# Car alarm project with GPS tracker and Windows software


University of Isfahan - Faculty of Engineering - Department of Electrical Engineering

Embedded Systems Class

Professor Masih Mohammad Shafiee

Members:

Amir Zeinali

Seyed Hossein Mortazavi

Semester: February – July 2025

# Table of Contents:



Note:

This project documentation was originally written in Persian. Due to the lack of time, AI translators have been used. There might be mistakes in using right words or grammar.

## Abstract:

In this project, which is the final project of the Embedded Systems Class, our effort was to design a system that has more intelligent capabilities than a regular alarm and also challenges the abilities of the group members. We completed this project using the basics we learned in class, what we knew, and the research we did. The distinguishing features of the project are; the ability to automatically open or lock the door, satellite positioning of the vehicle, alarm notification via SMS or call, remote control of the vehicle's location using software, and determining a safety zone for it.

## Introduction:

The project of the group in the Embedded Systems Class, Car Alarm and Smart Satellite Tracker, was actually chosen because it was both novel in a way at the university and sufficiently challenging. The ultimate goal of this project was to build a board for overall car control and especially its tracking using a software. The structure of this report is as follows: in the system design section, there are block diagrams, images, electronic components and modules, and explanations regarding the selection of components and modules and how the system works are given in this section. In the software design section, there are images and explanations regarding the Windows software that was designed for this project, and in the conclusion section, the experiences that this project added to the group, as well as the results and areas for further development of the project, are given. In the appendix section, the code of the electronic part and software of the project is also included.

## 1 - System design:

The overall block diagram of the system is as follows, and the entire project actually consists of 2 boards, images of which can be seen in images 2 and 3.

