

Design Manual



HAZARD HUNTER

Group 20

De Alwis K. K. M. (E/18/058)
Hariharan R. (E/18/128)
Karan R. (E/18/168)

Content

1. Introduction
2. System Overview
 - 2.1.1.1.Solution Architecture
 - 2.1.1.2.Technology stack
 - 2.1.1.3.Hardware design
3. Hardware Implementation
 - 3.1.1. Components
 - 3.1.2. Circuit Diagram
4. Server
5. Sign detection

1.Introduction

The number of deaths due to accidents has increased a considerable amount over the years. Drivers not being able to identify the road signs have a major contribution to these accidents. Also, not having anyone to help the injured person has increased the number of deaths in a considerable way.

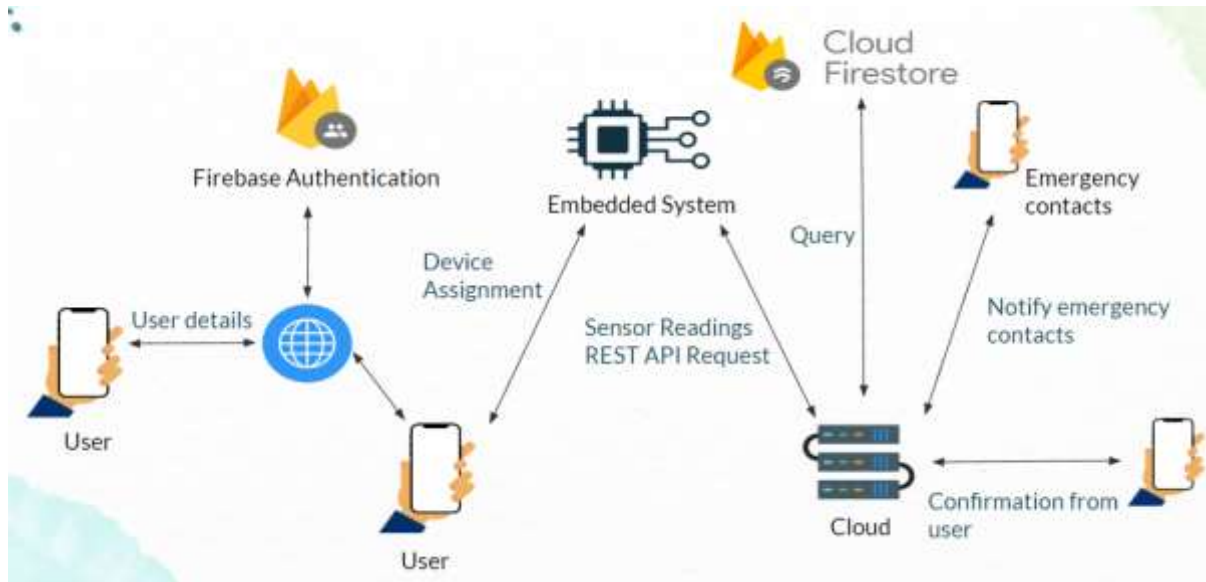
“Hazard Hunter” can detect accidents, create awareness among users when accidents happen, keep users alert about road signs, and notify emergency contacts.

This product has main two components. The hardware device and the mobile application. The hardware part of the product will be mainly used to detect accidents and sending notifications to emergency contacts. By using the mobile application, the users will be able to register to the system and start using the device. Also, users will be able to see the accidents that have happen and if they want, they can send help or they view trouble free routes for when they are going for important events.

Also, a website will be provided for the system admins to view current details about the users, accidents and other important details.

2. System Overview

2.1. Solution Architecture



The users will be registered to the system using firebase authentication and during the registration the user details including the emergency contacts will be taken. A particular device will be assigned to a user when they scan the QR code which is provided in that device. A QR code can be used to register to the system only once. In this way the security will be provided to the user. After registering and scanning the QR code the sensor readings and the rest API requests will be sent to the cloud and the cloud will query the Firestore. When an accident is detected the emergency contacts will be notified. Also, before sending the notifications to the emergency contacts a confirmation alert will be sent to the users. If the users confirm the alert request the notification will be sent. Otherwise, if they decline it notifications won't be sent. The other scenario here is, if the user is unconscious and unable to reply to the alert message. In such cases the notification will be automatically sent if the user does not reply during 15 seconds of time duration.

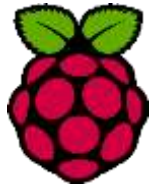
2.2. Technology Stack

- Python

For backend in cloud, and backend in raspberry pi as well.



- Raspberry Pi 3B



- Git

For version control because it is important to keep track of changes — and keep every team member working on the right version.



