

A PROJECT REPORT ON

ACCIDENT PREVENTION SYSTEM USING V2V COMMUNICATION

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

BACHELOR OF ENGINEERING

In

COMPUTER ENGINEERING

Of

SAVITRIBAI PHULE PUNE UNIVERSITY

By

ANKUSH CHILBULE	B150234232
AVANI FUTANE	B150234252
ASHRITHA GORAMANE	B150234261
AKSHIT ROHRA	B150234356

Under the guidance of
Prof. D.N.PATIL



Sinhgad Institutes

DEPARTMENT OF COMPUTER ENGINEERING
SINHGAD COLLEGE OF ENGINEERING, PUNE-41

Accredited by NAAC Grade 'A'
2018-19

Date:

CERTIFICATE

This is to certify that the project report entitled
“ACCIEDENT PREVENTION USING V2V COMMUNICATION”

Submitted by

ANKUSH CHILBULE	B150234232
AVANI FUTANE	B150234252
ASHRITHA GORAMANE	B150234261
AKSHIT ROHRA	B150234356

is a bonafide work carried out by him/her under the supervision of Prof. D.N. Patil and it is approved for the partial fulfillment of the requirements of Savitribai Phule Pune University, Pune for the award of the degree of Bachelor of Engineering (Computer Engineering) during the year 2018-19.

Prof. D.N. PATIL

Internal Guide

Prof. M.P. Wankhade

Head

Department of Computer Engineering

Dr. S.D. Lokhande

Principal

Sinhgad college of Engineering

Acknowledgement

The least we can do is express our gratitude to the following individuals. Their contribution has been noteworthy in the completion of the project “Accident Prevention System using V2V Communication”. It gives us immense pleasure to acknowledge the innumerable guidance extended by Prof. D. N. Patil. She has always enlightened us with her valuable suggestions and ideas. In this endeavor we acknowledge the contribution of Prof. M. P. Wankhede, HOD, Dept. of Computer Engineering, Sinhgad College of Engineering for encouraging us to perform this project. All his suggestions and advices have always been of great help. We are also grateful to Dr. S. D. Lokhande, Principal, Sinhgad College of Engineering, for his continued and unending support. We are thankful to Sinhgad College of Engineering, Pune for providing us all the necessary resources. We extend our sincere thanks to the Savitribai Phule Pune University, Pune for granting us the permission to work on this project along with our academics. We would also like to thank all those who helped us complete this project. Without them this project looked impossible to complete

Ankush Chilbule
Avani Futane
Ashritha Goramane
Akshit Rohra

Abstract

The main focus of Accident Prevention System is to reduce number of road accidents. The foremost scope of the system includes regions with fog and roads with blind turns. The system fulfills its aim by calculating distance between two nearby vehicles. For this purpose GPS, beacons, ultrasonic sensor, magnetometer and microcontroller is used.

The APS works in a peculiar radius. The GPS module finds the location which is traded with other vehicles through beacons. A plausible threat is realized by this. Then the ultrasonic sensors are used to get better affirmation of the threat. After confirmation of threat the driver is notified about it using LEDs, buzzers and display. A small screen is installed on which the surrounding region is exhibited. Along with this appropriate instructions are provided for better assistance. The system starts on ignition and works in a continuous loop till ignition is on.

List of Figures

2.1	External Overview	4
2.2	Internal Overview	5
2.3	Time Line Chart (May'18 - December'18)	11
2.4	Time Line Chart (January'19 - May'19)	11
3.1	Mapping Diagram	15
3.2	Use-case Diagram	17
3.3	Activity Diagram	18
3.4	Sequence Diagram	19
3.5	Class Diagram	20
3.6	Level-0 Data Flow Diagram	20
3.7	Level-1 Data Flow Diagram	21
3.8	Level-2 Data Flow Diagram	21
3.9	Deployment Diagram	22
4.1	Console output of transmitter	39
4.2	Console output of receiver	39
4.3	Console output of micro controller	40
7.1	Arduino UNO pin-out	50
7.2	Arduino Mega pin-out	50
7.3	NodeMCU pin-out	51
7.4	GPS SIM28 pin-out	51

7.5 ESP8266-01 pin-out 52

7.6 Magnetometer (HMC5883L) pin-out 52

7.7 Screen pin-out 53

List of Tables

2.1	Degree of complexity for EO	8
2.2	Degree of Complexity for ILF	9
2.3	Predefined Weights	9
2.4	Value Adjustment factor	10
3.1	IDEA Matrix	12
5.1	Test cases for unit testing	42
5.2	Test cases for integration testing-I	43
5.3	Test cases for integration testing-II	44
6.1	Response time of GPS module	46
6.2	Response time of Ultrasonic sensor	46
6.3	Values given by Magnetometer	47

Abbreviations

APS	Accident Prevention System
V2V	Vehicle to vehicle
VANET	Vehicular Ad-hoc Network
GPS	Global Positioning System
IDE	Integrated Development Environment
LED	Light Emitting Diode
ILF	Internal Logical File
EIF	External Interface File
EI	External Input
EO	External Output
EQ	External Inquiry
DFD	Data Flow Diagram
UFP	Unadjusted Function Point
FP	Function Point
VAF	Value Adjustment Factor
SLOC	Source Line of Code
Lat	Latitude
Long	Longitude

Contents

Certificate

Acknowledgement

Abstract **i**

List of Figures **ii**

List of Tables **iii**

Abbreviations **iv**

1 INTRODUCTION **1**

1.1 Background and Basics 1

1.2 Literature Survey 1

1.2.1 Paper I 1

1.2.2 Paper-II 2

1.2.3 Paper- III 2

1.3 Project Undertaken 3

1.3.1 Problem Definition 3

1.3.2 Scope Statement 3

1.4 Organization of Report 3

2 PROJECT PLANNING AND MANAGEMENT **4**

2.1 Detail System Requirement Specification (SRS) 4

2.1.1	System Overview	4
2.1.2	Functional Requirements	5
2.1.3	Non-Functional Requirements	6
2.1.4	Deployment Environment	6
2.1.5	External Interface Requirements	7
2.2	Project Process Modelling	7
2.3	Cost & Efforts Estimates	8
2.4	Project Scheduling	11
2.4.1	Time Line Chart	11
3	ANALYSIS & DESIGN	12
3.1	IDEA Matrix	12
3.2	Mathematical Model	12
3.3	Feasibility Analysis	16
3.4	UML Diagrams	17
3.4.1	Use Case Diagram	17
3.4.2	Activity Diagram	18
3.4.3	Sequence Diagram	19
3.4.4	Class Diagram	20
3.4.5	Data Flow Diagram	20
3.4.6	Deployment Diagram	21
4	IMPLEMENTATION AND CODING	23
4.1	Introduction	23
4.2	Operational Details	23

4.3	Major Classes	23
4.4	Code Listing	24
4.4.1	Transmit	24
4.4.2	Receive	27
4.4.3	Micro-controller	28
4.5	Screen Shots	38
5	TESTING	41
5.1	Unit Testing	41
5.2	Integration Testing	43
5.3	Acceptance Testing	44
6	RESULTS AND DISCUSSION	46
6.1	Prototype Snapshots	46
6.2	Results	46
6.3	Discussions	47
7	Conclusion and Future Work	48
7.1	Conclusion	48
7.2	Future Work	48

Chapter 1

INTRODUCTION

1.1 Background and Basics

A fast paced life has become a necessity in this ever growing and prospering world. Quick commute is the immediate effect of this change. Also the young generation finds it fashionable to drive swiftly. Both these things have given rise to speeding and reckless driving. As a result of this, lives of many innocent people seem to be at risk.

Our system aims at contributing to the society in this regard. The Accident Prevention System assists the driver to safeguard himself. The system works like an alarm by generating alerts when threshold is crossed. The driver is warned about the nearing vehicles with the help of LEDs and buzzers. It also has the facility to control the intensity of alerts in case of variable traffic densities.

Our system works similar to Vehicular Ad-hoc Networks (VANET). In the APS, exchange of location of vehicles takes place. The system then calculates distance from the GPS location with respect to itself and determines potential threats. To make the system more efficient, the threat is analyzed with the help of ultrasonic sensors. These sensors then notify the drivers. This alerts the drivers and helps in avoiding accidents

1.2 Literature Survey

1.2.1 Paper I

Title: An Adaptive Beacon-based Scheme for Warning Messages Dissemination in Vehicular Ad-hoc Networks

Conference: International Conference on Advanced Computing and Applications, 2017

Abstract: The simulation is performed for 3 categories: 10, 50, and 100 vehicles/sqkm. Cars are generated randomly, with a data rate of 18 mbps, transmission power of 20 mW and receiver sensitivity of -89 dBm. This achieves a range of approximately 510m. The system was evaluated based on number of warning messages received, number of beacons

and warning notification time. The results showed that the efficiency reached up to 95

Pros: Works best for sparse vehicle density.

Cons: As the vehicle density increases, the ABDDIs scheme does not perform very well. We can use the NSF scheme.

1.2.2 Paper-II

Title: Object Recognition using Horizontal Ray of Ultrasonic Sensors

Conference: International Conference on Communication and Signal Processing, 2016

Abstract: An array of ultrasonic sensors was used to find distance of object using time of flight principle. The sensors were connected to Arduino UNO. Delay between two measurements was of one second. Thus obtained input distance array was used to calculate difference matrix and determine the object. The method was evaluated based on angle and distance of the object from the sensor and its surface. The size of object and its distance from the sensor necessary for recognition was as follows: Cylinder- radius 20 cm, distance 120 cm; Cube- side 25 cm, distance 45 cm; Rectangular prism- side 35 cm, distance 30 cm.

Pros: The results are obtained by measuring target surfaces only one time

Cons: To recognize complex objects like cone and pyramid they had to be scanned from two different heights.

1.2.3 Paper- III

Title: ESP8266 based Implementation of Wireless Sensor Network with Linux Based Web Server

Conference: 2016 Symposium on Colossal Data Analysis and Networking (CDAN)

Abstract: There are temperature sensors and LED in the remote area, which are connected to the Raspberry pi module which acts as a Mini-computer in and continuously monitor the sensors and store it in the database using SQL. Croon Job is used as a job scheduler in UNIX like operating System. This helps in scheduling and updating of data in database at a fixed time interval that can be decided by the user. Wi-Fi Standard, Packet, Mode, Cost were the evaluation parameters. The proposed design for the event of a Wi-Fi based Wireless Sensor Network management exploitation using Linux board Raspberry pi and Internet of Things technology using ESP8266 Wi-Fi module.

Pros: ESP8266 solution for WSN support most features than other computing modules like RN-131c and Arduino shields

Cons: Arduino shield have better software support and documentation. Sparkfun and Adafruit are better than programming ESP8266 using Arduino IDE.

1.3 Project Undertaken

1.3.1 Problem Definition

Design a system to prevent road accidents by locating nearby vehicles. It calculates distance of adjacent vehicles to warn the driver of potential collision along with the direction.

1.3.2 Scope Statement

The system deals with the vehicles that may be in danger during blind turns and in the presence of fog. In case the vehicle is at a blind turn, the system makes a judgement of direction and speed of nearby vehicles. It also provides assistance to have control over own vehicle.

In case of presence of fog, the system notifies the driver about the neighboring vehicles and prohibits any possible collision.

1.4 Organization of Report

The report starts with the introduction of Accident Prevention System stating related work and project scope. The second chapter includes software requirement specifications which briefly describes the idea of what is to be done. It also includes project process modelling, cost and effort estimates and timeline. The report proceeds with IDEA Matrix, mathematical model, feasibility analysis and UML representations in the third chapter. The fourth chapter mentions the implementation and coding phase. It highlights the operational details and the major classes in the code. The fifth chapter discusses the testing methods viz. unit testing, integration testing and acceptance testing. The sixth chapter discusses the results of the system. The next chapter concludes the report and speaks briefly about the realizable future work. References are mentioned at the end of the report.