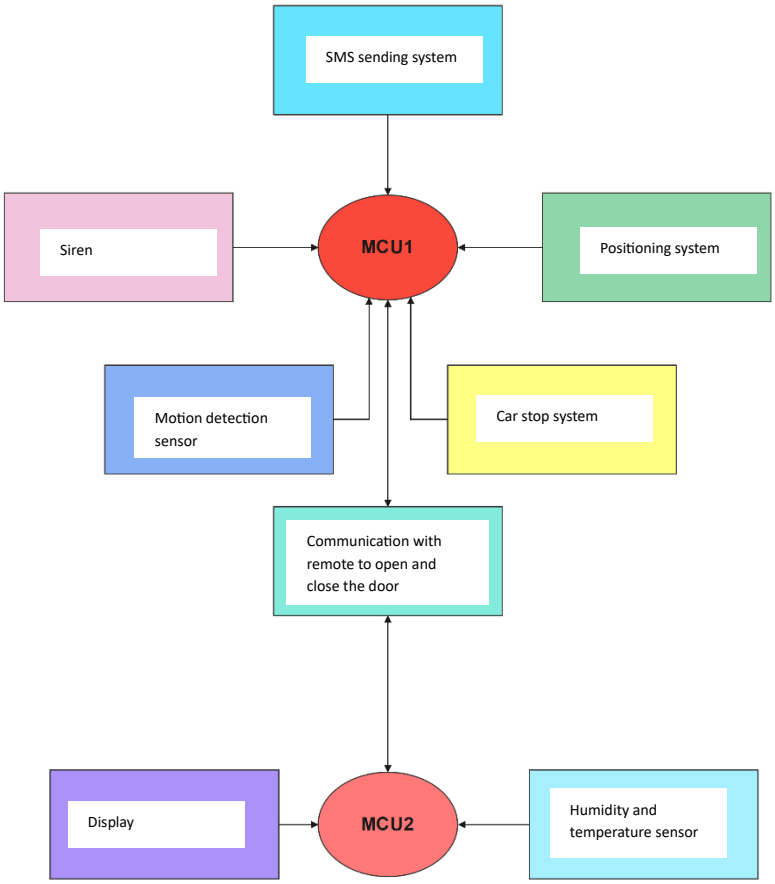



Image 1 – Overall project block diagram

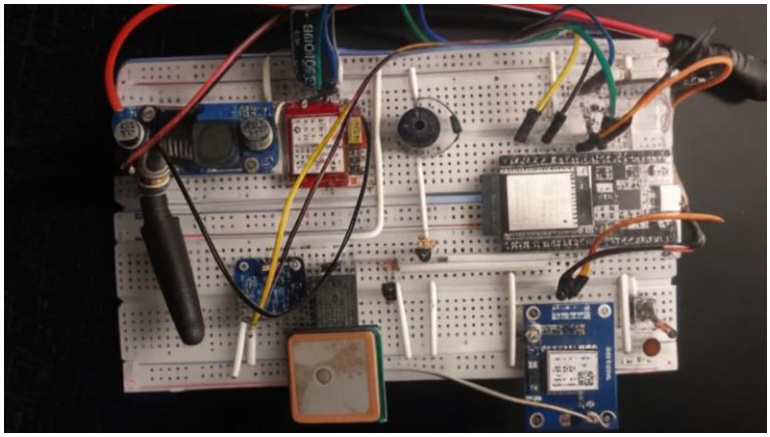
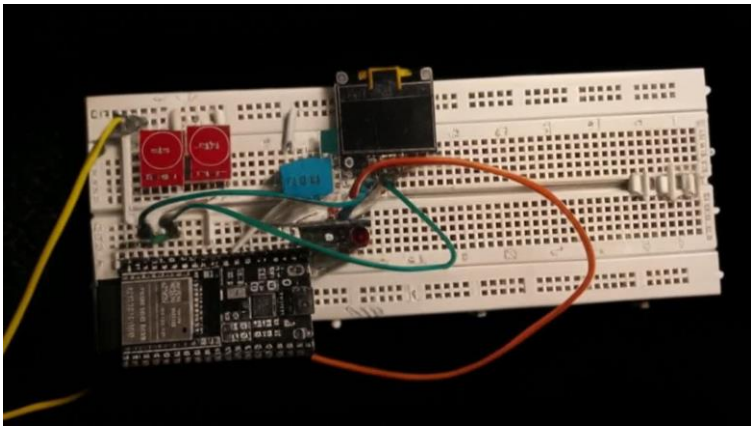


*Image 2*

*The main board designed for the car and includes GPS, GSM and RF receiver.*

*Image 3*

*Remote board to control whether the lock is open or closed and display temperature and humidity sensor information, next to the touch sensor for locking and unlocking.*



## GSM sim800L Module:

In order to communicate with the car owner and send messages such as moving or opening the door or movements in the car to the car owner, the sim800l module was used, which seemed sufficient in this project. Since in this project we do not use its other features such as its microphone and speaker, the hardware required will be two capacitors 2200uf and 100nf for stable and low-noise power supply, a GSM antenna, and two resistors k5.6 and k10 for RX. The reason for using these resistors is to create a resistance division because the GSM receiver does not tolerate 3.3 volts. Also, the GSM power supply is generally separate from the rest of the circuit and is provided with the lm2596 module because GSM requires a voltage range between 4.4 and 3.7, otherwise it will either burn out or not turn on. Therefore, in order for the GSM module and MCU to communicate with each other with a correct voltage reference, it is necessary for the ground of both circuits to be connected to each other so that communication can be established between the two. In the attachment where the esp code is given, the following are explanations about the functions related to gsm. bool sendATCommand() A general function for sending AT commands to the SIM800L module and checking the responses. bool sendSMS() Sends an SMS to the car owner's number using sendATCommand(). The messages are related to intrusion or movement

alerts. makeCall() Makes a voice call to the car owner if the car's movement is detected in alarm mode.

### http connection to connect to the server via GSM:

Used WIFI to send its information to the server, unless there is a nationwide local internet connection, which unfortunately is not available in Iran. Therefore, the aforementioned connection must be established through the SIM card inside the GSM. The GSM module has the ability to send information via SIM card data using http post. To do this, it is only necessary to send appropriate AT COMMANDS to the GSM before sending this information, which are clearly specified in the sendThingSpeakPost() function block in the appendix to these commands. For example, ("AT+SAPBR=3,1,\"Contype\",\"GPRS\"") This command sets the connection type to GPRS, or the command ("AT+HTTPINIT") starts the http connection. An important issue is that for the code to work properly, you must close this opened connection after sending information, because in many cases, operator companies automatically deactivate these types of connections if they do not see any activity, and we will encounter problems for subsequent transmissions. By executing this command, the entire process is closed ("AT+SAPBR=0,1").

