented at the macro level, such as smart farming, smart city and home, and tracking issues, is very important. There are several wireless communication protocols, each with its unique characteristics.

The Internet of Things is a new communication platform for communicating between intelligent objects. The introduction of this platform has provided

new facilities for solving problems, such as locating and tracking moving objects from vehicle attacks at the city, region, or country level. The Internet of Things (IoT) is rapidly gaining access to telecommunications scenarios, and it is expected that the exchange of information in global resource chains will facilitate.

Widely, Internet of Things can be used as the mainstay of pervasive systems and the activation of intelligent environments for ease in identifying objects and retrieving information from the Internet. From a conceptual point of view, the Internet of Things relies on three principles related to intelligent objects' ability: Ability to identify, transfer and interact either among themselves or with other users. Objects are usually classified either individually or as a member of a category.

One of the most important issues today is the tracking of moving objects, which refers to the immediate tracking of a specific moving object's current position. Moving object tracking systems is a solution to many problems, including security issues. It is a technology used to determine the location of an object.

One of the important applications of IoT is tracking systems that are used in various technologies. Tracking systems were developed for the transportation industry. One of the basic needs of the owners of this industry is to check the position of vehicles. The earliest systems for locating were passive systems that stored information in memory and could only be accessed when a vehicle was available. These systems are not suitable for real-time applications because they need to provide information to the user immediately. For solving this issue, active systems were created that allow the use of in-vehicle hardware and remote tracking servers.

Today, the safety of people and vehicles has become a public concern. The tracking system has provided reliable solutions to ensure people's and vehicles' safety. It has a significant impact on optimizing the quality of monitoring and management of public transportation, vehicle movements, people (children and the elderly), or It has every other moving object. Tracking system is a technology that makes it possible to determine the exact position and track of people, vehicles, or any other moving object using various methods such as the Global Positioning System [3].

In addition to vehicles, tracking systems also play an important role in remote monitoring and environmental monitoring applications. For example, tracking animals, humans, and locating objects are some of this system's applications. In human monitoring, this system can be beneficial for the elderly who have certain diseases such as Alzheimer's and are more likely to get lost, or for the safety of children. Families can use this system to find the position of their elderly or child [4].

The system we have implemented in this project can be used in various cases. One of the applications that can be imagined for the Internet of Things is implementing a system that can determine the exact position and path of each moving object at any time. In this project, we intend to build a tracking system that can identify a moving object's exact position and path.

In this project, our communication will be one-way in that the coordinates of the moving object are continuously measured by the GPS module and sent to a server. This module is continually connected to the satellite to get coordinates. GPS data is sent to the Arduino. Finally, the GSM modem sends this information to the software server. In this project, the software servers analyze the information after receiving it, and we will not have a request from the server since our communication will be one-way. In this part of the project, web-based software will be developed to process the submitted information, store it in the database, and finally convert the stored information into a display for users. An application written using stored data displays the location on a map. In this way, the object or person can be found at the current time. In the end, the current position and direction of the person will be displayed on the map. Another application of this system is to obtain high-traffic locations by analyzing the collected information.

In the continuation of this dissertation, we talk about the general architecture of the tracking system and its components. Then, we introduce the implementation method of this system using the mentioned components. Finally, we talk about the results of the system implementation and what can be done in the future based on this project.

# Chapter 2

## Tracking system components

The main goal of our project is to design and implement a system that can determine the exact position of each moving object at any time. The mentioned system should be cost-effective in addition to proper performance. To design such a system, we must first identify the system requirements and the overall architecture of the system we want. Then we use this architecture to implement the appropriate modules. In this chapter, in section 2.1, we first explain the general design of the tracking system and then in section 2.2, we introduce the components used in this design.

### 2.1 System design and architecture

In this section, we will design our system. According to the project's requirements, we must specify the transmitter and receiver modules, communication protocol, and application program for displaying information. The primary purpose of a tracking system is to track a specific object and gain its path. The tracking system provides information about the current location and speed of the object.

In this project, our communication has been one-way, in which the coordinates of the moving object are continuously measured and sent to a server. Then, the necessary processing of this information is performed on the server-side. According to the above explanations, we can mention three main parts in this system [5]:

- Obtain the location of a moving object using the GPS module
- Send location information to software servers by GSM modem
- Store location information on the server-side and implement an application to display the object's path on the map



As we have seen, the architecture of our system has four main parts. The first part is about locating the object from the satellite using the GPS module. The second part is related to sending the received information to the server using a GSM modem. The third part is the development of an application that uses the received data to display the location of the object on the map.

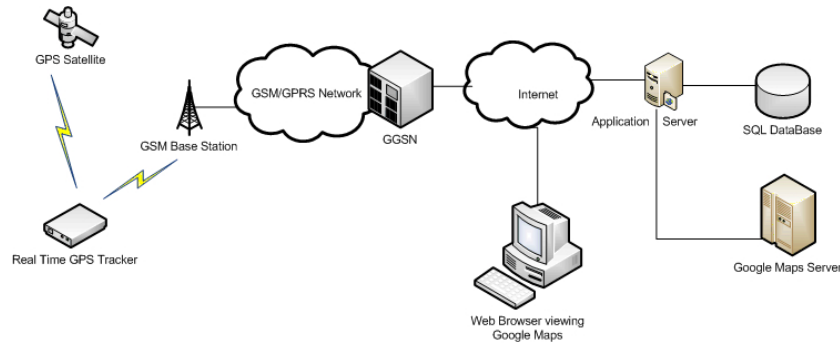


Figure 2.1: Block Diagram of Tracking System [6]

Figure 2.1 shows an overview of the designed system architecture and the relationship between its parts. For selecting the modules, it is necessary to know the task of each section accurately and choose the desired module for it [7].

- In the first part, we need to measure the location of the object continuously. As soon as the object moves, the GPS module consistently receives the moving object's coordinate from the satellite. The signal received from the satellite is weak, so we must use an antenna to amplify the desired signal, and at the end, sends the amplified signal to the Arduino board.
- In the second part, the information received from the GPS module is sent to the server by the GSM modem.
- Software servers analyze the information after receiving them. Our communication in this project is one-way, and we will not have a request from software servers. In this part of the project, web-based software will be developed to process the submitted information and store them in the database. In the last section, this information will be displayed in the designed web page.

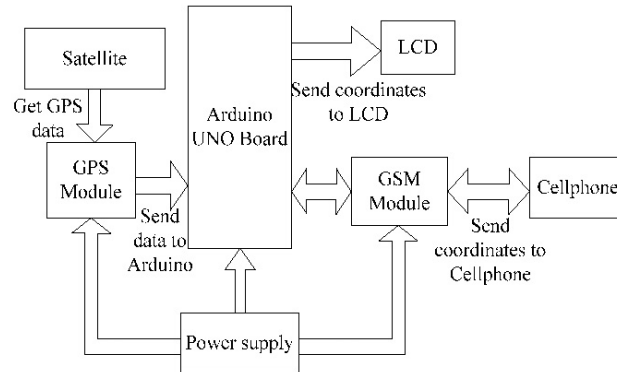


Figure 2.2: Architecture of Tracking System[3]

## 2.2 System components

In the previous section, we defined the system architecture. Now we will express and introduce the components of this architecture in detail.

### 2.2.1 Hardware components

The hardware components used to implement this system are:

- Arduino module
- SIM808 module
- GPS antenna
- GSM modem

#### 2.2.1.1 Arduino module

Arduino is an open-source microprocessor suitable for writing applications that interact with the environment and objects outside. This board is ideal for prototyping, and its software and hardware design are freely available to all people. Any interested person, even with a little knowledge and experience in electronics, can use Arduino to do their projects.

Arduino has a simple programming environment that anyone with a little knowledge of C and C++ can program in this environment and run the program written in Arduino. Various sensors can be connected to the Arduino microcontroller and controlled. The microprocessor used on the Arduino board is based on the Arduino programming language and does not require

any additional software or compiler for coding.

Figure 2.3 shows the Arduino Uno board:

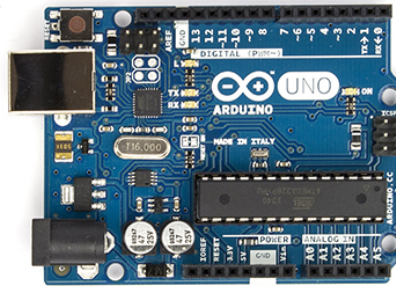


Figure 2.3: Arduino Uno board

#### 2.2.1.2 SIM808 module

The 808 SIM module is a combination of GSM / GPS PRS 9 module and GPS module with 1900 MHz support for data transmission, SMS and voice calling. This module has a SIM card socket in which the SIM card is inserted. Using the GSM / GPRS modem and the 808 SIM module, you can exchange data over the GSM network via the USB interface and access the information of devices located in remote locations.

In figure 2.4, You can see this chip.



Figure 2.4: SIM 808 module

## 2.2.2 Software components

### 2.2.2.1 Arduino IDE

The software used for programming is Arduino software, which you can see in Figure 2.5. Using the C language, you can write the required program, and after compiling, the generated hex code is uploaded on the Arduino. There are different libraries that make coding easier. The program is written using this software to receive data from satellite and send it to a mobile phone.