Chapter 2

PROJECT PLANNING AND MANAGEMENT

2.1 Detail System Requirement Specification (SRS)

2.1.1 System Overview

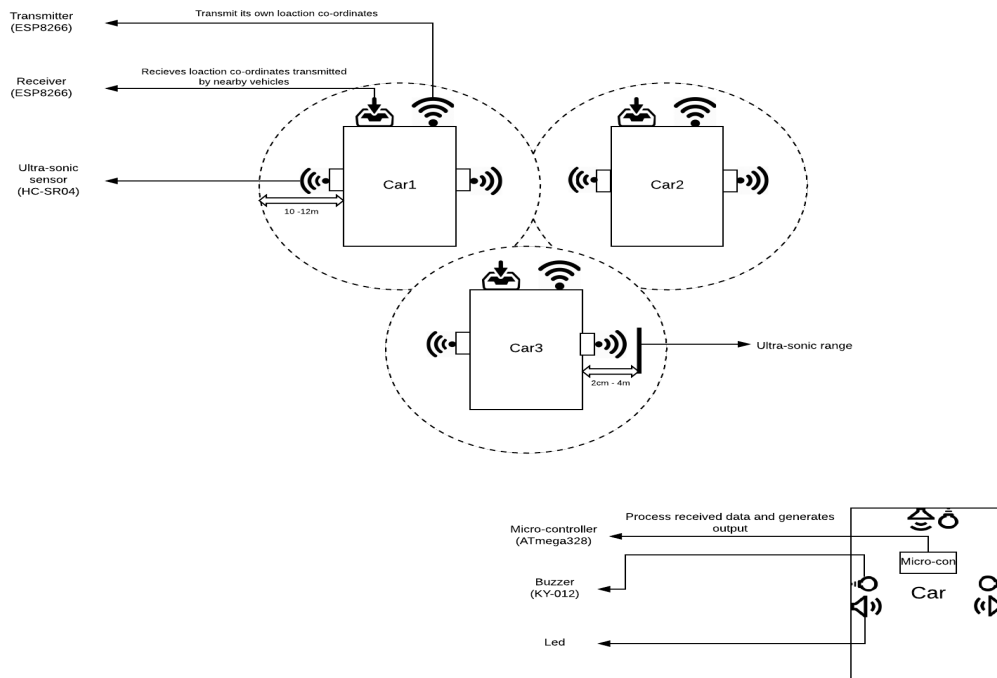


Figure 2.1: External Overview

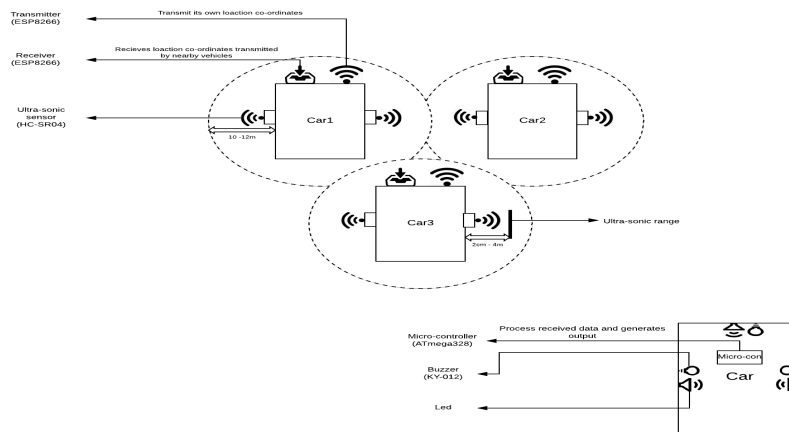


Figure 2.2: Internal Overview

2.1.2 Functional Requirements

The system should provide three main features:

- Exchanging location through beacons
- Calculating distance and direction
- Identifying potential threats and trigger alerts

System feature 1: Exchanging Locations

The exchange of locations between the nearby vehicles would be a continuous process which does not require any user intervention. A GPS module generates real time GPS location which is further transmitted and received via beacons as SSID. Unique identification of beacons would be done with the help of MAC addresses.

System feature 2: Distance and Direction Evaluation

Distance and direction calculation will be a high priority feature as it is the major step for threat analysis. This would be a continuous process as well and does not require any user intervention. The transmitted GPS locations are used in reference with the vehicle's location to calculate distance and direction.

System feature 3: Threat Analysis and Alert Generation

The user has a role to play after generation of alerts. He should respond to the alerts by taking appropriate actions like applying brakes, changing direction etc. It analyzes for the threats on the basis of the information obtained by calculation and notifies driver or user with instructions and details.

2.1.3 Non-Functional Requirements

- Reliability: There is no or minimal human intervention in the system which will make it highly reliable.
- Availability: The system should be provided pre-installed by car manufacturers so that every vehicle is equipped with it.
- Adaptability: In spite of change in traffic density, APS should adjust its threshold values and continue to function smoothly.
- Security: There is no regulation against sharing of a vehicle's location, which is the only requirement of the system.
- Robustness: A sudden change in traffic density should not affect the APS.
- Performance: The traffic density at different roads and at different time varies a lot. APS should not be affected by a change in traffic and should keep on functioning smoothly.

2.1.4 Deployment Environment

Hardware Requirements

- Beacons: ESP8266 module
- Receiver: ESP8266 module
- Ultrasonic sensors
- Arduino UNO
- Buzzer and LEDs
- GPS SIM28
- Magnetometer HMC5883L
- 3.5" LED Screen

Software Requirements

- Arduino IDE

Platform Requirements

- C

2.1.5 External Interface Requirements

User Interfaces

The user will have a screen as the interface on which the instructions and the details of the threat would be mentioned. The user has the privilege to adjust the intensity of sound according to his desire.

Hardware Interfaces

Arduino, GPS sensors, ultrasonic sensors and network interface are some hardware interfaces used for this system.

Software Interfaces

Arduino IDE is the main software interface for this product.

Communication Interfaces

Beacon frame is one of the management frames in IEEE 802.11 based WLANs. It contains all the information about the network. Beacon frames are transmitted periodically, they serve to announce the presence of a wireless LAN and to synchronize the members of the service set. Beacon frames are transmitted by the access point (AP) in an infrastructure basic service set (BSS). Communication in the system is done via beacons. It will carry the GPS location information of each vehicle and announce its presence.

2.2 Project Process Modelling

Waterfall model will be used for the development of project. The steps involved in the Waterfall Process Model is

- Requirement analysis- All the software and hardware requirements are recognized and gathered at this stage.

- Design- The model is actually designed.
- Implementation- The model is programmed at this stage
- Verification- Testing of the program is carried out at this stage
- Maintenance- If any bug are found, they are corrected.
- Deployment- The software is deployed for use.

2.3 Cost & Efforts Estimates

It begins with the decomposition of a project or application into its data and transactional functions. The data functions represent the functionality provided to the user by attending to their internal and external requirements in relation to the data, whereas the transactional functions describe the functionality provided to the user in relation to the processing this data by the application.

1. Internal Logical File (ILF)

2. External Interface File (EIF) The transactional functions are:

1. External Input (EI)

2. External Output (EO)

3. External Inquiry (EQ)

$$EI = 2$$

$$EO = 4$$

$$EQ = 1$$

$$ILF = 5$$

$$ELF = 2$$

The degree of complexity (simple, average or complex) was evaluated for each component.

Table 2.1: Degree of complexity for EO

Relative Element Types	Data Element Type		
	1-5	6-15	>15
0-1	Low	Low	Average
0-3	Low	Average	High
>3	Average	High	High

Table 2.2: Degree of Complexity for ILF

Relative Element Types	Data Element Type		
	1-5	6-15	>15
0-1	Low	Low	Average
2-5	Low	Average	High
>5	Average	High	High

Table 2.3: Predefined Weights

Rating	Values				
	EO	EQ	EI	ILF	ELF
Low	4	3	3	7	5
Average	5	4	4	10	7
High	6	5	6	15	10

From above table we get values of data functions and transaction functions

Calculation of Unadjusted Function Point (UFP)

$$UFP = \sum_{i=1}^n \sum_{j=1}^m Count_{i,j} * Weight_{i,j}$$

$$UFP = (2 * weight_{EI}) + (4 * weight_{EO}) + (1 * weight_{EQ}) + (5 * weight_{ILF}) + (2 * weight_{ELF})$$

$$UFP = (2 * 4) + (4 * 3) + (1 * 3) + (5 * 7) + (2 * 5) = 87$$

Calculation of Final Function point using formula

$$FinalFP = UFP * (0.65 + (VAF * 0.01))$$

Where, VAF = Value Adjustment factor Total VAF = 37

Table 2.4: Value Adjustment factor

	VAF's	Grade
1.	Data communications	5
2.	Distributed data processing	0
3.	Performance	4
4.	Heavily utilized configuration	3
5.	Transaction rate	5
6.	On-line data entry	0
7.	End-user efficiency	4
8.	On-line update	0
9.	Complex processing	1
10.	Reusability	4
11.	Installations ease	4
12.	Operational ease	4
13.	Multiple sites/organization	0
14.	Facilitate change	3

Therefore $FP = 68 * (0.65 + (37 * 0.01))$ $FP = 69.36$ Effort Calculation using lines of source code (SLOC) $Effort = FP * SLOC$ (as per Language) $Effort = 69.36 * 46$ $Effort = 3190.56$ hours 4As 160 days (20 hours/day) 4As 5 months

2.4 Project Scheduling

2.4.1 Time Line Chart

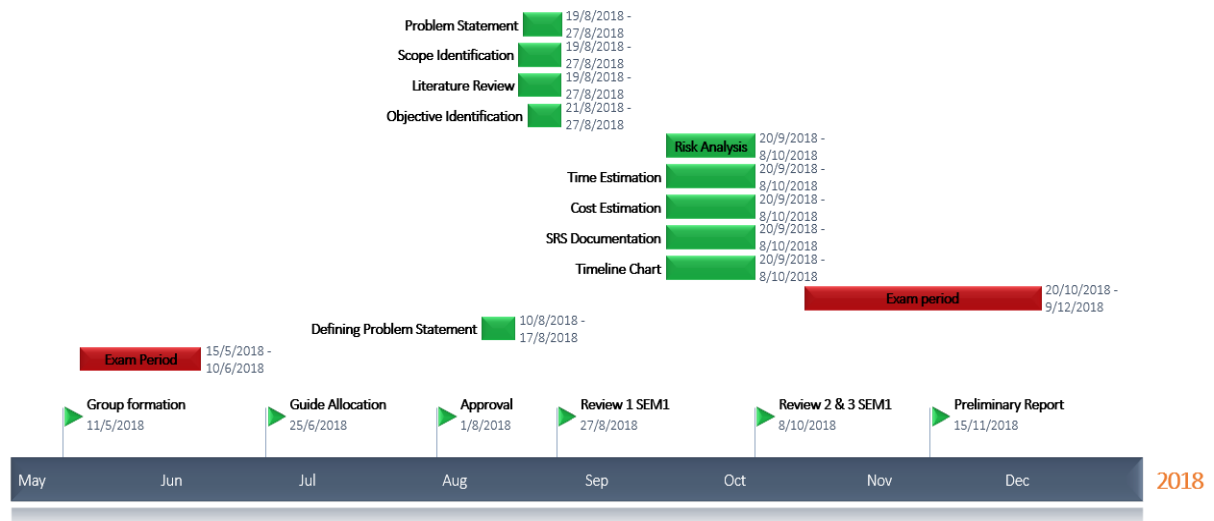


Figure 2.3: Time Line Chart (May'18 - December'18)

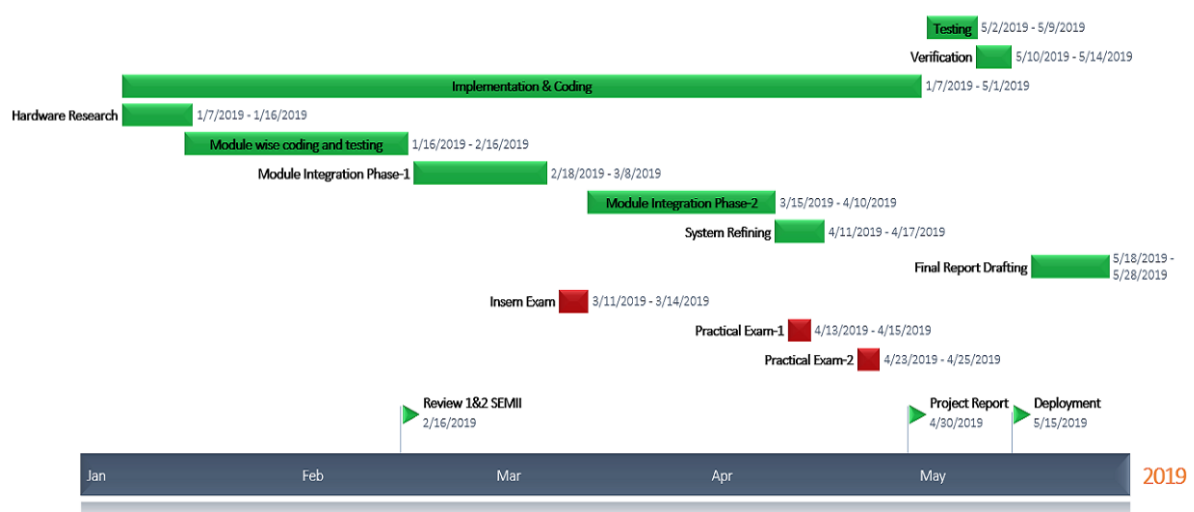


Figure 2.4: Time Line Chart (January'19 - May'19)