Regarding the server, it should be said that this project uses the free but limited Thingspeak student server from Mathworks. To send information along with the server's sending code, we send (Write api key) and to receive the corresponding code, we send (Get api key) to the server. Also, to specify the server URL for GSM, we must use this command ("AT+HTTPPARA=\"URL\",\"http://api.thingspeak.com/update\""), where http://api.thingspeak.com/update"" is the server address.

### ublox neo 6m GPS module:

To communicate with positioning satellites, a special chip and antenna are required. We used the ublox neo 6m module in the project, which has both a chip and a passive ceramic antenna to establish this connection. The reason for choosing this module was its more affordable price for a student project and its popularity in the market. This module has very good accuracy in determining its own location, and according to the datasheet of its manufacturer, its positioning error is 2 to 2.5 meters, which is very good for our project. Another thing about this module is that the ublox neo 6m only supports American GPS positioning satellites and is designed to communicate with them, while another module from the same brand, the ublox neo m8m, can communicate with multiple positioning systems simultaneously, for example, American GPS and Russian GLONASS, and as a result, it can position more accurately. UART communication was used to connect the module to the ESP32 board. As shown in the attachment, the RX module is connected to pin 4 of the ESP32 board and its TX to pin 5, and a Baud rate of 9600 Hz is used for communication. After

establishing the UART connection and supplying the required 3.3 V power, the module starts blinking when it establishes communication with the satellites and then continuously sends the relevant information. This information is in NMEA format, which is the American maritime standard and is used by all positioning systems. To translate this information, a library called tinygpsplus.h is used in Arduino, which can be used according to the gpsmodule() function block in the attachment. Lat and lng, speed, accuracy, number of available satellites, height above ground and many other information that GPS gives us are translated by this library and displayed in the software, which is explained in more detail below. In the mentioned function, we use the SIM card inside the GSM to communicate with the Thingspeak by Matlab server, and then this information is taken from the server in the software and displayed to the user every 20 seconds. This is explained in more detail in the section "http communication for communicating with the server with GSM".

## BLE (Bluetooth Low Energy):

To open the car door automatically, BLE is used, which the ESP32 module itself has, because it has a more limited range and lower energy than classic Bluetooth. Regarding the relevant codes, the appendix contains; class MyAdvertisedDeviceCallbacks BLE callback to detect the presence of the car owner's BLE device (based on RSSI). If the signal strength is sufficient, it is assumed that the owner is nearby. The handleBLELocking() function block is for this: if the BLE device is nearby, the door is unlocked, and if it moves away and disappears for a few seconds, the door is locked. All alarm variables are also reset. handleBLELocking() This function manages the BLE presence status: if BLE is detected (i.e. the car owner is nearby) and the door is currently locked, the door opening command is executed. If BLE disappears for a few seconds, and the door is currently open, the door is locked. Also, alarms and sirens are silenced so that the owner's entry into the car does not cause the alarm to remain active. In general, the BLEDevice.h library is used to perform these steps and all these communications.

## RF transmitter and receiver:

To open the door using RF, ASK transmitter and receiver modules are commonly used, both of which are powered by 5 volts and do not require additional hardware. uint8_t generateToken() This function block generates a simple security token in the attached code to make the RF communication with the remote more secure. The message and the secret key are combined as XOR to obtain a hash number. The same function is also used in the receiver to ensure that the sent token is correct and correct. In the RF remote section, the sendSecureCommand() function block is responsible for sending the RF command. First, a security token is generated using generateToken, then a packet is created as cmd:token and

sent via RH_ASK to the 315MHz RF ASK module. Finally, the value of the sent command is stored in the lastAction variable to be used for display on the OLED.