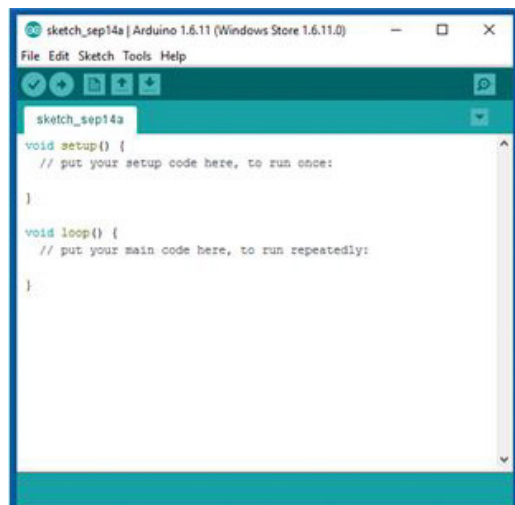


Figure 2.5: Arduino IDE

## Chapter 3

# Implementation and Performance Analysis

In the previous section, we described the system's general architecture and the modules required to implement the tracking system. In this chapter, we will explain how the different parts are connected, and we will explain the implementation of the system.

The tracking system designed in this project uses Arduino module and the SIM 808, including the GSM and GPS antennas, for tracking. The core of this project is the Arduino microcontroller. The geographical location of object is received using a GPS antenna, and then this information is sent to the webserver using GSM technology. A web application has been developed to view and track an object on a map. This application consists of two parts: Front-end and Back-end. The front-end part has developed using Angular framework, and the Back-end part has developed using Express framework. Initially, the SIM 808 module is initialized to get the location from the satellite. The initial settings of this device are done using AT commands. By connecting the GPS antenna, this module will be able to receive location coordinates from the satellite. Then the settings related to the GPRS network are done.

In figure 3.1 How to connect different modules in the tracking system is shown:



Figure 3.1: Designed Tracking System

## 3.1 Evaluate Tracking system performance

As mentioned, the hardware part of our system consists of four modules, SIM 808, GSM receiver, GPS receiver, and Arduino microcontroller. This section will explain the implementation of the designed system, how to connect the various components and the implemented code.

### 3.1.1 Circuit performance evaluation

Before dealing with the modules and how to connect them, it is necessary to observe the system microcontroller performance and processing information in the flowchart. The overall performance of the system was described in the previous section. Now we have a flowchart about the implemented algorithm, and we can have a better understanding of the workflow in the hardware circuit designed and the code written for it.

The following diagram shows the general process of the implemented code on the Arduino.

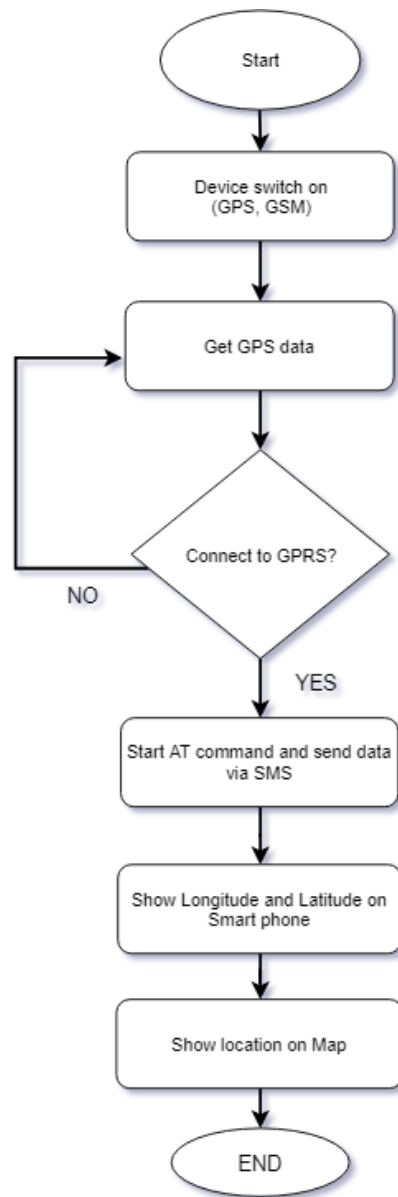


Figure 3.2: Performance of implemented code on Arduino [3]

Firstly, for testing the system, the GPS antenna is connected to the SIM 808 module to receive the object's location (latitude and longitude) from the satellite. For doing this, the Arduino IDE software is used to program the code written on the Arduino board.

The flowchart 3.3 shows how GPS works:

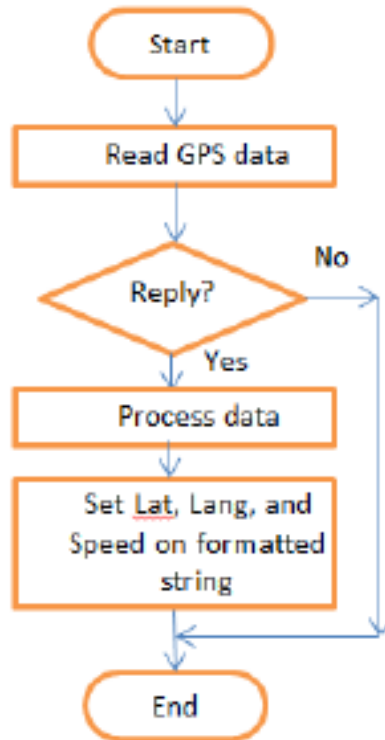


Figure 3.3: Read Information diagram [8]

For sending the object's location to the user via the GSM network, SIM 808 module and the Arduino microcontroller connected to it are used. For connecting 808 SIM module to the GSM network, we use AT commands to program and control it.



The flowchart 3.4 shows how GSM works:

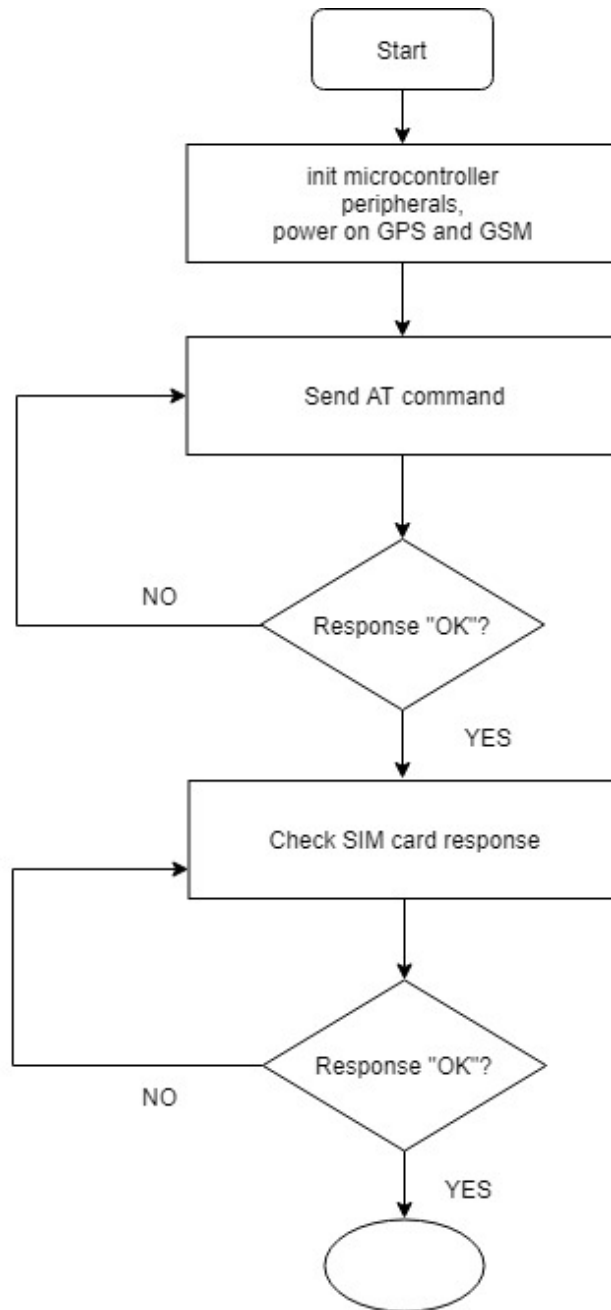


Figure 3.4: GSM [8]