Chapter 3

ANALYSIS & DESIGN

3.1 IDEA Matrix

Table 3.1: IDEA Matrix

I	Increase	The system focuses on increasing road safety
	Improve	Improves driving experience
	Integrate	Integrates 3 modules namely: GPS, Ultrasonic sensors, and Beacons
D	Deduce	Threats are been deduced by the system
	Deliver	Delivers Accident Prevention System
	Decrease	Road accidents
E	Educate	User needs to understand the instructions provided by the system
	Evaluate	Evaluates the possible threats using GPS and Ultrasonic sensors
	Eliminate	This system helps in eliminating the risk of road accidents
A	Accelerate	Accelerates decision making during potential threats
	Associate	The system associates with threat detection system with alert generation system
	Avoid	Security risks are been avoided as no personal information is being used

3.2 Mathematical Model

Let S be the Accident Prevention System

$$S = I, O, F, Success, Fail, Limitation$$

where,

$$I = I1, I2, I3$$

$$O = O1, O2, O3, O4, O5$$

$$F = F1, F2, F3, F4, F5$$

$$Success = Success1$$

$$Fail = Fail1$$

$$Limitation = Lim1$$

Let I be the inputs to S

$$I = I1, I2, I3,$$

where,

$$I1 = IgnitionOn|Off$$

$$I2 : SelfLocationLatitude, Longitude$$

$$I3 : UltrasonicSensorInputDistance, Direction$$

Let O be the outputs of S

$$O = O1, O2, O3, O4, O5,$$

where,

$O1 : \text{NearbyLocationLatitude, Longitude}$

$O2 : \text{RelativeDistance}$

$O3 : \text{PossibleThreatYes|No}$

$O4 : \text{ThreatDirection}$

$O5 : \text{Instruction}$

Let F be functions in S

$F = F1, F2, F3, F4, F5,$

where,

$F1 : \text{checkForBeacons}(I1) - O1$

$F2 : \text{transmitSelfLocation}(I1, I2)$

$F3 : \text{relativeDistanceCalculation}(I2, O1) - O2$

$F4 : \text{thresholdConfirmation}(I3, O2) - O3$

$F5 : \text{generateAlert}(O3) - O4, O5$

$$Success = Success1,$$

where,

$$Success1 = Appropriatealertgeneration$$

$$Failure = Fail1,$$

where $Fail1 = Hardwarefailure$

$$Limitation = Lim1,$$

where $Lim1 : Installedinallvehicles$

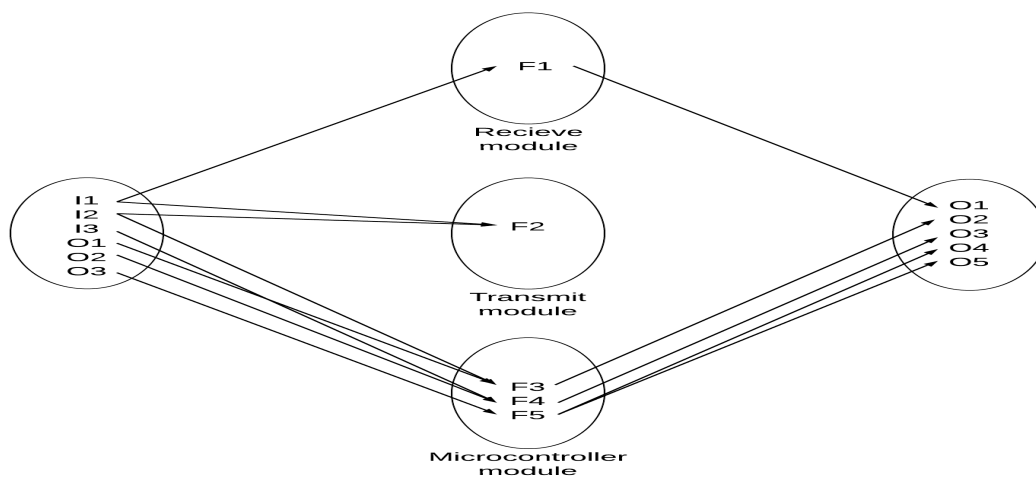


Figure 3.1: Mapping Diagram

3.3 Feasibility Analysis

The system consists of exchanging locations, distance and direction evaluation, and threat analysis and alert generation. The main requirement for a problem to be NP-Complete is that it should be solvable in polynomial time. All the above functions in the system satisfy this condition and hence the system is feasible.