## Other elements used:

To simulate the opening and closing of the car door, a push button was used, which used an nf100 capacitor to debounce it, and one side of it was pulled up with a resistor so that the key would not float. Also, to simulate the siren, a passive buzzer was used because different frequencies can be produced with it. However, since it has a self-load, although small, a Schottky diode was needed to neutralize the effect of the self-load and prevent damage to the circuit. The beepDouble() and beepOnce() function blocks are related to generating the buzzer sound. A relay was also used to simulate the car's cut-off relay. This relay cuts off the car's starter and after the car stops, if the car relay is active, it will not turn on again. This is a security option that is activated by sending an SMS. To activate the relay directly from the microcontroller, you must use a c945 transistor, the base of which is excited through a 4.7k resistor, and one of the relay coil terminals is connected to 5 volts, and since the other coil terminal is connected to ground, the relay is excited. Like the buzzer, the relay also requires a diode at both ends of the coil terminals.

Other features of this project include the remote display, which is a 0.96 inch OLED that simply works with I2C. Sensors have also been used in the project, and we used the dht11 and mpu6050 temperature sensors, which we learned how to work with each earlier during the semester.

To detect the presence of humans or living beings inside the cabin, a PIR sensor that uses infrared light is also used. Its power supply is 5 volts.

## Table of elements used:

| Elements | | |
|---|---|---|
| Ublox neo 6m | GSM SIM800L | ESP32WROOM-32D |
| LM2596 DC-DC CONV. | 315MHz RF Transmitter | 315MHz RF Reciever |
| DHT11 | OLED 0.96" | MPU6050 |
| 2x TTP223 | 5V Relay | Passive Buzzer |
| ..... | Resistors | Push button |

## 2- Software design:

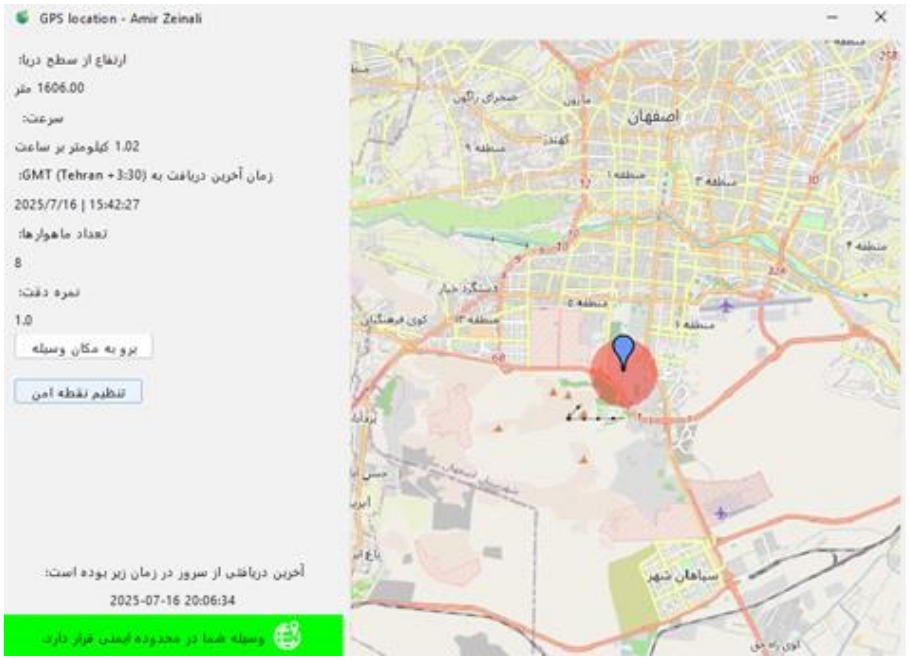



*Image 4*

To make the most of the board's capabilities and also to complete the project, a Windows software was created to display the device's location in this case the car. Normally there are websites that you can give lat and lng or directly raw gps information to and they do not display your location on the map. This may be suitable for a module test but not for a project. This software is designed with the java programming language, more precisely java swing and java fx. The common IDE among Java programmers, Eclipse IDE, is also used for compiling and catching errors.

