

# Positive Train Control, Command and Communication

A Thesis submitted to the Department of Electrical and Electronic Engineering of

**BRAC University**

By

Masharul Bin Mahfuz – 11221015

MD. Shakhawat Hossain - 11321052

Supervised by

**Avijit Das**

Lecturer

Department of Electrical and Electronic Engineering

BRAC University, Dhaka

in partial fulfilment of the requirements for the Bachelor of Science degree in

Electrical and Electronics Engineering



Spring 2017

BRAC University, Dhaka

# DECLARATION

We hereby declare that the thesis title “Positive Train Control, Communication & Command” submitted to the Department of Electrical and Electronics Engineering is our own work and has not been presented elsewhere for assessment. Where material has been used from other sources it has been properly acknowledged/referred.

Signature of  
Supervisor

---

Avijit Das

Signature of  
Authors

---

Masharul Bin Mahfuz

---

Md. Shakhawat Hossain

# ACKNOWLEDGMENT

We would like express our heartiest gratitude towards our supervisor Avijit Das, Lecturer of the department of Electrical and Electronic Engineering of BRAC University, for laying the foundation of our thesis project and providing valuable insight and guidance at each and every step of the development process. We are extremely fortunate and grateful to be able to work under his direct supervision. We want to thank him for continuous belief on us and frequent motivation to complete our project in time.

This is the project of Masharul Bin Mahfuz and Md. Shakhawat Hossain, students from the Electrical and Electronic Engineering department of BRAC University. We would like to express our gratitude to the Almighty who gave us strength, determination, intelligence, help and opportunity to complete this project. Furthermore, we would like to thank Zohair Mehtab Ali for his assistance in coding of the website for the Central Control System. We would also like to thank our family and friends for their continuous support and encouragement throughout the whole process of working for so long. Their cooperation made us becoming determined to achieve our goals. Last of all, our gratitude goes towards the faculty members of Department of Electrical and Electronic Engineering, BRAC University who provided us all of necessary and essential knowledge, concepts, ideas, support and tools that aided us to successfully complete our thesis project.

# ABSTRACT

Proposal of Positive Train Control, Command and Communication for Bangladesh Railway is envisioned to prevent railway hazards with the usage of Obstacle detection at level crossings, monitor locations and activities of trains by sending responsive commands back and forth from a central control unit using a private network. This system is solely expected to provide a comprehensive level of safety mechanism to eradicate human error, hence predicting and attempt to prevent any sort of major or minor accidents, loss of lives and properties.

**Keywords:** Railway, Automation, GPS, Communication, Safety

# CONTENTS

DECLARATION .....	i
ACKNOWLEDGMENT.....	ii
ABSTRACT .....	iii
CONTENTS .....	iv
LIST OF FIGURES .....	xi
LIST OF TABLES .....	xiv
ABBREVIATIONS .....	xv
CHAPTER 1: INTRODUCTION .....	1
1.1 – Railway System of Bangladesh.....	1
1.2 – Problems in the Existing System .....	2
1.3 – History of Railway accidents in Bangladesh .....	3
1.4 – Introduction to Positive Train Control .....	6
1.5 – Objective .....	7
1.6 – Motivation.....	8
1.7 – Project Overview.....	8
CHAPTER 2 – LITERATURE REVIEW .....	10
2.1 – Development of PTC by Union Pacific .....	10
2.2 – PTC works in Europe.....	10
2.3 – Planning of PTC in the United States .....	10
CHAPTER 3 – SYSTEM DESIGN .....	12
3.1 – Reformation of an Existing Design.....	12
3.2 – Components Used .....	13
3.2.1 – Microcontroller .....	13

3.2.2 – SIMCOM SIM808 Quad-Band GSM+GPRS+GPS Module .....	15
3.2.3 – LM2596 DC-DC Step-Down Buck Converter.....	17
3.2.4 – NE555 IC .....	18
3.2.5 – BD140 IC .....	20
3.2.6 – BC557 IC .....	20
3.2.7 – Resistors.....	21
3.2.8 – Capacitors.....	22
3.2.9 – 20X4 LCD.....	22
3.2.10 – 4X4 Matrix Keypad .....	23
3.2.11 – Potentiometer (10k ohm) .....	24
3.2.12 – SG90 Servo .....	25
3.2.13 – LED (5mm) .....	25
3.2.14 – Buzzer .....	26
3.2.15 – TSOP1738.....	27
3.2.16 – IR Transmitters .....	28
3.2.17 – Others .....	28
3.3 – Communication Flow.....	30
3.4 – Message Codes.....	31
CHAPTER 4 – HARDWARE IMPLEMENTATIONS .....	35
4.1 – Arduino Mega 2560 .....	35
4.2 – SIMCOM SIM808 GSM+GPS+GPRS Module .....	35
4.3 – LM2596 DC-DC Step-Down Buck Converter.....	35
4.4 – Peripheral Devices.....	36
4.4.1 – Peripheral Devices in the Train System.....	36
4.4.2 – Peripheral Devices in the Level Crossing System .....	37

CHAPTER 5 – SOFTWARE IMPLEMENTATIONS .....	38
5.1 – Arduino IDE.....	38
5.1.1. Software Libraries.....	39
5.1.2. Functions & Syntax.....	40
5.2 – Fritzing.....	41
5.3 – Proteus Professional .....	42
5.4 – AT Commands .....	44
5.4.1 – AT Command syntax .....	45
5.4.2 – Set of AT Commands used to program this system.....	48
5.5 – AT Command Tester Tool .....	52
5.5.1 – Port Configuration.....	53
5.5.2 – Command mode .....	53
5.5.3 – Script Mode.....	54
5.5.4 – Diagnostics.....	55
5.5.5 – Data Call .....	55
5.6 – HyperTerminal 7.....	56
5.7 – Adobe Dreamweaver.....	57
5.8 – PHPStorm .....	58
5.8.1 – Key features .....	59
5.9 – Apache Server .....	63
5.9.1 – HTTP server and proxy features .....	65
5.10 – Adobe Photoshop CS6 .....	66
CHAPTER 6 – THE TRAIN SYSTEM.....	67
6.1 – Hardware Configuration .....	67
6.1.1 – Dual Core Technology .....	67

6.1.2 – Master Core.....	68
6.1.3 – Slave Core.....	69
6.1.4 – Power Supply .....	70
6.2 – Code Algorithms .....	71
6.2.1 – Master Core.....	71
6.2.2 – Slave Core.....	82
6.2.3 – Master-Slave Communication.....	89
6.2.4 – Calculations.....	89
6.3 – PCB Design.....	90
6.4 – Working Principle .....	90
6.4.1 – Obtaining GPS Data .....	90
6.4.2 – Uploading data to Web Server .....	90
6.4.3 – Obtaining Data from Web Server .....	91
6.4.4 – Check for new SMS .....	91
6.4.5 – Functions of the ‘Start/Stop Journey’ Button .....	91
6.4.6 – Functions of the ‘Menu’ Button.....	92
6.4.7 – The User Interface.....	92
6.4.8 – Distance Calculation.....	95
6.4.9 – Search for Nearest Level Crossing.....	95
6.4.10 – Obtain Nearest Level Crossing Data.....	95
6.4.11 – Communicating with the nearest Level Crossing .....	95
6.4.12 – Responses to Obstacle Detection .....	95
CHAPTER 7 –THE LEVEL CROSSING SYSTEM .....	98
7.1 – Hardware Configuration .....	98
7.1.1 – Single Core Technology.....	98



7.1.2 – IR Transmitter Circuit.....	99
7.1.3 – IR Receiver Circuit .....	100
7.1.4 – Peripheral Devices .....	101
7.1.5 – Power Supply .....	102
7.2 – Code Algorithms .....	103
7.2.1 – Train Detection Algorithm.....	103
7.2.2 – Obstacle Detection Algorithm .....	106
7.2.3 – Road Controlling Algorithm .....	109
7.3 – PCB Design.....	113
7.3.1 – IR Transmitter Array.....	113
7.3.2 – IR Receiver Array .....	113
7.3.3 – Level Crossing Mainboard.....	114
7.4 – Working Principle .....	115
7.4.1 – The Train Detection Mechanism.....	115
7.4.2 – Obtaining and Storing Train Information .....	115
7.4.3 – Sensor Positions for Obstacle Detection.....	115
7.4.4 – Obstacle Detection Mechanism .....	117
7.4.5 – Closing and Opening Gates.....	120
7.4.6 – Control of Road Signals and Alarm.....	120
7.4.7 – Control of Train Track Signals .....	120
7.4.8 – The User Interface.....	120
7.4.9 – Functions of the ‘Menu’ Button.....	122
7.4.10 - Flow Chart of the Level Crossing System.....	123
7.5 – The Level Crossing Model.....	124
7.5.1 – Materials Used .....	124

7.5.2 – Pictures of the Model .....	124
CHAPTER 8 –THE CENTRAL CONTROL SYSTEM.....	126
8.1 – Introduction.....	126
8.2 – Specifications.....	129
8.3 – Database Structure .....	129
8.4 – Website Design .....	131
8.4.1 – User Navigation (Web Application) .....	131
8.4.2 – Location Upload (Web Application).....	132
8.4.3 – Location View (Web Application).....	133
8.4.4 – View Train Information .....	134
8.4.5 – Add Message.....	135
8.4.6 – Website Functions.....	136
8.4.7 – Site Map .....	137
8.5 – Software Used.....	138
8.5.1 – PHPStorm .....	138
8.5.2 – Adobe Dreamweaver.....	138
8.5.3 – Apache Server.....	138
CHAPTER 9 – TESTS & RESULTS .....	139
9.1 – SMS Messaging Test .....	139
9.1.1 – Test Type 1.....	139
9.1.2 – Test Type 2.....	140
9.2 – GPS Test .....	141
9.3 – Obstacle Detection Test .....	145
9.4 – Website Test.....	150
9.4.1 - Location Upload Test .....	150

9.4.2 – Test of sending SMS Command to trains .....	150
9.4.3 – Server Test .....	150
9.4.4 – Testing on Local Host .....	150
CHAPTER 10 – COSTS .....	151
10.1 – Train System .....	151
10.2 – Level Crossing System.....	152
10.3 – Central Control System.....	153
10.4 – Full Package .....	153
CHAPTER 11 – CONCLUSION.....	155
11.1 – Limitations .....	155
11.2 – Challenges Faced .....	155
11.3 – Future Works & Improvements .....	155
11.4 – Summary .....	156
REFERENCES .....	157
APPENDIX.....	161

# LIST OF FIGURES

Figure 1.1 Rail Accidents in Bangladesh 2008-2015 .....	6
Figure 1.2 Positive Train Control System Web Diagram .....	7
Figure 3.1 Arduino Mega 2560 .....	14
Figure 3.2 SIMCOM SIM808 Quad-Band GSM+GPRS+GPS Module .....	15
Figure 3.3 LM2596 DC-DC Step-Down Buck Converter .....	17
Figure 3.4 NE555IC .....	19
Figure 3.5 BD140 IC.....	20
Figure 3.6 BC557 IC.....	21
Figure 3.7 20X4 LCD .....	22
Figure 3.8 4X4 Matrix Keypad .....	23
Figure 3.9 100K Potentiometer .....	24
Figure 3.10 SG90 Servo.....	25
Figure 3.11 LED Lights .....	26
Figure 3.12 Buzzer .....	27
Figure 3.13 TSOP1738 IR Receiver .....	28
Figure 3.14 Communication Flow Diagram .....	30
Figure 5.1 Arduino IDE Screenshot.....	38
Figure 5.2 the Fritzing Desktop App .....	42
Figure 5.3 Proteus Screenshot.....	44
Figure 5.4 AT Command Tester User Interface .....	52
Figure 5.5 AT Command Tester Command Mode .....	53
Figure 5.6 AT Command Tester Script Mode .....	54
Figure 5.7 AT Command Tester Diagnostics Tab .....	55
Figure 5.8 Adobe Dreamweaver Screenshot.....	57
Figure 5.9 Zero-configuration web application debugging with Xdebug in PhpStorm. Darcula color scheme .....	59
Figure 5.10 Version Control Systems Integration .....	59
Figure 5.11 Viewing Data Source Structure and other SQL and Databases related features in PhpStorm.....	60
Figure 5.12 Apache Logo.....	63

Figure 5.13 Apache GUI.....	64
Figure 5.14 Adobe Photoshop Screenshot .....	66
Figure 6.1 Dual-Core System Architecture.....	67
Figure 6.2 Block Diagram of the Master Core.....	68
Figure 6.3 Block Diagram of the Slave Core.....	69
Figure 6.4a Flow chart of Power Distribution Diagram in the Train System Mainboard .....	70
Figure 6.4b Power Distribution on Train System Mainboard.....	71
Figure 6.5 Screenshot of GPS reply from the SIM808 Module.....	73
Figure 6.6 Flow Chart for Uploading data to the Web Server .....	77
Figure 6.7 Flow chart of the Master Core .....	79
Figure 6.8 Screenshot of a Received SMS Raw Output .....	83
Figure 6.9 Flow Chart of the Slave Core .....	86
Figure 6.10 PCB Design of Train System Mainboard .....	90
Figure 6.11 Menu Map of the Train System.....	92
Figure 6.12 Train System Mainboard .....	92
Figure 6.13 the Control Panel of Train System .....	93
Figure 6.14 Potentiometers to Control LCD Brightness.....	94
Figure 6.15 Emergency Alarm Button and LEDs .....	94
Figure 7.1 Block Diagram of the Level Crossing System .....	98
Figure 7.2 Circuit Schematic of Long Range IR Transmitter.....	99
Figure 7.3 IR Receiver Circuit Schematic .....	100
Figure 7.4 Peripheral Devices ports on Level Crossing Mainboard .....	101
Figure 7.5 Menu button and Gate LEDs on the Level Crossing Mainboard .....	102
Figure 7.6 Power Distribution diagram in the Level Crossing Mainboard.....	102
Figure 7.7 Screenshot of a Received SMS Raw Output .....	104
Figure 7.8 PCB Design of IR Transmitter Array .....	113
Figure 7.9 PCB Design of IR Receiver Array .....	113
Figure 7.10 PCB Design of Level Crossing Mainboard .....	114
Figure 7.11 Sensor Positions for Obstacle Detection at a Level Crossing .....	116
Figure 7.12 Flow Chart for Obstacle Detection at Level Crossings .....	119
Figure 7.13 Level Crossing Mainboard .....	121

Figure 7.14 Power Distribution on the Level Crossing Mainboard .....	121
Figure 7.15 Menu Map of the Level Crossing System .....	122
Figure 7.16 Flow Chart of the Level Crossing System.....	123
Figure 7.17 Level Crossing Model Side View.....	124
Figure 7.18 Level Crossing Model Front View .....	124
Figure 7.19 IR Transmitter and receiver’s position at the Level Crossing Model.....	125
Figure 7.20 IR Transmitters’ position in the Level Crossing Model .....	125
Figure 8.1 Login Page.....	127
Figure 8.2 Home page.....	127
Figure 8.3 Train Details and options to send message.....	128
Figure 8.4 Maintenance Team List .....	128
Figure 8.5 Database Schema: Entity Relationship Diagram.....	130
Figure 8.6 Flow Chart of User Navigation .....	131
Figure 8.7 Flow chart for uploading location in the web site .....	132
Figure 8.8 Flow Chart of viewing train location in the Web Site.....	133
Figure 8.9 Flow chart of viewing train information in the web site .....	134
Figure 8.10 Flow chart for adding messages to the database.....	135
Figure 8.11 Flow Chart of Website Functions .....	136
Figure 9.1 GPS Test, Set 1, Point A graph .....	141
Figure 9.2 GPS Test, Set 2, Point A graph .....	142
Figure 9.3 GPS Test, Set 1, Point B Graph.....	143
Figure 9.4 GPS test, Set 2, Point B.....	144
Figure 9.5 Obstacle Detection Test Setup Diagram.....	145
Figure 9.6 Obstacle Detection Test 1 Graph.....	146
Figure 9.7 Obstacle Detection Test 2 Graph.....	147
Figure 9.8 Obstacle Detection Test 3 Graph.....	148
Figure 9.9 Obstacle Detection Test 4 Graph.....	149
Figure 9.9 Obstacle Detection Combined Graph.....	149

# LIST OF TABLES

Table 1.1 Accidents by Category .....	3
Table 3.1 Message Codes and their corresponding actions .....	34
Table 5.1 Devices/ Modules and their corresponding libraries .....	40
Table 5.2 Functions used in programming and their corresponding tasks .....	41
Table 5.3. Types of AT commands and responses .....	46
Table 5.4. Set of AT Commands used in the programming of this system and their functions. ..	51
Table 6.1 Important variables used and their functions: .....	72
Table 6.2 AT+CGNSINF return Parameters .....	73
Table 6.3 AT Commands used to upload data to the Web Server .....	76
Table 6.4 AT Commands used to send or Receive SMS Messages .....	82
Table 7.1 AT Commands used to send or Receive SMS Messages .....	103
Table 7.2 SMS Codes sent to Train for different obstacle positions and its corresponding level crossing activity .....	118
Table 9.1 SMS Messaging Test Type-1 .....	139
Table 9.2 SMS Messaging Test Type-2.....	140
Table 9.3 GPS test, Set 1. Point A .....	141
Table 9.4 GPS test, Set 2. Point A .....	142
Table 9.5 GPS Test, Set 1, Point B .....	143
Table 9.6 GPS test, Set 2, Point B .....	144
Table 9.7 Obstacle Detection Test 1 .....	145
Table 9.8 Obstacle Detection Test 2 .....	146
Table 9.9 Obstacle Detection Test 3 .....	147
Table 9.10 Obstacle Detection Test 4 .....	148
Table 10.1 Table of Costs of Train System .....	151
Table 10.2 Table of Costs of Level Crossing System.....	153
Table 10.3 Table of Costs for Central Control System.....	153
Table 10.4 One Time Costs of entire System .....	153
Table 10.5 Monthly Cost of entire system.....	154

# **ABBREVIATIONS**

PTC – Positive Train Control

CCS – Central Control System

IR – Infra Red

GSM – Global System for Mobile Communications

GPS – Global Positioning System

GNSS – Global Navigation Satellite System

PCB – Printed Circuit Board

BR – Bangladesh Railway

API – Application Programming Interface

IC – Integrated Circuit

PNP – Hole-Electron-Hole or P-type-N-type-P-type

HTTP – Hyper-Text Transfer Protocol



# CHAPTER 1: INTRODUCTION

## 1.1– Railway System of Bangladesh

Although trains are considered as a very ‘convenient’ transport in Bangladesh, lack of management in this sector and also defected rail lines make huge loss in recent years not only to the asset but also to the human being. It is also mentioned that to resolve this problem ‘stand-alone sub system’ is introduced which is able to work without the help of human power and gives the assurance of safe crossing even in the crowded city like Dhaka. Railway communication is always considered as one of the convenient ways of communication especially on the overpopulated country to reduce the pressure of transport section. The country like Bangladesh where the number of population is very high, railway is getting improved by using the latest connection to make it ‘environment-friendly’, reasonable and ‘comfortable’, an effective plan is designed to reach the goal of making railway one of the public friendly transportation within next 25 years as there are lots of lacking in the current railway system. To make this system error free/perfect and more significant, there is no alternative to use the scientific digitalized technology rather than depending on the system where the station master takes the responsibility to send the message of train till it starts its journey then it reaches at the last station. Telecommunication system like landline is basically used in this message delivery process which is not trustworthy and danger free. At the same time, ‘signaling semaphore system’ and ‘color light signals’ are used to pass the signal which are not reliable at the present time(ibid). To move the ‘blocking signals’ and to ensure the position of a train, the combination of control central, signals and lines are inevitable, but still now, Bangladesh depends to control the rail networks by the help of signaller or station master. Bangladesh railway has 2,855 km route and 34,168 employees are involved in this sector. Even having this manpower, accidents in level crossing become a common incidents in this country which is definitely a matter of worry in the railway system. Dependency on the man power and uncontrolled level crossing are considered as the main reason of these accidents and also absence of ‘warning signal’ and ‘road signal’ making it too difficult to control the level crossing and among the 1,403 level crossing, only 250 level crossing gates are controlled by the gateman . As trains are operated by the train tracks, the low quality tracks can produce the heat below the running train and it can make major destruction/accidents. Only thinking over the issues are not enough rather it is important to take the initiative to solve this ‘faulty’ rail system.

After the emergence of Bangladesh as an independent country, the name of the railway was changed to Bangladesh Railway. It inherited 2858.73 km of tracks and 466 stations. The East Pakistan government had conducted feasibility studies for expansion of railways in the 1960s and accordingly, construction of a double line and various branch lines were initiated. The construction of double line track between Dhaka-Tangi and Chittagong-Mirsharai started in 1960s. These lines were opened in phases during the 1960s and 1970s. Under a realignment scheme, work on shifting the Dhaka railway station from the congested Phulbaria area to Kamalapur was started in late 1950s. The new station was opened to traffic on 1 May 1968. The station building is a marvel of architecture having canopies of hyperbolic-paraboloid shell structures. The station started providing container services with the establishment of the first inland container depot (ICD) of the country on 11 April 1987. [4]

## **1.2 – Problems in the Existing System**

Track maintenance at adequate standard is a fundamental requirement of permanent way (P-way) to avoid derailments and provide acceptable riding quality for traffic being carried. In general, derailment occurs because of twist faults, cyclic top faults, buckles and of course broken rail, along with switch and crossing layouts. However, priority must be given to elimination of defects on the open line where train speed is more. Therefore, Bangladesh Railway needs to make adequate arrangement for mechanical maintenance to ensure safety and adequate riding quality. [5]

The basic purpose of railway signals is to enhance safety and to give train drivers enough scope to slow down, stops or accelerate. The big advantage of rail over other forms of surface transport is the low friction of steel wheel on steel rail which contributes to large scale fuel efficiency. But this in turn makes it hard for the train to stop; a modern express train may require more than 2km to stop on the level and 3km to stop on a downhill slope. Although, railway signalling originated from the basic needs of safety, but its further development has permitted the exploitation of wider facilities, in the form of considerable economies and increased efficiency, coupled with the attainment of higher speeds and improved control. Colour-light signals were introduced in 1928 but were slow to take off. In recent years many older semaphore signals have been replaced by colour-light signals. However, both the Semaphore signals and Colour-light signals exist in the Bangladesh Railway system, but the share of Colour-light signals has been increasing. [5]

To move the ‘blocking signals ‘and to ensure the position of a train, the combination of control central, signals and lines are inevitable, but still now, Bangladesh depends to control the rail networks by the help of signalmen or station master. Bangladesh railway has 2,855 km route and 34,168 employees are involved in this sector. Even having this manpower, accidents in level crossing become a common incidents in this country which is definitely a matter of worry in the railway system. Dependency on the man power and uncontrolled level crossing are considered as the main reason of these accidents and also absence of ‘warning signal’ and ‘road signal’ making it too difficult to control the level crossing and among the 1,403 level crossing, only 250 level crossing gates are controlled by the gateman . As trains are operated by the train tracks, the low quality tracks can produce the heat below the running train and it can make major destruction/accidents. Only thinking over the issues are not enough rather it is important to take the initiative to solve this ‘faulty’ rail system.

### 1.3 – History of Railway accidents in Bangladesh

Rail is safer mode in comparison to road in Bangladesh. Presently, derailments of train are increasing but still the fatal accidents are less in number compared to other modes of transport. Railway is undoubtedly the most reliable traffic mode. The data on train accidents are available in the Information Books of Bangladesh Railway. The categories of train accidents are divided into collision, derailment, fire in the

Year	Collisions	Derailments	Fire in trains	Train running into obstructions	Total
1997-98	8	233	0	16	257
1998-99	5	304	0	49	358
1999-00	6	405	1	44	456
2000-01	5	510	0	37	552
2001-02	14	624	3	67	708
2002-03	13	482	2	27	524
2003-04	8	723	0	23	754
2004-05	7	592	30	78	707
2005-06	3	790	0	37	853

Table 1.1 *Error! No text of specified style in document.*

Unguarded level crossing, frequent mechanical and human failures, dilapidated rail tracks and outdated signalling system were the main reasons behind 590 rail accidents the year 2010

According to the Bangladesh Railway (BR) statistics, some 51 persons died while another 149 were injured in the accidents including the latest head-on collision between two trains in Narsingdi on Wednesday.

The recent train accident is considered the biggest head-on collision in ten years, BR officials said. They said this year the number of derailments is significantly high, but the majority of the deaths occurred at unguarded and unauthorized level crossings.

As many as 53 accidents occurred this year at level crossings around the country, killing about 22 persons. Mostly, collision between road traffic and train caused the accidents. In a few incidents, pedestrians were run over by trains while they were crossing the rail track at level crossings.

The rail officials also suspect that many train accidents at level crossings especially the minor ones went unreported.

"Accident at level crossing is our major concern since we don't have any control over it," said Mohammad Shahjahan, additional director general (operations) of BR.

The ADG said, "Level crossings sprouted up without any authorisation from the railway and it is very difficult to guard them. We protested this practice many times but no one listened."

At many level crossings, the railway has not been able to provide approach warning signals and road signals, he added. At present, the country's 2855 km rail network has some 1,403 level crossings whereas only 250 level crossing gates are operated by gatemen round the clock.

In Dhaka, negligent drivers and pedestrians are making the situation even worse.

He mentioned that the railway considers activities within 50 yards of rail tracks illegal but no one listens to this warning.

Moreover, the poor condition of tracks often causes derailments.

Yesterday in Kishoreganj, a Chittagong-bound mail train derailed while coming from Bahadurabad of Mymensingh.

Railway and local sources said four compartments of the train came off the track around 3:30pm, leaving 20 people injured.

They said faults in the rail track inadequate stone support and weak sleeper-- caused the accident.

The derailments most of the time happened on branch lines because the condition of those is worse.

There were 489 derailments this year.

Though the derailments did not cause any deaths, properties of BR were damaged in the accidents, Shahjahan noted.

"Restoration of tracks is going on at many points currently," the ADG said.

About human failure, he said shortage of manpower increased workload on technical hands like locomotive masters and stationmasters and this is the reason behind human failure.