3.4 UML Diagrams

3.4.1 Use Case Diagram

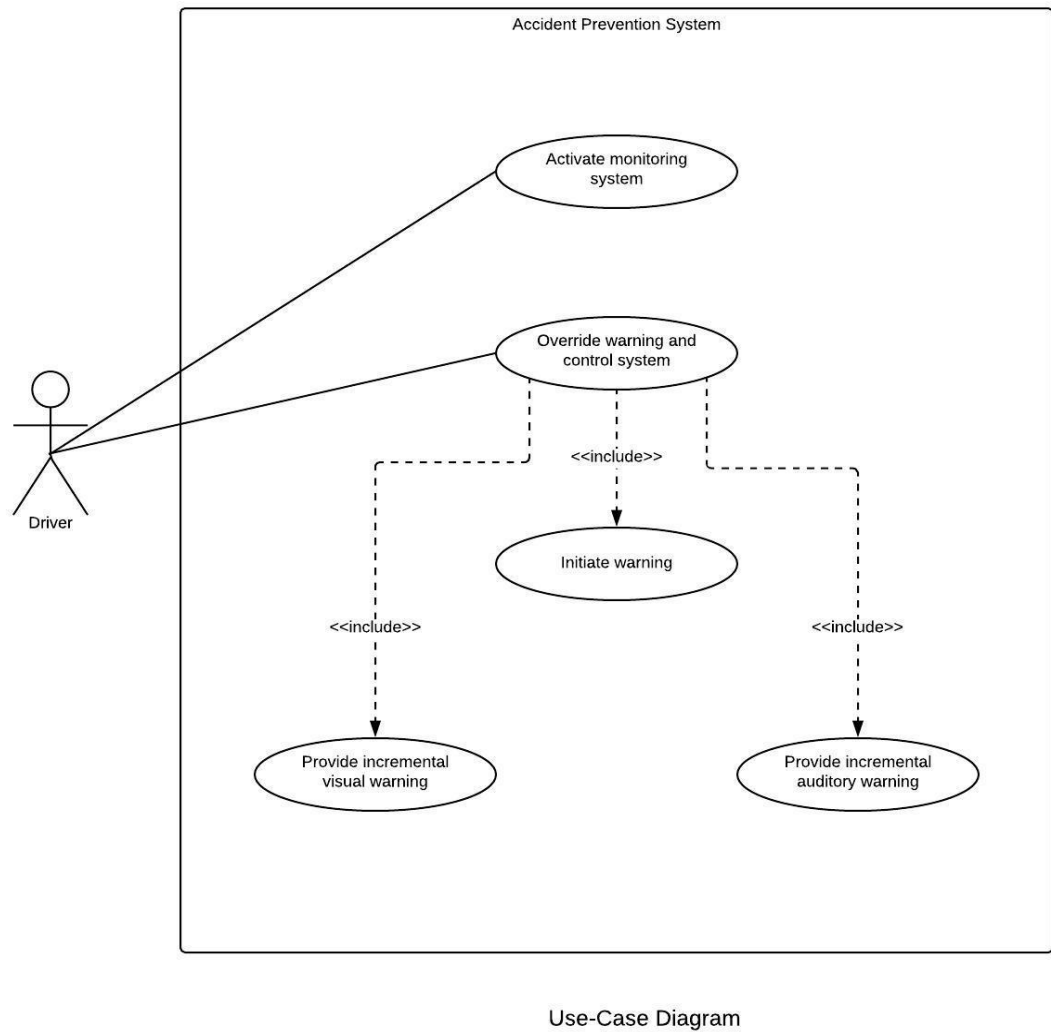


Figure 3.2: Use-case Diagram

3.4.2 Activity Diagram

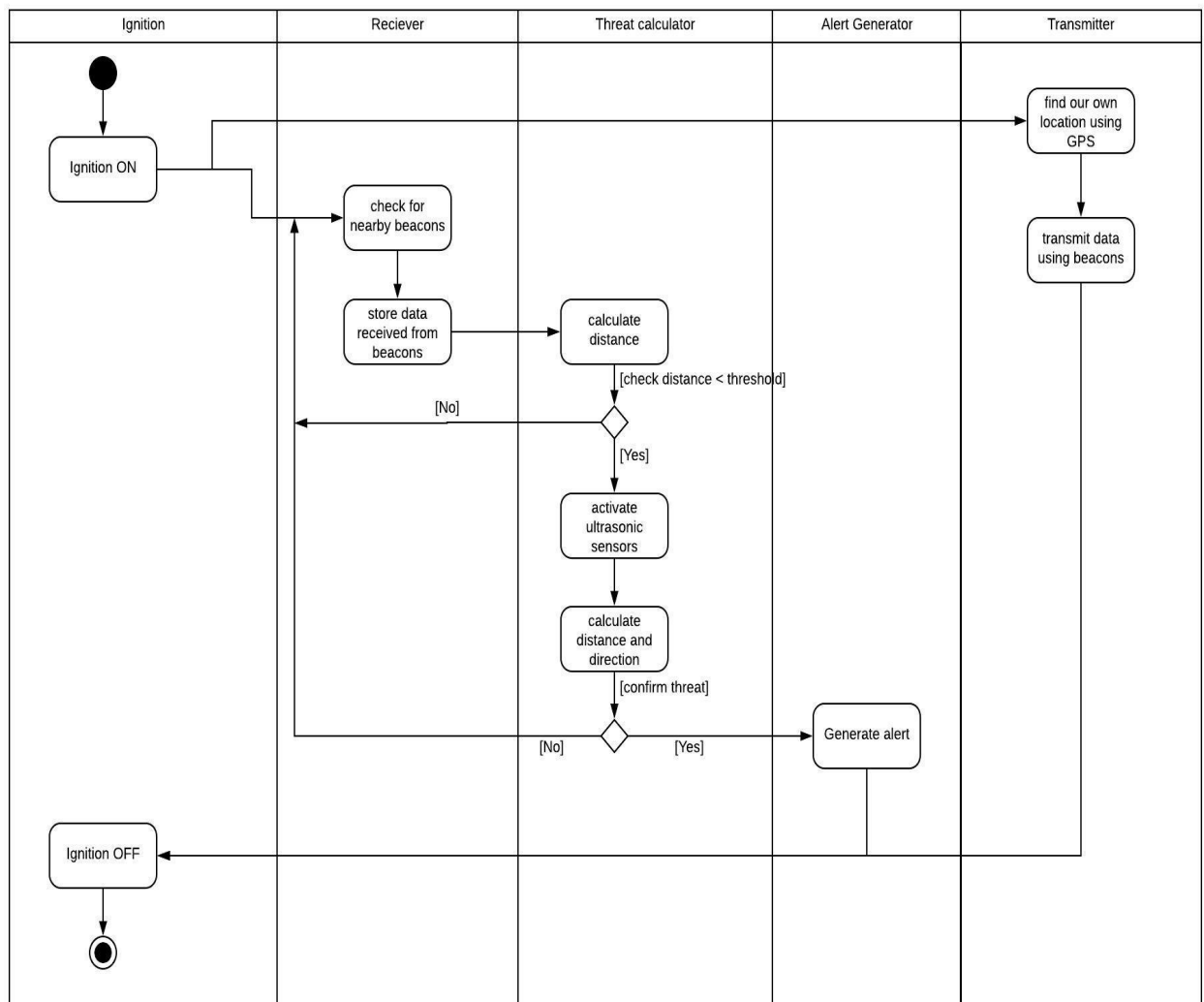


Figure 3.3: Activity Diagram

3.4.3 Sequence Diagram

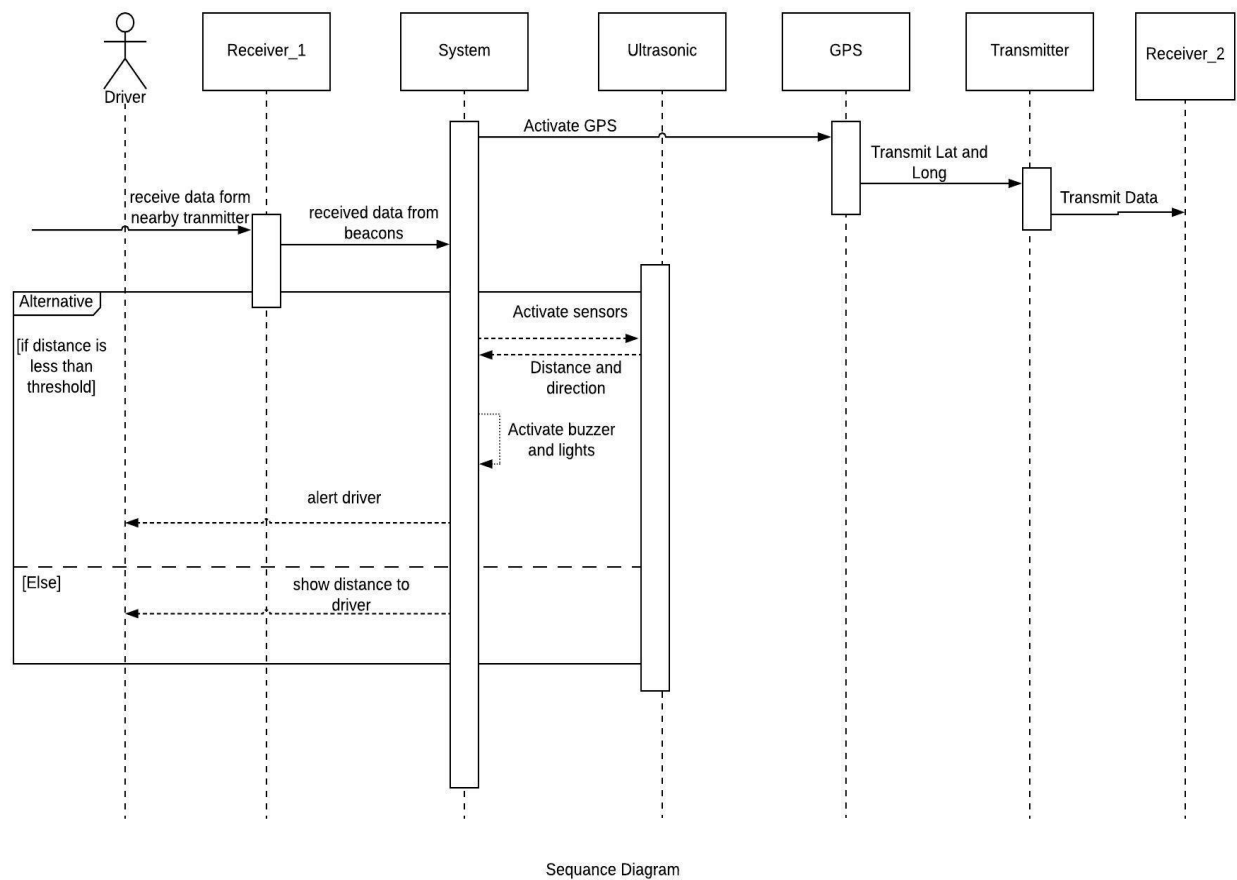
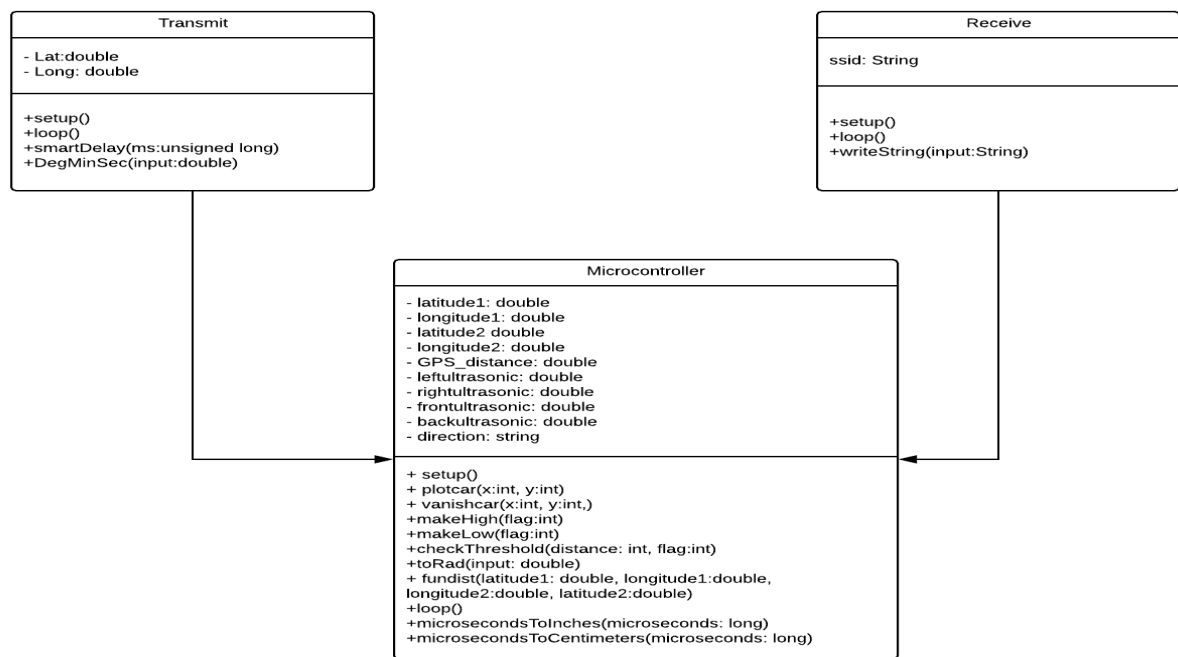


Figure 3.4: Sequence Diagram

3.4.4 Class Diagram



Class Diagram

Figure 3.5: Class Diagram

3.4.5 Data Flow Diagram

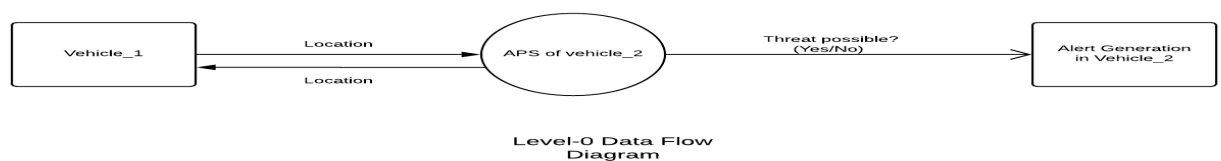
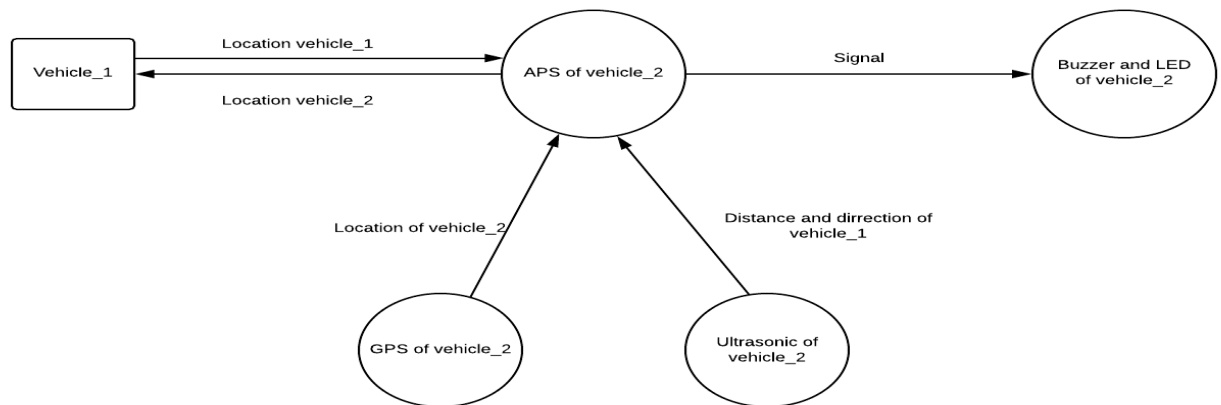
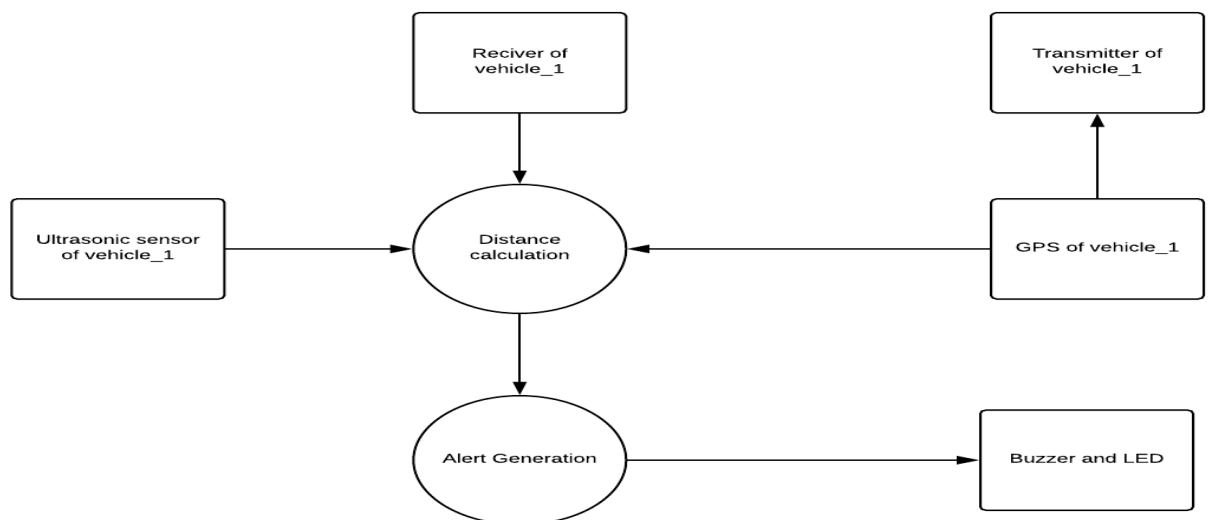


Figure 3.6: Level-0 Data Flow Diagram



Level-1 Data Flow Diagram

Figure 3.7: Level-1 Data Flow Diagram



Level-2 Data Flow Diagram

Figure 3.8: Level-2 Data Flow Diagram

3.4.6 Deployment Diagram

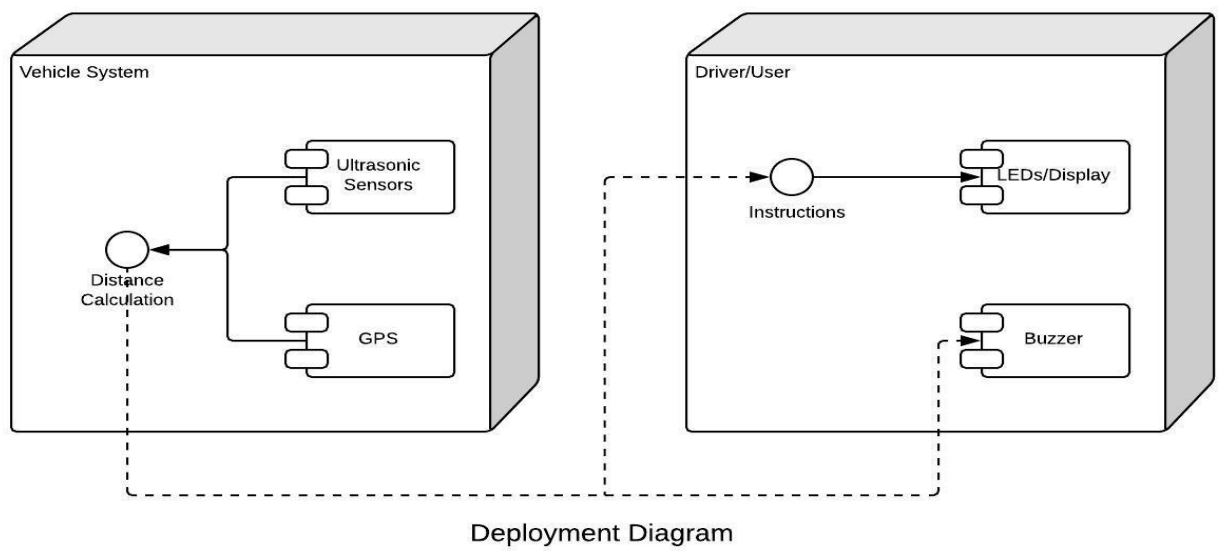


Figure 3.9: Deployment Diagram

Chapter 4

IMPLEMENTATION AND CODING

4.1 Introduction

The main aim of APS is to detect potential threats and generate alerts. For this, the function that it has to perform is calculation of distance and determining the direction of threat.