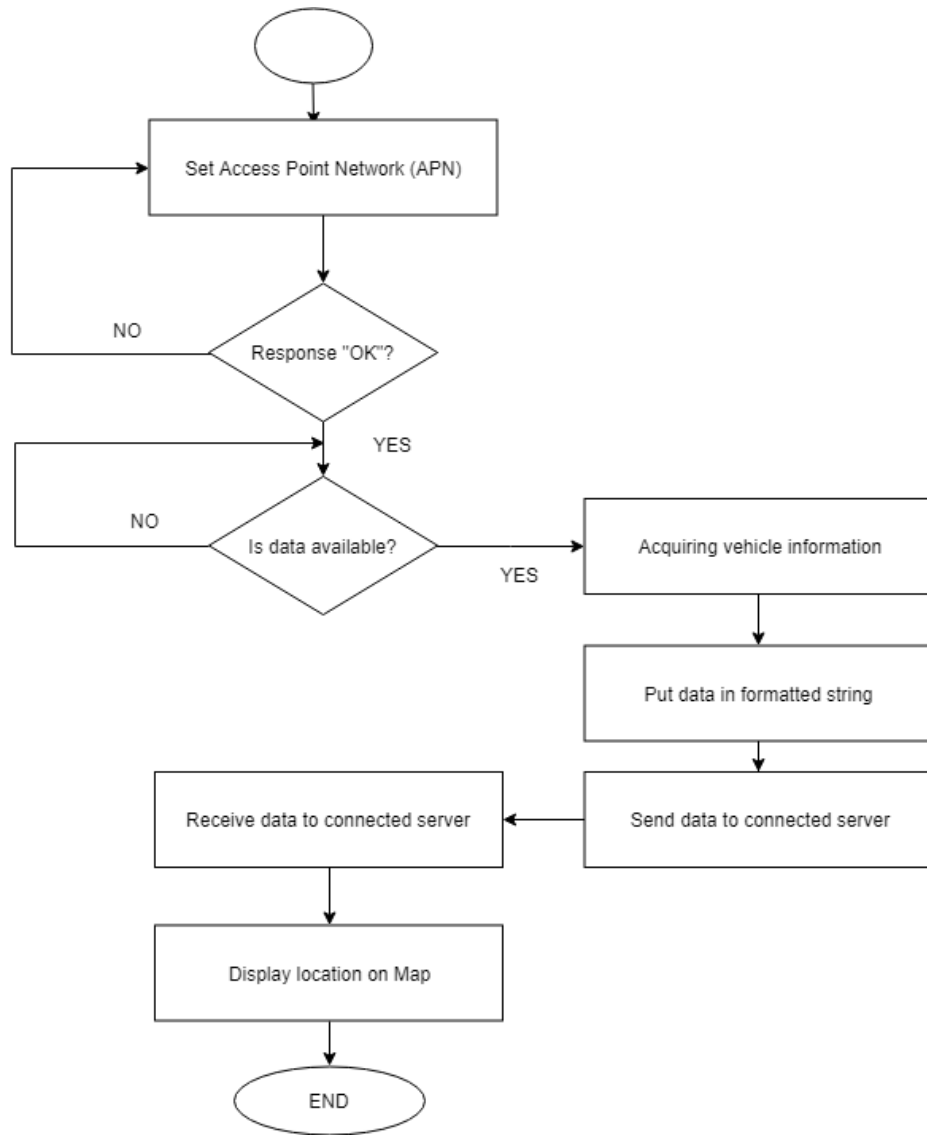


Figure 3.5: GSM [8]

## 3.2 Circuit Architecture evaluation

In this section, we explain the hardware part of the proposed system. As mentioned in the previous sections, using GPS antenna connected to the SIM 808 module, location information is received from the satellite every two minutes and then sent to the server via the GSM network.

In the following figure, you can see how to connect SIM 808 module to the Arduino.

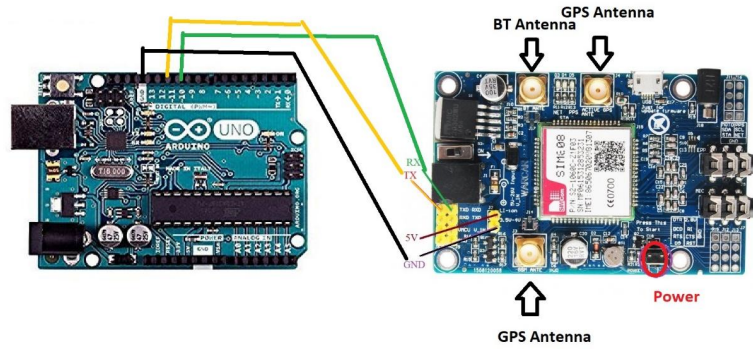


Figure 3.6: SIM 808 - Arduino

The 808 SIM module is connected to the Arduino using a serial interface; it has two TX and RX pins connected to the 10 and 11 Arduino digital pins, respectively. The ground pin of this module is also connected to the ground connection base of the Arduino board. We supply the required voltage of the 808 SIM module through a 9-volt output adapter.

## 3.3 Application

In the Back-end, a web service with RESTful architecture has been developed, which stores the received data in the database, and the web application receives and displays this information.

REST is a web service architecture that uses HTTP to exchange information between two systems. The basic idea of this architecture is to use HTTP to establish information between machines instead of using complex mechanisms to connect devices.

In this section, we first describe the structure of the database and how to communicate with it. Finally, we explain the written web application that shows the path of the object on the map.

### 3.3.0.1 Database

We have used Mongo database to store information and manage the server. Mongo DB is a NOSQL database that runs on various operating systems, including Windows and Macintosh Linux. It also supports most programming languages. Mongo DB provides high performance, accessibility, scalability, fast repeatability, and automatic sharing. Due to the NOSQL structure, Mongo DB only stores and searches data, thus significantly increasing data acquisition and storage speed.

The database received information from the hardware side. Then we store this information, including latitude, longitude, time, and date.

## 3.3.1 Display Information

### 3.3.1.1 Web Application

A web application has been developed to display the information stored in the database. The front-end of this web application is implemented using the Angular framework, and we use Google Maps to show the route.

This web application is available at the following address: [http:// 103.216.62.79](http://103.216.62.79)

Server-side processes were implemented using the Express framework. The server-side code works in such a way that as soon as new data is received from the hardware side, this data is stored in the Mongo database. The hardware sends an HTTP request to the server each time it sends new data to the server: `/api/device/add/`

The code written on the server side acts in such a way that it refers to the database once every 60 seconds. According to the last time of receiving information from it, it gets the received data from that time.

The following figure shows the designed application:

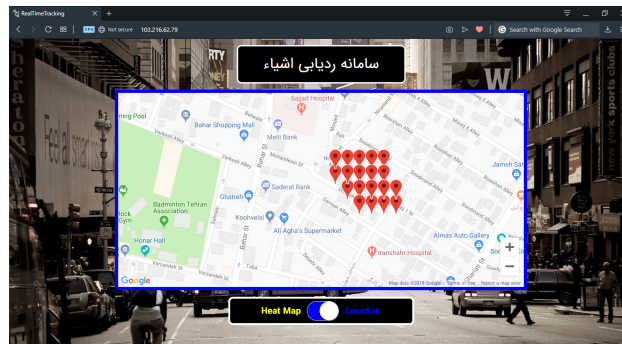


Figure 3.7: Show the path of the object in the application

If we click on any of the markers in the map that indicate the object's position at any time, we can see the speed of the object, time, and date in that position. The following figure shows a view of this output:

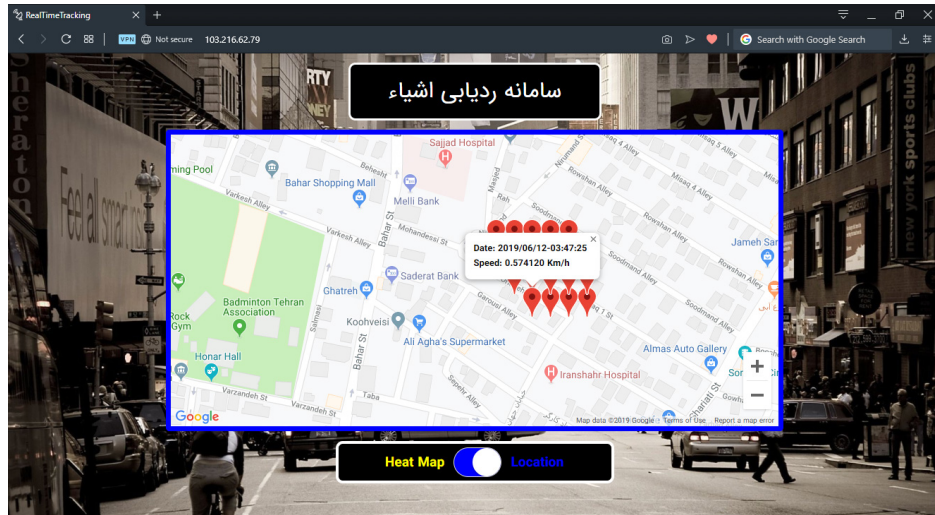
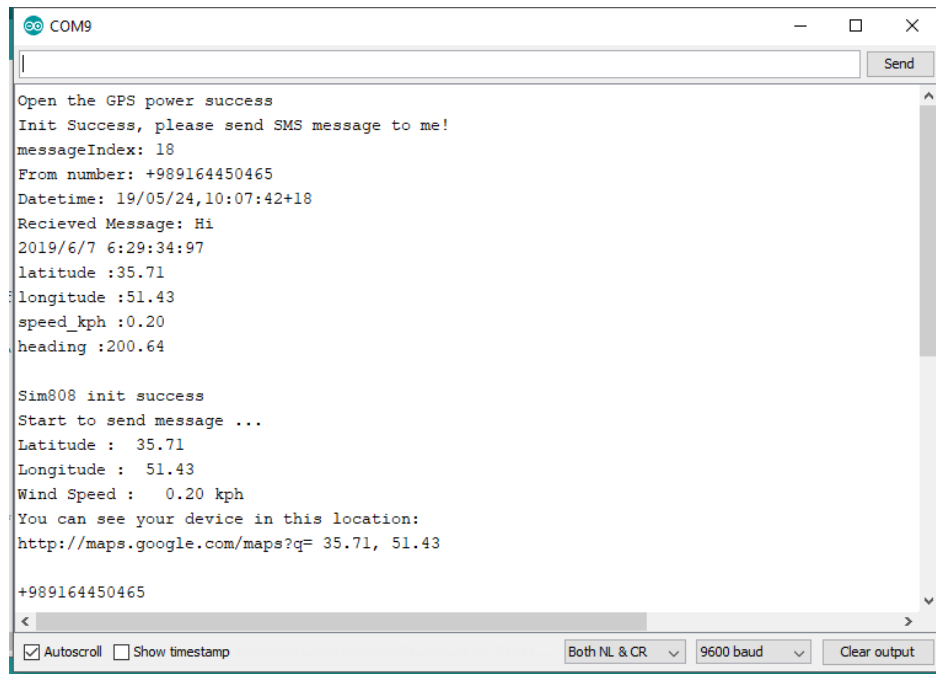