BR record shows there were eight incidents of disregarding signal this year.

In 2009, 60 people died in train accidents. Of them 51 died in collisions between train and road traffic at the level crossings. The death toll was 53 in 2008 in 893 train accidents.

On April 16, 2008, 17 people were killed and 25 others injured in a fatal accident when a Dhaka-bound intercity train rammed a passenger bus at a level crossing in Kalihati upazila of Tangail.

According to the Bangladesh Railway, around 5050 train accidents occurred in the country between 2000 and 2009. [3]

11 July 2006 – A train collided with a crowded bus at an unmanned railroad crossing at Akkelpur Upazila, Jaipurhat District, killing at least 33 people, leaving another 30 injured.

13 October 2007 – 4 people died and over 50 were injured when the rear carriages of the Probhati Express derails near Dhaka.

16 April 2008 – According to ATN Bangla television report, a Dinajpur–Dhaka Ekota Express train collided with a local bus on a level crossing on the outskirts of Kalihati, Tangail District killing 18 and injuring 30.

14 May 2008 – According to ATN Bangla television report, an Upaban Express train rams into the rear of Noakhali Express train at Ashuganj Upazila station, Brahmanbaria District killing 8 and injuring 100.

8 December 2010 – A collision between passenger trains killed at least 10 people. [2]

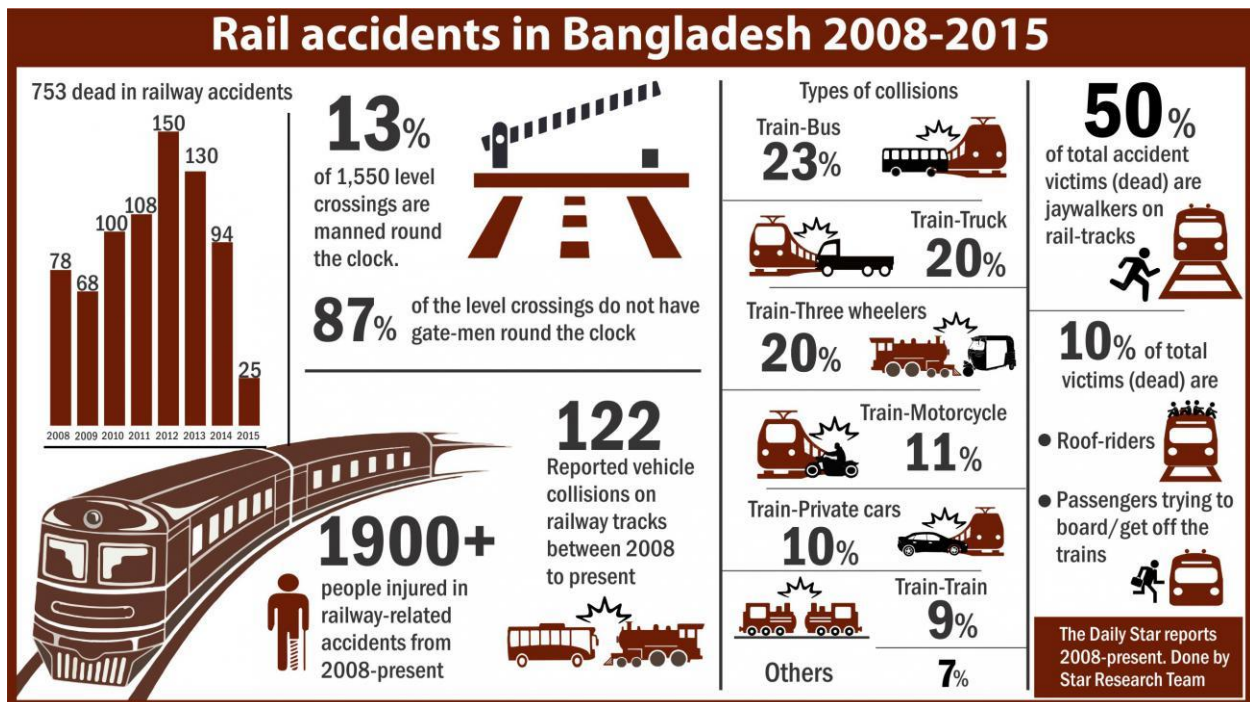


Figure 1.1 Error! No text of specified style in document.

## 1.4 – Introduction to Positive Train Control

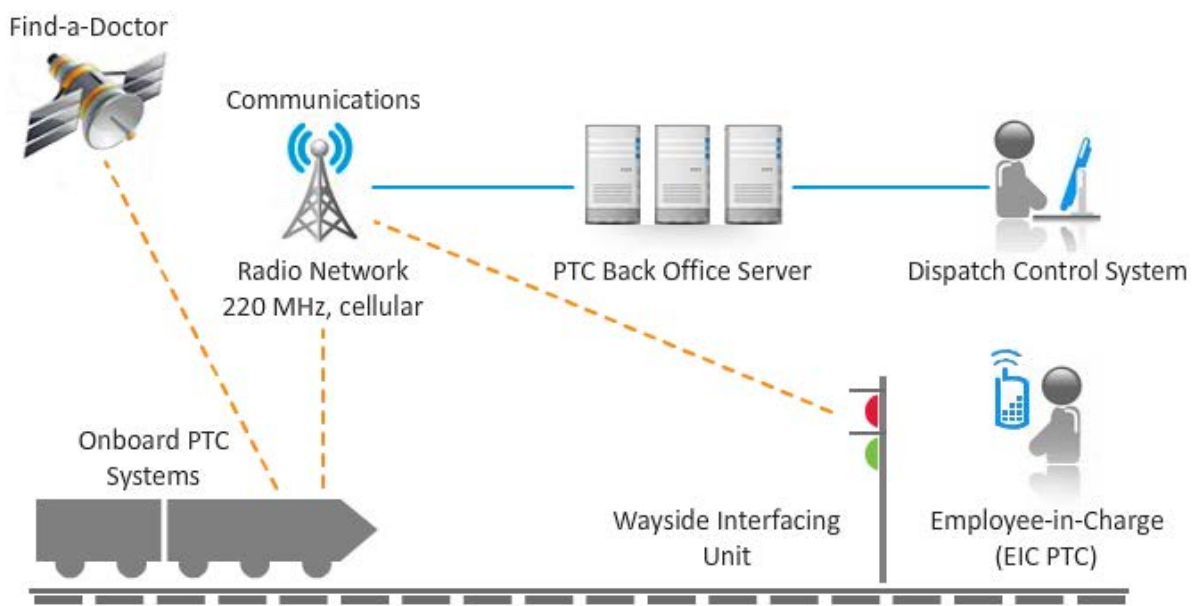
Positive train control (PTC) is a system of functional requirements for monitoring and controlling train movements and is a type of train protection systems. The term stems from Control Engineering. The train is only allowed to move in case of positive movement allowance. It generally improves the safety of railway traffic.

Positive Train Control (PTC) is a complex, nationwide system of newly developed technologies that continuously relays critical information such as speed limits, train movement authorization, switch positions, work zone locations and other operational data. It must factor in locomotive and rail car mix; train length, weight and speed; terrain and signal aspects to determine safe stopping distances. This conservatism in the "braking curve" slows the rail network's velocity and, thus, reduces capacity and ability to handle more freight. Additionally, any PTC hardware or software component failure also defaults to stopping the train, thus reducing rail network capacity. [1]

Implementing PTC properly requires integrating thousands of components across the telecommunications spectrum, such as GPS, Wi-Fi, radios, cellular technology, antennae, base stations and first-of-its-kind software that decides when to slow or stop a train. [1]

PTC is an advanced system of technologies designed to automatically stop a train before certain accidents occur. In particular, PTC is designed to prevent:

- Train-to-train collisions
- Derailments caused by excessive train speed
- Train movements through misaligned track switches
- Unauthorized train entry into work zones. [1]



*Figure 1.2 Positive Train Control System Web Diagram*

## 1.5 – Objective

The objective of our system is to provide a comprehensive level of safety to the railway system of a country specifically Bangladesh where railway safety has been a major issue over the past years as we have seen on the previous section. Keeping safety as the core objective of our system, we have decided to achieve it by preventing railway hazards, detect obstacles at level crossings and monitor trains live for 24/7.

## **1.6 – Motivation**

The motivation towards working on this research and development program was originated from the view that a massive modernization in this sector will not only improve the standard of the railway system in the country, but also reduce a significant number of accidents by predicting them prior to collisions hence saving a huge number of lives and properties.

Trains are thought to be uncontrollable once on the run. However, our system proves the fact otherwise. The concept of ability to control, command and communicate with the train through a private network.

## **1.7 – Project Overview**

The key concept of this project is to have a Central Control Unit which would be able to command, control and communicate with the trains that are on route. The locations of those trains can also be seen live in a Google map API and notify the operator of the Central Control Unit with an alert if any trains are speeding too much or closing up and becoming prone to collisions. Also without the permission of the Central Control System the trains would not be able to start its journey. The Central Control System ensures faster and a comprehensive level of safety throughout the journey of a train.

Gates at the level crossings are opened and closed automatically whenever a train is approaching a particular level crossing. The level crossing quickly checks if there are any obstacles present under the gates or along the course of the train and sends a quick response to the train to inform about any obstacle. The train reacts differently for each positions of the obstacle.

Level crossings use IR sensors which acts as ‘obstacle detector’ sends a signal to the microcontroller which signals the GSM module to send message to the approaching train. Information from the GSM module which is fitted on the train will constantly send the information about the distance of train and the level crossing and if any obstacle is sensed within this time, train can stop in a ‘safe distance’ and if everything goes well, the alarm to close the gate can be sent. Through GPS, it is possible to know the current position of the running train. To control the railway gate automatically, the use of ‘pressure sensor’ based project is designed to sense whether the train is coming or leaving the crossing automatically. whenever train’s arrival is sensed by using pressure sensory, message would be sent to the ‘microcontroller’ to let it check whether there



is any 'vehicle' within the gate or not. Although the use of pressure plate is more effective than IR sensors, it is difficult to implement pressure plates as it costs very high. That is why, GPS is more preferable as the way to send the necessary message to detect the position of a train.

# **CHAPTER 2 – LITERATURE REVIEW**

## **2.1 – Development of PTC by Union Pacific**

Union Pacific is committed to implementing Positive Train Control (PTC) carefully and thoroughly to enhance safety for employees and communities. Through Dec. 31, 2016, they have invested \$2.3 billion in PTC. Their current estimate for PTC's total cost is about \$2.9 billion.

## **2.2 – PTC works in Europe**

In Europe, the European Commission, under research program Horizon 2020, is sponsoring a project known as RHINOS, or “Railway High Integrity Navigation Overlay System,” as a starting point for integrating GNSS into the existing European and worldwide railway infrastructure to provide improved autonomy to existing mainline services and new capability in more austere regions.

Stanford has been performing research in this area to support both PTC and RHINOS by modifying techniques it helped develop for SBAS and GBAS certification for airborne users to fit the railway environment. This research includes comparing existing aviation and railway safety requirements, developing GNSS anomaly threat models that are specific to the railway environment, and the use of simulations to implement these threat models under railway conditions in order to determine the safety mitigations needed to meet railway performance and safety requirements. These research efforts are currently self-funded and supplemented through the RHINOS project. [6]

## **2.3 – Planning of PTC in the United States**

In 2008, Congress required Class I railroad main lines handling poisonous-inhalation-hazard materials and any railroad main lines with regularly scheduled intercity and commuter rail passenger service to fully implement Positive Train Control (PTC) by December 31, 2015. PTC uses communication-based/processor-based train control technology that provides a system capable of reliably and functionally preventing train-to-train collisions, overspeed derailments, incursions into established work zone limits, and the movement of a train through a main line switch in the wrong position.

In late 2015, Congress extended the deadline by at least three years to December 31, 2018, with the possibility for two additional years if certain requirements are met. The new legislation, the

PTC Enforcement and Implementation Act, required that railroads submit a revised PTC Implementation Plan (PTCIP) by January 26, 2016, outlining when and how the railroad would have a system fully installed and activated.

FRA continues to support railroads in implementing PTC, as well as rail carriers that are continuing to voluntarily implement PTC. That assistance includes:

Providing more than \$650 million to passenger railroads, including nearly \$400 million in Recovery Act funding.

Issuing a nearly \$1 billion loan to the Metropolitan Transportation Authority to implement PTC on the Long Island Rail Road and Metro-North Railroad.

Building a PTC testbed in Pueblo, Colorado.

Making \$25 million available in competitive grant funding to railroads, suppliers, and state and local governments.

Working directly with the Federal Communications Commission (FCC) and the Advisory Council on Historic Preservation to resolve issues related to spectrum use and improve the approval process for PTC communication towers.

Dedicating staff to continue work on PTC implementation in March 2010, including establishing a PTC task force. [7]

## CHAPTER 3 – SYSTEM DESIGN

### 3.1 – Reformation of an Existing Design

It was mentioned previously that PTC was designed to prevent:

- Train-to-train collisions
- Derailments caused by excessive train speed
- Train movements through misaligned track switches
- Unauthorized train entry into work zones. [1]

However, our system has been planned in a way where we replaced some of the existing features and mechanism with our own ones so as to match the conditions of Bangladesh. We also added additional feature in our design which is ‘Obstacle Detection’ at Level Crossings. The feature, ‘Train movements through misaligned track switches’ is replaced with ‘Train Detection at Level Crossings’. Therefore, the features contained in our design is summarized below:

- Train-to-train collision detections
- Excessive train speeds detection and response
- Live view of all trains on route of the country in a Google Map API in the Central Control System.
- Communicate, Command and Control Trains through the Central Control System
- Detect approaching trains at level crossings and open or close level crossing gates automatically.
- Detect obstacles at six different positions at the level crossing and warn the approaching train about it. Trains react differently for each of those six positions of obstacles at level crossing.

There are three separate sub-systems in the entire design. They are as follows:

- Train System
- Level Crossing System
- Central Control System

The Train System is meant to be installed inside the train's cockpit, on the dashboard so that it has easy access and is clearly visible to the drivers. With most of the operations being automated, the user interface are made very user friendly so that drivers have a relaxed and fast user experience.

The Level Crossing System is meant to be installed at every major or active level crossings of the country. This system will have two external circuit boards which are IR transmitters and Receivers responsible for obstacle detection. The Level Crossing system responds to trains, communicates and acts automatically. However there is also an option for manual control. In that case the user will experience a friendly environment with on-board display and keypad with speed-functions. Output devices such as LEDs, Buzzers and Servos are connected to this system for level crossing controls.

The Central Controls System is a website where locations of all trains on route can be viewed live along with their speed and GPS data. An option of sending commands to the trains and also receiving messages from the train is also there with several other highly useful options.

More details of sub-systems are discussed in the chapters later.

## **3.2 – Components Used**

### **3.2.1 – Microcontroller**

This system uses the Arduino Mega2560, which is a high-end microcontroller unit in comparison to most other similar boards, and has been chosen so that handling large amounts of data is not an issue, as it has a fairly large RAM. The device driver programs are solely responsible for controlling the hardware devices and executing low-level hardware specific routines. These custom device drivers running on the Arduino Mega2560 microcontroller are designed specifically to suit the needs of our proposed security system.

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. There are 54 input/output pins.

→ 15 PWM (Pulse width modulation) outputs

→ 16 analog inputs

→ 4 UARTs (hardware serial ports)

- A 16 MHz crystal oscillator
- USB connection
- A power jack
- An ICSP header
- A reset button

USB connection or external power supply is the main power source of Arduino Mega which is selected automatically and the external power can come through an AC to DC adapter or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack and a battery can be inserted in the Gnd and Vin pin headers of the power connection. The board can operate on an external supply of 6 to 20 volts and the recommended range is 7 to 12 volts. Also there are some restrictions of the board's power supply. If the power supply is less than 7 volts then the 5V pin may supply less than five volts and the board is unstable but if the supply is more than 12 volts then the voltage regulator may overheat and damage the board.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter it has 256 KB of flash memory for storing code (of which 8 KB is used for the boot loader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

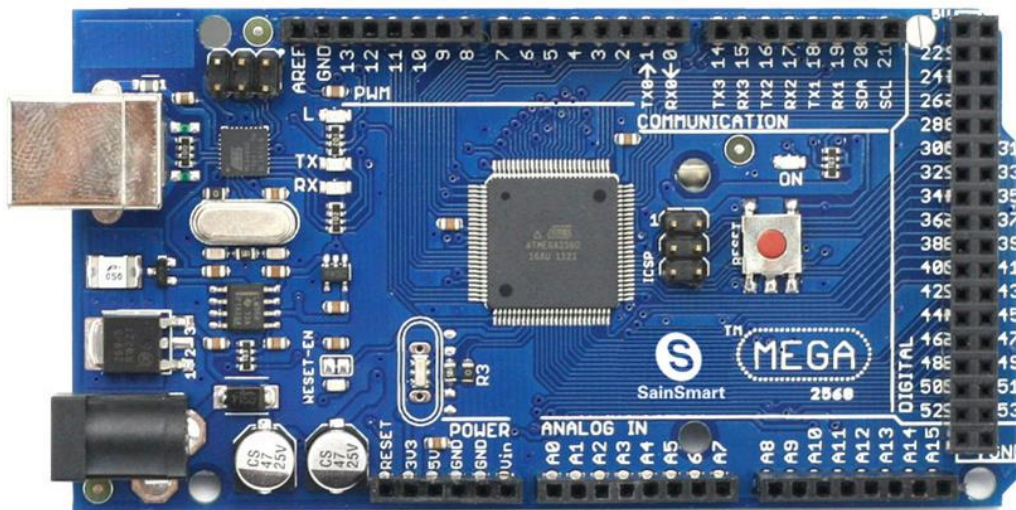


Figure 3.1 Error! No text of specified style in document.



SIM808 module is a complete Quad-Band GSM/GPRS module which combines GPS technology for satellite navigation. The compact design which integrated GPRS and GPS in a SMT package will significantly save both time and costs for customers to develop GPS enabled applications. Featuring an industry-standard interface and GPS function, it allows variable assets to be tracked seamlessly at any location and anytime with signal coverage.

It features ultra-low power consumption in sleep mode and integrated with charging circuit for Li-Ion batteries, that make it get a super long standby time and convenient for projects that use rechargeable Li-Ion battery. It has high GPS receive sensitivity with 22 tracking and 66 acquisition receiver channels. Besides, it also supports A-GPS that available for indoor localization.

The module is controlled by AT command via UART and supports 3.3V and 5V logical level.

### **Features:**

- Quad-band 850/900/1800/1900MHz
- GPRS multi-slot class12 connectivity: max. 85.6kbps (down-load/up-load)
- GPRS mobile station class B
- Controlled by AT Command (3GPP TS 27.007, 27.005 and SIMCOM enhanced AT Commands)
- Supports charging control for Li-Ion battery
- Supports Real Time Clock
- Supply voltage range 3.4V ~ 4.4V
- Integrated GPS/CNSS and supports A-GPS
- Supports 3.0V to 5.0V logic level
- Low power consumption, 1mA in sleep mode
- Supports GPS NMEA protocol
- Standard SIM Card

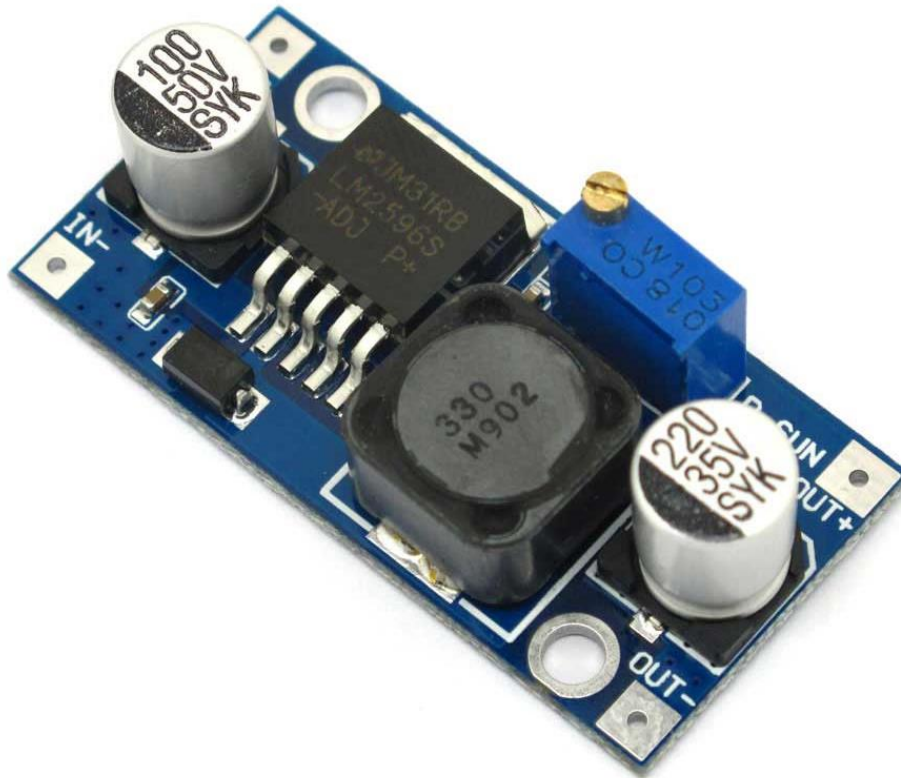
### **GPS Specifications**

- GPS Receiver channels: 22 tracking / 66 acquisition
- Coarse/Acquisition code: GPS L1
- Tracking sensitivity: -165dBm
- Time-To-First-Fix: Cold starts: 30s (typ.), Hot starts: 1s (typ.), Warm starts: 28s (typ.)



- Horizontal position accuracy: < 2.5m CEP
- Update rate: 5Hz

### 3.2.3 – LM2596 DC-DC Step-Down Buck Converter



*Figure 3.3 LM2596 DC-DC Step-Down Buck Converter*

The LM2596 regulator is monolithic integrated circuit ideally suited for easy and convenient design of a step-down switching regulator (buck converter). It is capable of driving a 3.0 A load with excellent line and load regulation. This device is available in adjustable output version and it is internally compensated to minimize the number of external components to simplify the power supply design. Since LM2596 converter is a switch-mode power supply, its efficiency is significantly higher in comparison with popular three-terminal linear regulators, especially with higher input voltages. The LM2596 operates at a switching frequency of 150 kHz thus allowing smaller sized filter components than what would be needed with lower frequency switching regulators. Available in a standard 5-lead TO-220 package with several different lead bend

options, and D2PAK surface mount package. The other features include a guaranteed 4% tolerance on output voltage within specified input voltages and output load conditions, and 15% on the oscillator frequency. External shutdown is included, featuring 80 A (typical) standby current. Self-protection features include switch cycle-by-cycle current limit for the output switch, as well as thermal shutdown for complete protection under fault conditions.

## **Features**

- Adjustable Output Voltage Range 1.23 V – 37 V
- Guaranteed 3.0 A Output Load Current
- Wide Input Voltage Range up to 40 V
- 150 kHz Fixed Frequency Internal Oscillator
- TTL Shutdown Capability
- Low Power Standby Mode, typ. 80 A
- Thermal Shutdown and Current Limit Protection
- Internal Loop Compensation
- Moisture Sensitivity Level (MSL) Equals 1
- Pb-Free Packages are Available

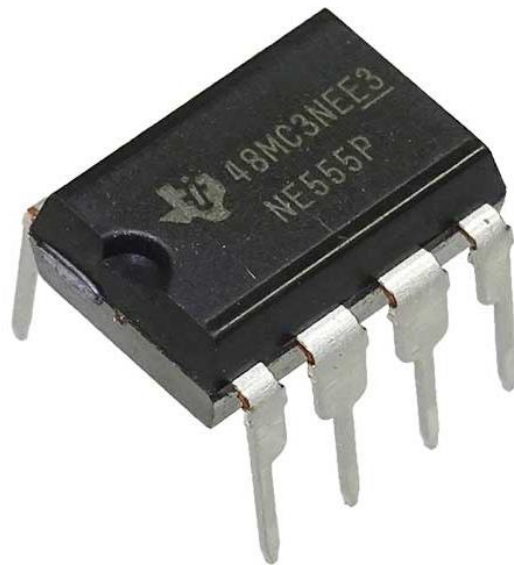
## **Applications**

- Simple High-Efficiency Step-Down (Buck) Regulator
- Efficient Pre-Regulator for Linear Regulators
- On-Card Switching Regulators
- Positive to Negative Converter (Buck-Boost)
- Negative Step-Up Converters
- Power Supply for Battery Chargers

### **3.2.4 – NE555 IC**

The 555 timer IC is an integrated circuit (chip) used in a variety of timer, pulse generation, and oscillator applications. The 555 can be used to provide time delays, as an oscillator, and as a flip-flop element. Derivatives provide two or four timing circuits in one package.

Depending on the manufacturer, the standard 555 package includes 25 transistors, 2 diodes and 15 resistors on a silicon chip installed in an 8-pin mini dual-in-line package (DIP-8).[8] Variants available include the 556 (a 14-pin DIP combining two 555s on one chip), and the two 558 & 559s (both a 16-pin DIP combining four slightly modified 555s with DIS & THR connected internally, and TR is falling edge sensitive instead of level sensitive).