- Flutter

For both mobile application and web app because Flutter is Google's free, open-source software development kit (SDK) for cross-platform mobile application development. Using a single codebase, Flutter helps developers build high-performance, scalable applications with attractive and functional user interfaces for Android or IOS. It has of the fastest growing community with good support. We are planning to use AWS EC2 instances, and AWS Lambda for cloud services. Amazon EC2 provides a wide selection of instance types optimized to fit different use cases.



- AWS

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. You can trigger Lambda from over 200 AWS services and only pay for what you use.



- Firebase

For user authentication, we are using Firebase Authentication. It aims to make building secure authentication systems easy, while improving the sign-in and onboarding experience for end users. It provides an end-to-end identity solution, supporting email and password accounts, phone auth, and Google, Twitter, Facebook, and GitHub login, and more.



2.3. Hardware Design



In the hardware there will be many ports open to the outside. Raspberry pi camera which is used for object detection will be open to the outside. Also, the ethernet port, power supply, cooling fan and the headphone jack will be open to outside. The ethernet port will be used for trouble shooting in the device. Sensors and all other components will be interconnected.

3. Hardware Implementation

3.1. Components

- Raspberry Pi 3B+



- 5V/2.5A DC power input
- Built-in wifi module
- Micro SD port
- CSI camera port
- Full-size HDMI

- Raspberry Pi camera



- 5-megapixel camera
- Pixel Count 2592 x 1944
- Capture video at 1080p30 with H.264 encoding
- Angle of View 54 x 41 degrees
- Field of View 2.0 x 1.33 m at 2 m

- Gyroscope



- GPS module



- Accelerometer



- Speakers



- Micro SD card

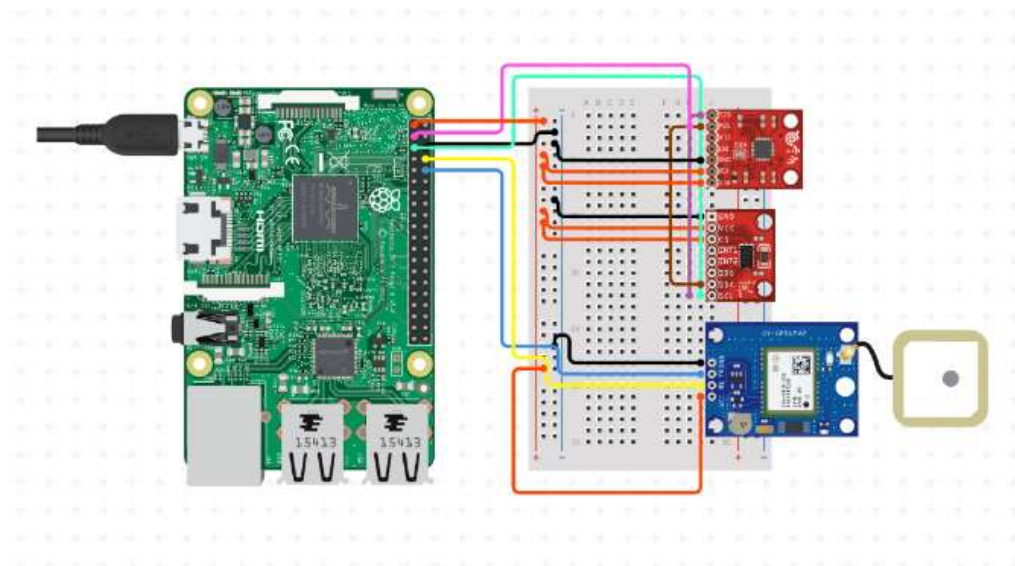


- 32 GB

- Cooling fan



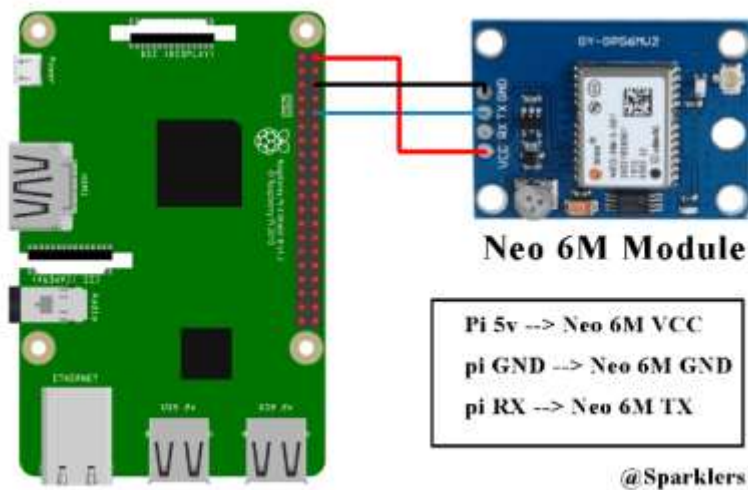
3.2. Circuit Diagram



3.2.1 GPS

Here, Neo 6M module is only needed to be connected with Raspberry Pi which is quite easy. The connections are shown below:

Neo 6M VCC => Raspberry pi 5v
Neo 6M GND => Raspberry pi GND
Neo 6M RX => Raspberry pi TX (gpio 14) //Not required in our case
Neo 6M TX => Raspberry pi RX (gpio 15)



The VCC of Neo 6M needs to be connected with 5v of Raspberry pi, GND of Neo 6M with GND of Raspberry pi, and TX of Neo 6M with RX of Raspberry Pi so that the GPS module can send data to the Raspberry pi through the serial connection.

Getting data from the GPS module:


1. Install the latest Raspbian OS on a memory card. For details visit www.raspberrypi.org/documentation/installation/installing-images/.
2. Insert the memory card into raspberry pi and power it up.
3. Now here we need to modify a few things. First, we need to edit the /boot/config.txt file. Now you need to open this file in any text editor. Here I am using nano:

```
sudo nano /boot/config.txt
```

At the end of the file add the following lines

```
dtoverlay=spi-on  
dtoverlay=pi3-disable-bt  
core_freq=250  
enable_uart=1  
force_turbo=1
```

It will look something like this



```
GNU nano 2.7.4 File: /boot/config.txt  
hdmi_group=2  
hdmi_mode=1  
hdmi_mode=87  
hdmi_cvt 480 320 60 6 0 0 0  
hdmi_drive=2  
start_x=1  
gpu_mem=128  
  
dtoverlay=spi-on  
dtoverlay=pi3-disable-bt  
core_freq=250  
enable_uart=1  
force_turbo=1
```

Now save this by typing ctrl +x, type y, and press enter.

4. Raspbian uses the UART as a serial console, so we need to turn off that functionality. To do so we need to change the /boot/cmdline.txt file. For safety before editing the file make a backup of that using the following command:

```
sudo cp /boot/cmdline.txt /boot/cmdline_backup.txt
```

Now to edit that file open that in a text editor:

```
sudo nano /boot/cmdline.txt
```

Replace the content with the following line (delete everything in it and write down the following content):

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4  
elevator=deadline fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-  
consoles
```

Now save this by typing ctrl +x, type y, and press enter.

Now reboot pi using the:

```
sudo reboot
```

5. After the reboot now it's time to check how our GPS module is working.

!!!before checking this out make sure that the blue LED in the Neo 6M is blinking. Basically, the blinking of a blue LED means that it's receiving the data perfectly. Generally, it takes a few minutes to blink the LED after giving power to the module. So wait until that. And if it's not blinking even after 5 minutes then go to a window side area or any place under the open sky. In the image below you can see how the blue LED should blink.



When the blue led is blinking, run the following command:

```
sudo cat /dev/ttyAMA0
```

Now you will see a lot of data shown in the below image. That basically means that it's working. To stop this type Ctrl + c.

```

$GPGGA,162733.00,2240.36183,N,08826.15723,E,2,07,1.26,-7.8,M,-54.0,M,,0000*5C
$GPGSA,A,3,31,32,40,10,14,20,25,,,,,,,,3.47,1.26,3.24*04
$GPGSV,3,1,12,10,63,065,27,12,11,063,,14,49,315,34,18,24,309,24*77
$GPGSV,3,2,12,20,46,111,28,21,17,169,17,25,27,100,22,26,05,186,*70
$GPGSV,3,3,12,27,03,235,,31,58,211,38,32,51,349,38,40,44,240,35*7A
$GPGLL,2240.36183,N,08826.15723,E,162733.00,A,D*63
$GPRMC,162734.00,A,2240.36182,N,08826.15718,E,0.116,,120719,,,D*7E
$GPVTG,,T,,M,0.116,N,0.215,K,D*26
$GPGGA,162734.00,2240.36182,N,08826.15718,E,2,07,1.26,-8.0,M,-54.0,M,,0000*55
$GPGSA,A,3,31,32,40,10,14,20,25,,,,,,,,3.47,1.26,3.24*04

```

Setup for writing the Python Code:

1. Now before writing the python code to get the GPS data, a few things needed to be set up again. By default, the Raspberry Pi uses a serial port for this “console” login so if the serial port is used to get data from the GPS module the console login need to be disabled again. Now there are two serial ports in Raspberry pi 3: serial0 and serial1. But in between them serial 0 will point to GPIO pins 14 and 15, so serial 0 needs to be used. Now to see which port is connected with serial0 use the following command:

```
ls -l /dev
```

There are two possible outputs:

- If the output looks like this:

```

crw-rw-r-- 1 root netdev 10, 58 Jul 10 16:07 rfkill
lrwxrwxrwx 1 root root    7 Jul 10 16:07 serial0 -> ttyAMA0
lrwxrwxrwx 1 root root    5 Jul 10 16:07 serial1 -> ttyS0
drwxrwxrwt 2 root root   40 Nov 3  2016 shm

```

Here, serial0 is linked with ttyAMA0. So to disable the console use the following commands:

```

sudo systemctl stop serial-getty@ttyAMA0.service
sudo systemctl disable serial-getty@ttyAMA0.service

```

- But if the output looks like below:

```

crw-rw-r-- 1 root root    10, 58 May 28 12:14 rfkill
lrwxrwxrwx 1 root root    5 May 28 12:14 serial0 -> ttyS0
lrwxrwxrwx 1 root root    7 May 28 12:14 serial1 -> ttyAMA0
drwxrwxrwt 2 root root   40 May 28 12:15 shm

```

That means serial0 is linked with ttyS0. So to disable the console use the following commands:

```

sudo systemctl stop serial-getty@ttyS0.service
sudo systemctl disable serial-getty@ttyS0.service

```

To learn more about the serial ports of Raspberry Pi you can visit the following link [configuring-gpio-serial-port-raspbian-jessie-including-pi-3](#)

Writing the Python code

1. Now python library needs to be installed:
`pip install pynmea2`
2. Now finally the code can be written:

```
import serial
import time
import string
import pynmea2

while True:
    port="/dev/ttyAMA0"
    ser=serial.Serial(port, baudrate=9600, timeout=0.5)
    dataout = pynmea2.NMEAStreamReader()
    newdata=ser.readline()

    if newdata[0:6] == "$GPRMC":
        newmsg=pynmea2.parse(newdata)
        lat=newmsg.latitude
        lng=newmsg.longitude
        gps = "Latitude=" + str(lat) + "and Longitude=" + str(lng)
        print(gps)
```

The above python code will give an output like shown in the below figure:

```
lat= 22.6726726667 lng= 88.4359028333
lat= 22.6726731667 lng= 88.4359005
lat= 22.6726733333 lng= 88.4359006667
lat= 22.6726736667 lng= 88.4359003333
lat= 22.6726741667 lng= 88.4358998333
lat= 22.6726743333 lng= 88.4359
lat= 22.6726741667 lng= 88.4359001667
lat= 22.6726745 lng= 88.435901
lat= 22.6726756667 lng= 88.4359
lat= 22.6726765 lng= 88.4358958333
lat= 22.672677 lng= 88.4358925
lat= 22.6726771667 lng= 88.435893
lat= 22.6726778333 lng= 88.4358925
lat= 22.6726786667 lng= 88.4358915
lat= 22.6726791667 lng= 88.4358918333
```

Here the device is not moving, that's why it shows the same GPS location. But as you can see it's working properly.

3.2.2 MPU-6050

Accessing MPU-6050 on Raspberry Pi using Python to get temperature, accelerometer, and gyroscope data.

Dependencies