Figure 3.8: Display the speed, time and date of movement of the object in each location on the map

### 3.3.1.2 SMS

One of the unique features of the implemented tracking system is that you can send a message to the SIM card number inserted on the SIM808 module to get the object's current position. After receiving the message, this module also receives the object's location via GPS and sends the necessary information for tracking in response to the received message. This information includes the latitude and longitude and speed of the object. There is also a link attached to this message that the user can click to view the object's location on the map.

In the following figures, you can see the message sent and its response through the designed system:



```
COM9
|
| Send
Open the GPS power success
Init Success, please send SMS message to me!
messageIndex: 18
From number: +989164450465
Datetime: 19/05/24,10:07:42+18
Recieved Message: Hi
2019/6/7 6:29:34:97
latitude :35.71
longitude :51.43
speed_kph :0.20
heading :200.64

Sim808 init success
Start to send message ...
Latitude : 35.71
Longitude : 51.43
Wind Speed : 0.20 kph
You can see your device in this location:
http://maps.google.com/maps?q= 35.71, 51.43

+989164450465
<
Autoscroll Show timestamp Both NL & CR 9600 baud Clear output
```

Figure 3.9: Receive the message sent by the user to receive the location

Figure 3-10 shows the message sent to the user.



Figure 3.10: The recieved message

By clicking on the received link, you can see the current position of the object on the map.

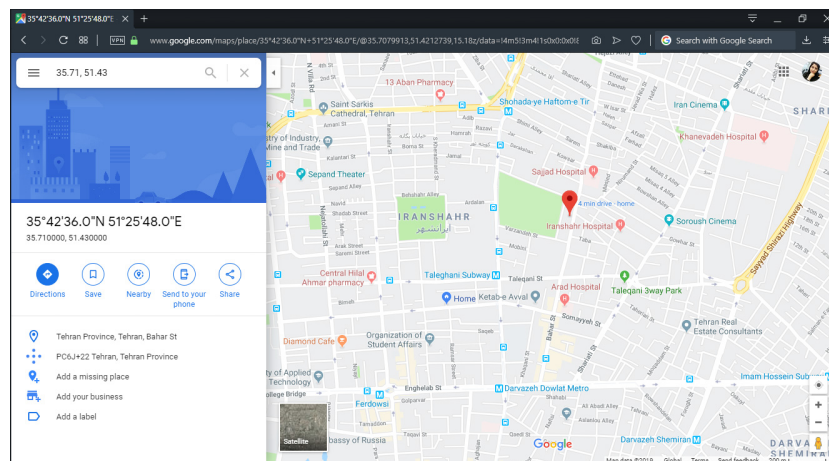


Figure 3.11: See the position of the object on the map

### 3.4 Information analysis

Using the designed real-time tracking system, we were able to continuously measure the location of the object and store this information in the database.

Another feature of this system is to get the frequently visited places of a person in a certain period. To do this, we used the heat map. Heatmaps are graphic and color data that enable us to identify the behavior of our users. In this project, we implemented the heatmap using stored information, which was the location of objects at different times. The most visited places of a person, which are more frequent in our data set than the others, will be visible as red areas on the map.

The designed application consists of two maps. First shows the object's positions on the map, and by selecting the Heatmap option, you can view the heatmap of these points. In Figure 3.12, you can see the high-traffic locations of the object shown in red color.

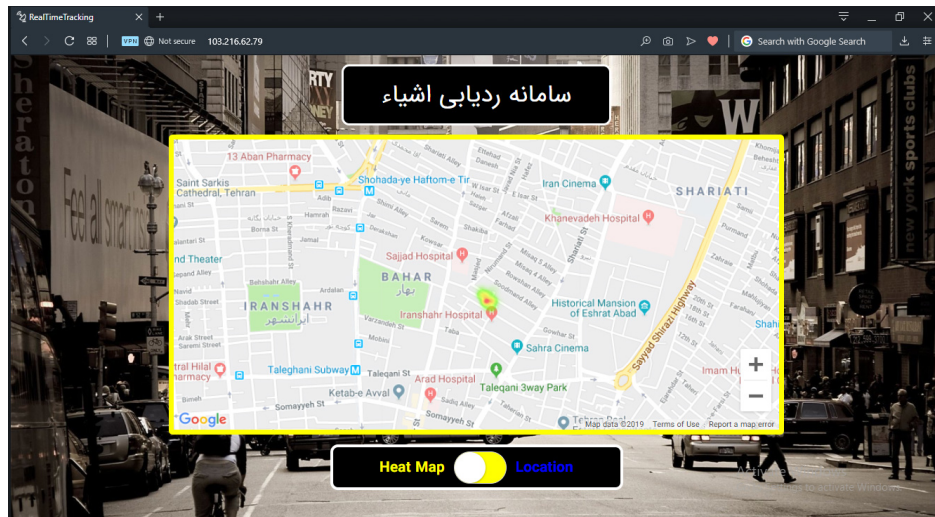


Figure 3.12: high-traffic locations on the heatmap



# Chapter 4

## Future Work

The GPS antenna has a relatively high power consumption. There are two proposed ways to reduce power consumption in this system. The first way that can be suggested for this system is to use machine learning algorithms to predict the object's next move. Another way is using LoRaWAN protocol instead of GPS to locate the object.

Another thing that can be done is to add sensors and actuators to this system. For example, an accelerometer sensor can be used to monitor the object's movement, and only when it is moving, GPS information is sent to the server, which will reduce power consumption.

A camera can also be installed on the designed system. Consequently, in addition to the information sent through GPS, we can also use this camera's images to monitor the location.

As we have said, one of the features of this system was receiving the position of the person by sending a message to the SIM card number inserted in this system. Despite its advantages, it can create security risks for people. Therefore, it is possible to improve the security of the desired tracking system and specify that if it receives a message from a specified and predetermined number, it will send the person's location. Otherwise, no information will be exchanged.

This system is designed in such a way that it can be customized for different applications. For example, we can specify the permissible range for the movement of the object. Therefore, if the object leaves the specified range, the system will automatically contact institutions such as police and minimize human errors.

# Chapter 5

## Conclusion

In this project, our goal was the implementation of a real-time tracking system. Therefore, in this project, we implemented a tracking system to monitor the position of a moving object via SMS as well as online on a map. The core of the designed system is Arduino board and SIM 808 module. The GSM modem is controlled using the AT commands and allows the exchange of information using the GSM network. In this project, GPS module has been used to find the position of the object. GPS Once every two minutes, the object's location is received from the satellite, and this information is sent to the server. The server sends the received information to the implemented web service. Finally, data is stored in the database and displayed in the application, and we can see the path of the object on the map.

The program implemented on the Arduino is written in such a way that it receives the location of the object using GPS every two minutes and sends it to the server. On the server side, this information is stored in the Mongo database. Using this information, we show the path of movement of an object on the map. The server-side code is written so that it visits the database every minute, and the map containing the path of the object is updated.

At the end of this project, we were able to design a real-time tracking system. It is used in various fields, such as tracking vehicles, children and the elderly, etc. By using it, we will be able to take the necessary actions in the fastest possible time.