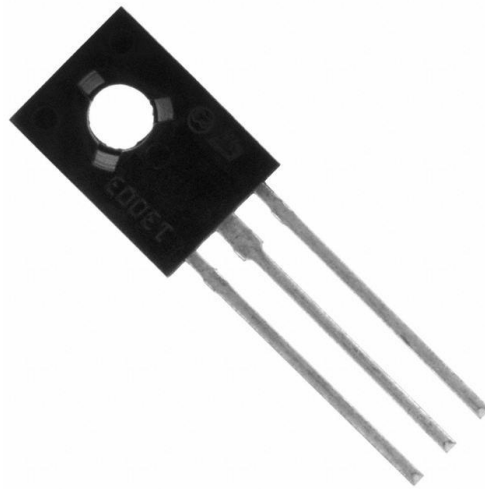
*Figure 3.4Error! No text of specified style in document. NE555IC*

The NE555 parts were commercial temperature range, 0 °C to +70 °C, and the SE555 part number designated the military temperature range, -55 °C to +125 °C. These were available in both high-reliability metal can (T package) and inexpensive epoxy plastic (V package) packages. Thus the full part numbers were NE555V, NE555T, SE555V, and SE555T. It has been hypothesized that the 555 got its name from the three 5 k $\Omega$  resistors used within, [9] but Hans Camenzind has stated that the number was arbitrary. [11]

Low-power versions of the 555 are also available, such as the 7555 and CMOS TLC555.[10] The 7555 is designed to cause less supply noise than the classic 555 and the manufacturer claims that it usually does not require a "control" capacitor and in many cases does not require a decoupling capacitor on the power supply. Those parts should generally be included, however, because noise produced by the timer or variation in power supply voltage might interfere with other parts of a circuit or influence its threshold voltages.

### 3.2.5 – BD140 IC

BD140 are epitaxial planar transistors which are mounted in the SOT-32 plastic package. They are designed for audio amplifiers and drivers utilizing complementary or quasi-complementary circuits. The NPN types are the BD135 and BD139, and the complementary PNP types are the BD136 and BD140.



*Figure Error! No text of specified style in document. BD140 IC*

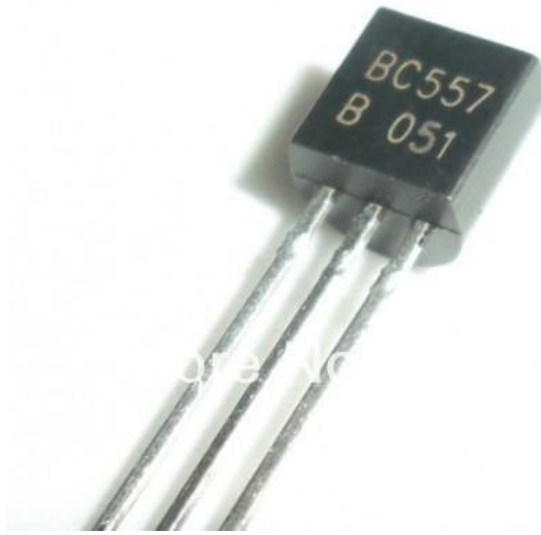
They are used in Medium Power Linear and Switching Applications. BD135, BD137 and BD139 are complement to BD136, BD138 and BD140 respectively.

#### **Key Features**

- Products are pre-selected in DC current gain

### 3.2.6 – BC557 IC

BC557 is a general purpose PNP transistor for switching and amplification. The NPN complements are: BC546 and BC547.



*Figure 3.6 BC557 IC*

**Features:**

- PNP Silicon Planar Epitaxial Transistors.
- Especially suited for use in Driver Stages of Audio Amplifiers, Low Noise Input Stages of Tape Recorders, HI-FI Amplifiers, Signal Processing Circuits of Television Receivers.
- Low current (max. 100 mA)
- Low voltage (max. 65 V).

**3.2.7 – Resistors**

The values of resistors used in this project are shown as follows:

- 150K $\Omega$ , ¼ Watt
- 10K $\Omega$ , ¼ Watt
- 1.5K $\Omega$ , ¼ Watt
- 1K $\Omega$ , ¼ Watt
- 560 $\Omega$ , ¼ Watt
- 470 $\Omega$ , ¼ Watt

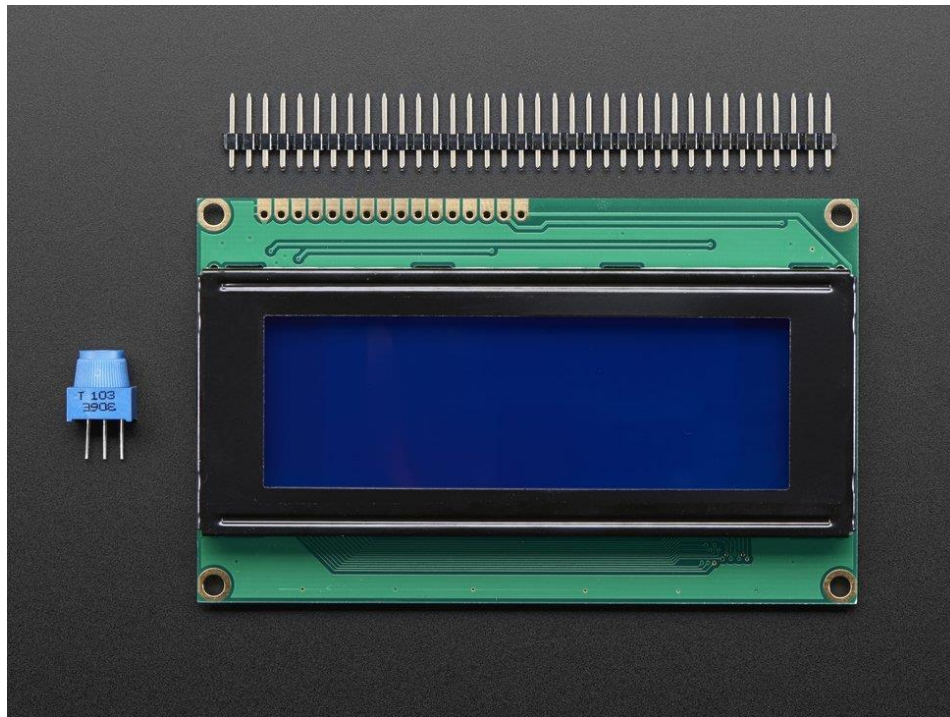
- 330 $\Omega$ , ¼ Watt
- 220 $\Omega$ , ¼ Watt
- 5.6 $\Omega$ , ¼ Watt
- 1 $\Omega$ , ¼ Watt

### 3.2.8 – Capacitors

The Values of capacitors used in this project are shown as follows:

- 0.01 $\mu$ F, Ceramic-Disc
- 1 $\mu$ F, 50V Electrolytic

### 3.2.9 – 20X4 LCD



*Figure 3.7 20X4 LCD*

This 20x4 Character LCD Display is built-in with RW1063 controller IC which are 6800, 4 line SPI or I2C interface options. The WH2004G 20x4 LCD Display have the same AA size and pin assignment as existing WH2004A and WH2004B character LCD modules but with smaller outline and VA size.

## Features:

- 20 characters wide, 4 rows
- White text on blue background
- Connection port is 0.1" pitch, single row for easy breadboarding and wiring
- Single LED backlight with a resistor included, you can power it directly from 5V. If it's too bright for you, it can be dimmed easily with a resistor or PWM and uses much less power than LCD with EL (electroluminescent) backlights
- Can be fully controlled with only 6 digital lines!
- Built in character set supports English/Japanese text, see the HD44780 datasheet for the full character set
- Up to 8 extra characters can be created for custom glyphs or 'foreign' language support (like special accents)

### 3.2.10 – 4X4 Matrix Keypad

This 16-button keypad provides a useful human interface component for microcontroller projects. Convenient adhesive backing provides a simple way to mount the keypad in a variety of applications. The module has two working modes. An 8 key mode that provides an independent 8 channel output or a 16 key mode that can be used with the I2C interface of the module, thereby saving even more pins and connections on the application Arduino or Microcontroller board.



Figure 3Error! No text of specified style in document. 8 4X4 Matrix Keypad

## Features

- Ultra-thin design
- Adhesive backing
- Excellent price/performance ratio
- Easy interface to any microcontroller
- Example programs provided for the BASIC
- Stamp 2 and Propeller P8X32A microcontrollers

### 3.2.11 – Potentiometer (10k ohm)

An adjustable potentiometer can open up many interesting user interfaces. Turn the pot and the resistance changes. Connect VCC to an outer pin, GND to the other, and the center pin will have a voltage that varies from 0 to VCC depending on the rotation of the pot. Hook the center pin to an ADC on a microcontroller and get a variable input from the user!



*Figure 3.9 100K Potentiometer*

This is a center-tap linear type potentiometer. The outer two pins will always show 100K resistance, the center pin resistance to one of the outer pins will vary from 100K Ohm to about 10K Ohm. The pot is linear meaning the resistance will vary linearly with its position. This is a good choice for general user interfaces.



### 3.2.12 – SG90 Servo

Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.



Figure 3Error! No text of specified style in document..10 SG90 Servo

#### Specifications:

- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgfcm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10  $\mu$ s
- Temperature range: 0 °C – 55 °C

### 3.2.13 – LED (5mm)

A light-emitting diode (LED) is a two-lead semiconductor light source. LEDs - those blinky things. A must have for power indication, pin status, opto-electronic sensors, and fun blinky displays.

This is a very basic 5mm LED with a red lens. It has a typical forward voltage of 2.0V and a rated forward current of 20mA.



*Figure 3.11 LED Lights*

**Applications:**

- Status indicator.
- Backlighting front panels.
- Light pipe sources.
- Lighted switches.

**Features:**

- High luminous intensity output.
- Low power consumption.
- Versatile mounting on PCB or panel.
- Popular T-1 diameter package.
- Reliable and rugged.
- RoHS compliant.

**3.2.14 – Buzzer**

Early devices were based on an electromechanical system identical to an electric bell without the metal gong. Similarly, a relay may be connected to interrupt its own actuating current, causing the contacts to buzz. Often these units were anchored to a wall or ceiling to use it as a sounding board. The word "buzzer" comes from the rasping noise that electromechanical buzzers made.



*Figure 3.12 Buzzer*

While technological advancements have caused buzzers to be impractical and undesirable, there are still instances in which buzzers and similar circuits may be used.

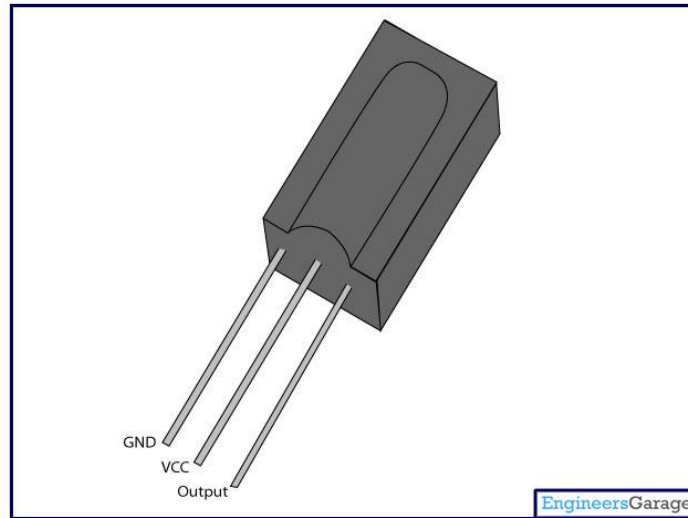
### **3.2.15 – TSOP1738**

The TSOP 1738 is a member of IR remote control receiver series. This IR sensor module consists of a PIN diode and a pre amplifier which are embedded into a single package. The output of TSOP is active low and it gives +5V in off state. When IR waves, from a source, with a center frequency of 38 kHz incident on it, its output goes low.

Lights coming from sunlight, fluorescent lamps etc. may cause disturbance to it and result in undesirable output even when the source is not transmitting IR signals. A bandpass filter, an integrator stage and an automatic gain control are used to suppress such disturbances.

TSOP module has an inbuilt control circuit for amplifying the coded pulses from the IR transmitter. A signal is generated when PIN photodiode receives the signals. This input signal is received by an automatic gain control (AGC). For a range of inputs, the output is fed back to AGC in order to adjust the gain to a suitable level. The signal from AGC is passed to a band pass filter to filter undesired frequencies. After this, the signal goes to a demodulator and this demodulated output drives an NPN transistor. The collector output of the transistor is obtained at pin 3 of TSOP module.

Members of TSOP17xx series are sensitive to different center frequencies of the IR spectrum. For example TSOP1738 is sensitive to 38 kHz whereas TSOP1740 to 40 kHz center frequency. [43]



*Figure 3.13 TSOP1738 IR Receiver*

### **3.2.16 – IR Transmitters**

The IR transmitter consists of the LED that emits the IR (Infra-Red) radiation. This is received by the photo diode, which acts as IR receiver at the receiving end. Since the IR radiation is invisible to human eye it is perfect for using in wireless communication.

An electronic remote device mainly consists of this IR transmitter and receiver. A remote control patterns a flash of invisible light which is turned into an instruction and is received by the receiver module.

The IR signal is modulated during transmission. Modulation means assigning pattern to the data to be sent to the receiver. The most commonly used IR modulation is about 38 kHz.

### **3.2.17 – Others**

Other components include:

- DC Power Plug
- DC Power Socket
- 12V, 3A DC Power Adapter
- 6V, 4500mAh DC Lead-Acid Battery
- 12V, 4500mAh DC Lead-Acid Battery
- Male-Male Jumper wires
- Male-Female Jumper wires
- Female-Female Jumper wires

- Male Connectors
- Female Connectors
- Arduino Stackable Header Pins
- SPDT Toggle Switches
- DPDT Button Switches
- DPDT Push Switches
- PVC Boards
- Tools

### 3.3 – Communication Flow

Communication is done by means of GSM and GPRS using the SIM808 module. Train system communicates with the Central Control System the level crossings and the Stations. Level Crossings Communicate with the Central Control System and trains. Central Control System communicates with the emergency services of the country and vice versa. We used two methods of communication. One is via SMS messages to and from the GSM modules, and another is HTTP, three-way handshake. Communication to and from the Central Control System is done using the HTTP. Trains and Level Crossings communicate by SMS messages.

The communication paths are described in the diagram below:

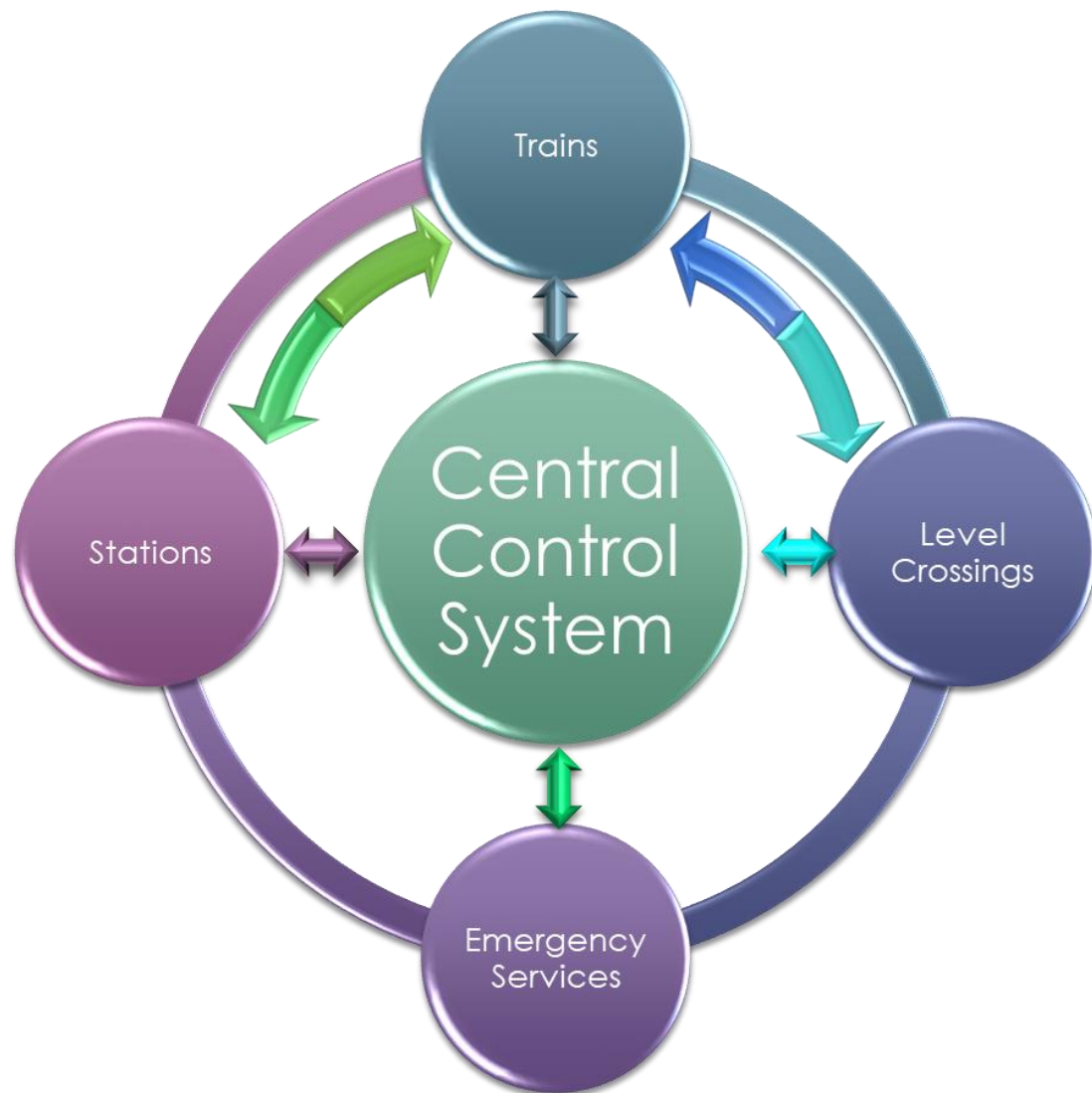


Figure 3Error! No text of specified style in document..14 Communication Flow Diagram

### 3.4 – Message Codes

Communication among each of the systems are done using SMS Messages using certain message codes. Those codes are then translated within the system and displayed to the user so that it is easy to understand. The Message Codes that are used in each of the systems are listed in the below table indicating the recipient and the client.

FROM	TO	SMS CODE	DESCRIPTION	ACTION
LEVEL CROSSING	TRAINS	OBS-010	“WARNING! OBSTACLE DETECTED”	TRAIN: ACT ACCORDING TO SPEED & DISTANCE LEVEL CROSSING: GATE 1 OR GATE 2 OR GATE 3 OR GATE 4 OPEN
LEVEL CROSSING	TRAINS	OBS-010	“WARNING! OBSTACLE DETECTED”	TRAIN: ACT ACCORDING TO SPEED & DISTANCE LEVEL CROSSING: GATE 1 OR GATE 2 OR GATE 3 OR GATE 4 OPEN
LEVEL CROSSING	TRAINS	OBS-111	“WARNING! OBSTACLE DETECTED”	TRAIN: ACT ACCORDING TO SPEED & DISTANCE LEVEL CROSSING: N/A
LEVEL CROSSING	TRAINS	OBS-100	“WARNING! OBSTACLE DETECTED”	TRAIN: ACT ACCORDING TO SPEED & DISTANCE LEVEL CROSSING: GATE 1 OR GATE 2 OR GATE 3 OR GATE 4 OPEN
LEVEL CROSSING	TRAINS	OBS-001	“WARNING! OBSTACLE DETECTED”	TRAIN: ACT ACCORDING TO SPEED & DISTANCE LEVEL CROSSING: GATE 1 OR GATE 2 OR GATE 3 OR GATE 4 OPEN
LEVEL CROSSING	TRAINS	CLR-000	“TRACK IS CLEAR TO GO”	TRAIN: GO
LEVEL CROSSING	TRAINS	EMG-111	“EMERGENCY STOP REQUESTED”	TRAIN: STOP IMMEDIATELY
LEVEL CROSSING	TRAINS	GATE-0001	“CAUTION! LEVEL CROSSING GATE 1 IS OPEN”	TRAIN: ACT ACCORDING TO SPEED & DISTANCE
LEVEL CROSSING	TRAINS	GATE-0010	“CAUTION! LEVEL CROSSING GATE 2 IS OPEN”	TRAIN: ACT ACCORDING TO SPEED & DISTANCE
LEVEL CROSSING	TRAINS	GATE-0100	“CAUTION! LEVEL CROSSING GATE 3 IS OPEN”	TRAIN: ACT ACCORDING TO SPEED & DISTANCE

LEVEL CROSSING	TRAINS	GATE-1000	"CAUTION! LEVEL CROSSING GATE 4 IS OPEN"	TRAIN: ACT ACCORDING TO SPEED & DISTANCE
TRAINS	LEVEL CROSSING	APP-TRAIN_ID-0001	"TRAIN APPROACHING"	LEVEL CROSSING: LOWER GATES, REMOVE OBSTACLE, RESPOND TO TRAIN
TRAIN	LEVEL CROSSING	HLT-001	"TRAIN ID: ____ IS PREPARING TO HALT"	LEVEL CROSSING: INSTRUCT TO CLEAR LINE
TRAIN	LEVEL CROSSING	HLT-111	"TRAIN ID: ____ HALTED DUE TO OBSTACLE"	LEVEL CROSSING: INSTRUCT TO CLEAR LINE
TRAIN	LEVEL CROSSING	HLT-000	"TRAIN ID: ____ FAILED TO HALT"	LEVEL CROSSING: EMERGENCY ALERT, INSTRUCT TO CLEAR LINE IMMEDIATELY. POSSIBILITY OF COLLISION
TRAIN	LEVEL CROSSING	RES-111	"TRAIN ID: ____ IS RESUMING ITS JOURNEY"	LEVEL CROSSING: ACKNOWLEDGE
TRAIN	LEVEL CROSSING	BRK-111	"TRAIN ID: ____ HAS BRAKE FAILURE"	LEVEL CROSSING: EMERGENCY ALERT, INSTRUCT TO CLEAR LINE IMMEDIATELY. POSSIBILITY OF COLLISION
TRAIN	LEVEL CROSSING	ENG-111	"TRAIN ID: ____ HAS ENGINE FAILURE"	LEVEL CROSSING: ACKNOWLEDGE
TRAIN	CCS	HLT-001	"TRAIN ID: ____ IS PREPARING TO HALT"	LEVEL CROSSING: INSTRUCT TO CLEAR LINE
TRAIN	CCS	HLT-111	"TRAIN ID: ____ HALTED DUE TO OBSTACLE"	LEVEL CROSSING: INSTRUCT TO CLEAR LINE
TRAIN	CCS	HLT-000	"TRAIN ID: ____ FAILED TO HALT"	LEVEL CROSSING: EMERGENCY ALERT, INSTRUCT TO CLEAR LINE IMMEDIATELY. POSSIBILITY OF COLLISION
TRAIN	CCS	RES-111	"TRAIN ID: ____ IS RESUMING JOURNEY"	LEVEL CROSSING: ACKNOWLEDGE
TRAIN	CCS	BRK-111	"TRAIN ID: ____ HAS BRAKE FAILURE"	LEVEL CROSSING: EMERGENCY ALERT, INSTRUCT TO CLEAR LINE



				IMMEDIATELY. POSSIBILITY OF COLLISION
TRAIN	CCS	ENG-111	"TRAIN ID: _____ HAS ENGINE FAILURE	LEVEL CROSSING: ACKNOWLEDGE
TRAIN	CCS	PPP-111	"TRAIN ID: _____ WILL BE STOPPING AT NEXT STATION"	CCS: AKCNOWLEDGE STATION: ACKNOWLEDEGE
TRAIN	CCS	STA-111	"TRAIN ID: _____ IS REQUESTING TO START JOURNEY. DESTINATION: _____"	CCS: ACKNOWLEDGE
TRAIN	CCS	STO-111	"TRAIN ID: _____ HAS REACHED ITS DESTINATION SAFELY."	CCS: ACKNOWLEDGE
TRAIN	CCS	DUR-111	"TRAIN ID: _____ - HALT DURATION: " + TIME	CCS: ACKNOWLEDGE, AND INSTRUCT NEXT TRAIN ACCORDINGLY
TRAIN	CCS	MED-111	"TRAIN ID: _____ REQUESTING MEDICAL ASSISTANCE"	CCS: CONTACT MEDICAL SERVICES AND SEND TRAIN'S LOCATION TO THEM
TRAIN	CCS	ACC-111	"TRAIN ID: _____ HAD AN ACCIDENT"	CCS: ACT ACCORDINGLY
TRAIN	CCS	HJK-111	"TRAIN ID: _____ HAS BEEN HIJACKED	CCS: ACKNOWLEDGE AND CONTACT LAW ENFORCEMENT AGENCY
TRAIN	CCS	FIR-111	"TRAIN ID: _____ IS ON FIRE"	CCS: ACKNOWLEDGE AND CONTACT FIRE BRIGADE
TRAIN	CCS	LAW-111	"TRAIN ID: _____ REQUESTING LAW ENFORCEMENT ASSISTANCE"	CCS: ACKNOWLEDGE AND CONTACT LAW ENFORCEMENT AGENCY
TRAIN	CCS	SER-111	"TRAIN ID: _____ IS REQUESTING MAINTENANCE SERVICE BACKUP"	CCS: ACKNOWLEDGE. SEND SERVICE BACKUP TO TRAINS LOCATION
CCS	TRAIN	SPD-DEC	"PLEASE DECREASE YOUR SPEED"	TRAIN: DECREASES SPEED
CCS	TRAIN	SPD-INC	"PLEASE INCREASE YOUR SPEED"	TRAIN: INCREASES SPEED
CCS	TRAIN	EMG-111	"EMERGENCY STOP REQUESTED"	TRAIN: STOP IMMEDIATELY
CCS	TRAIN	STP-111	"STOP AT NEXT STATION"	TRAIN: STOP AT NEXT STATION

CCS	TRAIN	GGO-111	“YOU ARE GOOD TO GO”	“TRAIN: ACKNOWLEDGE. START JOURNEY
CCS	TRAIN	MED-001	“MEDICAL ASSISTANCE IS ON THEIR WAY TO YOUR LOCATION”	TRAIN: ACKNOWLEDGE
CCS	TRAIN	SER-001	“SERVICE ASSISTANCE IS ON THEIR WAY TO YOUR LOCATION”	TRAIN: ACKNOWLEDGE
CCS	TRAIN	LAW-001	“LAW ENFORCEMENT ASSISTANCE IS ON THEIR WAY TO YOUR LOCATION”	TRAIN: ACKNOWLEDGE

*Table 3.1 Message Codes and their corresponding actions*

## **CHAPTER 4 – HARDWARE IMPLEMENTATIONS**

### **4.1 – Arduino Mega 2560**

The microcontroller used in this project is ATMEL ATMEGA 2560, which is an 8-bit AVR RISC-based microcontroller, embedded in a development board called Arduino Mega as a whole. The ease of usage and convenience of Arduino Mega development board is a fundamental aspect for projects or experiments, hence selected for such a project.