```
sudo apt install python3-smbus  
pip install mpu6050-raspberrypi
```

Run

```
Python3 mpu_6050.py
```

```
1  from mpu6050 import mpu6050  
2  import time  
3  mpu = mpu6050(0x68)  
4  
5  while True:  
6      print("Temp : "+str(mpu.get_temp()))  
7      print()  
8  
9      accel_data = mpu.get_accel_data()  
10     print("Acc X : "+str(accel_data['x']))  
11     print("Acc Y : "+str(accel_data['y']))  
12     print("Acc Z : "+str(accel_data['z']))  
13     print()  
14  
15     gyro_data = mpu.get_gyro_data()  
16     print("Gyro X : "+str(gyro_data['x']))  
17     print("Gyro Y : "+str(gyro_data['y']))  
18     print("Gyro Z : "+str(gyro_data['z']))  
19     print()  
20     print("-----")  
21     time.sleep(1)  
22
```

Acknowledgments

<https://pypi.org/project/mpu6050-raspberrypi/>

The library used in this code is written by <https://github.com/m-rtijn/mpu6050/>

3.2.3 GSM SIM800L

Internet connection is now becoming the most important thing after power supply for [Raspberry Pi based projects](#) and when we talk about a true [IoT project](#) then it becomes a must thing to consider. So, to interface GSM SIM800L for providing internet to our Pi just like a modem using GSM GPRS, we are using [Point to Point Protocol](#) (PPP) and [node to node communication](#) for communicating with the GSM serially. After doing this we don't need to connect your pi with Wi-Fi or ethernet.

Components required for Tether Internet on Raspberry Pi

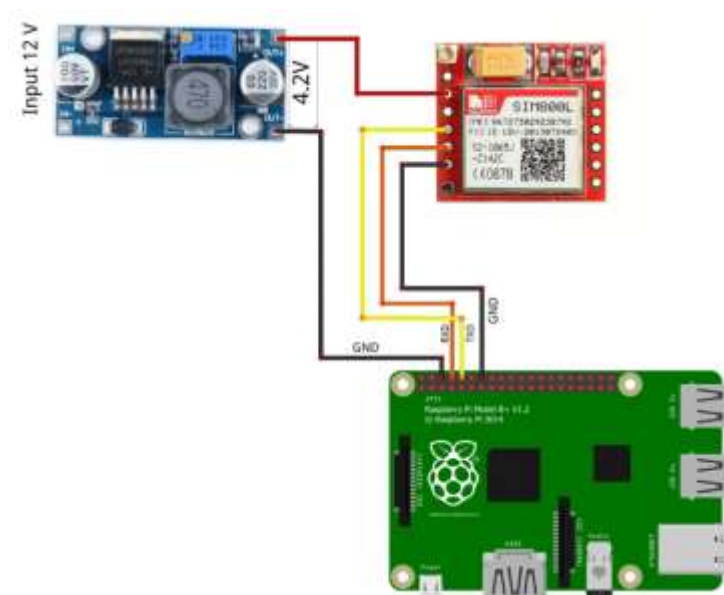
Component	Qty.
Raspberry Pi 3 (You can use Any)	1
GSM SIM800L Module	1
Jumper Wires F-F	As req.
Working networked SIM card	1
2596 buck converter with adaptor	1 each

Here, 2596 buck converter is used for providing a constant voltage of 4.2 Volts as the GSM SIM800L module is a power-hungry device that will require more current while using the GPRS.

Tether Internet on Raspberry Pi Circuit Diagram

GPS Module	Raspberry Pi 3
Rx	Tx of the Raspberry Pi
Tx	RX of the Raspberry Pi
GND	GND of the Raspberry Pi
	GND of the Buck
VCO	4.2 V (2596 Ic output)

Hence all the pins are connected accordingly. For a better understanding, refer to the connection diagram below.



Installing Library and Programming Raspberry Pi

Before going further, check the GSM with AT commands and confirm whether the GSM is responding to the AT commands or not. This can be done in the following way,

- [Enable the Serial Port](#)
- [Check the Serial Port](#)
- Run the following *check_commands.py* script
- It must return 'OK' in response to your code, if not, please check the connections of RX_TX and repeat the steps.
-

```
import time
import serial
import os, time
cmd=''
ser = serial.Serial('/dev/serial0', 9600, timeout=1)
ser.reset_input_buffer()
while True:
    smd=input("please enter a command: ")
    smd=smd+'\n'
    smds=smd.encode('utf-8')
    ser.write(smds)
    print("smd value is:",smd)
    line = ser.read(10).decode('utf-8').rstrip()
    print(line)
```

Note: Since python 3 is used on raspberry Pi, encoding and decoding of the data under “utf-8” should be taken into consideration.

Now the above code will send a command from your terminal and display the response on the terminal. In the beginning, write “AT” and then press ‘Enter’, it must return the “OK” message. Try out some more [AT commands](#) for fun. Now, PPP library which will help in using the Internet through the GSM module should be installed.

Write the following command in the terminal so that the latest update on the Pi will be available.