# References

- [1] M. Mukhtar, "*GPS based Advanced Vehicle Tracking and Vehicle Control System*," International Journal of Intelligent Systems and Applications, Feb. 2015.
- [2] S. Hussain Shah and I. Yaqoob, "*A survey: Internet of Things (IOT) technologies, applications and challenges*," 2016 IEEE Smart Energy Grid Engineering (SEGE), Aug. 2016.
- [3] Md. Rahman, J.Mou, K. Tara and Md. Sarkar, "*Real time Google map and Arduino based vehicle tracking system*," 2016 IEEE Smart Energy Grid Engineering (SEGE), Aug. 2016.
- [4] J. Saranya and J. Selvakumar, "*Implementation of children tracking system on android mobile terminals*," Implementation of children tracking system on android mobile terminals, Apr. 2013.
- [5] B. Bidabad and A. Tayebi, "*Design and implementation of vehicle tracking system using GPS/GSM/GPRS technology and smartphone application*," 2014.
- [6] M. Omar, H. Rashaand and T. Nicolae, "*Design and implementation of real time tracking system based on Arduino Intel Galileo*," 2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Jun. 2016.
- [7] T. Agrawal and M. Qadeer, "*Tracing Path with Arduino Uno using GPS and GPRS/GSM*," 2018 International Conference on Computing, Power and Communication Technologies (GUCON), Sep. 2018.
- [8] A. ElShafee, M. Menshawi and M. Saeed, "*Integrating Social Network Services with Vehicle Tracking Technologies*," International Journal of Computer Applications, Jun. 2013.

# Chapter 6

## Appendix

All codes are available on my GitHub: RealTimeTrackerSystem

1. The implemented code on Arduino board

```
1  /*****
2  * Bsc Final Project
3  * Real Time Tracking system
4  * Sareh Soltani Nejad, Email: sare.soltani74@gmail.com / sare1996@aut.ac.ir
5  * 13/june/2019
6  *****/
7
8  #include<SoftwareSerial.h>
9  extern uint8_t SmallFont[];
10
11 #define rxPin 11
12 #define txPin 10
13
14 SoftwareSerial mySerial(txPin, rxPin);
15
16 const char url[] = "http://103.216.62.79/api/device/add/";
17 char response[200];
18 char latitude[15];
19 char longitude[15];
20 char altitude[16];
21 char date[24];
22 char TTF[3];
23 char satellites[3];
24 char speedOTG[10];
25 char course[15];
26
27 void setup(){
28     mySerial.begin(9600);
29     Serial.begin(9600);
30     Serial.println("Starting...");
31     power_on();
```

```

32 start_GPS();
33 while (sendATcommand("AT+CREG?", "+CREG: 0,1", 2000) == 0);
34 sendATcommand("AT+SAPBR=3,1,\"Contype\",\"GPRS\"", "OK", 2000);
35 sendATcommand("AT+SAPBR=3,1,\"APN\",\"mtairancell\"", "OK", 2000);
36 sendATcommand("AT+SAPBR=3,1,\"USER\",\"\", \"OK\", 2000);
37 sendATcommand("AT+SAPBR=3,1,\"PWD\",\"\", \"OK\", 2000);
38 // gets the GPRS bearer
39 sendATcommand("AT+SAPBR=0,1", "OK", 20000);
40 while (sendATcommand("AT+SAPBR=1,1", "OK", 20000) == 0){
41     delay(5000);
42 }
43 }
44
45 void loop(){
46     // gets GPS data
47     get_GPS();
48     // sends GPS data to the script
49     send_HTTP();
50     //sendNMEAlocation("989164450465",frame);
51     delay(12000);
52 }
53
54 void power_on(){
55     uint8_t answer=0;
56     // checks if the module is started
57     answer = sendATcommand("AT", "OK", 2000);
58     if (answer == 0){
59         // waits for an answer from the module
60         while(answer == 0){
61             // Send AT every two seconds and wait for the answer
62             answer = sendATcommand("AT", "OK", 2000);
63         }
64     }

```

```

65     Serial.println("Power on");
66 }
67
68 int8_t start_GPS(){
69     // starts the GPS
70     while (sendATcommand("AT+CGNSPWR=1", "OK", 2000)==0);
71     while (sendATcommand("AT+CGPSRST=0", "OK", 2000)==0);
72     // waits for fix GPS
73     while(((
74         sendATcommand("AT+CGPSSTATUS?", "2D Fix", 5000) ||
75         sendATcommand("AT+CGPSSTATUS?", "3D Fix", 5000)) == 0 ) );
76     return 1;
77 }
78
79 int8_t get_GPS(){
80     int8_t answer;
81     char * auxChar;
82     // request Basic string
83     sendATcommand("AT+CGPSINF=0", "OK", 8000);
84     auxChar = strstr(response, "+CGPSINF:");
85     if (auxChar != NULL){
86         // Parses the string
87         memset(longitude, '\0', 15);
88         memset(latitude, '\0', 15);
89         memset(altitude, '\0', 16);
90         memset(date, '\0', 24);
91         memset(TTFF, '\0', 3);
92         memset(satellites, '\0', 3);
93         memset(speedOTG, '\0', 10);
94         memset(course, '\0', 15);
95         strcpy (response, auxChar);
96         Serial.println(response);
97     }

```

```

98     strtok(response, ",");
99     strcpy(latitude, strtok(NULL, ",")); // Gets longitude
100    strcpy(longitude, strtok(NULL, ",")); // Gets latitude
101    strcpy(altitude, strtok(NULL, ",")); // Gets altitude
102    strcpy(date, strtok(NULL, ",")); // Gets date
103    strcpy(TTFF, strtok(NULL, ","));
104    strcpy(satellites, strtok(NULL, ",")); // Gets satellites
105    strcpy(speedOTG, strtok(NULL, ",")); // Gets speed over ground. Unit is knots.
106    strcpy(course, strtok(NULL, "\r")); // Gets course
107    convert2Degrees(latitude);
108    convert2Degrees(longitude);
109    answer = 1;
110 }
111 else
112     answer = 0;
113 return answer;
114 }
115
116 void sendNMEALocation(char * cellPhoneNumber, char * message) {
117     char ctrlZString[2];
118     char sendSMSString[100];
119     // Started sendNMEALocation.
120     memset(ctrlZString, '\0', 2);
121     ctrlZString[0] = 26;
122     memset(sendSMSString, '\0', 100);
123     sprintf(sendSMSString, "AT+CMGS=\"%s\"", cellPhoneNumber);
124     // request Basic string
125     sendATcommand(sendSMSString, ">", 2000);
126     mySerial.println(message);
127     sendATcommand(ctrlZString, "OK", 6000);
128     //Ended sendNMEALocation.
129 }

```

```

131 int8_t send_HTTP() {
132     int8_t answer;
133     char aux_str[200];
134     char frame[200];
135     // Initializes HTTP service
136     answer = sendATcommand("AT+HTTPIPINIT", "OK", 10000);
137     if (answer == 1) {
138         // Sets CID parameter
139         answer = sendATcommand("AT+HTTTPARA=\"CID\",1", "OK", 5000);
140         if (answer == 1) {
141             // Sets url
142             memset(aux_str, '\0', 200);
143             sprintf(aux_str, "AT+HTTTPARA=\"URL\", \"%s\", url);
144             mySerial.print(aux_str);
145             Serial.println(aux_str);
146             sprintf(frame, "?visor=false&lat=%s&lon=%s&alt=%s&date=%s&TTFF=%s&sat=%s&speedOTG=%s&course=%s"
147             , latitude, longitude, altitude, date, TTFF, satellites, speedOTG, course);
148             Serial.println(frame);
149             mySerial.print(frame);
150             answer = sendATcommand("", "OK", 5000);
151             if (answer == 1) {
152                 // Starts GET action
153                 answer = sendATcommand("AT+HTTTPACTION=0", "+HTTTPACTION: 0,200", 30000);
154                 if (answer == 1) {
155                     Serial.println(F("Done!"));
156                 }
157                 else {
158                     Serial.println(F("Error getting url"));
159                 }
160             }
161             else {
162                 Serial.println(F("Error setting the url"));
163             }

```

```

164     }
165     else{
166         Serial.println(F("Error setting the CID"));
167     }
168 }
169 else{
170     Serial.println(F("Error initializing"));
171 }
172
173 sendATcommand("AT+HTTPTERM", "OK", 5000);
174 return answer;
175 }
176
177 int8_t sendATcommand(char* ATcommand, char* expected_answer1, unsigned int timeout){
178     uint8_t x=0, answer=0;
179     unsigned long previous;
180     char readVar[200];
181     char * auxChar;
182
183     memset(response, '\0', 200); // Initialize the string
184     memset(readVar, '\0', 200); // Initialize the string
185
186     while( mySerial.available() > 0) mySerial.read(); // Clean the input buffer
187     while( Serial.available() > 0) Serial.read(); // Clean the input buffer
188
189     mySerial.write(ATcommand); // Send the AT command
190     mySerial.write("\r\n\r\n"); // Send enter
191
192     Serial.println(ATcommand);
193
194     x = 0;
195     previous = millis();