The Arduino acts as the cores of each of the sub-systems where all other components are connected and controlled. Its huge number of output ports is also a reason to choose this microcontroller development board for this project. Also the high processing power compared to other project boards of this class was taken into account for choosing this board.

Two of these modules were used in the Train System and one in the Level Crossing System.

### **4.2 – SIMCOM SIM808 GSM+GPS+GPRS Module**

SIM808 module is a complete Quad-Band GSM/GPRS module which combines GPS technology for satellite navigation.

The compact design which integrated GPRS and GPS in a SMT package will significantly save both time and costs for customers to develop GPS enabled applications. Featuring an industry-standard interface and GPS function, it allows variable assets to be tracked seamlessly at any location and anytime with signal coverage. [12]

The ability of wide range of applications and features compared to other GSM modules makes it best for the use in the purpose of this project.

We used these modules in the Train System as well as the Level Crossing System for GSM, GPRS and GPS. Two of these modules were used in the Train System and one in the Level Crossing System.

### **4.3 – LM2596 DC-DC Step-Down Buck Converter**

These converters come in very handy when we need multiple number of voltage sources in one board. Its wide range of adjustable Output Voltage Range which is 1.23 V – 37 V makes it very

convenient to use for converting higher voltages to our desired ones. Remarkably, these modules also have a wide range of input voltages which is up to 40V.

One of these converters were used to convert 12V input voltages to 5V in the Train System and two of these were used in the Level Crossing System to convert 5V and 9V outputs respectively.

## **4.4 – Peripheral Devices**

### **4.4.1 – Peripheral Devices in the Train System**

The Peripheral Devices used in the Train System are as follows:

- **Signal LEDs:** LEDs of three colors – Green, Yellow and Red are used as signals for the train driver whether to Go, Go Slow or Stop respectively. These LEDs are mounted on-board for providing expedient user experience for the driver.
- **New Message LEDs:** Two blue LEDs are used to indicate the driver there is a new message either from the Central Control System or a nearby Level Crossing. These LEDs are also mounted on-board.
- **Emergency Alert LEDs:** Nine Red LEDs were used for maximum illumination for any emergency alert to grab the attention of the driver or any person in the train's cockpit. These LED's are also mounted on board.
- **Power LEDs:** Two Red LEDs were used to indicate which power sources are ON or OFF on the Train System Mainboard.
- **Buzzer:** A 40dB buzzer is used for emergency alert to generate sound for grabbing the attention of the driver of the train or any persons in the train's cockpit.
- **20X4 LCDs:** Four LCDs each showing different information were mounted on board of the Train System Mainboard. The LCDs shows current status and location of the train, new messages, arrival and departure information, obstacle information respectively.
- **4X4 Matrix Keypad:** Two of these keypads were used to control the Menu, for example, scrolling or going to next page and fast-send messages to Central Control System or a nearby Level Crossing manually.
- **100K Ohm Potentiometers:** These potentiometers are installed to control the brightness and contrast of each of the LCDs in the Train System Mainboard.

#### 4.4.2 – Peripheral Devices in the Level Crossing System

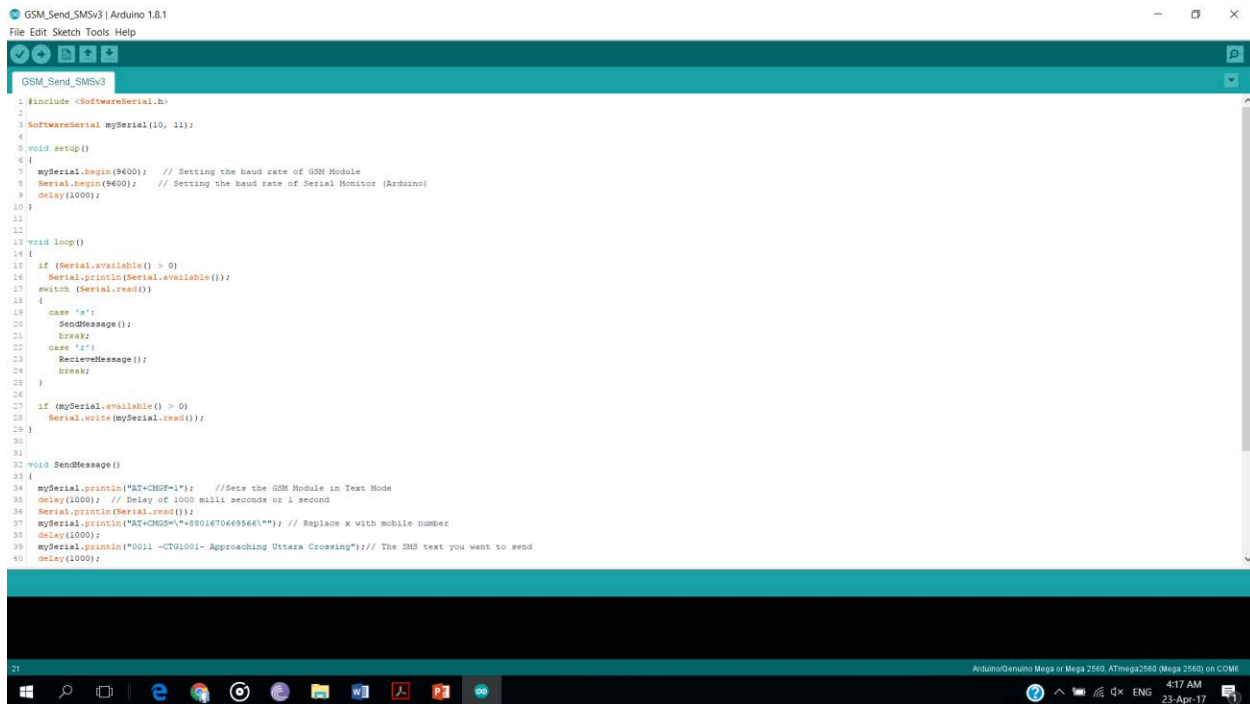
The peripheral devices used in the Level Crossing system are described as follows:

- **Gate LEDs:** Four Red LEDs were used to see the status of the Gates at the Level Crossing. This allows the operator at the level crossing to know the status of the gates without going outside as they are mounted on the mainboard. The LEDs glow upon the opening of the gates.
- **Buzzer:** The 40dB buzzer is used to signal the traffic if a train is arriving is not. The Buzzer is placed at the road divider and is connected with the mainboard with extension of wires.
- **Road Signal LEDs:** LEDs of two colors-Red and Green- are used to signal the traffic whether a train is arriving or not. The red LEDs glow whenever the level crossing knows that a train is arriving. Otherwise the Green LED stays ON. Two sets of these LEDs were used for two sides of the road.
- **Train Signal LEDs:** LEDs of three colors-Green, Yellow and Red- are used to signal the train whether to go, go slow or stop respectively. These LEDs are placed between two train tracks a few distances further away from the level crossing. Two sets of these LEDs are used to signal the trains from both sides.
- **Servo Motors:** Four Servo motors are used which act as the Level Crossing Gates for stopping the traffic whenever a train is incoming at the Level Crossing.
- **IR Transmitter and Receiver Array:** These are used for obstacle detection at the level crossing. Details are discussed in the chapters later.

# CHAPTER 5 – SOFTWARE IMPLEMENTATIONS

## 5.1 – Arduino IDE

One of the major components of this security and safety system is the software coding in order to make the system fully functional. The core of the system is the Arduino Mega development board which is programmed using Arduino IDE version 1.8.1. The programming language supported by the board is based on C and C++ languages and is coded using the Arduino development environment. The usage of programming in this project facilitates effective communications and fast processing between hardware components that are connected to the microcontroller.



```
GSM_Send_SMSv3 | Arduino 1.8.1
File Edit Sketch Tools Help
GSM_Send_SMSv3
1 #include <SoftwareSerial.h>
2
3 SoftwareSerial mySerial(10, 11);
4
5 void setup()
6 {
7   mySerial.begin(9600); // Setting the baud rate of GSM Module
8   Serial.begin(9600); // Setting the baud rate of Serial Monitor (Arduino)
9   Delay(1000);
10 }
11
12
13 void loop()
14 {
15   if (Serial.available() > 0)
16     Serial.println(Serial.available());
17   switch (Serial.read())
18   {
19     case 's':
20       SendMessage();
21       break;
22     case 'r':
23       ReceiveMessage();
24       break;
25   }
26
27   if (mySerial.available() > 0)
28     Serial.write(mySerial.read());
29 }
30
31
32 void SendMessage()
33 {
34   mySerial.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
35   Delay(1000); // Delay of 1000 milli seconds or 1 second
36   Serial.println(Serial.read());
37   mySerial.println("AT+CMSS="+8901670669566"); // Replace x with mobile number
38   Delay(1000);
39   mySerial.println("0011 -CTD1001- Approaching Utkara Crossing");// The SMS text you want to send
40   Delay(1000);
41 }
```

Figure 5.1 Arduino IDE Screenshot

Arduino IDE is written in Java and is a cross-platform application. It is designed to help programmers to configure the Arduino microcontrollers to their preference. Arduino programs are written in C or C++ but require two functions, setup() and loop(), to make a runnable cyclic executive program. The setup() function initializes all the settings and runs once at the start of the program. The loop() function executes the microcontrollers main job and is called repeatedly until the board powers off.

The ATmega2560 on the Arduino Mega 2560 comes preprogrammed with a bootloader that allows users to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500. It can also be able to bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode. Then the Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) can be used to load a new firmware. Or the ISP header with an external programmer (overwriting the DFU bootloader) can also be used.

### **5.1.1. Software Libraries**

Programming also enables the microcontroller to learn about different devices that are attached with it. For each of the different modules used for the architecture of the system, different software libraries were used.

A software library is a suite of data and programming code that is used to develop software programs and applications. It is designed to assist both the programmer and the programming language compiler in building and executing software.

A software library generally consists of pre-written code, classes, procedures, scripts, configuration data and more. Typically, a developer might manually add a software library to a program to achieve more functionality or to automate a process without writing code for it. For example, when developing a mathematical program or application, a developer may add a mathematics software library to the program to eliminate the need for writing complex functions. All of the available functions within a software library can just be called/used within the program body without defining them explicitly. Similarly, a compiler might automatically add a related software library to a program on run time.

In other words, libraries are a collection of code that makes it easy to connect to a variety of modules or devices such as a sensor, display etc. For example, the built-in LiquidCrystal library makes it easy to talk to character LCD displays.

In this project a number of libraries were required so as to interface different devices and modules which are included in this process of software implementation to make the system functional. Libraries were used for the GSM Module, Keypad and the Liquid Crystal Display.

The aforementioned devices or modules and their corresponding library files that were used to program the system are shown in the table below.

Sl.no.	Devices/ Modules	Library files
3	GSM Module	SIM808, SoftwareSerial.h
4	Keypad	Keypad.h
5	Liquid Crystal Display	LiquidCrystal.h

*Table 5.1 Devices/ Modules and their corresponding libraries*

### 5.1.2. Functions & Syntax

Several built-in functions from each of the libraries and default Arduino functions are used to operate different tasks by the devices connected to the Arduino.

The functions and their corresponding tasks are shown in the following table.

Sl. no.	Functions	Tasks
1	<code>#include&lt;SoftwareSerial.h&gt;</code>	Imports the Software serial library file
2	<code>#include&lt;Keypad.h&gt;</code>	Imports the Keypad library file
5	<code>#include &lt;LiquidCrystal.h&gt;</code>	Imports the LCD library file
3	<code>SoftwareSerialmySerial(10, 11);</code>	Creates an object named mySerial for GSM module. Indicates that the RX pin of the module is connected to pin 10 and TX pin is connected to pin 11 of the Arduino.
4	<code>mySerial.begin(9600);</code>	Sets Baud-rate of the GSM Module
5	<code>mySerial.println();</code>	Sends commands to the GSM Module
6	<code>Keypad myKeypad = Keypad(makeKeymap(keymap), rowPins, colpins, 4, 4);</code>	Initializes the Keypad. Makes a map of the keypad using rows and columns
7	<code>myKeypad.waitForKey();</code>	Waits until a key is pressed from the keypad
8	<code>lcd.begin(16, 2);</code>	Initializing LCD. Setting number of rows and columns of the LCD.



9	<code>lcd.setCursor(column, row);</code>	Setting the cursor of the LCD
10	<code>lcd.print(" ");</code>	Displaying message on the LCD
11	<code>lcd.write(" ");</code>	Displaying a single character on the LCD
12	<code>lcd.clear();</code>	Clears the LCD screen
13	<code>pinMode(pin number, INPUT/OUTPUT);</code>	Configures the specified pin to behave either as an input or an output.
14	<code>digitalWrite(pin number, HIGH/LOW);</code>	Writes a HIGH (5V) or a LOW (0V) value to a digital pin.
15	<code>digitalRead(pin number)</code>	Reads the value from a specified digital pin, either HIGH (5V) or LOW (0V).
16	<code>analogRead(pin number)</code>	Reads the value from the specified analog pin.
17	<code>delay(ms)</code>	Delays by the value mentioned in microseconds
18	<code>attachInterrupt(digital pin to interrupt, interrupt method, mode)</code>	Attaches Interrupt to a digital pin. The pins that support interrupt are 2, 3, 18, 19, 20, 21
19	<code>&lt;include Wire.h&gt;</code>	Includes the wire library which enables I2C communication. Typically used to connect another arduino.

*Table 5.2 Functions used in programming and their corresponding tasks*

## 5.2 – Fritzing

Fritzing is an open source initiative to develop amateur or hobby CAD software for the design of electronics hardware, to support designers and artists ready to move from experimenting with a prototype to building a more permanent circuit. It was developed at the University Of Applied Sciences Of Potsdam. [13]

The software is created in the spirit of the Processing programming language and the Arduino microcontroller [14] and allows a designer, artist, researcher, or hobbyist to document their Arduino-based prototype and create a PCB layout for manufacturing. The associated website helps users share and discuss drafts and experiences as well as to reduce manufacturing costs.

Fritzing can be seen as an electronic design automation (EDA) tool for non-engineers: the input metaphor is inspired by the environment of designers (the breadboard-based prototype), while the

output is focused on accessible means of production. As of December 2, 2014 Fritzing has made a code view option, where one can modify code and upload it directly to an Arduino device. [15]

Component images are distributed under CC-BY-SA, which will also be the license for any generated breadboard views.

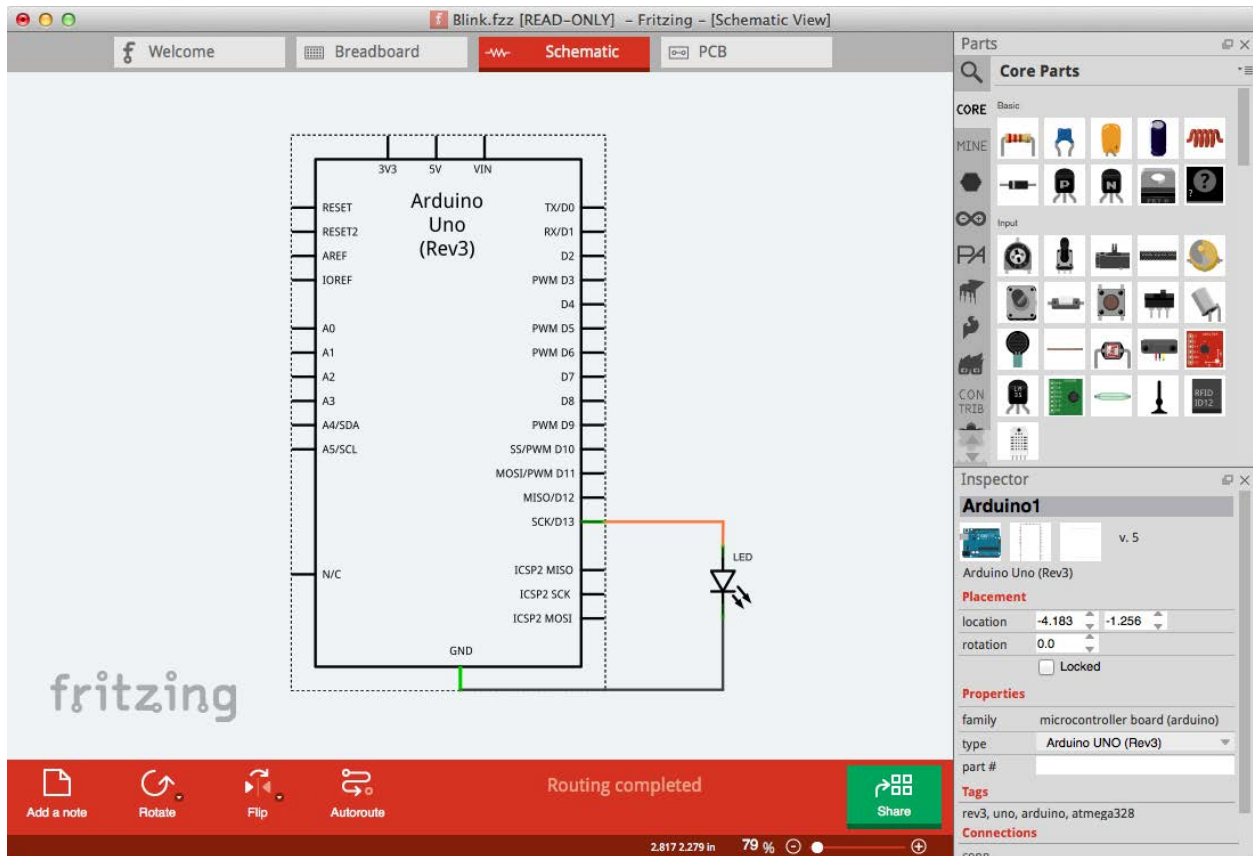


Figure 5.2 the Fritzing Desktop App

In our project Fritzing was mainly used to design the PCBs because of its user friendly interface. Fritzing is also fast to understand as it offers a unique real-life "breadboard" view, and a parts library with many commonly used high-level components. Fritzing makes it very easy to communicate about circuits, as well as to turn them into PCB layouts ready for production.

### 5.3 – Proteus Professional

The Proteus Design Suite is an Electronic Design Automation (EDA) tool including schematic capture, simulation and PCB Layout modules. It is developed in Yorkshire, England by Labcenter Electronics Ltd with offices in North America and several overseas sales channels. The software

runs on the Windows operating system and is available in English, French, Spanish and Chinese languages.

Schematic capture in the Proteus Design Suite is used for both the simulation of designs and as the design phase of a PCB layout project. It is therefore a core component and is included with all product configurations.

The micro-controller simulation in Proteus works by applying either a hex file or a debug file to the microcontroller part on the schematic. It is then co-simulated along with any analog and digital electronics connected to it. This enables it's used in a broad spectrum of project prototyping in areas such as motor control, [16] [17] temperature control [18] [19] and user interface design. [20] It also finds use in the general hobbyist community [21] [22] and, since no hardware is required, is convenient to use as a training [23] [24] or teaching tool. [25][26] Support is available for co-simulation of:

Microchip Technologies PIC10, PIC12, PIC16, PIC18, PIC24, dsPIC33 Microcontrollers.

Atmel AVR (and Arduino), 8051 and ARM Cortex-M3 Microcontrollers

NXP 8051, ARM7, ARM Cortex-M0 and ARM Cortex-M3 Microcontrollers.

Texas Instruments MSP430, PICCOLO DSP and ARM Cortex-M3 Microcontrollers.

Parallax Basic Stamp, Freescale HC11, 8086 Microcontrollers.

Proteus was used to create circuit schematics and simulate before it was implemented in practical.

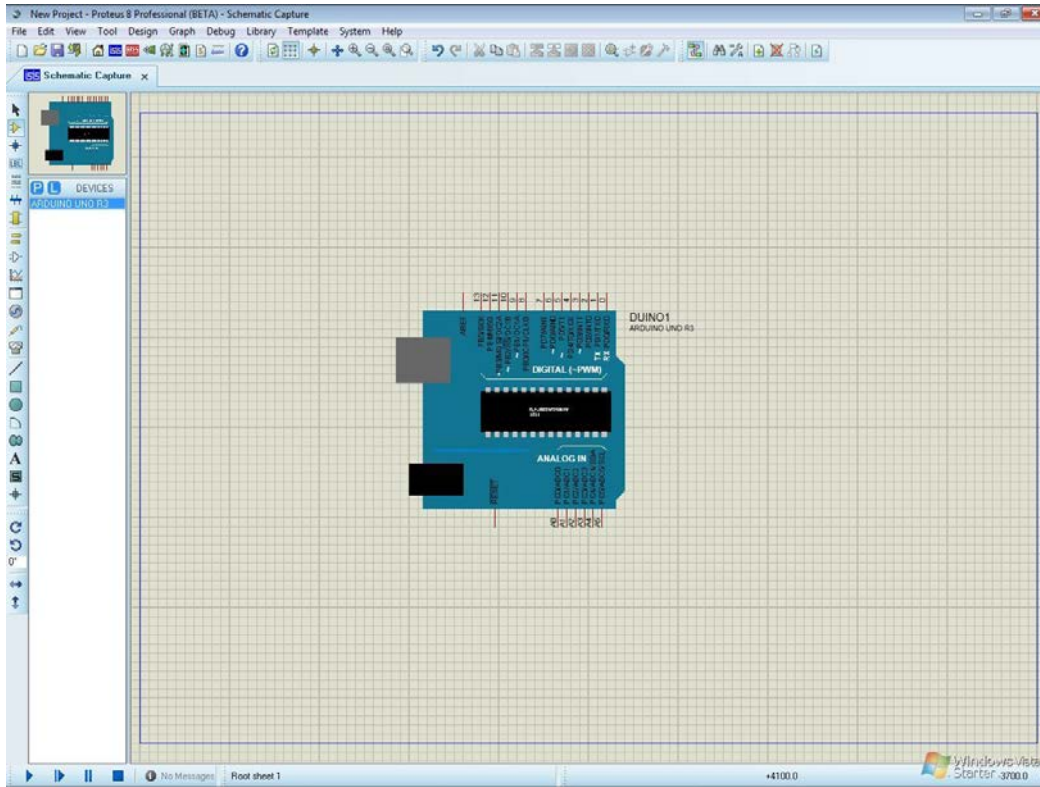


Figure 5.3 Proteus Screenshot

## 5.4 – AT Commands

There are special sets of commands to control the GSM module from the computer or controller. These commands are called AT commands. AT commands are used to control modems. AT is the abbreviation for Attention. Every command line starts with "AT" or "at". That's why modem commands are called AT commands. These commands come from Hayes commands that were used by the Hayes smart modems. The Hayes commands started with AT to indicate the attention from the modem. The dial up and wireless modems (devices that involve machine to machine communication) need AT commands to interact with a computer. These include the Hayes command set as a subset, along with other extended AT commands. For this system, module SIM900a is being used. It has its certain group of commands. As sending SMS between different cells of the system is a prime need, the SMS commands are used.

Point to be noted is that the starting "AT" is the prefix that informs the modem about the start of a command line. It is not part of the AT command name.

Mobile phone manufacturers usually do not implement all AT commands, command parameters and parameter values in their mobile phones. Also, the behaviour of the implemented AT

commands may be different from that defined in the standard. In general, GSM/GPRS modems designed for wireless applications have better support of AT commands than ordinary mobile phones.

There are two types of AT commands: basic commands and extended commands.

*Basic commands* are AT commands that do not start with "+". For example, D (Dial), A (Answer), H (Hook control) and O (Return to online data state) are basic commands.

*Extended commands* are AT commands that start with "+". All GSM AT commands are extended commands. For example, +CMGS (Send SMS message), +CMSS (Send SMS message from storage), +CMGL (List SMS messages) and +CMGR (Read SMS messages) are extended commands.

The GSM engines are referred to as following term:

- 1) ME (Mobile Equipment);
- 2) MS (Mobile Station);
- 3) TA (Terminal Adapter);
- 4) DCE (Data Communication Equipment) or facsimile DCE (FAX modem, FAX board);

In application, controlling device controls the GSM engine by sending AT Command via its serial interface. The controlling device at the other end of the serial line is referred to as following term:

- 1) TE (Terminal Equipment);
- 2) DTE (Data Terminal Equipment) or plainly "the application" which is running on an embedded system;

#### **5.4.1 – AT Command syntax**

The "AT" or "at" prefix must be set at the beginning of each Command line. To terminate a Command line enter <CR>.

Commands are usually followed by a response that includes. "<CR><LF><response><CR><LF>" Throughout this document, only the responses are presented, <CR><LF> are omitted intentionally.

A HEX string such as "00 49 49 49 49 FF FFFFFFFF" will be sent out through serial port at the baud rate of 115200 immediately after SIM900 is powered on. The string shall be ignored since it is used for synchronization with PC tool. Only enter AT Command through serial port after SIM900 is powered on and Unsolicited Result Code "RDY" is received from serial port. If auto-bauding is enabled, the Unsolicited Result Codes "RDY" and so on are not indicated when you start up the

ME, and the "AT" prefix, not "at" prefix must be set at the beginning of each command line. All these AT commands can be split into three categories syntactically: "basic", "S parameter", and "extended". These are as follows:

**Basic syntax:** These AT commands have the format of "AT<x><n>", or "AT&<x><n>", where "<x>" is the Command, and "<n>" is/are the argument(s) for that Command. An example of this is "ATE<n>", which tells the DCE whether received characters should be echoed back to the DTE according to the value of "<n>". "<n>" is optional and a default will be used if missing.

**S Parameter syntax:** These AT commands have the format of "ATS<n>=<m>", where "<n>" is the index of the S register to set, and "<m>" is the value to assign to it. "<m>" is optional; if it is missing, then a default value is assigned.