```
sudo apt-get upgrade
sudo apt-get install ppp screen eLinks
```

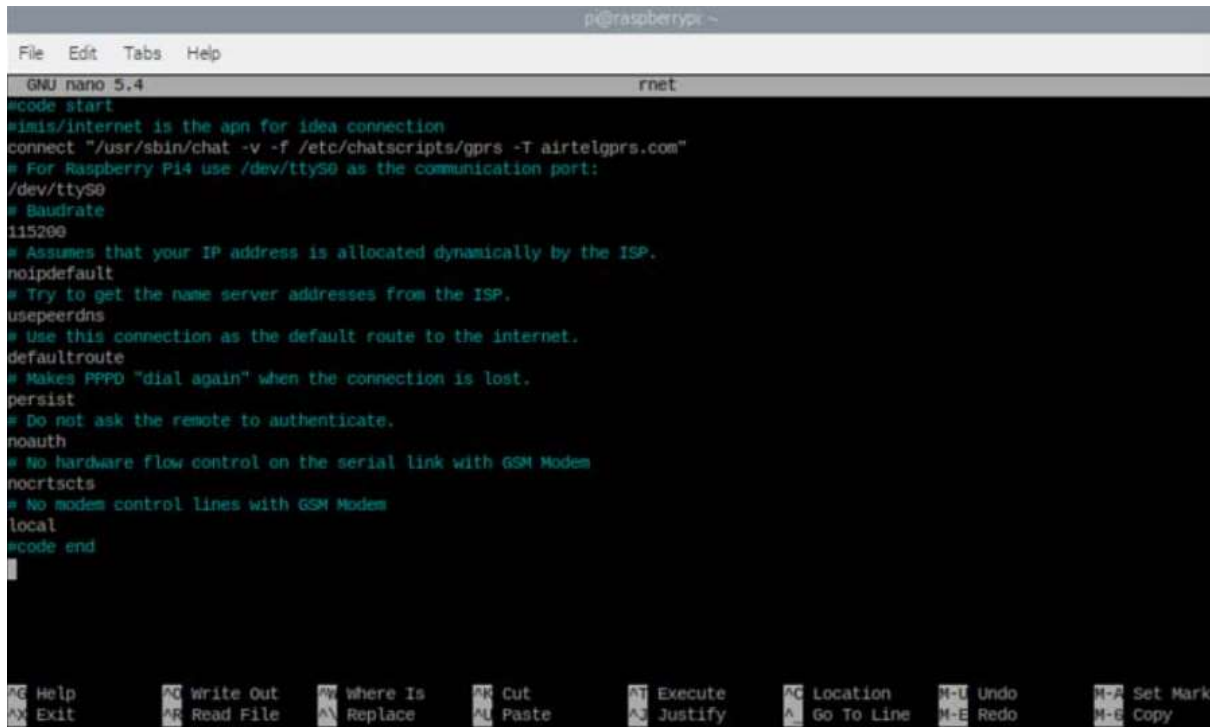
This way you will download and install the ppp library. Now go to the sudo -i by typing the same on the terminal. Then you will have to go to the **peers** folder by using the following command.

```
cd /etc/ppp/peers*
```

Then type “**nano rnet**” so that you can create a file named ‘**rnet**’ in the ‘**peers**’ folder. There you have to write the following lines of the code. (just copy and paste it).