```

```

196 // this loop waits for the answer
197 do{
198     if(mySerial.available() != 0){
199         readVar[x] = mySerial.read();
200         x++;
201         // check if the desired answer is in the response of the module
202         auxChar = strstr(readVar, expected_answer1);
203         if (auxChar != NULL){
204             if( strstr(readVar, "+CGPSINF:") == NULL)
205                 strcpy (response, auxChar);
206             else
207                 strcpy (response, readVar);
208             answer = 1;
209         }
210     }
211     // Waits for the asnwer with time out
212 }
213 while((answer == 0) && ((millis() - previous) < timeout));
214 if(auxChar == NULL)
215     Serial.println(readVar);
216 return answer;
217 }
218
219 void convert2Degrees(char* input){
220     char res[40];
221     float deg;
222     float minutes;
223     float seconds;
224     char aux[10] = "";
225     //latitude format: DDmm.mmmmm'
226
227     // get 'degrees' from input parameter

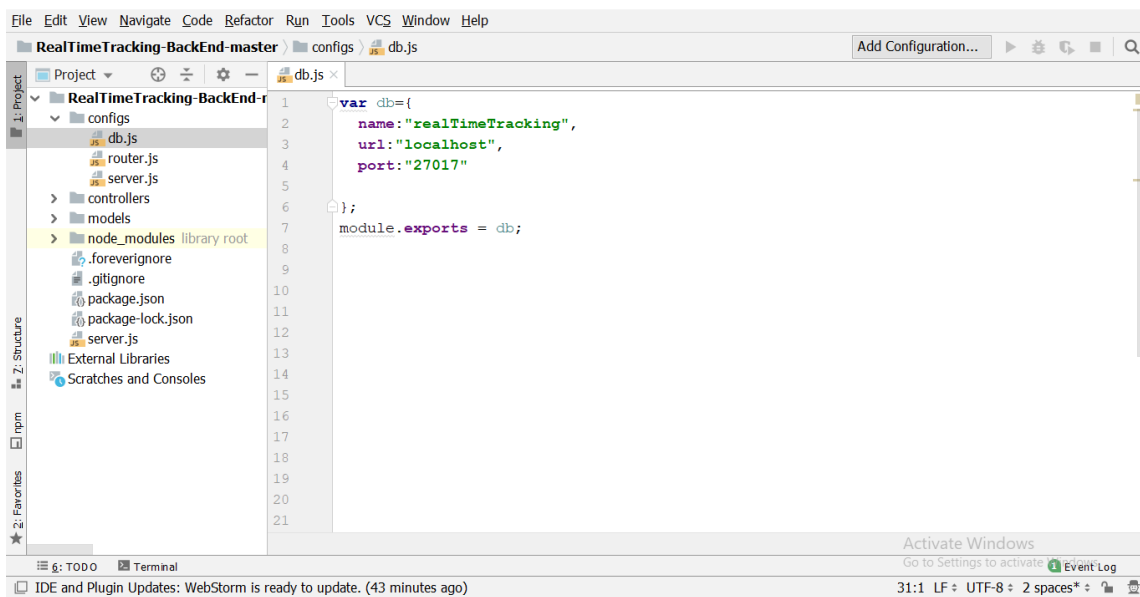
```

```

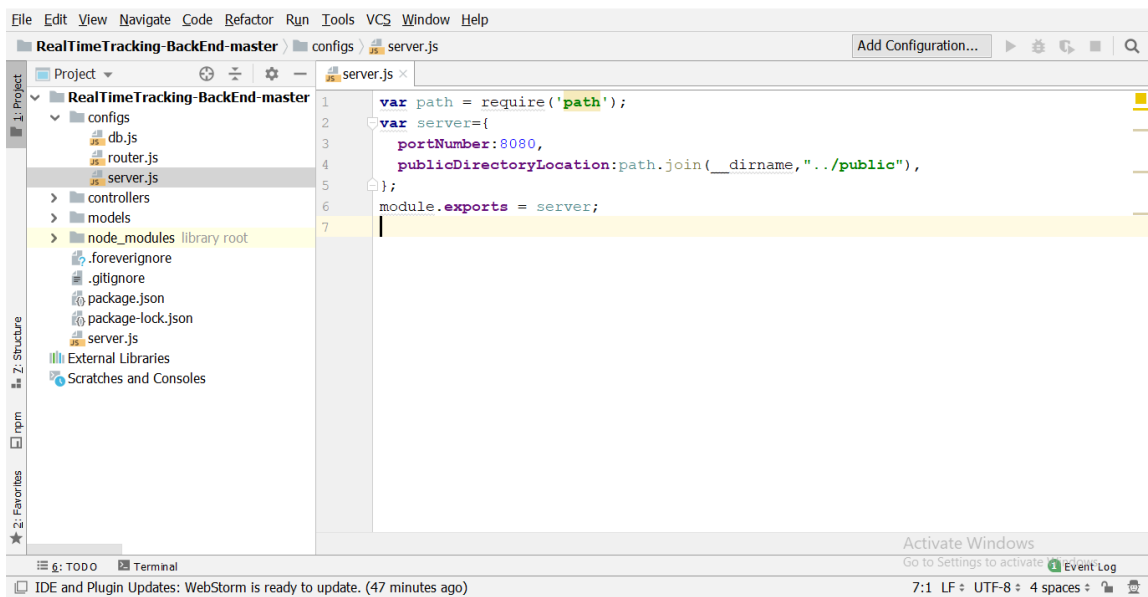
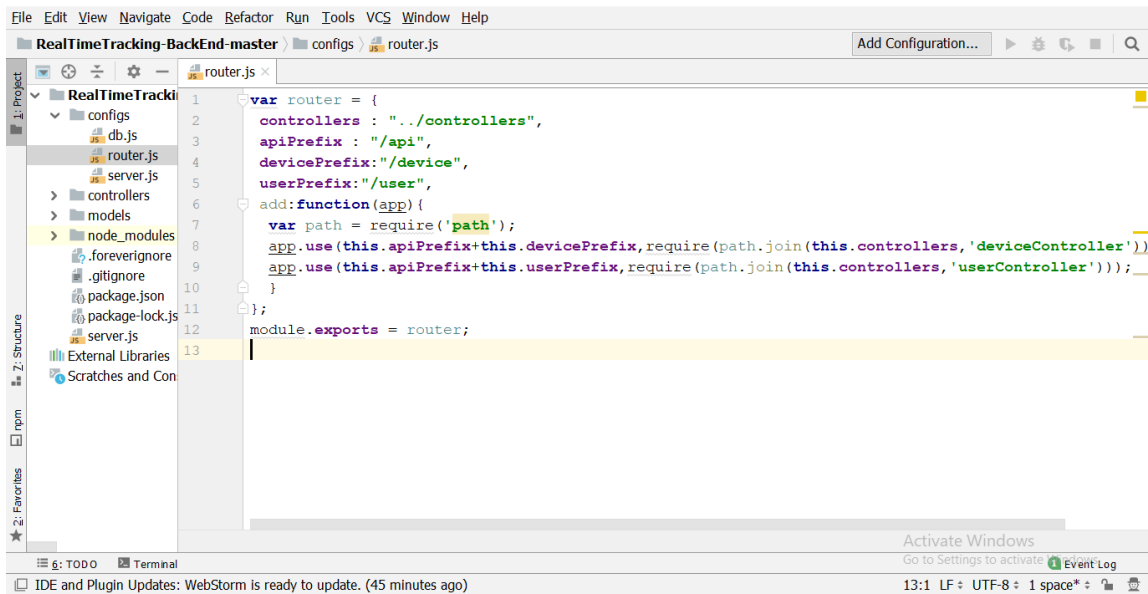
227 // get 'degrees' from input parameter
228 aux[0] = input[0];
229 aux[1] = input[1];
230 aux[2] = '\0';
231
232 // convert string to integer and add it to final float variable
233 deg = atoi(aux);
234 // get 'minutes' from input parameter
235 for ( int i=0; i<7; i++){
236     aux[i] = input[i+2];
237 }
238 aux[7] = '\0';
239 // convert string to integer and add it to final float variable
240 minutes = atoi(aux);
241
242 // get 'seconds' from input parameter
243 for ( int i=0; i<2; i++){
244     aux[i] = aux[i+3];
245 }
246 aux[3] = '\0';
247 seconds = atoi(aux);
248
249 // add minutes to degrees
250 deg = deg + minutes/60 + seconds/3600;
251 dtostrf(deg, 2, 10, res);
252 strncpy(input, res,9);
253 }

```

## 2. Back-end development







The screenshot shows the VS Code editor interface with the project 'RealTimeTracking-BackEnd-master' open. The file explorer on the left shows the project structure, including 'controllers' and 'models' folders. The 'deviceController.js' file is selected in the 'controllers' folder. The code in the editor is as follows:

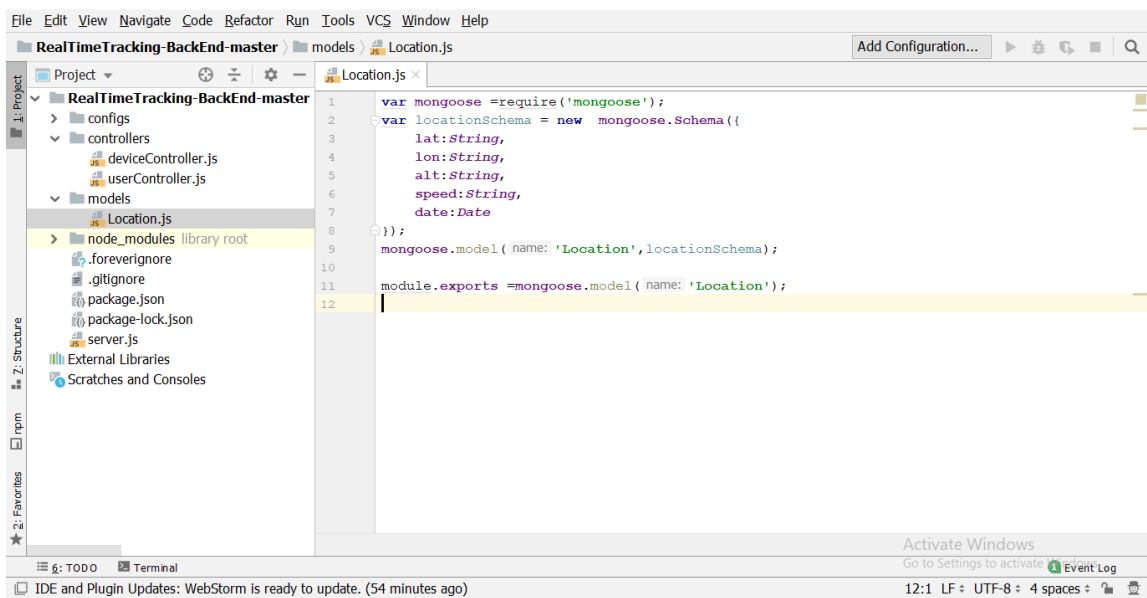
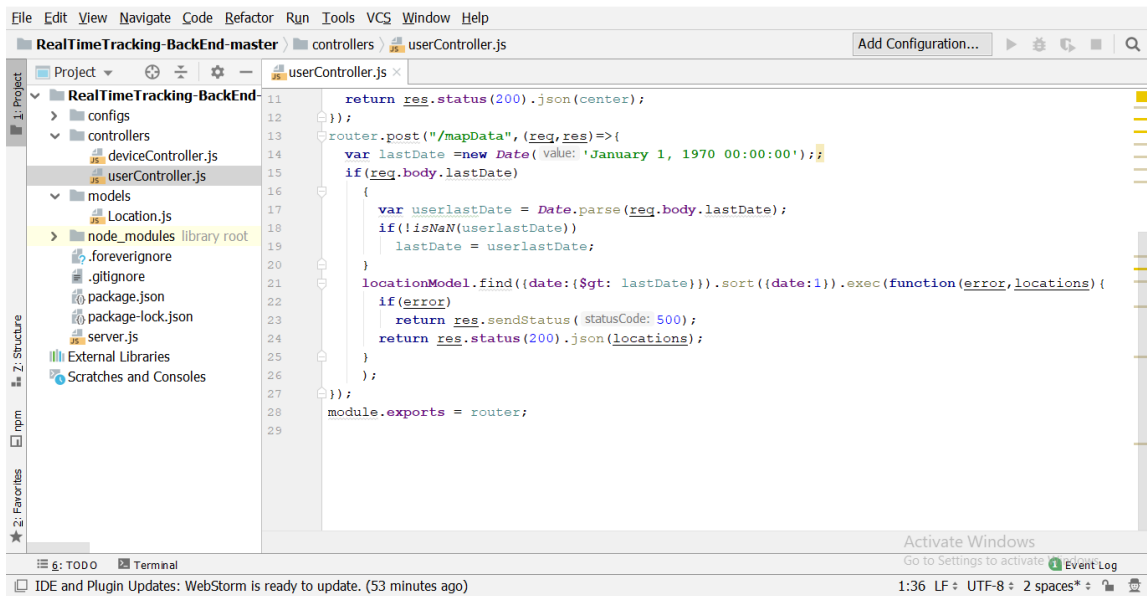
```
1 const express = require('express');
2 const fs = require('fs');
3 const path = require('path');
4 const moment = require('moment');
5 var locationModel = require('../models/Location');
6 var router = express.Router();
7 router.get("/add", (req, res) => {
8   if (!req.query.lat || !req.query.lon || !req.query.alt || !req.query.date || !req.query.speedOTG)
9     return res.sendStatus( statusCode: 400 );
10   var date = moment( req.query.date, "YYYYMMDDHHmmss.SSS" ).add( amount: 4.5, unit: 'h' );
11   if (!date.isValid())
12     return res.sendStatus( statusCode: 400 );
13   locationModel.create({
14     lat: req.query.lat,
15     lon: req.query.lon,
16     alt: req.query.alt,
17     speed: req.query.speedOTG * 1.852,
18     date: date.format()
19   }, function( err, location ) {
20     if( err )
21       return res.sendStatus( statusCode: 500 );
22     return res.sendStatus( statusCode: 200 );
23   });
24 });
```

The status bar at the bottom indicates '26:1 LF UTF-8 2 spaces'.

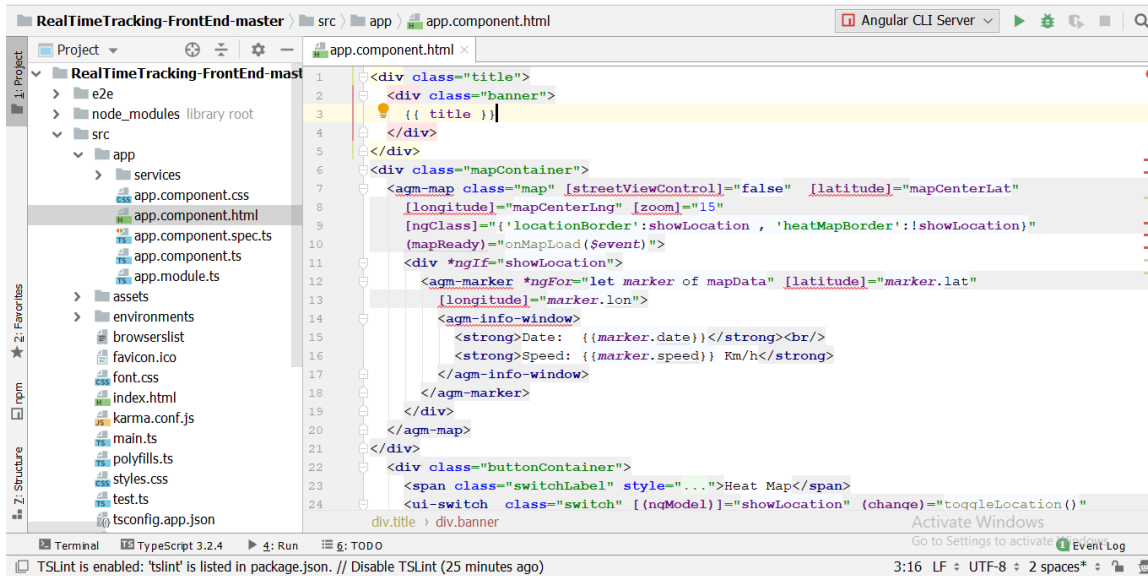
The screenshot shows the VS Code editor interface with the project 'RealTimeTracking-BackEnd-master' open. The file explorer on the left shows the project structure, including 'controllers' and 'models' folders. The 'userController.js' file is selected in the 'controllers' folder. The code in the editor is as follows:

```
1 const express = require('express');
2 const fs = require('fs');
3 const path = require('path');
4 var locationModel = require('../models/Location');
5 var router = express.Router();
6 router.post("/mapCenter", (req, res) => {
7   var center = {
8     lat: 35.706421,
9     lon: 51.426771
10  };
11  return res.status(200).json(center);
12 });
13 router.post("/mapData", (req, res) => {
14   var lastDate = new Date( value: 'January 1, 1970 00:00:00' );
15   if( req.body.lastDate )
16   {
17     var userlastDate = Date.parse( req.body.lastDate );
18     if( !isNaN( userlastDate ) )
19       lastDate = userlastDate;
20   }
21   locationModel.find( { date: { $gt: lastDate } } ).sort( { date: 1 } ).exec( function( error, locations ) {
22     if( error )
23       return res.sendStatus( statusCode: 500 );
24     return res.status(200).json(locations);
25   });
26 });
```

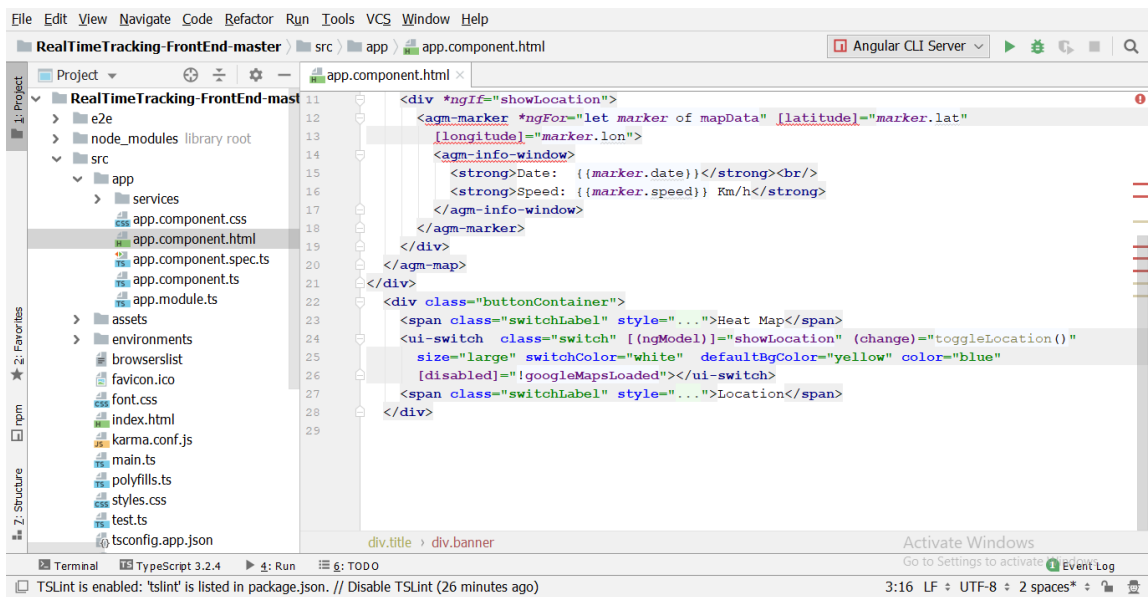
The status bar at the bottom indicates '1:36 LF UTF-8 2 spaces'.