**Extended Syntax:** These commands can operate in several modes, as in the following table:

Test Command	AT+<x>=?	The mobile equipment returns the list of parameters and value ranges set with the corresponding Write Command or by internal processes.
Read Command	AT+<x>?	This command returns the currently set value of the parameter or parameters.
Write Command	AT+<x>=<...>	This command sets the user-definable parameter values.
Execution Command	AT+<x>	The execution command reads non-variable parameters affected by internal processes in the GSM engine.

*Table Error! No text of specified style in document..3. Types of AT commands and responses*

### Combining AT commands on the same Command line

You can enter several AT commands on the same line. In this case, you do not need to type the "AT" or "at" prefix before every command. Instead, you only need type "AT" or "at" the beginning of the command line. Please note to use a semicolon as the command delimiter after an extended

command; in basic syntax or S parameter syntax, the semicolon need not enter, for example:  
ATE1Q0S0=1S3=13V1X4+IFC=0,0;+IPR=115200; &W.

The Command line buffer can accept a maximum of 556 characters. If the characters entered exceeded this number then none of the Command will executed and TA will return "**ERROR**".

### **Entering successive AT commands on separate lines**

When you need to enter a series of AT commands on separate lines, please Note that you need to wait the final response (for example OK, CME error, CMS error) of last AT Command you entered before you enter the next AT Command.

### **Supported character sets**

The SIM900 AT Command interface defaults to the **IRA** character set. The SIM900 supports the following character sets:

- GSM format
- UCS2
- HEX
- IRA
- PCCP
- PCDN
- 8859-1

The character set can be set and interrogated using the "**AT+CSCS**" Command (GSM 07.07). The character set is defined in GSM specification 07.05.

The character set affects transmission and reception of SMS and SMS Cell Broadcast messages, the entry and display of phone book entries text field and SIM Application Toolkit alpha strings.

### **Flow control**

Flow control is very important for correct communication between the GSM engine and DTE. For in the case such as a data or fax call, the sending device is transferring data faster than the receiving side is ready to accept. When the receiving buffer reaches its capacity, the receiving device should be capable to cause the sending device to pause until it catches up.

There are basically two approaches to achieve data flow control: software flow control and hardware flow control. SIM900 support both two kinds of flow control. In Multiplex mode, it is recommended to use the hardware flow control.

#### 5.4.2 – Set of AT Commands used to program this system

The AT Command set which is used to program this system is shown in the table below:

Demonstration	Syntax	Expect Result
Check if GSM Module is connected	AT	OK
Set SMS system into text mode, as opposed to PDU mode.	AT+CMGF=1	OK
Send an SMS to Admin.	AT+CMGS="+880167066 9566" >This is a test <Ctrl+Z>	OK +CMGS:34 OK
Unsolicited notification of the SMS arriving	+CMTI: "SM",1	
Read SMS message that has just arrived.  Note: the number should be the same as that given in the +CMTI notification.	AT+CMGR=1	+CMGR: "REC UNREAD","+880167066 9566","","02/01/30,20:40: 31+00"  This is a test  OK
List all SMS messages.  Note: "ALL" must be in uppercase.	AT+CMGL="ALL"	+CMGL: 1,"REC READ","+880167066956 6","","02/01/30,20:40:31 +00"  This is a test



		+CMGL: 2,"REC UNREAD",",","+8801670 669566",",", "02/01/30,20: 45:12+00"
Delete an SMS message.	AT+CMGD=1	OK
Know the Power status of the GPS	AT+CGNSPWR=?	Response  +CGNSPWR: (list of supported <mode>s )
Turns GPS ON or OFF	AT+CGNSPWR=<mode>	Response  GNSS POWER CONTROL ON/OFF  OK  ERROR  Parameters  <mode> 0 Turn off GNSS power supply  1 Turn on GNSS power supply

Obtain GPS Information	<p>Execution Command</p> <p>AT+CGNSINF</p>	<p>Response</p> <p>+CGNSINF: &lt;GNSS run status&gt;,&lt;Fix status&gt;,&lt;UTC date &amp; Time&gt;,&lt;Latitude&gt;,&lt;Longitude&gt;,&lt;Smart Machine Smart Decision&gt;</p> <p>SIM800</p> <p>Seires_GNSS_Application Note_V1.00 1 1 2015-04-10</p> <p>&lt;MSL Altitude&gt;,&lt;Speed Over Ground&gt;,&lt;Course Over Ground&gt;,&lt;Fix Mode&gt;,&lt;Reserved1&gt;,&lt;HDOP&gt;,&lt;PDOP&gt;,&lt;VDOP&gt;,&lt;Reserved2&gt;,&lt;GNSS Satellites in View&gt;,&lt;GNSS Satellites Used&gt;,&lt;GLONASS Satellites Used&gt;,&lt;Reserved3&gt;,&lt;C/N0 max&gt;,&lt;HPA&gt;,&lt;VPA&gt;</p>
------------------------	--	---

		OK
<i>Others (See Appendix Section)</i>		

*Table Error! No text of specified style in document..4. Set of AT Commands used in the programming of this system and their functions.*

## 5.5 – AT Command Tester Tool

AT Command Tester is a serial monitor based software which analyses the AT commands being sent and received from a device as well as providing a look into the underlying process that takes place during those interchanges, which gave us a proper understanding of what is happening inside in response to our input commands. The tool was used to test the GSM module before implementing in the main circuit. The user friendly interface allows developers to:

- Configure and connect to modem ports
- Send single or batch of AT commands
- Perform modem diagnostics
- Establish 3G or GPRS call
- Collect and save modem logs

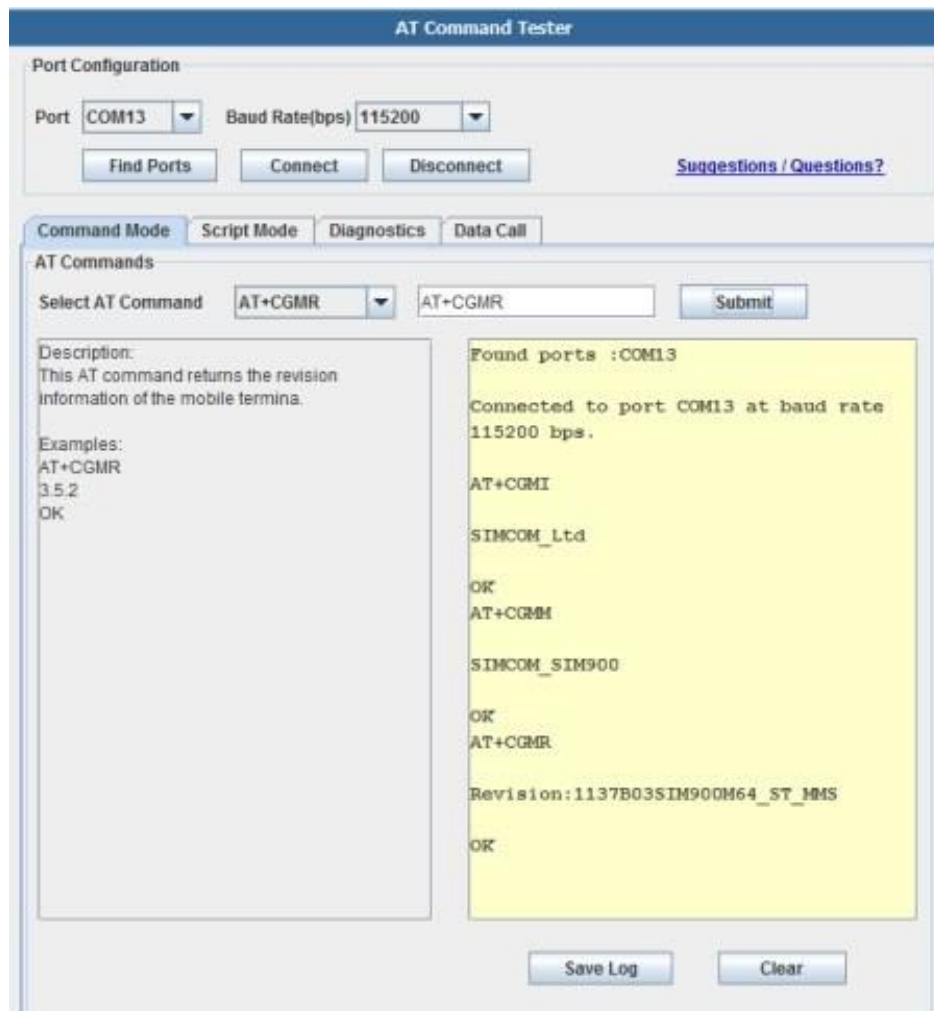


Figure Error! No text of specified style in document..4 AT Command Tester User Interface

### 5.5.1 – Port Configuration

AT Command Tester uses Java-based serial drivers to interface to the modem. The ‘Find Ports’ button will automatically find all ports available in the system. The user can connect the appropriate modem port with the desired port speed.

### 5.5.2 – Command mode

After connecting successfully to the modem, users can send single AT commands under ‘Command Mode’ tab. The drop down provides a list of AT commands with description and examples. The users can modify or enter their own AT command in the text box.

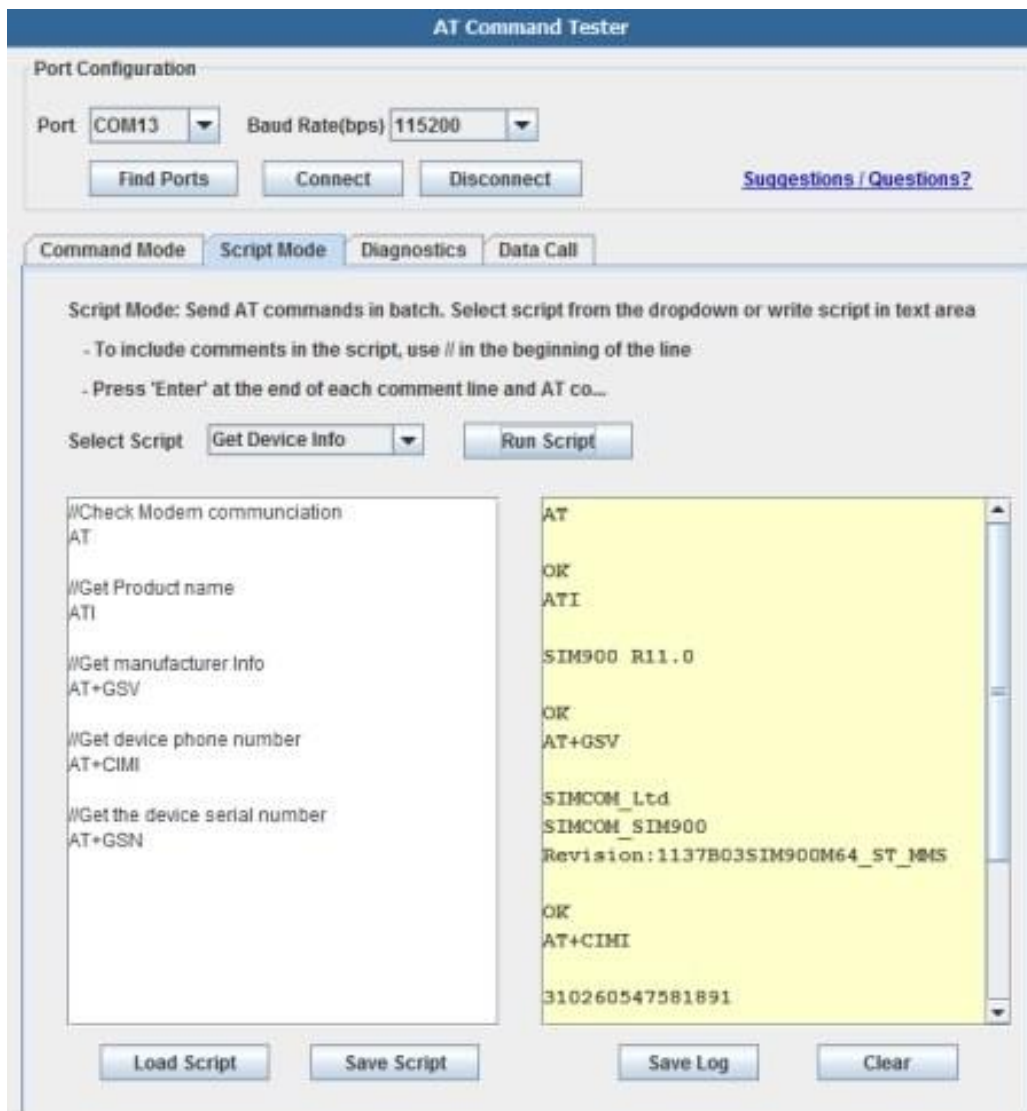


Figure Error! No text of specified style in document..5 AT Command Tester Command Mode

### 5.5.3 – Script Mode

Users can send batch of AT commands under the ‘Script Mode’ tab. They can also save and load the script from the local machine. Users can develop their own scripts for specific set of tasks such as call setup, send SMS, HTTP access etc. Users can also include descriptive comments in their script.

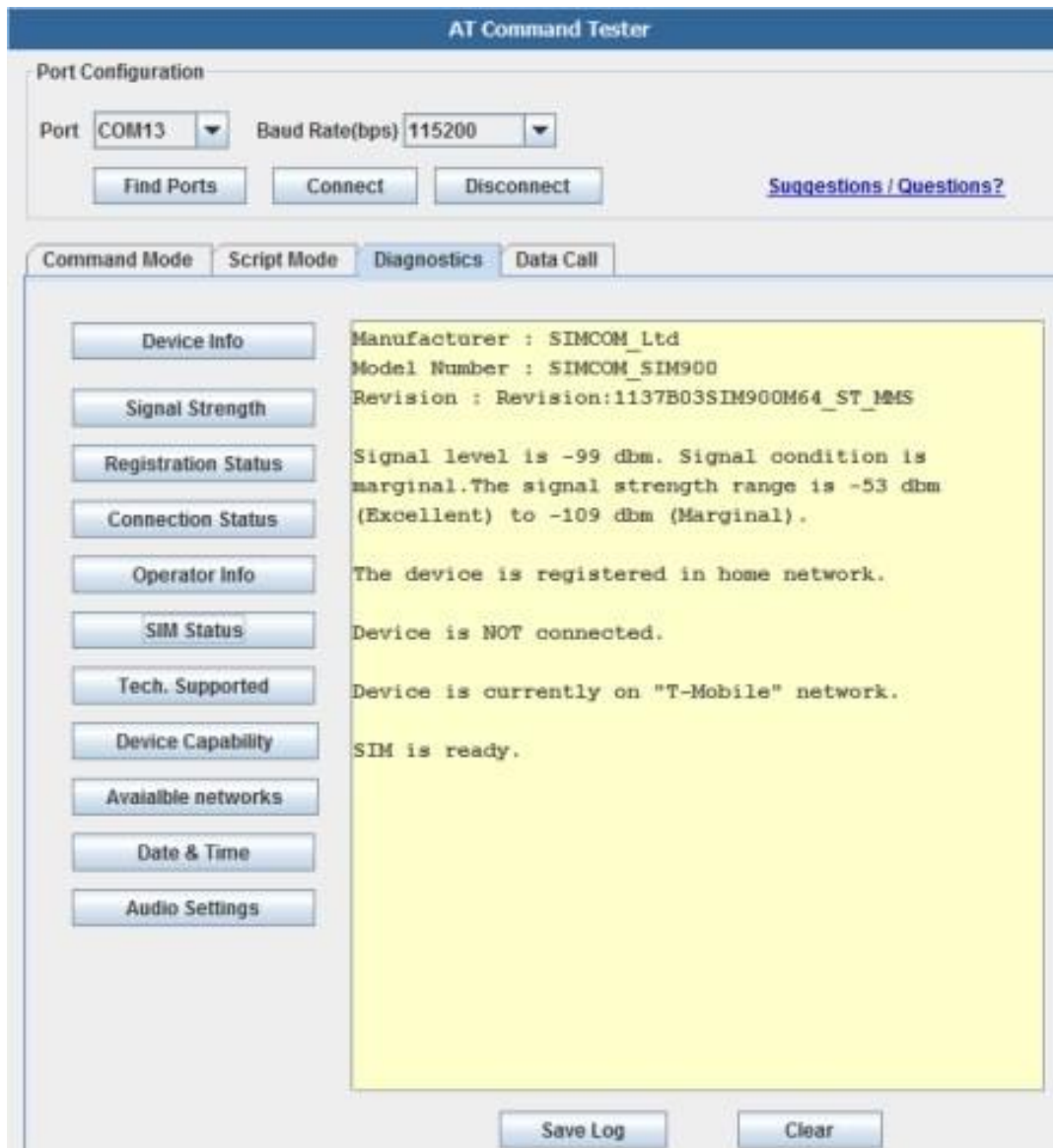


Figure Error! No text of specified style in document..6 AT Command Tester Script Mode

### 5.5.4 – Diagnostics

Users can perform basic troubleshooting of the modem under the ‘Diagnostics’ tab. Here the AT Command Tester tool sends the required AT commands and provides descriptive output about the state of the modem.

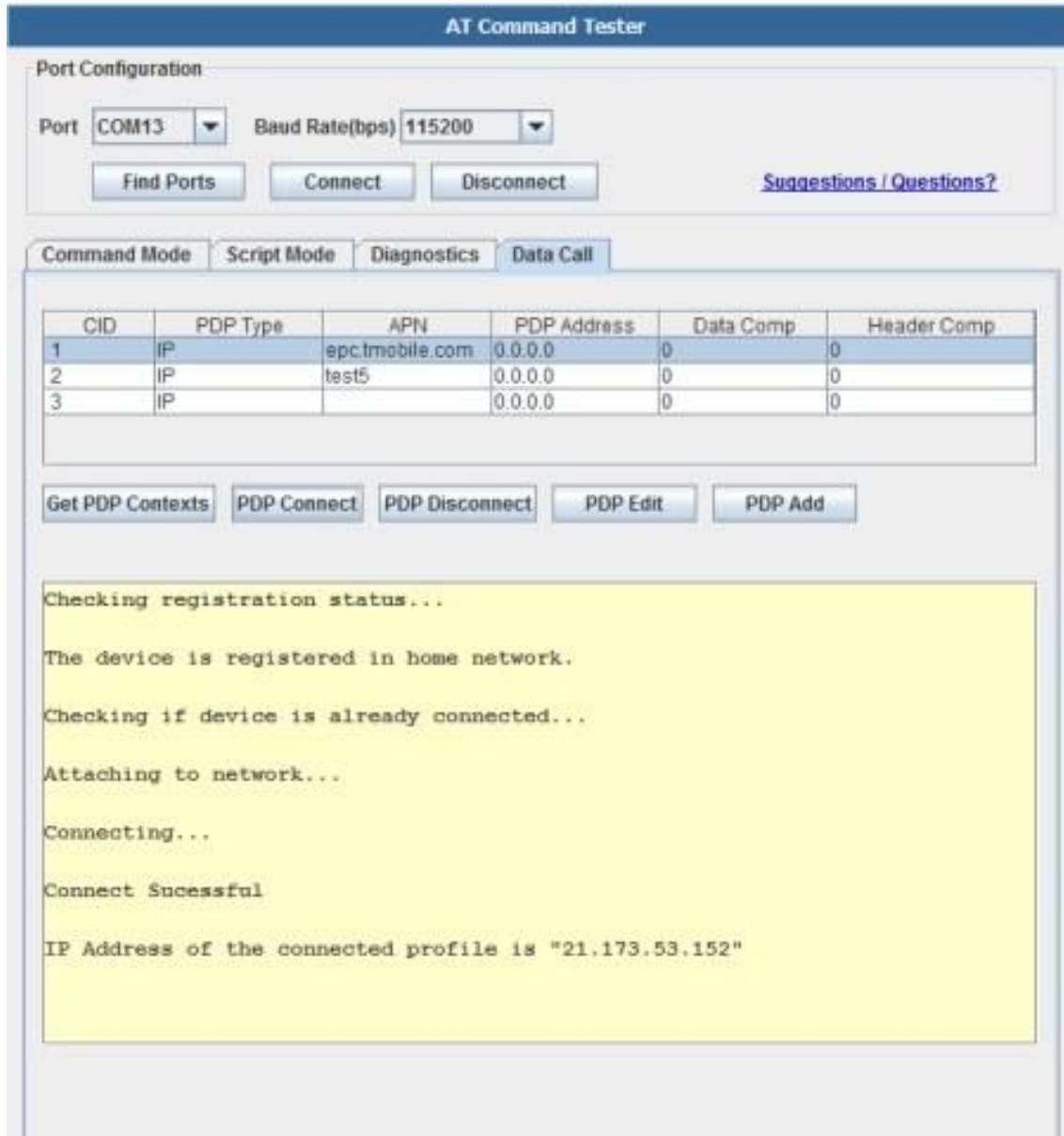


Figure 5.7 AT Command Tester Diagnostics Tab

### 5.5.5 – Data Call

The ‘Data Call’ tab provides the interfaces to setup data call with the GSM network. ‘Get PDP Contexts’ button will list all the PDP context profiles stored on the SIM. Users can also add or

update new PDP context profile. Users can then connect to the selected profile. AT Command Tester will first check whether the device is registered on the network. If so, it will attach and connect to the network with the selected PDP context credentials. More feature additions such as voice call, SMS, Phone book functions, HTTP, FTP, and TCP/IP are planned for AT Command Tester tool.

## **5.6 – HyperTerminal 7**

HyperTerminal is a windows terminal emulation program capable of connecting to systems through TCP/IP Networks, Dial-Up Modems, and COM ports.

It is a program that you can use to connect to other computers, Telnet sites, bulletin board systems (BBSs), online services, and host computers. HyperTerminal connections are made using a modem, a null modem cable (used to emulate modem communication), or an Ethernet connection.

HyperTerminal has capabilities beyond making connections to other computers. It can, for example, transfer large files from a computer onto your portable computer using a serial port rather than requiring you to set up your portable computer on a network. It can help debug source code from a remote terminal. It can also communicate with many older, character-based computers.

HyperTerminal records the messages passed to and from the computer or service on the other end of your connection. It can therefore serve as a valuable troubleshooting tool when setting up and using your modem. To make sure that your modem is connected properly or to view your modem's settings, you can send commands through HyperTerminal and check the results. HyperTerminal also has scroll functionality that enables you to view received text that has scrolled off the screen.

### **Some uses of HyperTerminal:**

- Use a TCP/IP network to connect to systems on the Internet or your network using Telnet or Secure Shell (SSH)
- Use a Dial-Up modem to dial into modem based systems
- Talk directly to many different types of devices using RS232 serial COM ports.

HyperTerminal was used to test the GSM modules through the COM ports using AT Commands.



## 5.7 – Adobe Dreamweaver

Adobe Dreamweaver is a proprietary web development tool developed by Adobe Systems. Dreamweaver was created by Macromedia in 1997, [28] and was maintained by them until Macromedia was acquired by Adobe Systems in 2005. [29]

Adobe Dreamweaver is available for macOS and for Windows.

Adobe Dreamweaver CC is a web design and development application that combines a visual design surface known as Live View and a code editor with standard features such as syntax highlighting, code completion, and code collapsing as well as more sophisticated features such as real-time syntax checking and code introspection for generating code hints to assist the user in writing code. Combined with an array of site management tools, Dreamweaver lets its users design, code and manage websites as well as mobile content. Dreamweaver is positioned as a versatile web design and development tool that enables visualization of web content while coding.

Dreamweaver, like other HTML editors, edits files locally then uploads them to the remote web server using FTP, SFTP, or WebDAV. Dreamweaver CS4 now supports the Subversion (SVN) version control system.

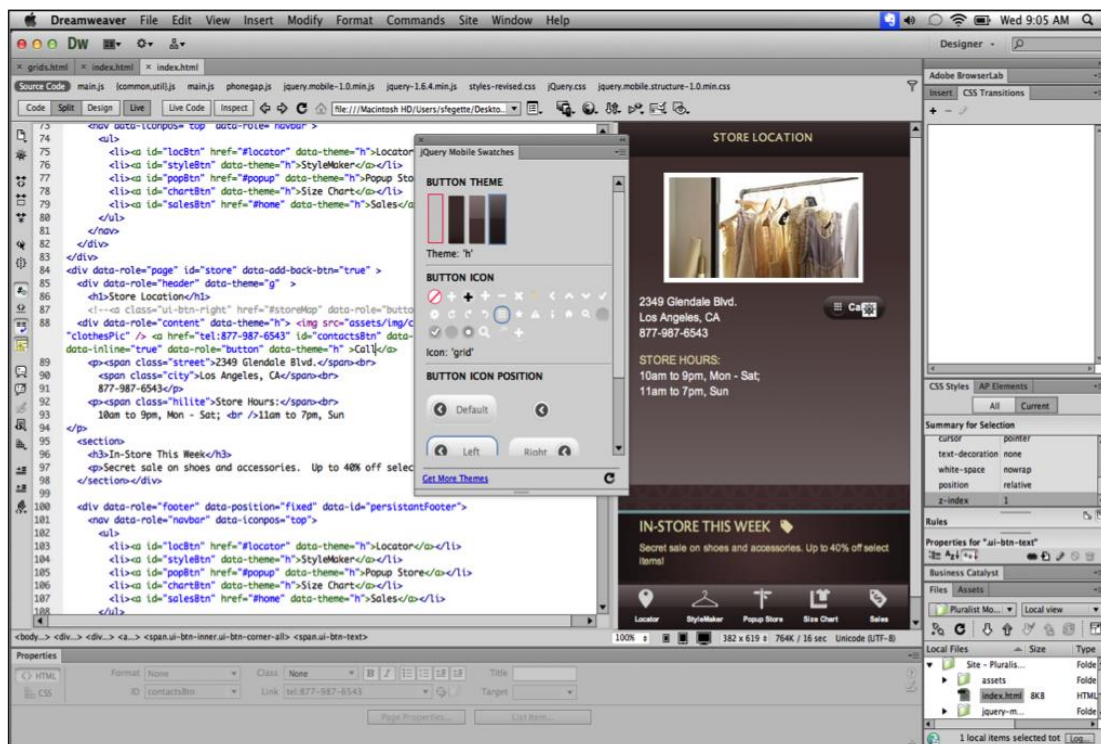


Figure 5.8 Adobe Dreamweaver Screenshot

Since version 5, Dreamweaver supports syntax highlighting for the following languages out of the box:

- ActionScript
- Active Server Pages (ASP).
- C#
- Cascading Style Sheets (CSS)
- ColdFusion
- EDML
- Extensible HyperText Markup Language (XHTML)
- Extensible Markup Language (XML)
- Extensible Stylesheet Language Transformations (XSLT)
- HyperText Markup Language (HTML)
- Java
- JavaScript
- PHP
- Visual Basic (VB)
- Visual Basic Script Edition (VBScript)
- Wireless Markup Language (WML)

Support for ASP.NET and JavaServer Pages was dropped in version CS5. [30]

Users can add their own language syntax highlighting. In addition, code completion is available for many of these languages.