```
#code start
#imis/internet is the apn for idea connection
connect "/usr/sbin/chat -v -f /etc/chatscripts/gprs -T airtelgprs.com"
# For Raspberry Pi4 use /dev/ttyS0 as the communication port:
/dev/ttyS0
# Baudrate
115200
# Assumes that your IP address is allocated dynamically by the ISP.
noipdefault
# Try to get the name server addresses from the ISP.
usepeerdns
# Use this connection as the default route to the internet.
defaultroute
# Makes PPPD "dial again" when the connection is lost.
persist
# Do not ask the remote to authenticate.
noauth
# No hardware flow control on the serial link with GSM Modem
nocrtscts
# No modem control lines with GSM Modem
local
#code end
```

Since I am using the **Airtel** network sim card hence, we should know the APN of the same. As of now, it is “**airtelgprs.com**”. You can easily find out yours on Google.



Here you will have to change your **APN** in the line ‘connect’. Mine is “**airtelgprs.com**”. Also, my Serial port is “**ttys0**” hence I have mentioned it. Please note that you do not need to change the Baud rate. Then save it and close it by pressing “**Ctrl+X**”.

Now restart the terminal. Then, in the terminal type “**ifconfig**” and press enter.

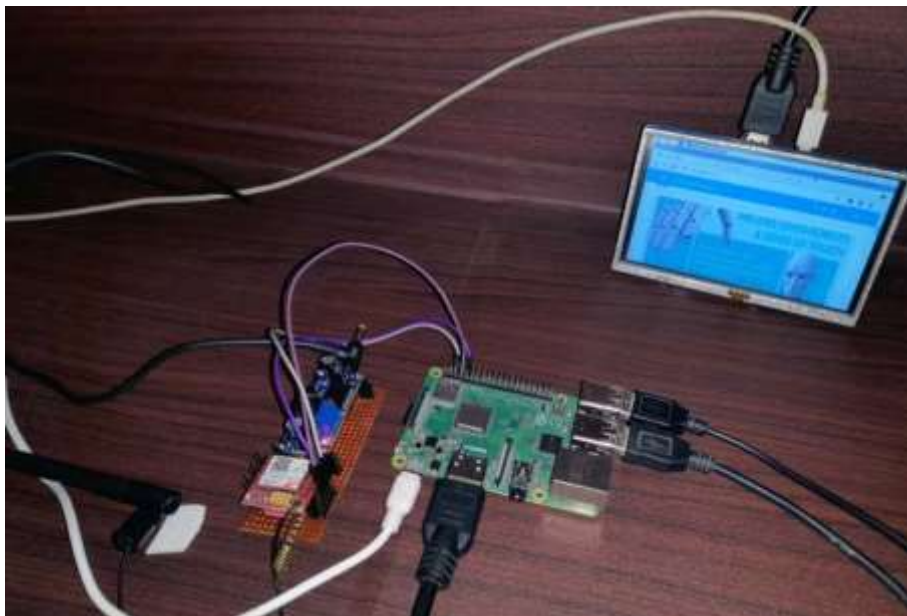
```
pi@raspberrypi:~$ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether b8:27:eb:9f:58:06 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 30 bytes 2887 (2.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 30 bytes 2887 (2.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ppp0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 100.126.235.50 netmask 255.255.255.255 destination 192.168.254.254
    ppp txqueuelen 3 (Point-to-Point Protocol)
    RX packets 5219 bytes 7415676 (7.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 5232 bytes 306908 (299.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.137.217 netmask 255.255.255.0 broadcast 192.168.137.255
    inet6 fe80::aeb4:26c8:399e:7a85 prefixlen 64 scopeid 0x20<link>
    ether b8:27:eb:ca:0d:53 txqueuelen 1000 (Ethernet)
    RX packets 4888 bytes 331829 (324.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4489 bytes 1226432 (1.1 MiB)
```

Now, let the fun begins by entering the following command i.e. “*sudo pon rnet*”. Nothing will happen on the screen but the behavior of your GSM LED will change. This time the frequency of toggling the LED on and off will increase. Here you can stop the internet by entering “*sudo poff rnet*”. Now you can enjoy the GPRS net on your Raspberry Pi. Although the internet speed is not so good in my case due to some signal strength, that can be neglected if you want to make an IoT project.



4. Server

Our system has lots of requirements that have to be handled by an server.

Those things are

- Find nearest accidents for all the users, and notify them.
- Call emergency contacts, and send them an SMS with accident location after accident verification.

For these requirements, we have APIs hosted in AWS EC2 instance which will handle the requests coming from our devices to server during the accident time.

These APIs are created by Flask from Python which take GPS coordinates as arguments.

Once an accident is verified, device will send the accident coordinates to the server. When this request comes, server will query the emergency contact details of the user according to the device id, and server will call the emergency contact saying about accident, and its location, and server will also send location in a text message as well. Another part, that is handled by the server is finding the nearby accidents to all the users, and notifying them. Accident coordinates will also be sent to this api end point. With those coordinates it will query the locations of users nearby, and notify them about the latest, nearest accidents. This api will also delete new accidents after one day.

5. Sign Detection

Below code was used for detecting the road signs. The xml files which were used for traffic detection are in the github repository

```
1  import cv2
2  from playsound import playsound
3  old_val=""
4  new_val=""
5  import pygame
6  pygame.mixer.init()
7
8
9
10 # Stop Sign Cascade Classifier xml
11 stop_sign = cv2.CascadeClassifier('cascade_stop_sign.xml')
12 yieldsigns = cv2.CascadeClassifier('yeildstages.xml')
13 speedlimit_sign= cv2.CascadeClassifier('speedlimit.xml')
14 trafficlight_sign= cv2.CascadeClassifier('trafficlights.xml')
15 left_turn= cv2.CascadeClassifier('left.xml')
16
17 cap = cv2.VideoCapture(0)
18 cap.set(cv2.CAP_PROP_FRAME_WIDTH, 256)
19 cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 144)
20
21
22 while cap.isOpened():
23     _, img = cap.read()
24     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
25     stop_sign_scaled = stop_sign.detectMultiScale(gray, 1.3, 5)
26     stop_sign_scaled1 = yieldsigns.detectMultiScale(gray, 1.3, 5)
27     speedlimit_scaled = speedlimit_sign.detectMultiScale(gray, 1.3, 5)
28     trafficlight_scaled = trafficlight_sign.detectMultiScale(gray, 1.3, 5)
29     left_scaled = left_turn.detectMultiScale(gray, 1.3, 5)
30
31     # Detect the stop sign, x,y = origin points, w = width, h = height
32     for (x, y, w, h) in stop_sign_scaled:
33
34         # Draw rectangle around the stop sign
35         stop_sign_rectangle = cv2.rectangle(img, (x,y),
36                                             (x+w, y+h),
37                                             (0, 255, 0), 3)
38         # Write "Stop sign" on the bottom of the rectangle
39         stop_sign_text = cv2.putText(img=stop_sign_rectangle,
40                                     text="Stop Sign",
41                                     org=(x, y+h+30),
42                                     fontFace=cv2.FONT_HERSHEY_SIMPLEX,
43                                     fontScale=1, color=(0, 0, 255),
44                                     thickness=2, lineType=cv2.LINE_4)
45         new_val="stop"
```