4.2 Operational Details

The APS has latitude and longitude values of itself as well as nearby vehicles. This enables the system to obtain distance between itself and other vehicles. The distance is checked with threshold value. If it surpasses the threshold value, ultrasonic sensors are used to confirm the distance with better efficiency.

The magnetometer helps in finding the direction of the threat by calculating relative orientations. This tells the system which direction the threat is approaching from.

As soon as a confirmation is received from the ultrasonic sensors, the LEDs and buzzers are switched on as determined by the magnetometer. They help in notifying the driver about the detected threats and their direction.

A small display is used in which we can see the vehicles in the neighboring area. It allows the driver to take quick action just by looking at screen and analyzing threats.

4.3 Major Classes

There are 3 major classes working in this system viz. Transmit, Receive, and Microcontroller .

The Transmit module acquires its GPS location and transmits it to other vehicles along with its MAC Id. This module keeps on updating the values received from the GPS module.

The Receive module receives the GPS locations of nearby vehicles. It then calculates the distance between them and direction of each vehicle.

The Microcontroller starts its work when the system has its own and nearby vehicle's locations. It first calculates the distance from GPS locations. If this distance has crossed the threshold value, it again verifies the threat with the help of ultrasonic sensors. Ultrasonic sensors have a better efficiency and hence avoid false alarms.

When ultrasonic sensors send an affirmative signal regarding threat, the alert system is activated. The LEDs start to blink and buzzer is switched on. For better understanding the environment and making quick decisions, the driver may take look at the screen which will plot the neighboring area.

4.4 Code Listing

4.4.1 Transmit

```
#include "TinyGPS++.h";
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

SoftwareSerial GPS_SoftSerial(5, 4); /* (Rx, Tx) */

TinyGPSPlus gps;

double temp_lat, temp_long;

volatile float minutes, seconds;
volatile int degree, secs, mins;

int n=0;
void setup() {
    Serial.begin(9600); /* Define baud rate for serial communication */
    GPS_SoftSerial.begin(9600); /* Define baud rate for software serial c
```

```
WiFi.mode(WIFI_AP);
IPAddress myIP = WiFi.softAPIP();
WiFi.persistent(false);
}

void loop() {
    smartDelay(1000); /* Generate precise delay of 1ms */
    unsigned long start;
    double lat_val, lng_val, alt_m_val;
    uint8_t hr_val, min_val, sec_val;
    bool loc_valid, alt_valid, time_valid;
    lat_val = gps.location.lat(); /* Get latitude data */
    loc_valid = gps.location.isValid(); /* Check if valid location
    lng_val = gps.location.lng(); /* Get longitude data */
    alt_m_val = gps.altitude.meters(); /* Get altitude data in meters
    alt_valid = gps.altitude.isValid(); /* Check if valid altitude
    hr_val = gps.time.hour(); /* Get hour */
    min_val = gps.time.minute(); /* Get minutes */
    sec_val = gps.time.second(); /* Get seconds */
    time_valid = gps.time.isValid(); /* Check if valid time data
    if (!loc_valid)
    {
        Serial.print("Latitude : ");
        Serial.println("*****");
        Serial.print("Longitude : ");
        Serial.println("*****");
    }
    else
    {
        String myString=String(n);

        temp_lat = lat_val;
        temp_long = lng_val;
        String latString = String(temp_lat,6);
        String lngString = String(temp_long,6);

        String ssid1 = String("()("+latString+", "+lngString+")");
        int len = ssid1.length();
        char copy[25];
```

```
        ssid1.toCharArray(copy, len+2);

        Serial.println("(" + latString + "," + lngString + ")");
        WiFi.softAP(copy);
        n+=1;
        delay(750);

    }
    if (!alt_valid)
    {
        Serial.print("Altitude : ");
        Serial.println("*****");
    }
    if (!time_valid)
    {
        Serial.print("Time : ");
        Serial.println("*****");
    }
}

static void smartDelay(unsigned long ms)
{
    unsigned long start = millis();
    do
    {
        while (GPS_SoftSerial.available()) /* Encode data read from GPS w
            gps.encode(GPS_SoftSerial.read());

    } while (millis() - start < ms);
}

void DegMinSec( double tot_val) /* Convert data in decimal degrees in
{
    degree = (int)tot_val;
    minutes = tot_val - degree;
    seconds = 60 * minutes;
    minutes = (int)seconds;
    mins = (int)minutes;
    seconds = seconds - minutes;
```

```
seconds = 60 * seconds;  
secs = (int)seconds;  
}
```

4.4.2 Receive

```
#include "ESP8266WiFi.h"
```

```
void setup() {  
    Serial.begin(9600);  
    WiFi.mode(WIFI_STA);  
    WiFi.disconnect();  
    delay(100);  
}
```

```
void loop() {  
    int n = WiFi.scanNetworks();  
    if (n == 0) {  
        Serial.println("no networks found");  
    }  
    else {  
        for (int i = 0; i < n; ++i) {  
  
            String mac=WiFi.BSSIDstr(i);  
            if (mac!="DE:4F:22:3B:10:3C")  
            {  
                writeString(WiFi.SSID(i));  
                writeString("\n");  
            }  
            delay(10);  
        }  
    }  
    Serial.println("");  
    delay(1000);  
}
```

```
void writeString(String stringData) { // Used to serially push out a St
```

```
    for (int i = 0; i < stringData.length(); i++)
```

```

    {
        Serial.write(stringData[i]);    // Push each char 1 by 1 on each loop
    }

} // end writeString

```

4.4.3 Micro-controller

```

#include <SoftwareSerial.h>
#include <math.h>
#include <Wire.h>
#include <QMC5883L.h>
char getdirections[9][3] = {{ 'N' }, { 'N', 'E' }, { 'E' }, { 'S', 'E' }, { 'S' }, { 'S', 'W' }, { 'W' }, { 'N', 'W' }, { 'N' } };
#define LCD_CS A3 // Chip Select goes to Analog 3
#define LCD_CD A2 // Command/Data goes to Analog 2
#define LCD_WR A1 // LCD Write goes to Analog 1
#define LCD_RD A0 // LCD Read goes to Analog 0
#define LCD_RESET A4 // Can alternately just connect to Arduino's reset
#include <SPI.h> // f.k. for Arduino-1.5.2
#include "Adafruit_GFX.h" // Hardware-specific library
#include <MCUFRIEND_kbv.h>
#include <math.h>
MCUFRIEND_kbv tft;
#define BLACK 0x0000
#define BLUE 0x001F
#define RED 0xF800
#define GREEN 0x07E0
#define CYAN 0x07FF
#define MAGENTA 0xF81F
#define YELLOW 0xFFE0
#define WHITE 0xFFFF
#define RADIUS 10
int my_x, my_y, h, w;
int x[10]={0}, y[10]={0}, counter= -1;
double val = M_PI/180;
QMC5883L compass;

double endlat1 = 19.075983, endlng1 = 72.877655; //Mumbai

```

```
double startlat2 = 18.520430, startlng2 = 73.856743; //Pune

double lat1 ,lng1;
double lat2 ,lng2;

const int pingPinF = 25; // Trigger Pin of Ultrasonic Sensor
const int echoPinF = 27; // Echo Pin of Ultrasonic Sensor

const int pingPinR = 46; // Trigger Pin of Ultrasonic Sensor
const int echoPinR = 48; // Echo Pin of Ultrasonic Sensor

const int pingPinB = 47; // Trigger Pin of Ultrasonic Sensor
const int echoPinB = 49; // Echo Pin of Ultrasonic Sensor

const int pingPinL = 33; // Trigger Pin of Ultrasonic Sensor
const int echoPinL = 35; // Echo Pin of Ultrasonic Sensor

const int LEDF = 29;
const int LEDB = 51;
const int LEDL = 37;
const int LEDR = 50;

void setup ()
{
  Serial3.begin(9600);
  Serial1.begin(9600);
  Serial.begin(9600);
  Wire.begin();
  compass.init();
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
  pinMode(LEDF,OUTPUT);
  pinMode(LEDB,OUTPUT);
  pinMode(LEDL,OUTPUT);
  pinMode(LEDR,OUTPUT);

  // Display setup
  uint16_t ID = tft.readID();
```



```

    if (ID == 0xD3D3) ID = 0x9481;
    tft.begin(ID);
    tft.fillScreen(BLACK);
    h = tft.height();
    w = tft.width();
    tft.setCursor((w-175)/2,(h-50)/2);
    tft.setTextSize(5);
    tft.println("Jaguar");
    tft.setTextSize(2);
    tft.println("          Inspired");
    delay(5000);

    tft.fillScreen(BLACK);
    tft.setCursor(85, 0);
    tft.setTextColor(WHITE);    tft.setTextSize(3);
    tft.println("!!!MAP!!!");
    int x1, x2, y1, y2;
    y1 = y1 + h/10;
    x1 = x1 + w/5;
    y2 = y1 + h - h/5;
    x2 = x1;
    tft.drawLine(x1,y1,x2,y2,WHITE);
    x1 = w-w/5;
    x2 = x1;
    tft.drawLine(x1,y1,x2,y2,WHITE);
    my_x = w/2;
    my_y = h/2;
    tft.fillCircle(my_x,my_y,RADIUS,BLUE);
}

void plotcar(int x, int y, uint16_t color) {
    tft.fillCircle(x,y,RADIUS,color);
    delay(10);
}

void vanishcar(int x, int y, uint16_t color) {
    tft.fillCircle(x,y,RADIUS,color);
}

void makeHigh(int flag)

```

```
{
    if ( flag ==0)
        digitalWrite (LEDL, HIGH);
    else if ( flag ==1)
        digitalWrite (LEDR, HIGH);
    else if ( flag ==2)
        digitalWrite (LEDF, HIGH);
    else if ( flag ==3)
        digitalWrite (LEDB, HIGH);
}
void makeLow(int flag)
{
    if ( flag ==0)
        digitalWrite (LEDL, LOW);
    else if ( flag ==1)
        digitalWrite (LEDR, LOW);
    else if ( flag ==2)
        digitalWrite (LEDF, LOW);
    else if ( flag ==3)
        digitalWrite (LEDB, LOW);
}
bool checkThreshold(int cm,int flag)
{
    if (cm<10)
    {
        if ( flag ==0)
            makeHigh (0);
        else
            makeHigh (1);
    }
    else
    {
        if ( flag ==0)
            makeLow (0);
        else
            makeLow (1);
    }
}
double toRad(double val)
```

```

{
    return val*3.1452/180;
}
double fundist(double lat1 ,double lng1 , double lat_val , double lng_val)
{
    long double R=6371000;
    double lat_dif=toRad(lat1-lat_val);
    double lng_dif=toRad(lng1-lng_val);
    lat1=toRad(lat1);
    lat_val=toRad(lat_val);
    double a=sin(lat_dif/2)*sin(lat_dif/2)+cos(lat_val)*cos(lat1)*sin(lng
    double c=2*asin(sqrt(a));
    double distance=R*c;
    return distance;
}
void loop(){
    char seq;
    boolean StringReady = false;
    String latString;
    String lngString;

    if (Serial3.available())
    {
        String temp=Serial3.readStringUntil('(');
        temp = Serial3.readStringUntil(',',');
        if(temp.startsWith(")("))
        {
            latString = temp.substring(2);
            lngString = Serial3.readStringUntil(')');
            lat1=latString.toFloat();
            lng1=lngString.toFloat();
            Serial.println("otherslocation: ");
            Serial.println(lat1,6);
            Serial.println(lng1,6);
        }
    }

    if(Serial1.available())
    {