This was used in designing the website for the Central Control System. HTML, Java Script and PHP were used.

## **5.8 – PHPStorm**

JetBrains PhpStorm is a commercial, cross-platform IDE for PHP built on JetBrains' IntelliJ IDEA platform.

PhpStorm provides an editor for PHP, HTML and JavaScript with on-the-fly code analysis, error prevention and automated refactorings for PHP and JavaScript code. PhpStorm's code completion supports PHP 5.3, 5.4, 5.5, 5.6 & 7.0[31] (modern and legacy projects), including generators, coroutines, the finally keyword, list in foreach, namespaces, closures, traits and short array syntax. It includes a full-fledged SQL editor with editable query results. [32][33]

PhpStorm is built on IntelliJ IDEA, which is written in Java. Users can extend the IDE by installing plugins created for the IntelliJ Platform or write their own plugins.

All features available in WebStorm are included in PhpStorm, [34] which adds support for PHP and databases. [35] WebStorm ships with pre-installed JavaScript plugins (such as for Node.js), which are available for PhpStorm as well at no cost. [36]

### 5.8.1 – Key features

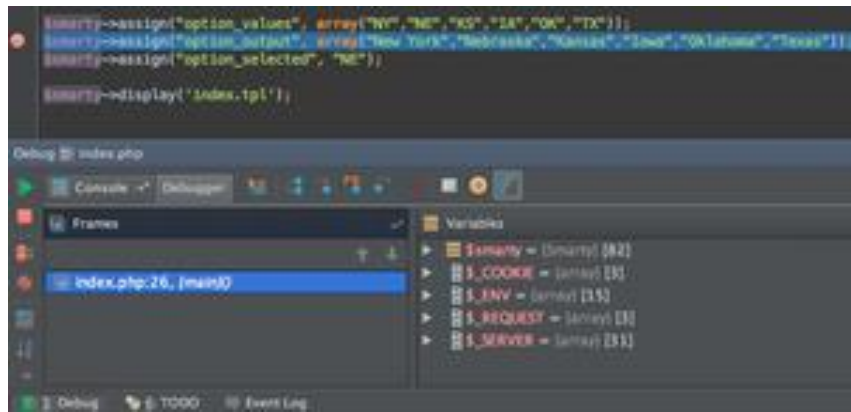


Figure 5.9 Zero-configuration web application debugging with Xdebug in PhpStorm. Darcula color scheme

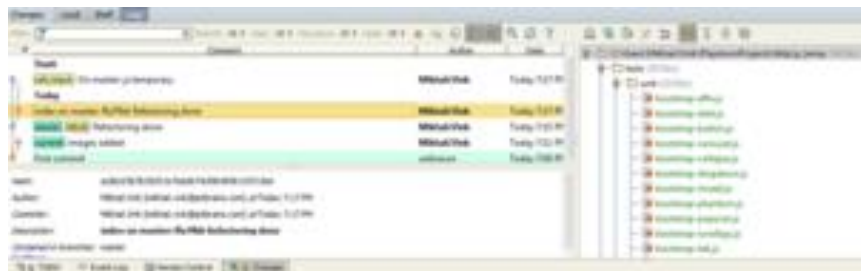


Figure 5.10 Version Control Systems Integration

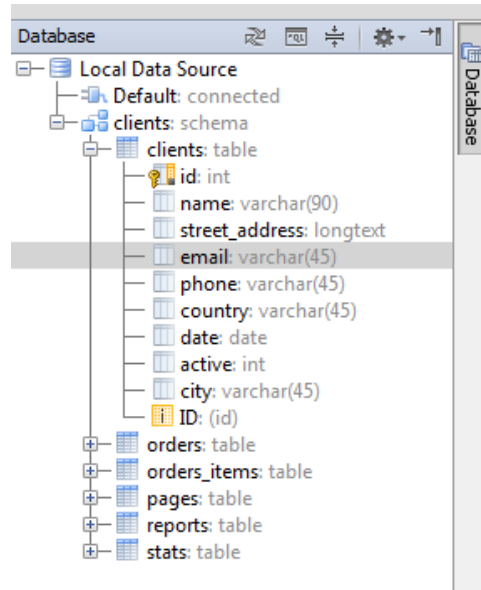


Figure 5.11 Viewing Data Source Structure and other SQL and Databases related features in PhpStorm

## PHP editor

- PhpStorm provides a rich code editor [37] for PHP with syntax highlighting, extended code formatting configuration, on-the-fly error checking, and code completion.
- PHP 5.3, 5.4, 5.5, 5.6, 7.0, 7.1 support, including generators, coroutines, the finally keyword, list in foreach, using empty() on the result of function calls and other expressions, traits, closures, class member access on instantiation, short array syntax, array dereferencing on function call, binary literals, expressions in static calls, it supports return types and scalar type hints and constant visibility, etc. It can be used for both modern and legacy PHP-based projects.
- Code autocompletion finalizes classes, methods, variable names, and PHP keywords, plus commonly used names for fields and variables depending on their type.
- Coding style Support (PSR1/PSR2, Drupal, [38] Symfony2, Zend).
- PHPDoc support. The IDE provides code completion suggestions based on @property, @method and @var annotations.
- Duplicated Code Detector.
- PHP Code Sniffer (phpcs) that checks for code smells on the fly.

- Refactorings (Rename, Introduce Variable, Introduce Constant, Introduce Field, Inline Variable, Move Static Member, Extract Interface).
- Smarty and Twig templates editing (Syntax errors highlighting; Smarty functions and attributes completion; automatic insertion of paired braces, quotes and closing tags; and more).
- MVC view for Symfony2, Symfony3, and Yii frameworks.
- PHAR support.

### **Development environment**

- SQL and databases support (live database schema refactoring, generation of schema migration scripts, export query result to file or clipboard, editing of stored procedures, etc.).
- Remote deployment over FTP, SFTP, FTPS etc. with automatic synchronization.
- Version control systems integration (Git (including specific GitHub features), Subversion, Mercurial, Perforce, CVS, TFS) allowing you to perform actions (commit, merge, diff, etc.) right from the IDE.
- Local History (tracks any changes in the code locally).
- PHP UML (UML class diagrams for PHP code with refactorings invoked right from the diagram).
- Phing support (autocompletion, checks standard tags, properties, target names, path attribute values in build files).
- Issue tracker integration.
- Support for Vagrant, SSH console & remote tools
- Google App Engine for PHP Support

## **Debugging and testing**

- Easy-to-configure visual debugger (Xdebug, Zend Debugger)<sup>[10]</sup> for inspecting context-relevant local variables and user-defined watches, including arrays and complex objects, and editing values on the fly.
- Scripts can be profiled right from PhpStorm with either XDebug or Zend Debugger. An aggregated report is available, and the user can jump from the execution statistics directly to the function in PHP code.
- PHPUnit tests can be developed in PhpStorm and run instantly from a directory, file or class using the context menu options) with code coverage.

## **JavaScript, CSS and HTML features**

- Code completion for JavaScript, HTML and CSS (for tags, keywords, labels, variables, parameters and functions).
- HTML5 support.
- Live Edit: changes in the code can be immediately viewed in the browser without reloading the page.
- CSS/SASS/SCSS/LESS support (code completion, error highlighting, validation, etc.).
- Zen Coding.
- Code navigation and usages search (Go to declaration/symbol, Find Usages).
- ECMAScript Harmony Support.
- JavaScript refactoring (Rename, Extract Variable/Function, Inline Variable/Function, Move/Copy, Safe delete, Extract embedded script into file).
- JavaScript debugger and unit testing.

## **IntelliJ IDEA PHP support**

The Ultimate Edition of JetBrains polyglot IDE IntelliJ IDEA supports the same functionality as PhpStorm by plugins.

## 5.9 – Apache Server

The Apache HTTP Server, colloquially called Apache, is the world's most used web server software. Originally based on the NCSA HTTPd server, development of Apache began in early 1995 after work on the NCSA code stalled. Apache played a key role in the initial growth of the World Wide Web, [40] quickly overtaking NCSA HTTPd as the dominant HTTP server, and has remained most popular since April 1996. In 2009, it became the first web server software to serve more than 100 million websites. [41]



*Figure 5.12 Apache Logo*

Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes. Some common language interfaces support Perl, Python, Tcl, and PHP. Popular authentication modules include `mod_access`, `mod_auth`, `mod_digest`, and `mod_auth_digest`, the successor to `mod_digest`. A sample of other features include Secure Sockets Layer and Transport Layer Security support (`mod_ssl`), a proxy module (`mod_proxy`), a URL rewriting module (`mod_rewrite`), custom log files (`mod_log_config`), and filtering support (`mod_include` and `mod_ext_filter`).

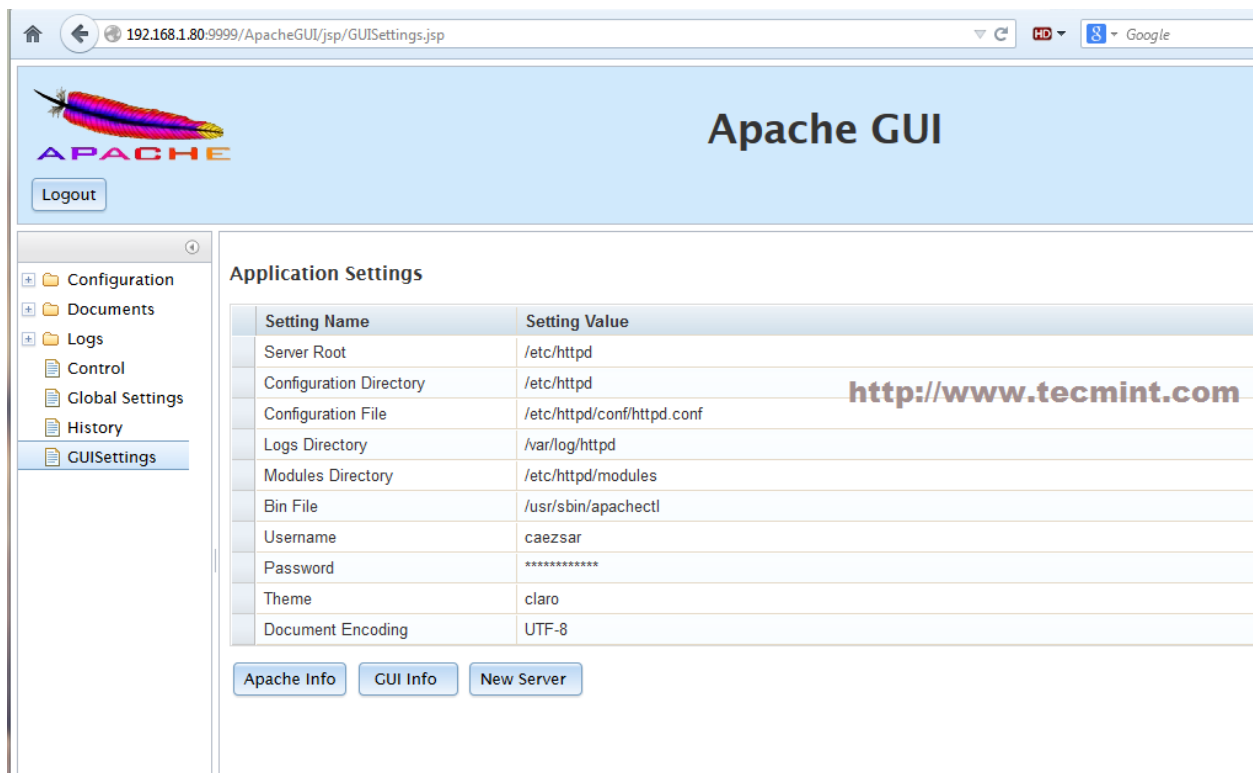
Popular compression methods on Apache include the external extension module, `mod_gzip`, implemented to help with reduction of the size (weight) of Web pages served over HTTP.

ModSecurity is an open source intrusion detection and prevention engine for Web applications. Apache logs can be analyzed through a Web browser using free scripts, such as AWStats/W3Perl or Visitors.

Virtual hosting allows one Apache installation to serve many different Web sites. For example, one machine with one Apache installation could simultaneously serve `www.example.com`, `www.example.org`, `test47.test-server.example.edu`, etc.

Apache features configurable error messages, DBMS-based authentication databases, and content negotiation. It is also supported by several graphical user interfaces (GUIs).

It supports password authentication and digital certificate authentication. Because the source code is freely available, anyone can adapt the server for specific needs, and there is a large public library of Apache add-ons. [42]



The screenshot shows a web browser window with the URL `192.168.1.80:9999/ApacheGUI/jsp/GUISettings.jsp`. The page title is "Apache GUI" and features the Apache logo and a "Logout" button. A left sidebar contains a navigation menu with items: Configuration, Documents, Logs, Control, Global Settings, History, and GUISettings (which is selected). The main content area is titled "Application Settings" and contains a table with the following data:

Setting Name	Setting Value
Server Root	/etc/httpd
Configuration Directory	/etc/httpd
Configuration File	/etc/httpd/conf/httpd.conf
Logs Directory	/var/log/httpd
Modules Directory	/etc/httpd/modules
Bin File	/usr/sbin/apachectl
Username	caezsar
Password	*****
Theme	claro
Document Encoding	UTF-8

Below the table are three buttons: "Apache Info", "GUI Info", and "New Server". A watermark "http://www.tecmint.com" is visible on the right side of the settings table.

Figure 5.13 Apache GUI



### 5.9.1 – HTTP server and proxy features

- Loadable Dynamic Modules
- Multiple Request Processing modes (MPMs) including Event-based/Async, Threaded and Prefork.
- Highly scalable (easily handles more than 10,000 simultaneous connections)
- Handling of static files, index files, auto-indexing and content negotiation
- .htaccess support
- Reverse proxy with caching
- Load balancing with in-band health checks
- Multiple load balancing mechanisms
- Fault tolerance and Failover with automatic recovery
- WebSocket, FastCGI, SCGI, AJP and uWSGI support with caching
- Dynamic configuration
- TLS/SSL with SNI and OCSP stapling support, via OpenSSL.
- Name- and IP address-based virtual servers
- IPv6-compatible
- HTTP/2 protocol support
- Fine-grained authentication and authorization access control
- gzip compression and decompression
- URL rewriting
- Headers and content rewriting
- Custom logging with rotation
- Concurrent connection limiting
- Request processing rate limiting
- Bandwidth throttling
- Server Side Includes
- IP address-based geolocation
- User and Session tracking
- WebDAV
- Embedded Perl, PHP and Lua scripting

- CGI support
- public\_html per-user web-pages
- Generic expression parser
- Real-time status views
- XML support

## 5.10 – Adobe Photoshop CS6

Adobe Photoshop is a raster graphics editor developed and published by Adobe Systems for macOS and Windows. It can edit and compose raster images in multiple layers and supports masks, alpha compositing and several color models including RGB, CMYK, CIELAB, spot color and duotone. Photoshop has vast support for graphic file formats but also uses its own PSD and PSB file formats which support all the aforementioned features. In addition to raster graphics, it has limited abilities to edit or render text, vector graphics (especially through clipping path), 3D graphics and video. Photoshop's feature set can be expanded by Photoshop plug-ins, programs developed and distributed independently of Photoshop that can run inside it and offer new or enhanced features.

In this project Adobe Photoshop was used to invert colors of the masks of the PCB designs that were created using Fritzing.

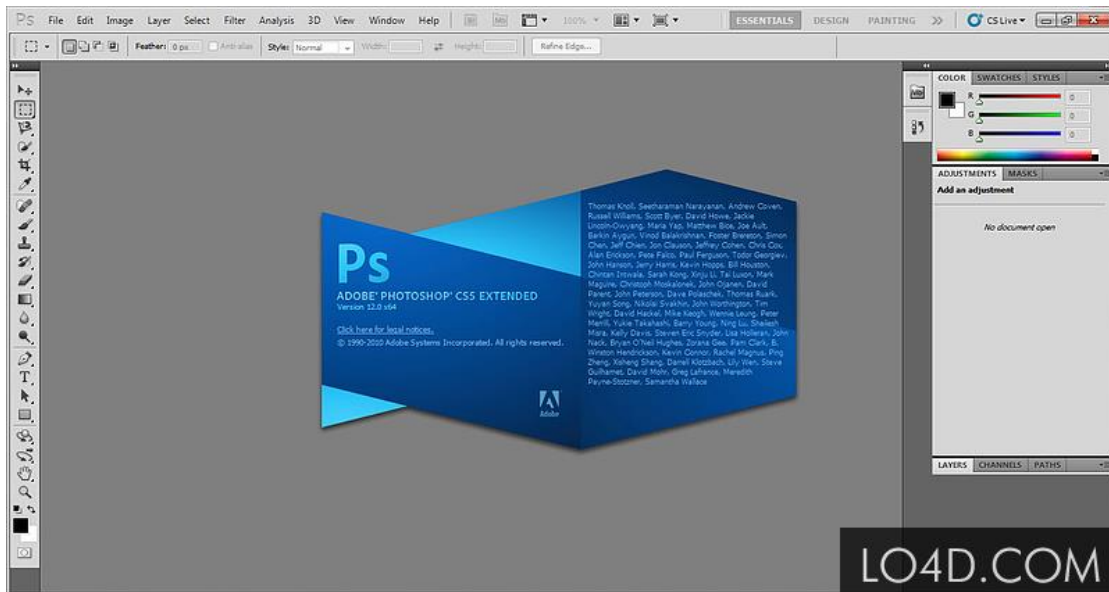


Figure 5.14 Adobe Photoshop Screenshot

# CHAPTER 6 – THE TRAIN SYSTEM

The Train System is meant to be installed inside the train’s cockpit, on the dashboard so that it has easy access and is clearly visible to the drivers. With most of the operations being automated, the user interface are made very user friendly so that drivers have a relaxed and fast user experience.

## 6.1 – Hardware Configuration

### 6.1.1 – Dual Core Technology

The Train System is based on dual core microcontroller architecture which uses two ATMEGA2560 microcontrollers, where one acts as a master and another as a slave. Since these microcontrollers are of single core each having single threads, they are not allowed to perform multi-tasking or any high speed calculations. We have tested that in a single core system the GSM modules require about 7 to 10 seconds time to send a SMS message and 25-30 seconds time to send an HTTP request. The system also couldn’t perform these two tasks at the same time. By using the dual core system we have minimized those time delays and also perform both the tasks together simultaneously. With each microcontroller the GSM+GPS+GPRS modules are attached each performing separate tasks.

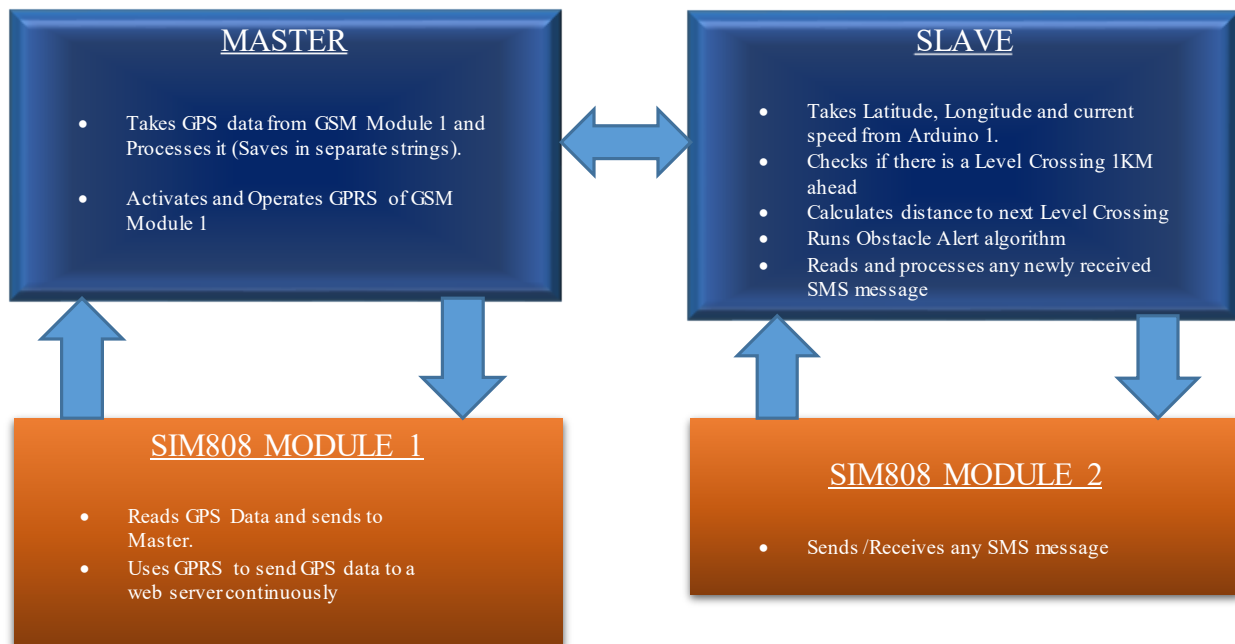


Figure 6.1 Dual-Core System Architecture

### 6.1.2 – Master Core

The GSM module attached with the master is responsible for obtaining GPS data from the satellites using the 1575.42 MHz GPS antenna attached with it and uploading the data to the web server. This GSM module is connected using the TTL pins and uses RS232 mode of communication. The RX and TX pins of the GSM module are connected to the master microcontroller’s digital pins 11 and 10 respectively.

The ‘Start/ Stop Journey’ with its LED and the ‘Menu’ Buttons are also connected to the master Arduino’s two of the interrupt pins so as to function any time when pressed and the LED is connected to a digital pin. All LEDs are attached with a current limiting resistor in series to prevent burn out

LCD 1 and LCD 3 are connected to the Master so that the variables or calculations done by the master can directly be shown on the displays. Dedicated potentiometers to control the brightness of the LCDs are connected to each of the LCDs

The Keypads are programmed to function only after the menu button is pressed, therefore they are also connected to the master Arduino.

The Master Arduino is connected to the Slave Arduino through the SDA and SCL pins.

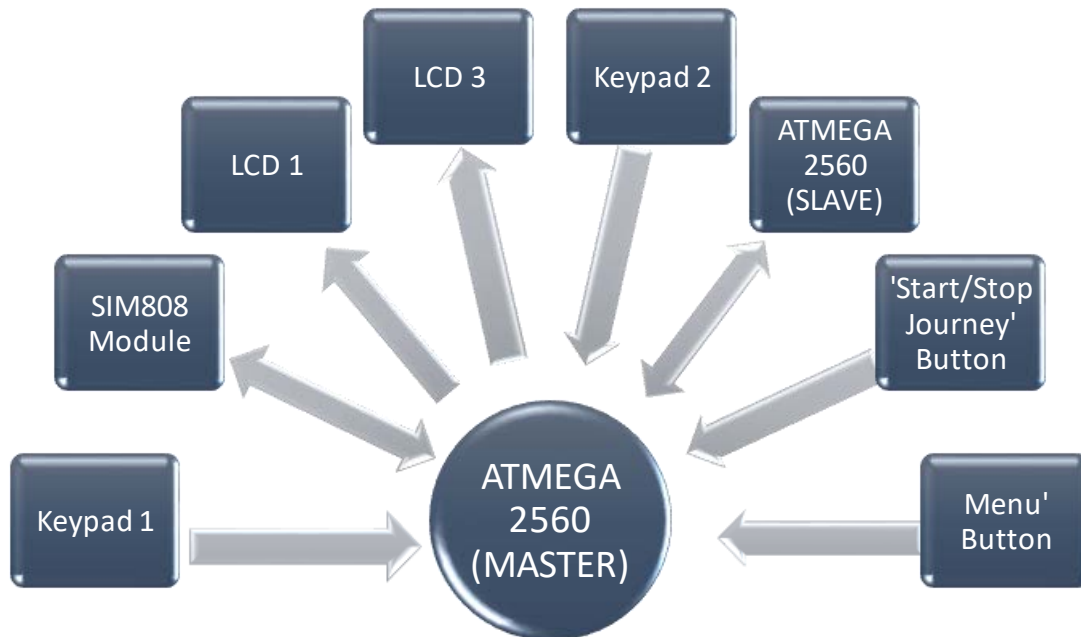


Figure 6.2 Block Diagram of the Master Core

### 6.1.3 – Slave Core

The GSM module connected to the Slave core is only responsible for sending and receiving SMS to communicate with the other sub-systems. Similar to the master core, the GSM module with the slave is connected using the TTL pins and uses RS232 mode of communication. The RX and TX pins of the GSM module are connected to the slave microcontroller's digital pins 11 and 10 respectively.

LCD 2 and LCD 4 are connected to the Slave so that the variables or calculations done by the Slave can directly be shown on its displays. Dedicated potentiometers to control the brightness of the LCDs are connected to each of the LCDs.

The 'Emergency Alarm' switch is connected to the one of the Slave's interrupt pins so that it can act whenever the switch is pressed. Consequently, the 'Emergency Alarm' LEDs were formed a bundle and connected to one of the digital pins of the slave.

The 'New Message' LEDs and the 'Signal' LEDs were also connected to five digital pins of the Slave since it is in control of the SMS messages and notifying whether to go slow, go or stop.

The Slave is connected to the Master through the SDA and SCL pins of the microcontrollers.

All LEDs are attached with a current limiting resistor in series to prevent burn out.