```

46         if ((old_val=="")|(new_val!=old_val)):
47             pygame.mixer.music.load("/home/pi/Desktop/Sign/stop.wav")
48             pygame.mixer.music.play()
49             while pygame.mixer.music.get_busy() == True:
50                 continue
51             old_val="stop"
52
53
54     for (x, y, w, h) in stop_sign_scaled1:
55         # Draw rectangle around the stop sign
56         stop_sign_rectangle1 = cv2.rectangle(img, (x,y),
57                                             (x+w, y+h),
58                                             (0, 255, 0), 3)
59         # Write "Stop sign" on the bottom of the rectangle
60         stop_sign_text1 = cv2.putText(img=stop_sign_rectangle1,
61                                     text="Yield",
62                                     org=(x, y+h+30),
63                                     fontFace=cv2.FONT_HERSHEY_SIMPLEX,
64                                     fontScale=1, color=(0, 0, 255),
65                                     thickness=2, lineType=cv2.LINE_4)
66

```

```

67     new_val="yield"
68     if ((old_val=="")|(new_val!=old_val)):
69         pygame.mixer.music.load("/home/pi/Desktop/Sign/yield_sign.mp3")
70         pygame.mixer.music.play()
71         while pygame.mixer.music.get_busy() == True:
72             continue
73         old_val="yield"
74
75
76     for (x, y, w, h) in speedlimit_scaled:
77         # Draw rectangle around the stop sign
78         speedlimit_rectangle = cv2.rectangle(img, (x,y),
79                                             (x+w, y+h),
80                                             (0, 255, 0), 3)
81         # Write "Stop sign" on the bottom of the rectangle
82         speedlimit_text1 = cv2.putText(img=speedlimit_rectangle,
83                                     text="speed limit",
84                                     org=(x, y+h+30),
85                                     fontFace=cv2.FONT_HERSHEY_SIMPLEX,
86                                     fontScale=1, color=(0, 0, 255),
87                                     thickness=2, lineType=cv2.LINE_4)
88     new_val="speed limit"

```

```

89         if ((old_val=="")|(new_val!=old_val)):
90             pygame.mixer.music.load("/home/pi/Desktop/Sign/Speed Limit.mp3")
91             pygame.mixer.music.play()
92             while pygame.mixer.music.get_busy() == True:
93                 continue
94             old_val="speed limit"
95
96     for (x, y, w, h) in trafficlight_scaled:
97         # Draw rectangle around the stop sign
98         trafficlightr_rectangle = cv2.rectangle(img, (x, y),
99                                                  (x + w, y + h),
100                                                  (0, 255, 0), 3)
101         # Write "Stop sign" on the bottom of the rectangle
102         trafficlightr_text1 = cv2.putText(img=trafficlightr_rectangle,
103                                           text="traffic_light",
104                                           org=(x, y + h + 30),
105                                           fontFace=cv2.FONT_HERSHEY_SIMPLEX,
106                                           fontScale=1, color=(0, 0, 255),
107                                           thickness=2, lineType=cv2.LINE_4)
108         new_val="traffic_light"
109         if ((old_val=="")|(new_val!=old_val)):
110             pygame.mixer.music.load("/home/pi/Desktop/Sign/traffic_lights.mp3")
111             pygame.mixer.music.play()
112             while pygame.mixer.music.get_busy() == True:
113                 continue
114             old_val="traffic_light"

```

```

116     for (x, y, w, h) in left_scaled:
117         # Draw rectangle around the stop sign
118         left_rectangle = cv2.rectangle(img, (x, y),
119                                        (x + w, y + h),
120                                        (0, 255, 0), 3)
121         # Write "Stop sign" on the bottom of the rectangle
122         left_text1 = cv2.putText(img=left_rectangle,
123                                 text="Left Turn",
124                                 org=(x, y + h + 30),
125                                 fontFace=cv2.FONT_HERSHEY_SIMPLEX,
126                                 fontScale=1, color=(0, 0, 255),
127                                 thickness=2, lineType=cv2.LINE_4)
128         new_val="Left Turn"
129         if ((old_val=="")|(new_val!=old_val)):
130             pygame.mixer.music.load("/home/pi/Desktop/Sign/Left Turn.mp3")
131             pygame.mixer.music.play()
132             while pygame.mixer.music.get_busy() == True:
133                 continue
134             old_val="Left Turn"
135
136     cv2.imshow("img", img)
137     key = cv2.waitKey(30)
138     if key == ord('q'):
139         cap.release()
140         cv2.destroyAllWindows()
141     break

```