```

```
Serial.println("HELLO");
seq = Serial1.read();
while(seq != '(')
{
    seq = Serial1.read();
}
if(seq == '(')
{
    String temp = Serial1.readStringUntil(',');
    if(temp.startsWith(")("))
    {
        String latitude, longitude;
        latitude = temp.substring(2);
        longitude = Serial1.readStringUntil(')');
        lat2 = latitude.toFloat();
        lng2 = longitude.toFloat();
        Serial.println(lat2, 6);
        Serial.println(lng2, 6);
    }
}
}
double distance;
distance = fundist(lat1, lng1, lat2, lng2);
Serial.println("Distance=" + String(distance) + "meters");

int x1, y1, z1;
compass.read(&x1, &y1, &z1);
float heading = atan2(y1, x1);
float declinationAngle = 0.0037;
heading += declinationAngle;
if(heading < 0)
    heading += 2*PI;
if(heading > 2*PI)
    heading -= 2*PI;
float headingDegrees = heading * 180/M_PI;

Serial.print("x: ");
Serial.print(x1);
Serial.print("    y: ");
```

```
Serial.print(y1);
Serial.print("      z: ");
Serial.print(z1);
Serial.print("      heading: ");
Serial.print(heading);
Serial.print("      Radius: ");
Serial.print(headingDegrees);
Serial.println();

// Direction of the incoming vehicle
double radians;
double y2 = endlng1-startlng2;
double x2 = endlat1-startlat2;

radians = atan2(y2,x2);
double compassreading = radians * (180/M_PI);
int coorindex = round(compassreading/45);
if(coorindex<0)
{
    coorindex += 8;
}
Serial.print("Compass Reading: ");
Serial.print(compassreading);
Serial.print(" - Direction: ");
Serial.print(getdirections[coorindex]);
Serial.println();

// Display code

double d = 50, degree = compassreading, phi = headingDegrees;
int i=0;
if(counter!=-1)
{
    for(i=0;i<=counter;i++)
    {
        vanishcar(x[i],y[i],BLACK);
    }
    counter = -1;
}
```

```
if (phi!=0 || phi!=360)
{
    if (phi <=180)
    {
        degree = degree+phi;
    }
    else //between 180 & 360
    {
        phi = 360 - phi;
        degree = degree - phi;
    }
}
if (degree < -180)
{
    degree = 360 + degree;
}
else if (degree > 180)
{
    degree = degree - 360;
}
if ((int) degree == 0)
{
    counter++;
    x[counter] = my_x;
    y[counter] = my_y - d;
    plotcar(x[counter], y[counter], RED);
}
else if (degree > 0 && degree < 90)
{
    counter++;
    x[counter] = my_x + d*sin(degree*val);
    y[counter] = my_y - d*cos(degree*val);
    plotcar(x[counter], y[counter], RED);
}
else if ((int) degree == 90)
{
    counter++;
    x[counter] = my_x + d;
```

```
        y[counter] = my_y;
        plotcar(x[counter],y[counter],RED);
    }
    else if (degree>90 && degree<180)
    {
        counter++;
        x[counter] = my_x + abs(d*cos((degree-90)*val));
        y[counter] = my_y + abs(d*sin((degree-90)*val));
        plotcar(x[counter],y[counter],RED);
    }
    else if ((int)degree==180)
    {
        counter++;
        x[counter] = my_x;
        y[counter] = my_y + d;
        plotcar(x[counter],y[counter],RED);
    }
    else if (degree<0 && degree>-90)
    {
        degree = degree * -1;
        counter++;
        x[counter] = my_x - d*sin(degree*val);
        y[counter] = my_y - d*cos(degree*val);
        plotcar(x[counter],y[counter],RED);
    }
    else if ((int)degree==-90)
    {
        counter++;
        x[counter] = my_x - d;
        y[counter] = my_y;
        plotcar(x[counter],y[counter],RED);
    }
    else if (degree<-90 && degree>-180)
    {
        degree = degree*-1;
        counter++;
        x[counter] = my_x - abs(d*cos((degree-90)*val));
        y[counter] = my_y + abs(d*sin((degree-90)*val));
        plotcar(x[counter],y[counter],RED);
    }
```

```
}
else if ((int) degree == -180)
{
    counter++;
    x[counter] = my_x;
    y[counter] = my_y + d;
    plotcar(x[counter], y[counter], RED);
}

//ULTRASONIC
long durationL, inchesL, cmL;
long durationR, inchesR, cmR;
long durationF, inchesF, cmF;
long durationB, inchesB, cmB;
pinMode(pingPinL, OUTPUT);
digitalWrite(pingPinL, LOW);
digitalWrite(pingPinL, HIGH);
digitalWrite(pingPinL, LOW);
pinMode(echoPinL, INPUT);
durationL = pulseIn(echoPinL, HIGH);
cmL = microsecondsToCentimeters(durationL);
Serial.print(cmL);
Serial.print("cmL");
Serial.println();
checkThreshold(cmL, 0);

pinMode(pingPinR, OUTPUT);
digitalWrite(pingPinR, LOW);
digitalWrite(pingPinR, HIGH);
digitalWrite(pingPinR, LOW);
pinMode(echoPinR, INPUT);
durationR = pulseIn(echoPinR, HIGH);
cmR = microsecondsToCentimeters(durationR);
Serial.print(cmR);
Serial.print("cmR");
Serial.println();
checkThreshold(cmR, 1);

pinMode(pingPinF, OUTPUT);
```



```
digitalWrite (pingPinF , LOW);
digitalWrite (pingPinF , HIGH);
digitalWrite (pingPinF , LOW);
pinMode(echoPinF , INPUT);
durationF = pulseIn(echoPinF , HIGH);
cmF = microsecondsToCentimeters(durationF);
Serial.print(cmF);
Serial.print("cmF");
Serial.println();
checkThreshold(cmF,2);

pinMode(pingPinB , OUTPUT);
digitalWrite (pingPinB , LOW);
digitalWrite (pingPinB , HIGH);
digitalWrite (pingPinB , LOW);
pinMode(echoPinB , INPUT);
durationB = pulseIn(echoPinB , HIGH);
cmB = microsecondsToCentimeters(durationB);
Serial.print(cmB);
Serial.print("cmB");
Serial.println();
checkThreshold(cmB,4);

}

long microsecondsToInches(long microseconds) {
    return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds) {
    return microseconds / 29 / 2;
}
```

4.5 Screen Shots

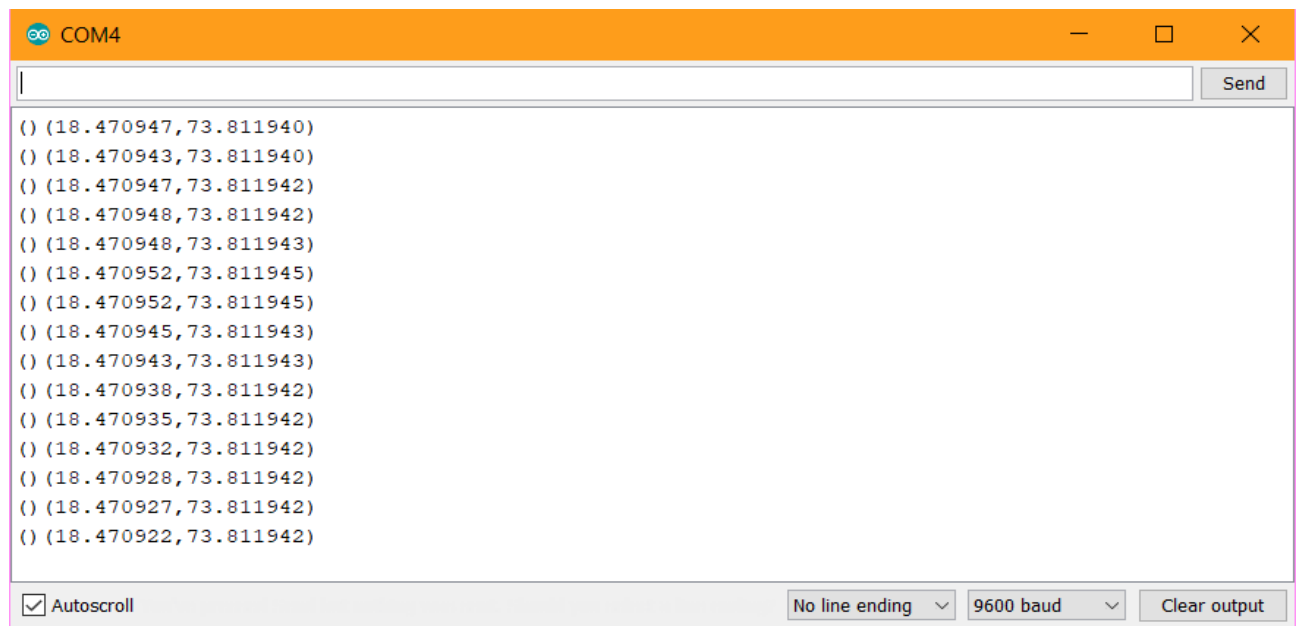


Figure 4.1: Console output of transmitter

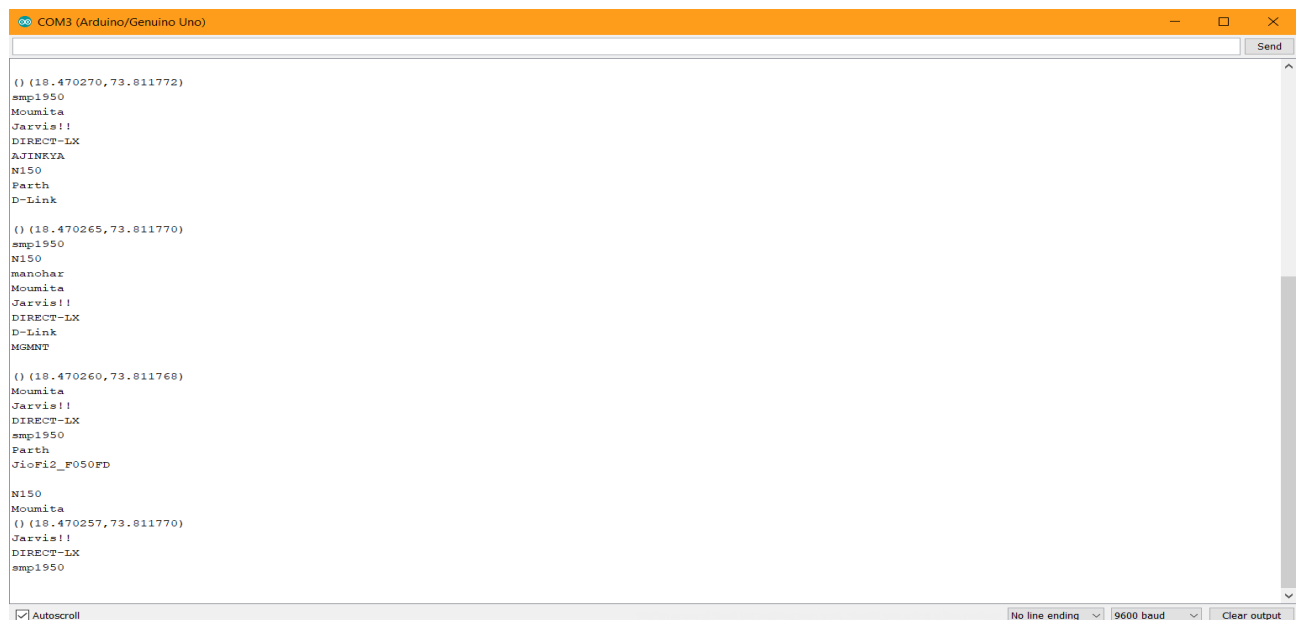
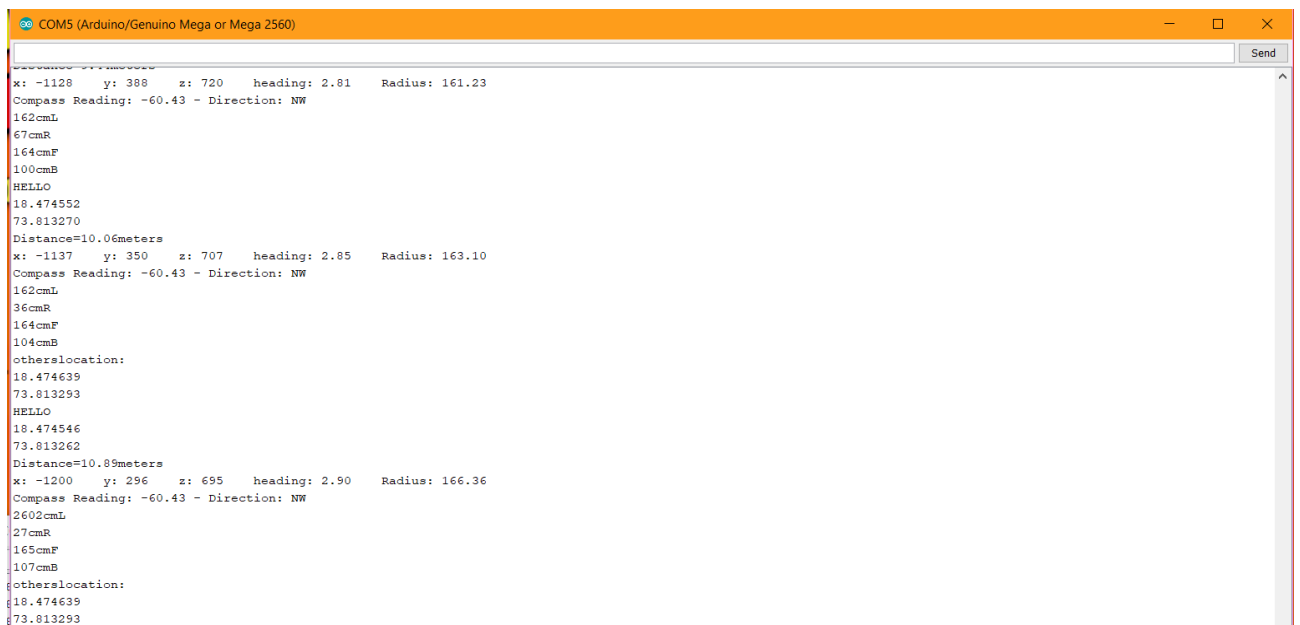