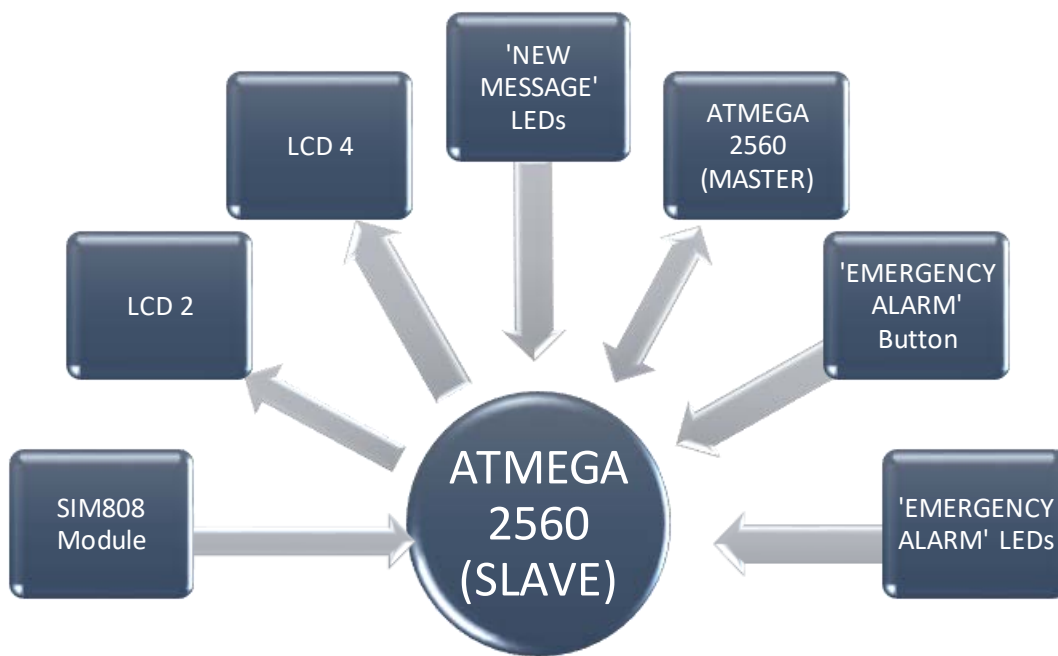


Figure 6.3 Block Diagram of the Slave Core

### 6.1.4 – Power Supply

The Train System is fed by a DC power adapter of 12V, 3A through a 5mm DC jack into a DC socket. However, a secondary power option has also been kept by means of a 6V, 4500mAh DC Lead-Acid Battery.

The microcontrollers require 5V DC power to operate whereas the SIM808 modules has a wide range of input voltages ranging from 6V DC to 30V DC. The SIM808 module consumes a current of 1A to 1.5A when it is functional, for example to send or receive SMS messages or to connect to the web server using HTTP. Otherwise, the SIM808 modules use a very low power when in sleep mode (*See Appendix Section*). All other components operate in 5V DC power.

Since the components require two types of input voltages, we used the LM2596 DC-DC Step-Down Buck Converter for supplying these two separate voltages. The specialty of LM2596 of supporting a wide range of input voltages and delivering a wide range of output voltages were taken advantage of. The SIM808 modules were fed directly from the 12V input source and all other components were converted to 5V by regulating the potentiometer in the LM2596 module. This is how two separate voltages were supplied in the Train System Mainboard.

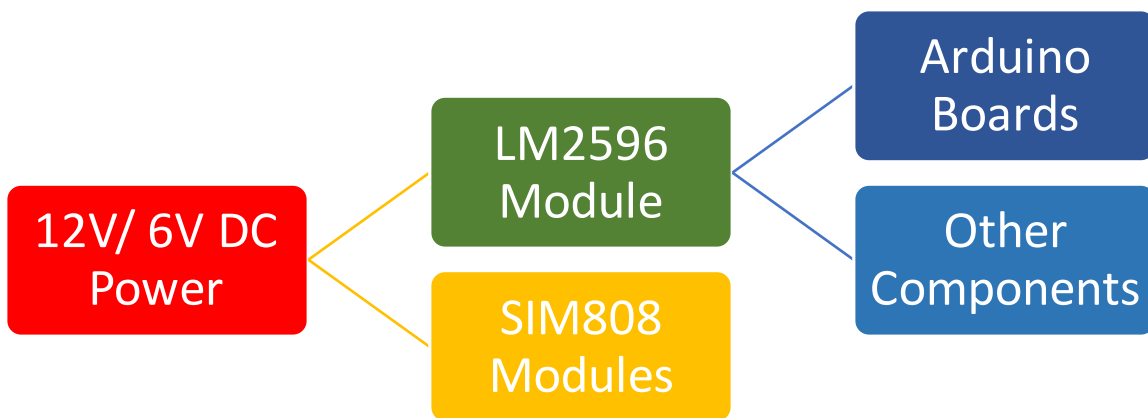


Figure 6.4a Flow chart of Power Distribution Diagram in the Train System Mainboard

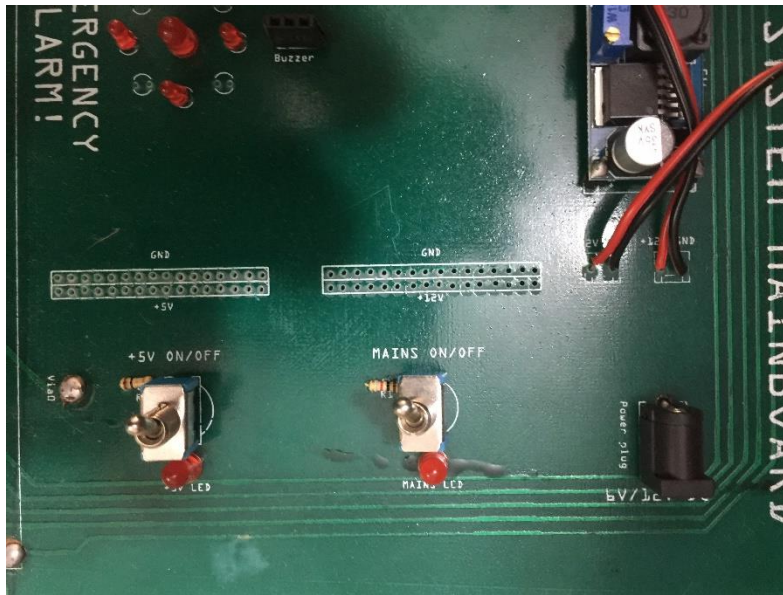


Figure 6.4b Power Distribution on Train System Mainboard

## 6.2 – Code Algorithms

Both the cores are programmed using the Arduino IDE version 1.8.1. The programming language supported by the board is based on C and C++ languages and is coded using the Arduino development environment.

There are two modes of operations in each of the cores they are:

- Debug Mode: Prints which task is being executed and its results for all tasks written in the whole program.
- Normal Mode: Prints output of specified tasks only.

### 6.2.1 – Master Core

Important variables used and their functions:

Sl. No	Name of Variable	Variable Type	Function
1	trainName	String	Stores the code name of that particular train to which the system is installed into
2	trainKey	String	An auto-generated key for that specific train to which the system is installed into
3	train_departing	String	Used to control mode of journey.

			“true” = On Route “false” = Idle
4	CCSMMessageCodes[]	String	Stores all possible message codes that will be sent to the Central Control System in an array of Strings.
5	levelCrossingNames[]	String	Stores all possible location names of the level crossings on the train’s route in an array of Strings.
6	levelCrossingNumbers[]	String	Stores all possible SIM numbers of the level crossings on the train’s route in an array of Strings.
7	levelCrossingLat[]	String	Stores all possible latitudes of level crossings that are on the train’s route in an array of Strings.
8	levelCrossingLon[]	String	Stores all possible longitudes of level crossings that are on the train’s route in an array of Strings.
9	levelCrossingCodes[]	String	Stores all possible message codes that will be sent to the Level Crossings in an array of Strings.

Table 6.1 Important variables used and their functions:

### Obtaining GPS Data:

GPS data from the SIM808 module is obtained by sending the AT Command, “AT+CGNSINF” to the SIM808 module from the Master core. The SIM808 module replies back with a line in the following format:

**+CGNSINF: <GNSS run status>, <Fix status>, <UTC date & Time>, <Latitude>, <Longitude>, <MSL Altitude>, <Speed over Ground>, <Course over Ground>, <Fix Mode>, <Reserved1>, <HDOP>, <PDOP>, <VDOP>, <Reserved2>, <GNSS Satellites in View>, <GNSS Satellites Used>, <GLONASS Satellites Used>, <Reserved3>, <C/N0 max>, <HPA>, <VPA>**  
**OK**



Example of a Raw GPS data Output:

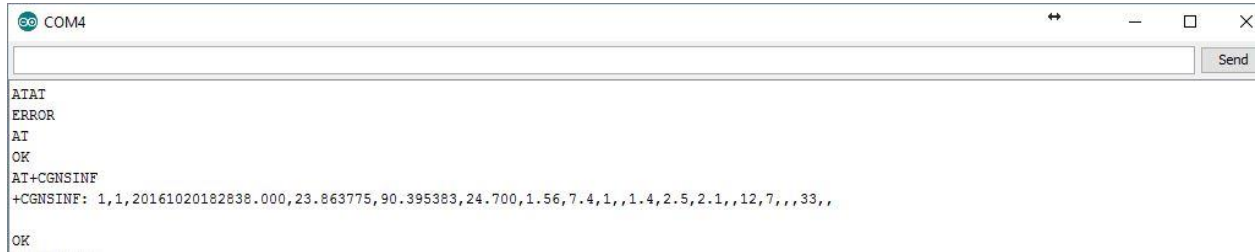


Figure 6.5 Screenshot of GPS reply from the SIM808 Module

Index	Parameter	Unit	Range	Length
1	GPS run status	--	0-1	1
2	Fix status	--	0-1	1
3	UTC date & Time	yyyyMMddhh mmss.sss	yyyy: [1980,2039] MM : [1,12] dd: [1,31] hh: [0,23] mm: [0,59] ss.sss:[0.000,60.999]	18
4	Latitude	±dd.dddddd	[-90.000000,90.000000]	10
5	Longitude	±ddd.ddddd	[-180.000000,180.000000]	11
6	MSL Altitude	meters		8
7	Speed Over Ground	Km/hour	[0,999.99]	6
8	Course Over Ground	degrees	[0,360.00]	6
9	Fix Mode	--	0,1,2 <sup>[1]</sup>	1
10	Reserved1			0
11	HDOP	--	[0,99.9]	4
12	PDOP	--	[0,99.9]	4
13	VDOP	--	[0,99.9]	4
14	Reserved2			0
15	GPS Satellites in View	--	[0,99]	2
16	GNSS Satellites Used	--	[0,99]	2
17	GLONASS Satellites in View	--	[0,99]	2
18	Reserved3			0
19	C/N0 max	dBHz	[0,55]	2
20	HPA <sup>[2]</sup>	meters	[0,9999.9]	6
21	VPA <sup>[2]</sup>	meters	[0,9999.9]	6

Table 6.2 AT+CGNSINF return Parameters

## **Extraction of GPS Data:**

After obtaining the GPS data, some parameters are stored in separate variables of String type so that they can be used later. The parameters that are extracted from the GPS data are:

- Latitude
- Longitude
- Altitude
- UTC Date and Time
- Speed over Ground
- Course over Ground

These variables are separated by using the ‘Comma Detection’ Algorithm.

The Comma Detection Algorithm is as follows:

```
for (int i = 0; i < value.length(); i++) { //Start searching for commas to split String
    if (value[i] == ',' || i == value.length() - 1) { //For detection of comma or for the last position of the String
        debugPrintln("Comma detected");
        String text = ""; //Text extracted from GPS response
        debugPrintln("Extracting the content:");
        for (int j = start; j <= i; j++) { //Split it from Starting point till current position
            if (value[j] != ',') text = text + value[j]; //Add each character to the result
        }
    }
}
```

## **Transmitting Data to Slave:**

The variables transmitted to the Slave core are as follows:

- Latitude
- Longitude
- Altitude
- UTC Date and Time
- Speed over Ground
- Course over Ground

Transmission of data to slave is done using a separate method. The code is given below:

```
//Transmits data to SLAVE
boolean transmitData(String value){
  if(transmitToSlave){
    delay(5000);
    debugPrintln("Beginning Transmission");
    Wire.beginTransmission(8); // transmit to device #8
    debugPrint("Transmitting: ");
    debugPrintln(value);
    for(int i=0; i < value.length(); i++){

      Wire.write(value[i]);    // sends data
    }
    Wire.endTransmission(); // stop transmitting
    debugPrintln("Transmission Complete");
    delay(500);
    return true;
  }
  else{
    debugPrintln("Transmission to Slave not enabled");
  }
}
```

### **Uploading Data to Web Server:**

The variables that are uploaded to the Web Server are as follows:

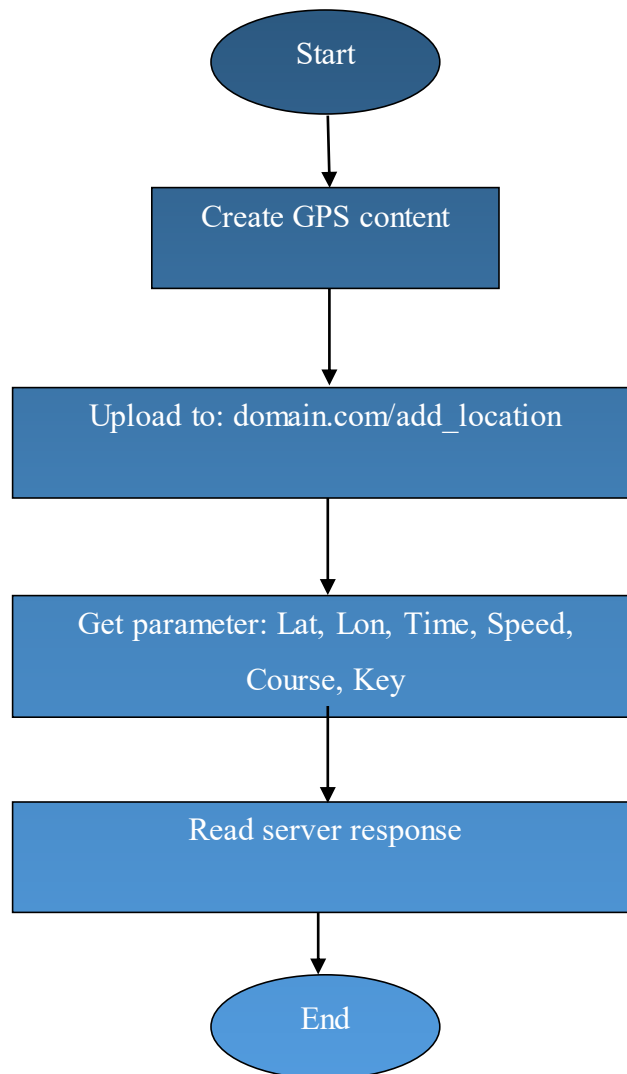
- Latitude
- Longitude
- Altitude
- UTC Date and Time
- Speed over Ground
- Course over Ground
- Train Name
- Train ID
- Train Key
- Train SIM Number
- Train Track of Travel

HTTP is used for uploading data to the web server. The transmission of data to the web server is done by sending specific AT Commands to the SIM808 Module. The AT Commands used to send the HTTP request are as follows:

Function	Syntax
Signal Quality Report	AT+CSQ
Attach or Detach from GPRS Service	AT+CGATT?
Initialize HTTP service	AT+HTTPINIT
Set HTTP Parameters value	AT+HTTTPARA="\URL\","http://www.ptc-ccs.com/web/update_train_info?visor=false&latitude=" + lat + "&longitude=" + lon + "&altitude=" + alt + "&time=" + dat + "&satellites=" + sat + "&arrival=" + train_arriving + "&departure=" + train_departing + "&speedOTG=" + spe + "&train="+ trainName + "&key=" + trainKey + "&course=" + cou + "\"
Set HTTP Parameters value	AT+HTTTPARA="\URL\","http://www.ptc-ccs.com/web/add_location_updated?visor=false&latitude=" + lat + "&longitude=" + lon + "&altitude=" + alt + "&time=" + dat + "&satellites=" + sat + "&speedOTG=" + spe + "&train="+ trainName + "&key=" + trainKey + "&course=" + cou + "\"
HTTP Method Action	AT+HTTPACTION=0
Read HTTP Server Response	AT+HTTPREAD
Terminate HTTP Service	AT+HTTPTERM
<i>(For more details See Appendix Section)</i>	

Table 6.3 AT Commands used to upload data to the Web Server

**Flow Chart for uploading location to the Web Server:**



*Figure 6.6 Flow Chart for Uploading data to the Web Server*

**Interrupt:** The Master is also responsible for controlling the action of two buttons which are: ‘Start/ Stop Journey’ button and the ‘Menu’ button. Input from these buttons are taken using the function ‘attachInterrupt ()’.

The first parameter to attachInterrupt is an interrupt number. Normally we should use digitalPinToInterrupt(pin) to translate the actual digital pin to the specific interrupt number. For example, if we connect to pin 3, use digitalPinToInterrupt(3) as the first parameter to attachInterrupt.

Digital Pins Usable For Interrupts: 2, 3, 18, 19, 20, and 21

Inside the attached function, delay() won't work and the value returned by millis() will not increment. Serial data received while in the function may be lost. Therefore we declared any variables as volatile that were modified within the attached function.

### Syntax

```
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);
```

```
attachInterrupt(interrupt, ISR, mode)
```

```
attachInterrupt(pin, ISR, mode) ;
```

### Parameters

interrupt:               the number of the interrupt (int)

pin:                      the pin number

ISR:                     the ISR to call when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.

mode:                    defines when the interrupt should be triggered. Four constants are predefined as valid values:

LOW to trigger the interrupt whenever the pin is low,

CHANGE to trigger the interrupt whenever the pin changes value

RISING to trigger when the pin goes from low to high,

FALLING for when the pin goes from high to low.

Flow Chart of the Master is given on the next page.