### Software panel:

The software panel is located on the right side of the software, as shown in Figure 4. This panel contains 5 of the 7 most important pieces of information that the GPS module gives us. Altitude above sea level, speed in kilometers per hour, the last activity sent by the module in UTC Greenwich Mean Time, the number of satellites in the module's view, and the accuracy and strength of the connection. Regarding time, the time of receiving information from the server, i.e. the last request by the software to receive information from the server, is displayed at the bottom of the screen in local time. If the module is always working properly every 20 seconds, these two times will be exactly the same. And regarding accuracy, it should be said that according to the NMEA structure of the information, the closer this value is to 1 and zero, the higher the accuracy, and the closer it is to 99, the lower the accuracy and the ability to connect to satellites has encountered problems. For example, the module is in a very closed environment, which was previously mentioned, which means that this module is not suitable for very precise work in crowded and closed areas, and there is a need to purchase modules or parts with higher prices. Also, a bar has been designed inside the panel, which is currently green in the image above, and it shows whether the device is within the specified safety zone or not (red circle on the map), which is referred to in the settings below. There are also two buttons in the panel that, by clicking on (Go to

device location), if we have moved on the map, we can return to the location of the device or view the device with a greater zoom. However, every 20 seconds, the map position returns to the initial location and default zoom, and for further zooming, like other maps, the mouse must be used. By clicking on the (Set Safe Point) button, the following page opens, through which the safety zone can be determined:

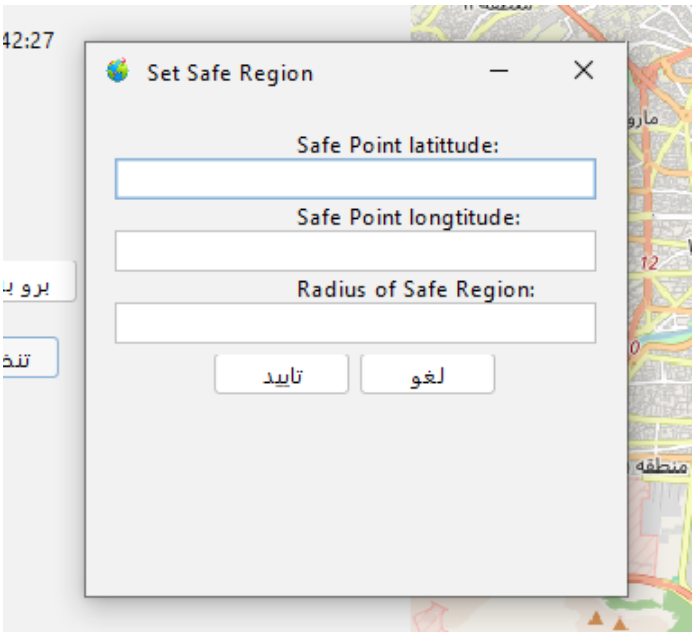



*Figure 5*

*Safepoint Setup Window*

## Setting a safe range for the device:

To do this, after selecting (Set Safe Point), the image above will open, you can select a location, for example, a car park as a safe place, then answer the question to yourself about how much radius is considered safe for that vehicle, for example, it is night and the car must be in the neighborhood. After answering these questions, enter your lat location in the first field, your lng location in the second field, and the radius of this safety limit in meters in the third field. Then click on the (Confirm) option and close the window, the software will check the location of your vehicle in the next round of its own check and obtaining information from the server and will show you the result through the right panel bar of the program mentioned earlier. For example, we search the location of Isfahan's Naqsh-e Jahan Square on the Internet and get its geographical latitude and longitude:

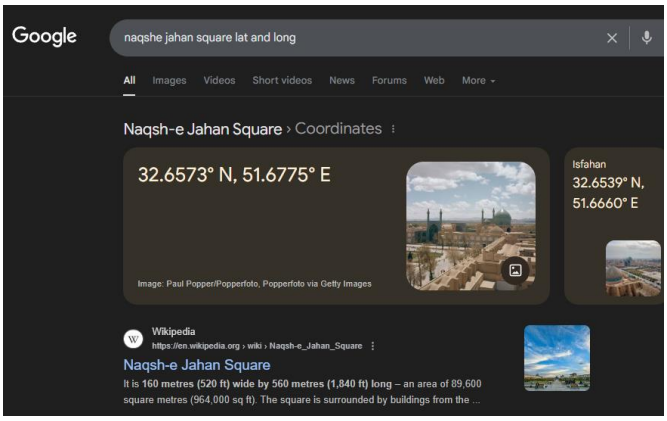



*Image 6: Coordinates of Naqsh-e Jahan Square*

Copy the Lat (longitude) and Lng (latitude) and select a radius of 1000 meters (1 kilometer) for the safety zone as shown:

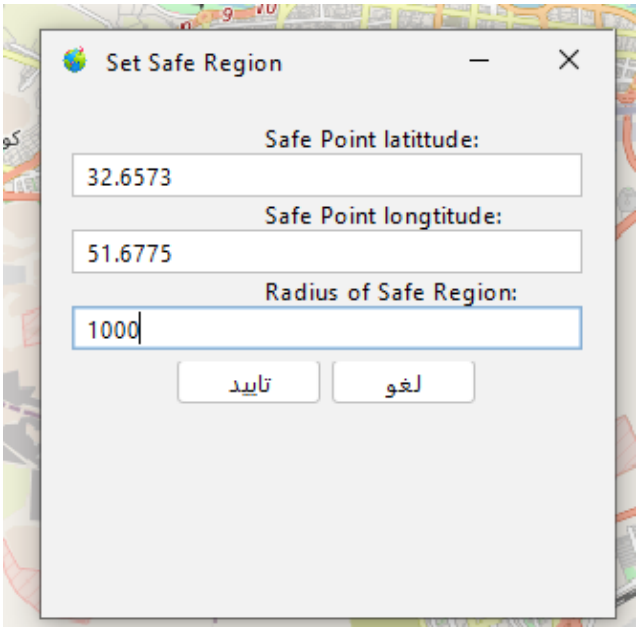



*Image 7*

By clicking on the Confirm button, our settings are saved and the program will show the changes in its message after a maximum of 20 seconds in the next round of checking and retrieving information from the server:

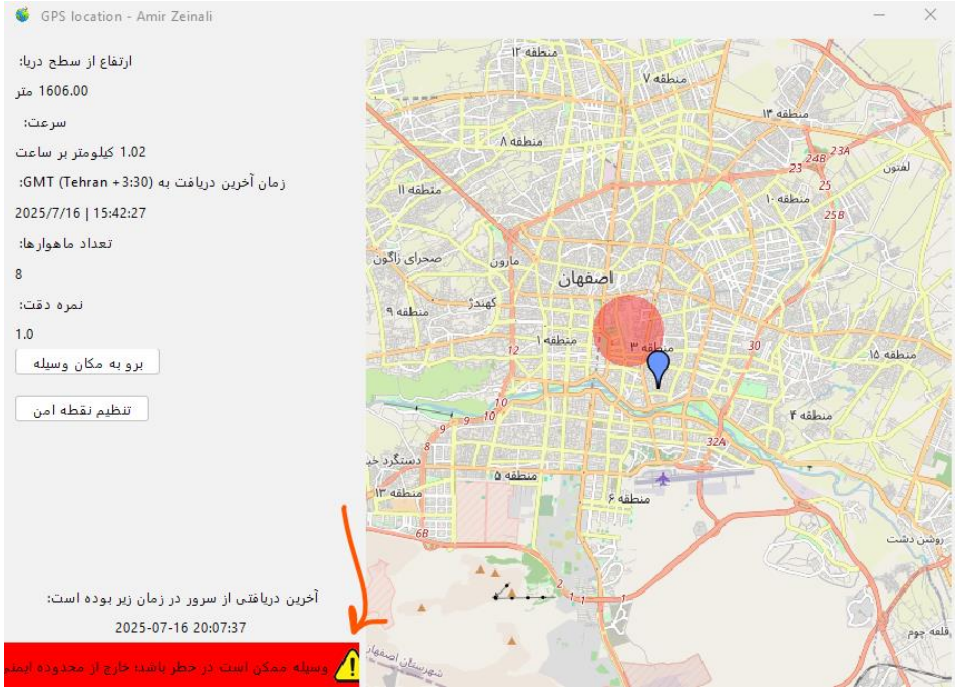



*Figure 8*

*Note the change in the device message at the bottom indicating danger.*

## Haversine's theorem (distance formula on a sphere):

To calculate the distance of a location from the center of the safe point, the Haversine theorem (the formula for distance on a sphere) is used because the maps themselves maintain the spherical structure of the Earth within their identity, even though the maps are displayed flat on the pages:

$$\varphi = lattitude\ , \lambda = longtitude$$

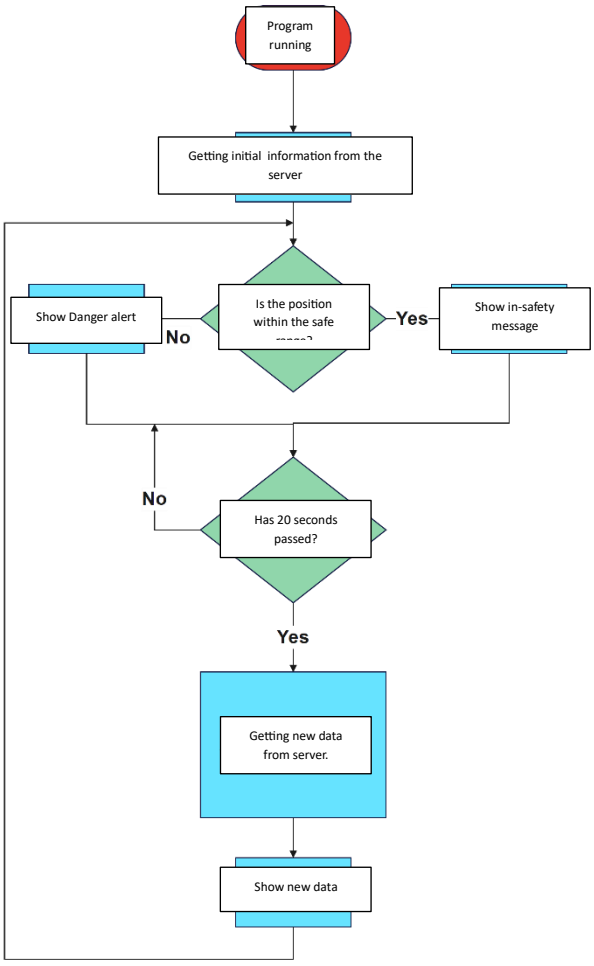
$$\Delta\varphi\ = \varphi_2\ - \varphi_1\ \ \Delta\lambda\ = \lambda_2\ \ - \lambda_1$$

$$a\ =\ \ sin^2\ \left(\frac{\Delta\varphi}{2}\right) + \ cos(\varphi_1) \cdot\ cos(\varphi_2) \cdot\ sin^2\ \ \left(\frac{\Delta\lambda}{2}\right)$$

$$c\ =\ 2\ \cdot arctan2\left(\sqrt{a}, \sqrt{1-a}\right)$$

$$d\ =\ R\ \cdot c$$

## Software operation flowchart:



## 3 – Analysis and Conclusion:

The project was successfully implemented and the GPS, GSM, RF, etc. modules performed the expected tasks well and the information was also received or sent to the server well. But the issues that could help the development of the project are the use of a more powerful module for GPS or an active antenna that gives us higher accuracy. In addition, a more powerful GSM module can solve problems such as internet connection and antenna coverage. Because the SIM800L module provides older communications such as 2G, which is gradually becoming obsolete in many countries. The software could also provide a more appropriate user interface for determining the safe point and also send an SMS to the person when the vehicle was outside the limits, apart from the notification in the software itself, from communicating through the server with ESP.

## 4- Attachments:

• The hardware code file in the Arduino environment is attached to this file.

carAlarm_and_gpsTraking_project.ino

• The rar file containing the sources of parts of the software is attached to this file as an example:

smartGPS_tracking_app.rar