Figure 4.2: Console output of receiver



```
COM5 (Arduino/Genuino Mega or Mega 2560)
x: -1128 y: 388 z: 720 heading: 2.81 Radius: 161.23
Compass Reading: -60.43 - Direction: NW
162cmL
67cmR
164cmF
100cmB
HELLO
18.474552
73.813270
Distance=10.06meters
x: -1137 y: 350 z: 707 heading: 2.85 Radius: 163.10
Compass Reading: -60.43 - Direction: NW
162cmL
36cmR
164cmF
104cmB
otherlocation:
18.474639
73.813293
HELLO
18.474546
73.813262
Distance=10.89meters
x: -1200 y: 296 z: 695 heading: 2.90 Radius: 166.36
Compass Reading: -60.43 - Direction: NW
2602cmL
27cmR
165cmF
107cmB
otherlocation:
18.474639
73.813293
```

Figure 4.3: Console output of micro controller

Chapter 5

TESTING

5.1 Unit Testing

Unit testing is the first level of software testing where individual units are tested by the developers themselves. Here we have three main modules or units:

1. Distance Calculation Module
2. Threat Analysis Module
3. Alert Generation Module

In unit testing we will test all the modules individually.

Table 5.1: Test cases for unit testing

ID	Test Case	Input	Expected O/P	Observed O/P	Remark
1	Distance Calculation, input two GPS locations	Sample Lat-18.47010 Long-73.82952 Sample Lat-18.47221 Long- 73.82952	230 m		
2		Sample Lat-18.47010 Long-73.82952 Sample Lat-18.47020 Long-73.82952	10m		
3		Sample Lat-18.47010 Long-73.82952 Sample Lat-18.47040 Long-73.82952	30 m		
4	Potential Threat or not, input is distance with GPS and sensors	GPS:5m, Sensor:4m	No		
5		GPS:1m, Sensor:0.5m	Yes		
6		GPS:9m, Sensor:7m	No		
7	Alerts or generated properly or not, input is direction of threat	Left	Sound+Left signal		
8		Right	Sound+Right Signal		
9		Rear	Sound+Rear Signal		

5.2 Integration Testing

Integration testing is the second level of testing, here first two modules which are distance calculation and threat analysis will be integrated and tested. The test cases for it are as follows:

Table 5.2: Test cases for integration testing-I

ID	Test Case	Input	Expected O/P	Observed O/P	Remark
1	Inputs: GPS location, Ultrasonic sensor readings. Output: Whether the vehicle is a potential threat or not.	Sample Lat-18.47010 Long-73.82952 Sample Lat-18.47221 Long- 73.82952 Sensor Reading: 0.5 m	Yes		
2		Sample Lat-18.47010 Long-73.82952 Sample Lat-18.47020 Long-73.82952 Sensor Reading: 9 m	No		
3		Sample Lat-18.47010 Long-73.82952 Sample Lat-18.47040 Long-73.82952 Sensor Reading: 6 m	No		

After this the third module will also be combined and checked whether the output generated is appropriate or not.

Table 5.3: Test cases for integration testing-II

ID	Test Case	Input	Expected O/P	Observed O/P	Remark
1	Inputs: GPS location, Ultrasonic sensor readings. Output: Whether the vehicle is a potential threat or not.	Sample Lat-18.47010 Long-73.82952 Sample Lat-18.47221 Long- 73.82952 Sensor Reading: 0.5 m	Threat from Behind, Speed Up or Give Side		
2		Sample Lat-18.47010 Long-73.82952 Sample Lat-18.47020 Long-73.82952 Sensor Reading: 9 m	No		
3		Sample Lat-18.47010 Long-73.82952 Sample Lat-18.47040 Long-73.82952 Sensor Reading: 6 m	No		

5.3 Acceptance Testing

Acceptance testing is the last phase of software testing, it will be performed by the development team which is known as alpha testing and then it will also be performed by the project guide which can be termed as beta testing. The main issues that will be tested in acceptance testing are:

- The system gives proper alerts about the potential threats. The system provides alerts in a reasonable time i.e. the alerts should be generated at least 1 second before the collision. The alerts generated by the system should be in a user friendly manner such that the driver is able to understand the direction of the potential threat easily and be able to take required

actions for it such as applying brakes, speeding up, changing direction, giving sides etc.

Chapter 6

RESULTS AND DISCUSSION

6.1 Prototype Snapshots

6.2 Results

For the demonstration purpose, we made a prototype using the required hardware. The response time of each hardware was recorded. The distance calculation with GPS had was done in an average time of 0.9 seconds. The average response time of Magnetometer and Ultrasonic sensors was found to be 2.3 and 1.2 seconds respectively.

Table 6.1: Response time of GPS module

Distance Calculated by GPS	Response Time
234.84 m	1.1 sec
11.04 m	0.8 sec
33.34 m	0.9 sec

Table 6.2: Response time of Ultrasonic sensor

Ultrasonic Sensor Position	Response Time
Left	1.6 sec
Right	1.3
Front	1.2 sec
Back	0.8 sec

Table 6.3: Values given by Magnetometer

X	Y	Z	Degree	Relative Direction	Response Time
157	-1580	466	-60.43	North-West	2 sec
48	-1555	197	-60.43	North-West	2 sec
726	-1386	360	-60.43	North-West	3 sec

6.3 Discussions

The Accident Prevention System is observed to have a quick response time. The main reason behind this feature is lack of necessity to establish connection. The system works with the vehicles available in the viable radius. This makes the aim of a rapid system attainable. The use of display to plot nearby region and accompanying instructions allow the driver to take quick decisions.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Thus the APS can be implemented to reduce accidents due to blind turns or presence of fog. The APS uses GPS module to detect location and calculates relative distance. If this distance crosses the threshold value, ultrasonic sensors confirm the threat and generate alerts accordingly.

The ultrasonic sensor determines the distance and magnetometer determines the direction of the vehicle which can pose a danger. The drivers of both the vehicles are warned about each other with help of buzzers, LEDs and screen. The screen maps the surrounding area and also displays warnings. Mapping of surrounding area on screen makes it simpler for the driver to make decisions to avoid wreckage.

7.2 Future Work

Currently the Accident Prevention System is developed mainly for regions with high fog density and roads with blind turns. For the purpose of demonstration, the APS is built with ordinary hardware.

With further use of efficient hardware it can be made usable for regular circumstances. We can use latest GPS module, magnetometer and better sensors for locating the entities precisely.

We can use high-cost technology like LIDAR (Light Detection and Ranging) and circumvent the use of discrete hardware. This technology gives us the real-time view of the surrounding area.

The actions to be taken in extreme conditions could be displayed on a screen. This would give an extra assistance for the driver to respond to particular threat.

References

[1] Truc D.T. Nguyen, Quang-Bao Huynh and Hoang-Anh Phan, “An Adaptive Beacon-based Scheme for Warning Messages Dissemination in Vehicular Ad-hoc Networks”, International Conference on Advanced Computing and Applications, 2017.

[2] Apurva Rajendra Patkar, Pankaj P. Tasgaonkar, “Object Recognition Using Horizontal Array of Ultrasonic Sensors”, International Conference on Communication and Signal Processing, 2016.

[3] Tejas Thaker, “ESP8266 based Implementation of Wireless Sensor Network with Linux based Web-Server”, 2016 Symposium on Colossal Data Analysis and Networking(CDAN), 2016.

[4] <http://www2.tku.edu.tw/~tkjse/13-4/05-EE9713.pdf>

[5] <https://www.lucidchart.com/>

[6] https://www.authorea.com/users/23901/articles/27506-software-requirements-engineering-for-pedestrian-backup-assist-system-pbs-/_/_show_article