### 3. Front-end development



```
1 <div class="title">
2   <div class="banner">
3     {{ title }}
4   </div>
5 </div>
6 <div class="mapContainer">
7   <agm-map class="map" [streetViewControl]="false" [latitude]="mapCenterLat"
8     [longitude]="mapCenterLng" [zoom]="15"
9     [ngClass]="{'locationBorder': showLocation, 'heatMapBorder': !showLocation}"
10    (mapReady)="onMapLoad($event)">
11     <div *ngIf="showLocation">
12       <agm-marker *ngFor="let marker of mapData" [latitude]="marker.lat"
13         [longitude]="marker.lon">
14         <agm-info-window>
15           <strong>Date: {{marker.date}}</strong><br/>
16           <strong>Speed: {{marker.speed}} Km/h</strong>
17         </agm-info-window>
18       </agm-marker>
19     </div>
20   </agm-map>
21 </div>
22 <div class="buttonContainer">
23   <span class="switchLabel" style="...">Heat Map</span>
24   <ui-switch class="switch" [(ngModel)]="showLocation" (change)="toggleLocation()"
25     [disabled]="!googleMapsLoaded">
26     <div>
27       <span class="switchLabel" style="...">Location</span>
28     </div>
29   </ui-switch>
30 </div>
```



```
11 <div *ngIf="showLocation">
12   <agm-marker *ngFor="let marker of mapData" [latitude]="marker.lat"
13     [longitude]="marker.lon">
14     <agm-info-window>
15       <strong>Date: {{marker.date}}</strong><br/>
16       <strong>Speed: {{marker.speed}} Km/h</strong>
17     </agm-info-window>
18   </agm-marker>
19 </div>
20 </agm-map>
21 </div>
22 <div class="buttonContainer">
23   <span class="switchLabel" style="...">Heat Map</span>
24   <ui-switch class="switch" [(ngModel)]="showLocation" (change)="toggleLocation()"
25     size="large" switchColor="white" defaultBgColor="yellow" color="blue"
26     [disabled]="!googleMapsLoaded">
27     <div>
28       <span class="switchLabel" style="...">Location</span>
29     </div>
30   </ui-switch>
31 </div>
```

The screenshot shows the Visual Studio Code editor with the project 'RealTimeTracking-FrontEnd-master' open. The file explorer on the left shows the project structure, with 'data-api.service.ts' selected under 'src/app/services/data-api/'. The editor displays the following TypeScript code:

```
1 import { Injectable } from '@angular/core';
2 import { HttpClient, HttpHeaders } from '@angular/common/http';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class DataApiService {
8   prefix:string = "/api/user"
9   constructor(private http: HttpClient) {
10
11   }
12   getMapCenter()
13   {
14     return this.http.post<any>(url: this.prefix+"/mapCenter", body: null);
15   }
16   getMapData(date)
17   {
18     return this.http.post<any>(url: this.prefix+"/mapData", body: {lastDate:date});
19   }
20 }
21
```

The status bar at the bottom indicates 'TypeScript 3.2.4', 'Run', '6: TODO', and '1:1 LF UTF-8 2 spaces\*'. A message at the bottom states 'TSLint is enabled: 'tslint' is listed in package.json. // Disable TSLint (29 minutes ago)'.

The screenshot shows the Visual Studio Code editor with the project 'RealTimeTracking-FrontEnd-master' open. The file explorer on the left shows the project structure, with 'index.html' selected under 'src/app/assets/'. The editor displays the following HTML code:

```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>RealTimeTracking</title>
6   <base href="/">
7
8   <meta name="viewport" content="width=device-width, initial-scale=1">
9   <link rel="icon" type="image/x-icon" href="favicon.ico">
10 </head>
11 <body>
12   <app-root></app-root>
13 </body>
14 </html>
15
```

The status bar at the bottom indicates 'TypeScript 3.2.4', 'Run', '6: TODO', and '15:1 LF UTF-8 2 spaces\*'. A message at the bottom states 'TSLint is enabled: 'tslint' is listed in package.json. // Disable TSLint (35 minutes ago)'.

#### 4. The implemented code for sending and receiving an SMS

```
tracking-v2 $
1 #include <sim808.h>
2 #include <DFRobot_sim808.h>
3 #include <SoftwareSerial.h>
4 #define MESSAGE_LENGTH 160
5 char message[MESSAGE_LENGTH];
6 int messageIndex = 0;
7 char MESSAGE[300];
8 char lat[12];
9 char lon[12];
10 char wspeed[12];
11 char phone[16];
12 char datetime[24];
13
14 #define PIN_TX 10
15 #define PIN_RX 11
16
17 SoftwareSerial mySerial(PIN_TX,PIN_RX);
18 DFRobot_SIM808 sim808(&mySerial);//Connect RX,TX,PWR,
19
20 void setup(){
21   mySerial.begin(9600);
22   Serial.begin(9600);
23   //***** Initialize sim808 module *****
24   while(!sim808.init()){
```

```
25     Serial.print("Sim808 init error\r\n");
26     delay(1000);
27   }
28   delay(3000);
29   if( sim808.attachGPS())
30     Serial.println("Open the GPS power success");
31   else
32     Serial.println("Open the GPS power failure");
33   Serial.println("Init Success, please send SMS message to me!");
34 }
35
36 void loop(){
37   //***** Detecting unread SMS *****
38   messageIndex = sim808.isSMSUnread();
39   //***** At least, there is one UNREAD SMS *****
40   if (messageIndex > 0){
41     Serial.print("messageIndex: ");
42     Serial.println(messageIndex);
43     sim808.readSMS(messageIndex, message, MESSAGE_LENGTH, phone, datetime);
44     //*****In order not to full SIM Memory, is better to delete it*****
45     sim808.deleteSMS(messageIndex);
46     Serial.print("From number: ");
47     Serial.println(phone);
48     Serial.print("Datetime: ");
```

```
tracking-v2 $
48 Serial.print("Datetime: ");
49 Serial.println(datetime);
50 Serial.print("Recieved Message: ");
51 Serial.println(message);
52 while(!sim808.getGPS()){}
53 Serial.print(sim808.GPSdata.year);
54 Serial.print("/");
55 Serial.print(sim808.GPSdata.month);
56 Serial.print("/");
57 Serial.print(sim808.GPSdata.day);
58 Serial.print(" ");
59 Serial.print(sim808.GPSdata.hour);
60 Serial.print(":");
61 Serial.print(sim808.GPSdata.minute);
62 Serial.print(":");
63 Serial.print(sim808.GPSdata.second);
64 Serial.print(":");
65 Serial.println(sim808.GPSdata.centisecond);
66 Serial.print("latitude :");
67 Serial.println(sim808.GPSdata.lat);
68 Serial.print("longitude :");
69 Serial.println(sim808.GPSdata.lon);
70 Serial.print("speed_kph :");
71 Serial.println(sim808.GPSdata.speed_kph);
```

```
tracking-v2 $
71 Serial.println(sim808.GPSdata.speed_kph);
72 Serial.print("heading :");
73 Serial.println(sim808.GPSdata.heading);
74 Serial.println();
75
76 float la = sim808.GPSdata.lat;
77 float lo = sim808.GPSdata.lon;
78 float ws = sim808.GPSdata.speed_kph;
79
80 dtostrf(la, 6, 2, lat); //put float value of la into char array of lat
81 dtostrf(lo, 6, 2, lon); //put float value of lo into char array of lon
82 dtostrf(ws, 6, 2, wspeed); //put float value of ws into char array of wspeed
83
84 sprintf(MESSAGE, "Latitude : %s\nLongitude : %s\nWind Speed : %s kph\nYou can see your device in this location:\nhttp://maps.google.com/maps?q=%s,%s\n",
85 lat, lon, wspeed, lat, lon);
86 Serial.println("Sim808 init success");
87 Serial.println("Start to send message ...");
88 Serial.println(MESSAGE);
89 Serial.println(phone);
90 sim808.sendSMS(phone,MESSAGE);
91 //***** Turn off the GPS power *****
92 sim808.detachGPS();
93 }
94 }
```