[7] <https://www.softwaremetrics.com/>

[8] <https://cs.uwaterloo.ca/~apidduck/CS846/Seminars/abbas.pdf>

[9] <http://www.easyengineeringclasses.com/>

[10] <https://www.movable-type.co.uk/scripts/latlong.html>

Appendix A

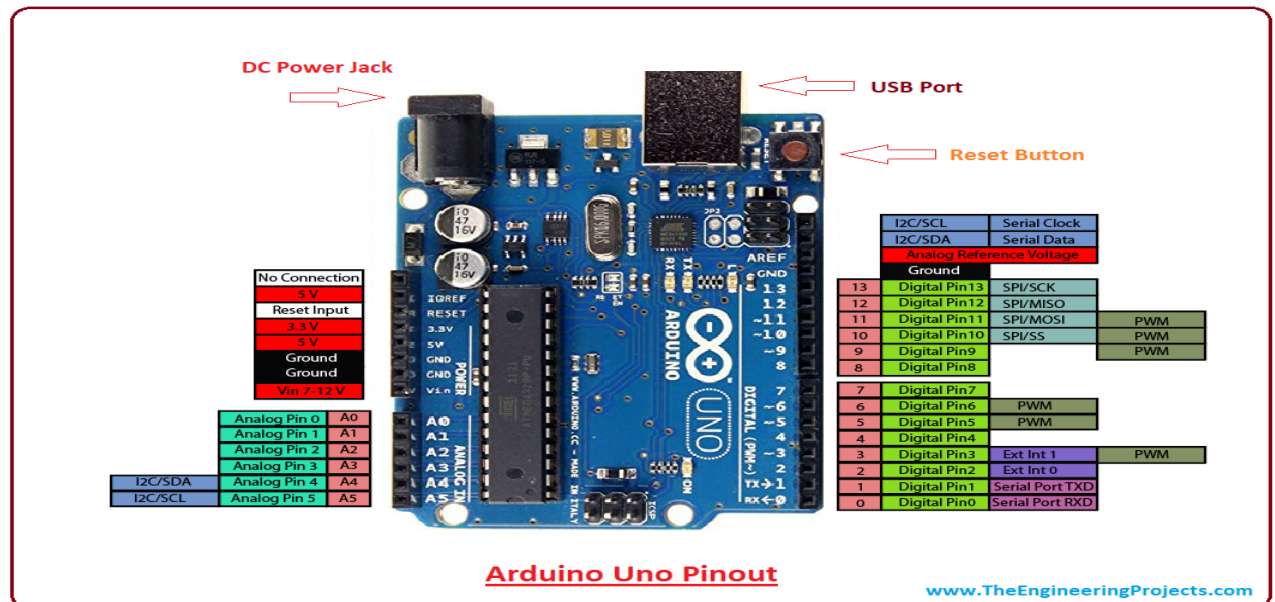


Figure 7.1: Arduino UNO pin-out

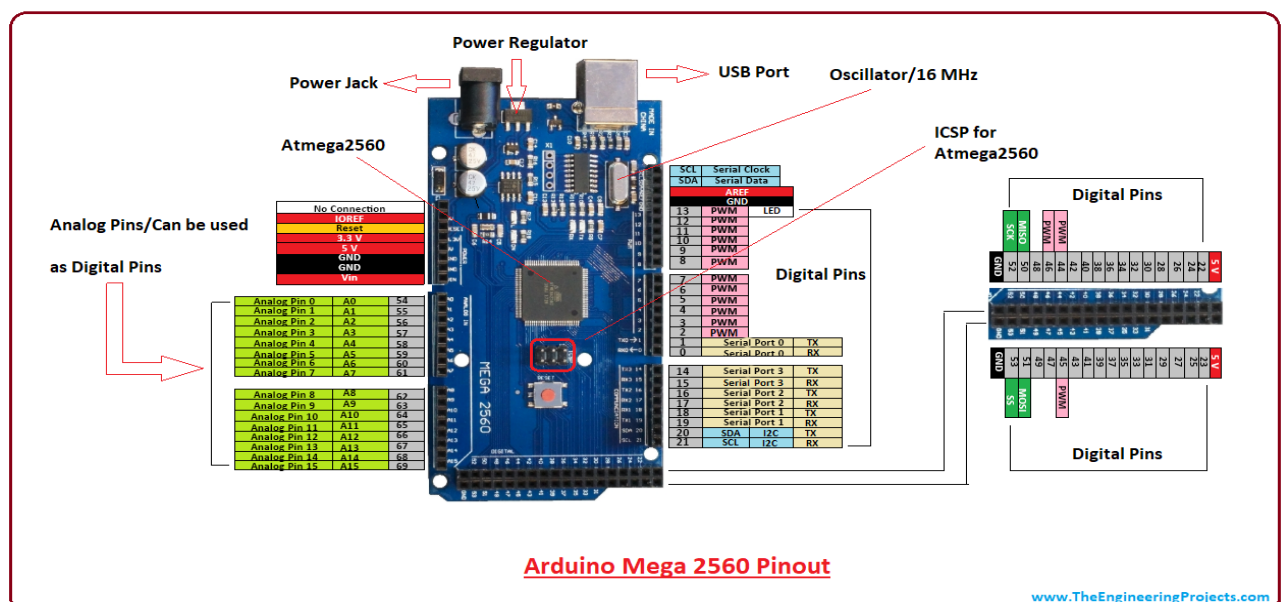


Figure 7.2: Arduino Mega pin-out

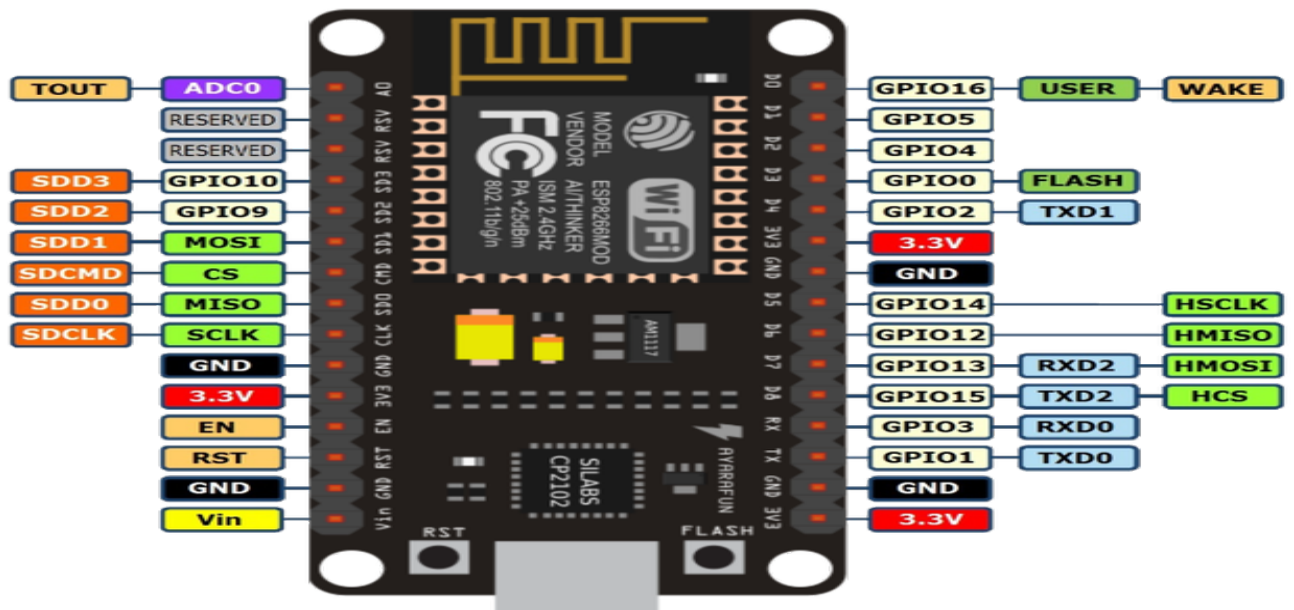


Figure 7.3: NodeMCU pin-out



Figure 7.4: GPS SIM28 pin-out

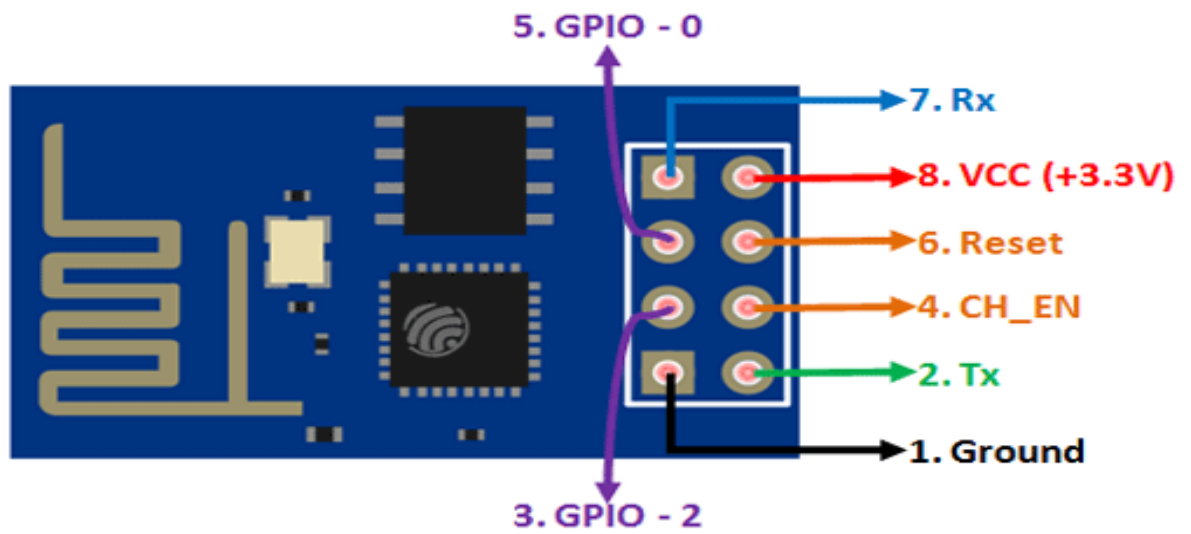


Figure 7.5: ESP8266-01 pin-out

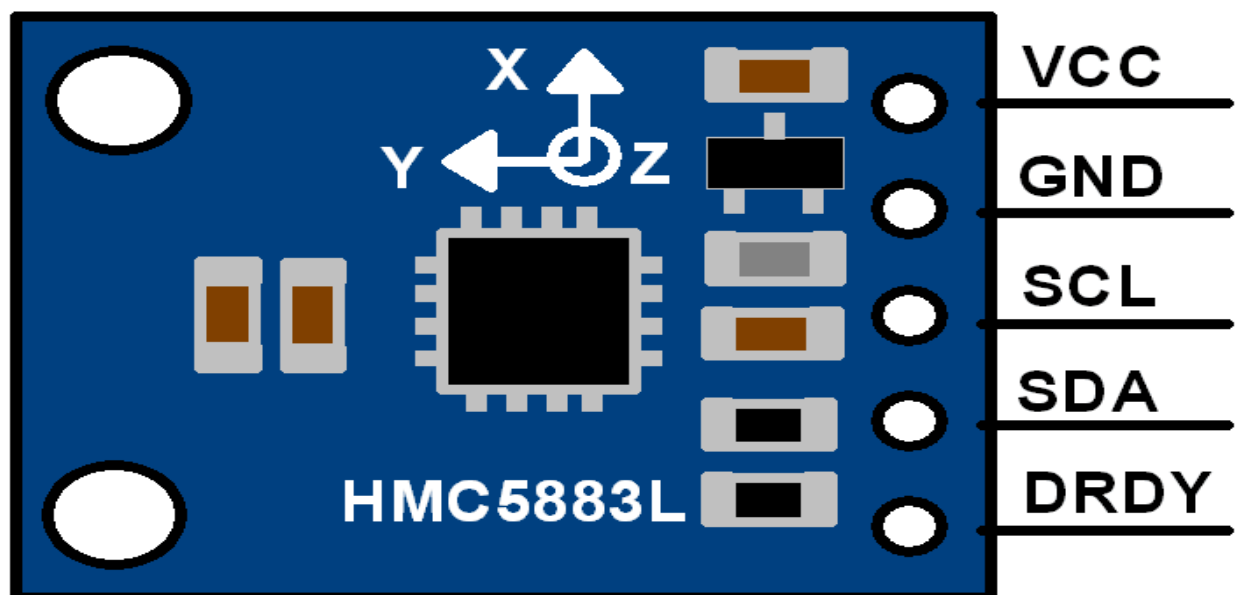


Figure 7.6: Magnetometer (HMC5883L) pin-out

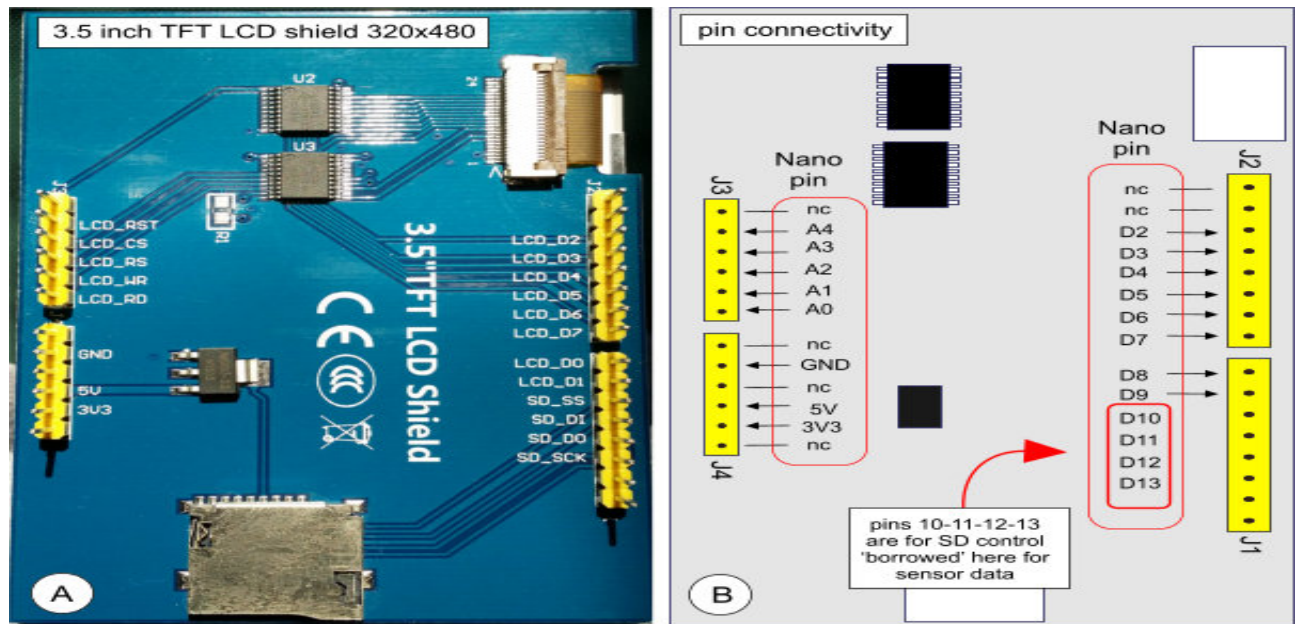


Figure 7.7: Screen pin-out

Appendix B

TinyGPS++ TinyGPS++ is a new Arduino library for parsing NMEA data streams provided by GPS modules. Like its predecessor, TinyGPS, this library provides compact and easy-to-use methods for extracting position, date, time, altitude, speed, and course from consumer GPS devices. However, TinyGPS++'s programmer interface is considerably simpler to use than TinyGPS, and the new library can extract arbitrary data from any of the myriad NMEA sentences out there, even proprietary ones.

SoftwareSerial The SoftwareSerial library has been developed to allow serial communication on other digital pins of the Arduino, using software to replicate the functionality (hence the name "SoftwareSerial"). It is possible to have multiple software serial ports with speeds up to 115200 bps.