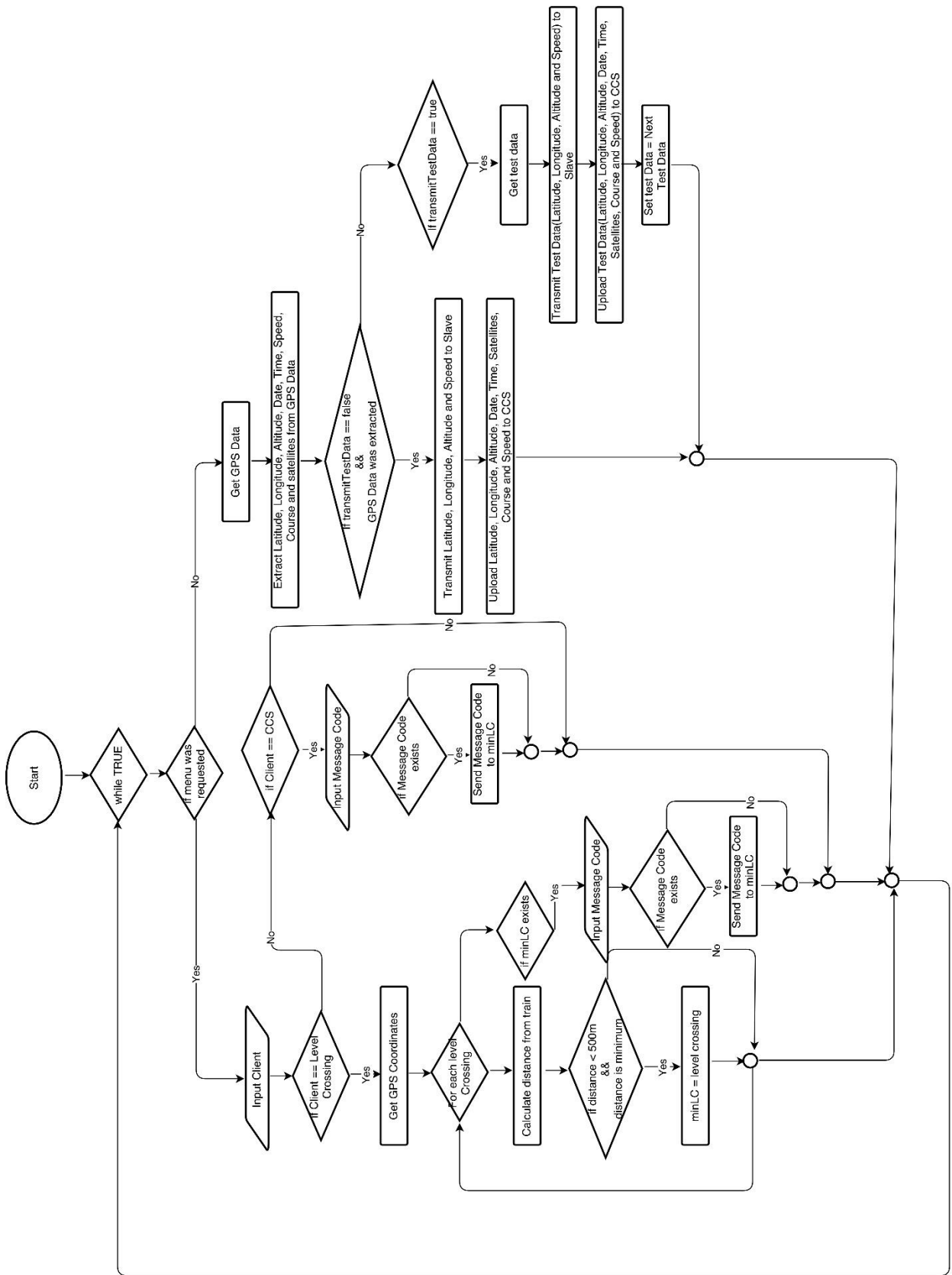


Figure 6.7 Flow chart of the Master Core

## Code: [Note: Due to Copyright reasons, full code is not provided]

master | Arduino 1.8.1  
File Edit Sketch Tools Help

```
1 //--IMPORTING DEPENDENCIES START--
2
3 //Including Libraries - START
4 #include <SoftwareSerial.h>
5 #include <Wire.h>
6 #include <Keypad.h>
7 #include <LiquidCrystal.h>
8 //Including Libraries - END
9
10 //Initialising Global Variables - START
11 SoftwareSerial mySerial(10, 11); // (RX, TX of GSM Module)
12
13 // initialize LCD
14 LiquidCrystal lcd_1(33, 31, 29, 27, 25, 23);
15 LiquidCrystal lcd_3(36, 34, 32, 30, 28, 26);
16
17
18
19 //--IMPORTING DEPENDENCIES END--
20
21 //--INITIALIZING VARIABLES START--
22
23 //Information about individual Train
24 String trainName = "SUB-701"; //Codename of the train
25 String trainKey = ""; //Key for the train! Auto-Generated
26 //Controller for START/STOP Interrupt request
27 const byte interruptPin = 18; //Switch to indicate START/STOP
28 String train_departing = "false"; //Mode of Journey: Start or Stop
29 volatile boolean interrupted = false; //Interrupt request: pressed or NOT pressed
30
31 //Information about Message Codes and Level Crossing
32 int lcdCodeArraySize = 7;
33 int lcdNumberArraySize = 2;
34 int lcdCodeArraySize = 16;
35
36 String CCSMessageCodes[] = {"HLI-001", "HLI-111", "HLI-000", "RES-111", "BRK-111", "ENG-111", "PPP-111", "STA-111", "STO-111", "DUR-111", "MED-111", "ACC-111", "HJK-111", "FIR-111", "LAW-111", "SER-111"};
37 String levelCrossingNames[] = {"MOHAKHALI", "NIKETAN"};
38 String levelCrossingNumbers[] = {"*8801676964718", "*8801670669566"};
39 String levelCrossingLat[] = {"23.7799341", "23.7738008"};
40 String levelCrossingLon[] = {"80.4088973", "80.4104637"};
41 String levelCrossingCodes[] = {"APP-001", "HLT-001", "HLT-111", "HLT-000", "RES-111", "BRK-111", "ENG-111"};
42
43 //Controller for Menu View Interrupt request
44 const byte menuInterruptPin = 19; //Switch to Display Menu
45 volatile boolean showMenu=false; //Menu Display: TRUE or FALSE
46
47 //Keypad for Menu
48 const byte ROWS = 4; //four rows
49 const byte COLS = 4; //four columns
50 //define the symbols on the buttons of the keypad
51 char hexaKeys[ROWS][COLS] = {
52   {'1','4','7','*'},
53   {'2','5','8','0'},
54   {'3','6','9','#'},
55   {'A','B','C','D'}
56 };
57 byte rowPins[ROWS] = {44, 42, 40, 38}; //connect to the row pinouts of the keypad
58 byte colPins[COLS] = {52, 50, 48, 46}; //connect to the column pinouts of the keypad
59
60 //Initialize an instance of class NewKeypad
61 Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
62
63 //Controller for debug mode
64 boolean debugHTTP = true;
65 boolean debugSlave=true;
66 boolean transmitToSlave = true; //Set it false if you don't want to send data to slave
67 boolean debugMode = false; //Controller of Debug Mode [Default: OFF]
68 boolean transmitOnlyTestData = true; //Set it to true in order to skip actual GPS coordinates and use Demo Coordinates
69
70
71 //Controller for GPS Coordinates
72 String Location = "";
73
74 String latitude; //Saves current latitude of GPS
75 String longitude; //Saves current longitude of GPS
76 String altitude; //Saves current longitude of GPS
77 String datetime; //Saves current datetime of GPS
78 String speedOTG = "0.0"; //Saves current speed of GPS
79 String course = "N/A"; //Saves current course of GPS
80 String satellites = "N/A"; //Saves current satellites of GPS
81
82
83 //Demo GPS coordinates for debug: From 'Mohakhali Flyover' to 'Niketan Road No. 5'
84 String demoLatitudes[] = {"23.7806097", "23.7804821", "23.7801679", "23.7798341", "23.7793334", "23.7788516", "23.7783668", "23.7746558", "23.7748759", "23.7738008", "23.7728852", "23.7731614"};
85 String demoLongitudes[] = {"80.3590559", "80.4010677", "80.4050027", "80.4088973", "80.4105281", "80.4105281", "80.4136777", "80.4128938", "80.4119121", "80.4105603", "80.4104637", "80.4102957"};
86 String demoAltitude = "0.41";
87 String demoDatetime = "201703081250.00";
88 String demoSatellite = "1.44(Demo)";
89 String demoCourse = "N/A(Demo)";
90 String demoSpeedOTG[] = {"0.50", "0.60", "0.70", "0.80", "0.90", "1.0", "1.1", "1.2", "1.3", "1.4", "1.5", "1.6"};
91 int counterHTTP = 0;
92
93 //--INITIALIZING VARIABLES END--
94
95 //--INITIALIZING SETUP START--
96 void setup() {
97   // put your setup code here, to run once:
98   //Initializing Required Modules - START
99   Serial.begin(9600); // Setting the baud rate of Serial Monitor (Arduino)
100   mySerial.begin(9600); // Setting the baud rate of GSM Module
101
102   lcd_1.begin(20, 4);
103   lcd_3.begin(20, 4);
104   Wire.begin(); // join i2c bus (address optional for master)
105   // pinMode(sw, INPUT);
106
107   //attachInterrupt(0, StartOfJourney, RISING);
108
109   turnDebugOff(); //Turning Debug Mode OFF: turnDebugOff() || Turning Debug Mode ON: turnDebugOn()
110   debugPrintln("Debug Mode is turned ON. Turn it off to stop viewing these messages.");
111
112   debugPrintln("Setting Key for the train");
113   setTrainKey();
114   String msg = "Setup Complete. The Key is " + trainKey;
115   debugPrintln(msg);
116
117   debugPrintln("Executing AT commands to set up the GSM Module");
```



```

844 }
845 else{
846     led_1.clear();
847     displayInLCDLine1(led_1, "NEAREST LEVEL CROSSING NOT FOUND");
848     Serial.println("Nearest Level Crossing could not be found");
849 }
850 }
851 }
852 else{
853     led_1.clear();
854     displayInLCDLine1(led_1, "INVALID COMMAND");
855     Serial.println("Invalid Command..Preparing to Exit");
856 }
857 }
858 }
859 }
860 //Send SMS
861 boolean sendSMS(String number, String message){
862     mySerial.write("AT+CMGF=1\r"); //GSM Module in Text Mode
863     delay(1000);
864     ShowSerialData();
865 }
866 mySerial.println("AT+CMGS=\""+number+"\r"); // Replace x with mobile number
867 delay(1000);
868 ShowSerialData();
869 }
870 mySerial.println(message);// The SMS text you want to send
871 delay(1000);
872 mySerial.println((char)26);// ASCII code of CTRL+Z
873 delay(1000);
874 }
875 ShowSerialData();
876 }
877 return true;
878 }
879 //Upload Code to CCS
880 void SubmitHttpRequest(String text)
881 {
882     debugPrintln("Setting CSQ Mode");
883     mySerial.println("AT+CSQ");
884     delay(100);
885 }
886 ShowSerialData();// this code is to show the data from gprs shield, in order to easily see the process of how the gprs shield submit a http request, and the following is for this purpose too.
887 }
888 debugPrintln("Setting CGATT");
889 mySerial.println("AT+CGATT");
890 delay(1000);
891 }
892 ShowSerialData();
893 }
894 debugPrintln("Setting Connection type for GPRS");
895 mySerial.println("AT+SAPBR=3,1,\"CTYPE\",\"GPRS\");//setting the SAPBR, the connection type is using gprs
896 delay(1000);
897 }
898 ShowSerialData();
899 }
900 debugPrintln("Setting local Access Point for GPRS");
901 mySerial.println("AT+SAPBR=3,1,\"API\",\"CMNET\");//setting the API, the second need you fill in your local apn server
902 delay(1000);
903 }
904 ShowSerialData();
905 }
906 debugPrintln("Enabling GPRS");
907 mySerial.println("AT+SAPBR=1,1");//setting the SAPBR, for detail you can refer to the AT command manual
908 delay(2000);
909 }
910 ShowSerialData();
911 }
912 debugPrintln("Initializing HTTP request");
913 mySerial.println("AT+HTTPIII");//init the HTTP request
914 }
915 delay(2000);
916 ShowSerialData();
917 }
918 debugPrintln("Preparing HTTP content");
919 Serial.println("AT+HTTPPARA=\"URL\", \"http://www.ptc-ccs.com/web/add_message/\" + text + "\");// setting the httppara, the second parameter is the website you want to access
920 mySerial.println("AT+HTTPPARA=\"URL\", \"http://www.ptc-ccs.com/web/add_message/\" + text + "\");// setting the httppara, the second parameter is the website you want to access
921 delay(1000);
922 String preparedData = getSerialData();
923 debugPrintln(preparedData);
924 }
925 }
926 debugPrintln("Sending HTTP content");
927 mySerial.println("AT+HTTFACTION=0");//submit the request
928 delay(10000);//the delay is very important, the delay time is base on the return from the website, if the return datas are very large, the time required longer.
929 //while(!mySerial.available());
930 }
931 String sentData = getSerialData();
932 debugPrintln(sentData);
933 }
934 debugPrintln("Reading Server response");
935 mySerial.println("AT+HTTPREAD");// read the data from the website you access
936 delay(1000);
937 }
938 String serverResponse = getSerialData();
939 }
940 debugPrintln(serverResponse);
941 }
942 debugPrintln("Terminating HTTP request");
943 mySerial.println("AT+HTTPTERM");//init the HTTP request
944 delay(2000);
945 }
946 String terminatedHTTP = getSerialData();
947 debugPrintln(terminatedHTTP);
948 }
949 debugPrintln("Disabling GPRS");
950 mySerial.println("AT+SAPBR=0,1");//setting the SAPBR, for detail you can refer to the AT command manual
951 delay(2000);
952 }
953 ShowSerialData();
954 }
955 mySerial.println("");
956 delay(1000);
957 //Deleting uploaded message
958 }
959 }
960 //CORE FUNCTIONS END--

```

## 6.2.2 – Slave Core

**Receive Data From Master:** The prime task of the Slave is to obey its master, which means, receiving data from the Master will be the first priority of the slave core. Data received from the master includes the following:

- Latitude
- Longitude
- Altitude
- UTC Date and Time
- Speed over Ground
- Course over Ground

**Send or Receive SMS:** The secondary task of the slave is to send or receive messages from other systems. SMS that has been received, is deleted automatically after it has been read by the user. This is done so that the message memory of the SIM doesn't get full over time and block new SMS from incoming. This reduces the chances of miscommunication and the probability of any hazard. The AT Commands that were used to send and receive SMS messages are given in the following table.

Command	Description
AT+CMGD	Delete SMS message
AT+CMGF	Select SMS message format
AT+CMGL	List SMS messages from preferred store
AT+CMGR	Read SMS message
AT+CMGS	Send SMS message
AT+CMGW	Write SMS message to memory
AT+CMSS	Send SMS message from storage
AT+CNMI	New SMS message indications
AT+CPMS	Preferred SMS message storage
AT+CRES	Restore SMS settings
AT+CSAS	Save SMS settings
AT+CSCA	SMS service center address
AT+CSCB	Select cell broadcast SMS messages
AT+CSDH	Show SMS text mode parameters
AT+CSMP	Set SMS text mode parameters
AT+CSMS	Select message service

*Table 6.4 AT Commands used to send or Receive SMS Messages*

**Extraction of Received SMS Messages:** Upon sending the AT Command, AT+CNMI=<mode>[,<mt>[,<bm>[,<ds>[,<bfr>]]]] notification of any new SMS messages are received. SMS Messages that are received are displayed in the format below:

+CMT: <oa>, <scts>[,<toa>,<fo>,<pid>,<dcs>,<sca>,<tosca>,<length>]<CR><LF><data>

Example Output:

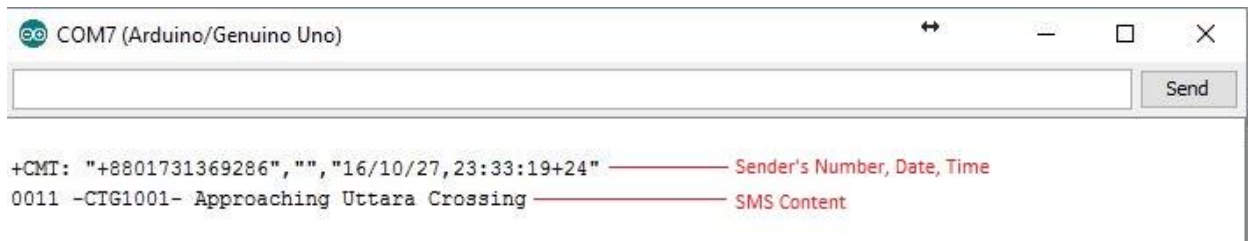


Figure 6.8 Screenshot of a Received SMS Raw Output

Extraction of SMS is done by the ‘Comma Detection’ Algorithm. This means every content between each comma are extracted and stored in a separate variable. The parameters that are extracted from a SMS raw output are:

- Sender’s Number
- Date & Time at which the SMS was received
- SMS Message content

These variables are then used to perform different other tasks.

SMS Extraction Code is as follows:

```
boolean ExtractSMSdata(String value) {
  debugPrintln("Extracting SMS contents from the provided value:");
  int start = 0; //Start position of trim
  int counter = 0; //Number of commas detected

  for (int i = 0; i < value.length(); i++) { //Start searching for commas to split String

    if (value[i] == ',' || i == value.length() - 1) { //For detection of comma or for the last position of the String
      debugPrintln("Comma detected");
      String text = ""; //Text extracted from SMS response
      debugPrintln("Extracting the content:");
      for (int j = start; j <= i; j++) { //Split it from Starting point till current position
```

```

    if (value[j] != ',' && value[j] != '"' && value[j] != '/' && value[j] != '\\' && value[j] != '\n') text = text + value[j]; //Add each character to
the result
    else{
        if(value[j] == '/' || value[j] == '\\'){
            text = text + '-';
        }
    }
}
debugPrint("Content Extracted: ");
debugPrintln(text);

if (counter == 2) {
    debugPrintln("Sender Number recieved");
    sim_number = text; //1st comma indicates Sender Number
}
else if (counter == 4) {
    debugPrintln("Date recieved");
    sms_date = text;
}
else if (counter == 5) {
    debugPrintln("Time recieved");
    sms_time = text;
}
else if (counter == 6) {
    debugPrintln("Message recieved");
    sms_message = text;
    counter++; //Increase counter
    start = i; //Set next starting point of trim
}
}
debugPrint("Total Data recieved: ");
debugPrintln(counter);
if (counter >= 5) {
    debugPrintln("SMS contents extracted");
    return true;
}
else {
    debugPrintln("SMS contents not extracted");
    return false;
}
}
}

```

**Interrupt:** The Slave is also responsible for controlling the action of the ‘Emergency Alert’ button. Input from the button is taken using the function ‘attachInterrupt ()’.

The first parameter to attachInterrupt is an interrupt number. Normally we should use digitalPinToInterrupt(pin) to translate the actual digital pin to the specific interrupt number. For example, if we connect to pin 3, use digitalPinToInterrupt(3) as the first parameter to attachInterrupt.

Digital Pins Usable For Interrupts: 2, 3, 18, 19, 20, and 21

Inside the attached function, delay() won't work and the value returned by millis() will not increment. Serial data received while in the function may be lost. Therefore we declared any variables as volatile that were modified within the attached function.

### Syntax

```
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);
```

```
attachInterrupt(interrupt, ISR, mode)
```

```
attachInterrupt(pin, ISR, mode) ;
```

### Parameters

interrupt:           the number of the interrupt (int)

pin:                 the pin number

ISR:                the ISR to call when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.

mode:               defines when the interrupt should be triggered. Four constants are predefined as valid values:

LOW to trigger the interrupt whenever the pin is low,

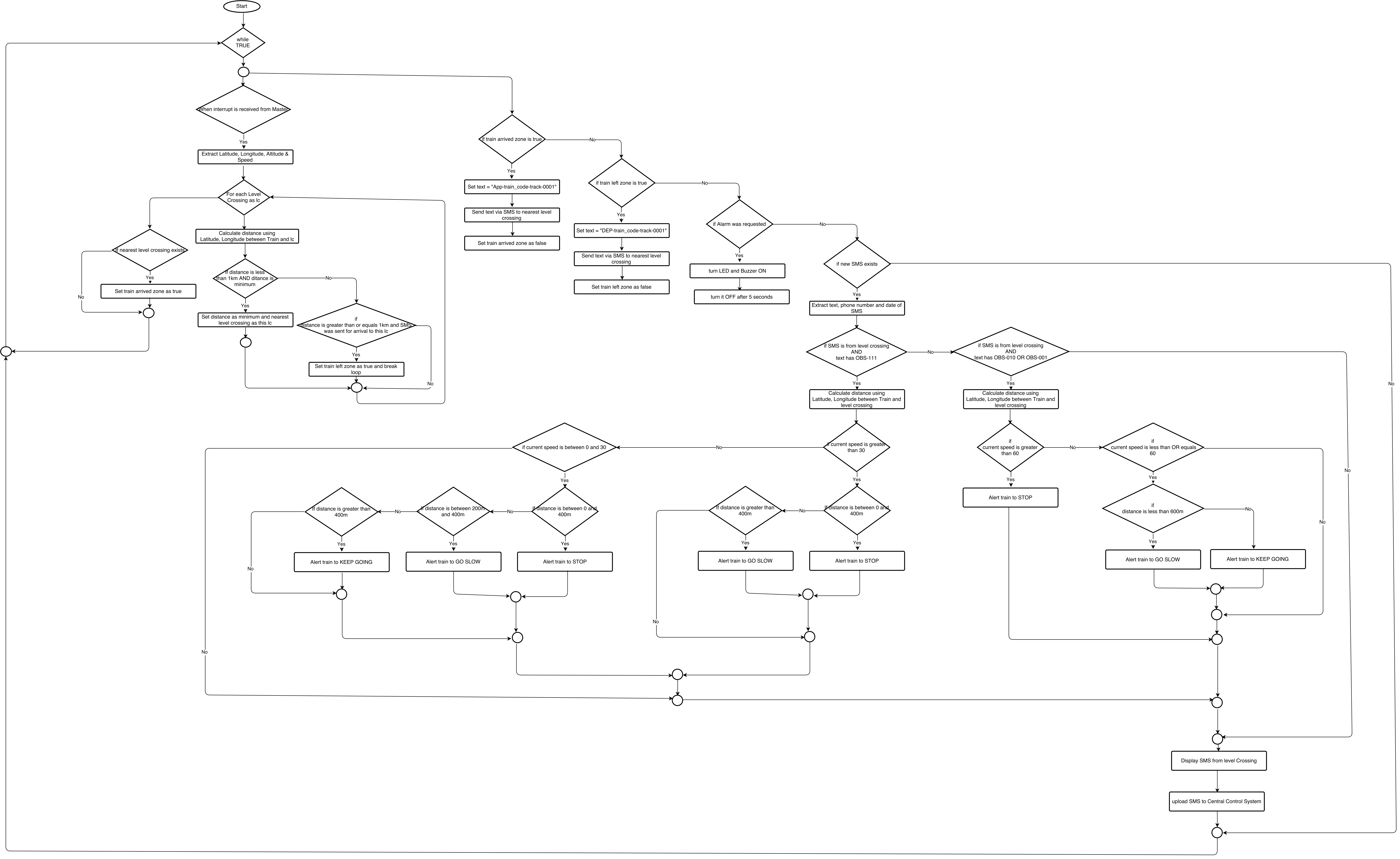
CHANGE to trigger the interrupt whenever the pin changes value

RISING to trigger when the pin goes from low to high,

FALLING for when the pin goes from high to low.

**Control of Peripherals:** The Slave also changes the state of the status LEDs connected to it, on different circumstances.

Flow Chart of the Slave is given on the next page.



**Code:** [Note: Due to Copyright reasons, full code is not provided]

```
slave | Arduino 1.8.1
File Edit Sketch Tools Help

slave
1 //Including Libraries - START
2 #include <SoftwareSerial.h>
3 #include <Wire.h>
4 #include <LiquidCrystal.h>
5 //Including Libraries - END
6
7 //Initializing Global Variables - START
8 SoftwareSerial mySerial(11, 10); //RX, TX of GSM Module
9
10 LiquidCrystal lcd_2(23, 31, 29, 27, 25, 23);
11 LiquidCrystal lcd_4(45, 43, 41, 39, 37, 35);
12
13 //Code of train
14 String train_code = "SUB-701";
15
16 //Information about Message Codes and Level Crossing
17 int lcCodeArraySize = 7;
18 int lcNumberArraySize = 2;
19 int ccsCodeArraySize = 16;
20
21 String ccsNumber = "8001700339963";
22 String ccsMessageCodes[] = {"HLT-001", "HLT-111", "HLT-000", "RES-111", "BRK-111", "EMG-111", "FFF-111", "STA-111", "STO-111", "OUR-111", "MED-111", "ACC-111", "SNK-111", "FIR-111", "LAW-111", "SKN-111"};
23 String levelCrossingNames[] = {"MOMNAHALLI", "HIKETAN"};
24 String levelCrossingNumbers[] = {"8801676964719", "8801670669566"};
25 String levelCrossingLat[] = {"23.7798341", "23.7738008"};
26 String levelCrossingLon[] = {"90.4088973", "90.4105603"};
27 String levelCrossingCodes[] = {"RFP-001", "HLT-001", "HLT-111", "HLT-000", "RES-111", "BRK-111", "EMG-111"};
28
29 //Alarm Control
30 volatile boolean alarmOn = false; //Controls Alarm Option
31
32 //Interrupt Pins
33 const byte alarmSwitch = 19;
34 const int alarmLED = 18;
35 const int alarmBuzzer = 17;
36 const int newMSGLED = 52;
37 const int stopLED = 48;
38 const int goSlow = 40;
39 const int go = 36;
40 //GO SLOW: 40, GO: 36, STOP: 48
41 //SMS upload Control
42 boolean newText=false;
43 String SMSmessage="";
44
45 //SMS Notification
46 int smsFlag = 0;
47 int a = 0;
48 int b = 0;
49
50 String trainTrack = "A";
51 String location = "";
52 String startingLocation = "";
53 float distance = 1.15;
54
55 String sim_number;
56 String sms_date;
57 String sms_time;
58 String sms_message;
59
60 String location;
61 String latitude; //Saves current latitude of GPS
62 String longitude; //Saves current longitude of GPS
63 String altitude; //Saves current longitude of GPS
64 String speedOTG = "0.0"; //Saves current speed of GPS
65
66 float Latitude;
67 float Longitude;
68 float Altitude;
69 float Speed;
70 float CourseOverGround;
71
72 float departureLat;
73 float departureLon;
74
75 //Flag to Check
76 boolean arrivedZone = false;
77 boolean leftZone = false;
78 boolean inZone = false;
79 int sendMsgTo = -1;
80 float lcToTrainDistance = -1;
81
82 boolean debugMode=false; //Controller of Debug Mode [Default: OFF]
83 boolean testSpeedInclude = true; //Checks Demo Speed
84
85 void setup() {
86 // put your setup code here, to run once:
87 //Initializing Required Modules - START
88 Serial.begin(9600); // Setting the baud rate of Serial Monitor (Arduino)
89 mySerial.begin(9600); // Setting the baud rate of GSM Module
90
91 lcd_2.begin(20, 4);
92 lcd_4.begin(20, 4);
93
94 // pinMode(sw, INPUT);
95
96 //attachInterrupt(0, StartOfJourney, RISING);
97
98 turnDebugOff(); //Turning Debug Mode OFF: turnDebugOff(). || Turning Debug Mode ON: turnDebugOn()
99 debugPrintln("Debug Mode is turned ON. Turn it off to stop viewing these messages.");
100
101 debugPrintln("Executing AT commands to set up the GSM Module");
102 debugPrintln("");
103 debugPrintln("Communicating with GSM Module");
104
105 mySerial.write("AT\r"); //Communicating with GSM
106 delay(1000);
107 ShowSerialData();
108
109 debugPrintln("Setting Default Configurations of GSM");
110 mySerial.write("ATZ\r"); //Reset Default
111 delay(2000);
112 ShowSerialData();
113
114 debugPrintln("Setting Baud rate of Data flow to 9600");
115
116 mySerial.write("AT+IFR=9600\r"); //Baud Rate set
117 delay(2000);
118 ShowSerialData();
```

```

1037     }
1038     else if (distance > 400 && distance <= 600){
1039         Serial.println("KEEP GOING!!!");
1040         keepGoing();
1041     }
1042     else if (distance > 600 && distance <= 800){
1043         Serial.println("KEEP GOING!!!");
1044         keepGoing();
1045     }
1046     else if (distance > 800 && distance <= 1000){
1047         Serial.println("KEEP GOING!!!");
1048         keepGoing();
1049     }
1050     else if (distance > 1000 && distance <= 1200){
1051         Serial.println("KEEP GOING!!!");
1052         keepGoing();
1053     }
1054     }
1055 }
1056 else if (txt == "085-010" || txt == "085-001"){
1057     if (curSpeed > 60){
1058         if (distance > 0 && distance <= 1200){
1059             Serial.println("STOP NOW!!!");
1060             stopNow();
1061         }
1062     }
1063 }
1064 else if (curSpeed > 50 && curSpeed <= 60){
1065     if (distance > 0 && distance <= 600){
1066         Serial.println("GO SLOWLY!!!");
1067         goSlowly();
1068     }
1069     else if (distance > 600 && distance <= 1200){
1070         Serial.println("KEEP GOING!!!");
1071         keepGoing();
1072     }
1073     else if (distance > 0 && distance <= 600){
1074         Serial.println("KEEP GOING!!!");
1075         keepGoing();
1076     }
1077 }
1078 }
1079 }
1080 int isFromOD(String no){
1081     Serial.println("Checking if SMS is from OD");
1082     for (int i=0; i < lcNumberArraySize; i++){
1083         if (no == levelCrossingNumbers[i]){
1084             return i;
1085         }
1086     }
1087     return -1;
1088 }
1089 void SMS_Mohakhali_UP() {
1090     mySerial.write("AT+CMGF=1"); // GSM Module in Text Mode
1091     delay(1000);
1092     mySerial.println("AT+CMGS=\"" + 880177549536 + "\""); // Replace x with mobile number
1093     delay(1000);
1094     mySerial.println("0011 - TRAIN:CTG1001 Approaching Mohakhali Crossing");// The SMS text you want to send
1095     delay(1000);
1096     mySerial.println((char)26);// ASCII code of CTRL+Z
1097     delay(1000);
1098 }
1099
1100 void SMS_Mohakhali_DOWN() {
1101     mySerial.println("AT+CMGF=1"); // GSM Module in Text Mode
1102     delay(1000);
1103     mySerial.println("AT+CMGS=\"" + 880177549536 + "\""); // Replace x with mobile number
1104     delay(1000);
1105     mySerial.println("0000 - TRAIN:CTG1001 Crossed Mohakhali Crossing");// The SMS text you want to send
1106     delay(1000);
1107     mySerial.println((char)26);// ASCII code of CTRL+Z
1108     delay(1000);
1109     //mySerial.println("AT+CMGF=0");
1110 }
1111 void SMS_Uttara_UP() {
1112     mySerial.println("AT+CMGF=1"); // GSM Module in Text Mode
1113     delay(1000);
1114     //Serial.println(Serial.read());
1115     mySerial.println("AT+CMGS=\"" + 880177549536 + "\""); // Replace x with mobile number
1116     delay(1000);
1117     mySerial.println("0011 - TRAIN:CTG1001 Approaching Uttara Crossing");// The SMS text you want to send
1118     delay(1000);
1119     mySerial.println((char)26);// ASCII code of CTRL+Z
1120     delay(1000);
1121     //mySerial.println("AT+CMGF=0");
1122 }
1123 }
1124
1125 void SMS_Uttara_DOWN() {
1126     mySerial.println("AT+CMGF=1"); // GSM Module in Text Mode
1127     delay(1000);
1128     //Serial.println(Serial.read());
1129     mySerial.println("AT+CMGS=\"" + 880177549536 + "\""); // Replace x with mobile number
1130     delay(1000);
1131     mySerial.println("0000 - TRAIN:CTG1001 Crossed Uttara Crossing");// The SMS text you want to send
1132     delay(1000);
1133     mySerial.println((char)26);// ASCII code of CTRL+Z
1134     delay(1000);
1135     //mySerial.println("AT+CMGF=0");
1136 }
1137 //Send SMS
1138 boolean sendSMS(String number, String message){
1139     mySerial.write("AT+CMGF=1\r"); // GSM Module in Text Mode
1140     delay(1000);
1141     ShowSerialData();
1142 }
1143 mySerial.println("AT+CMGS=\"" + number + "\""); // Replace x with mobile number
1144 delay(1000);
1145 ShowSerialData();
1146 mySerial.println(message);// The SMS text you want to send
1147 delay(1000);
1148 mySerial.println((char)26);// ASCII code of CTRL+Z
1149 delay(1000);
1150 ShowSerialData();
1151 return true;
1152 }
1153 }
1154 }

```



### 6.2.3 – Master-Slave Communication

The Master and Slave are connected to each other through SDA and SCL pins which are pin numbers 20 and 21 respectively. The method of communication between them is called I2C. This communication requires the use of Arduino's "Wire" Library.

This library allows us to communicate with I2C / TWI devices. On the Arduino boards with the R3 layout (1.0 pinout), the SDA (data line) and SCL (clock line) are on the pin headers close to the AREF pin.

There are both 7- and 8-bit versions of I2C addresses. 7 bits identify the device, and the eighth bit determines if it's being written to or read from. The Wire library uses 7 bit addresses throughout.. However the addresses from 0 to 7 are not used because are reserved so the first address that can be used is 8. Please note that a pull-up resistor is needed when connecting SDA/SCL pins. MEGA 2560 board has pull-up resistors on pins 20 - 21 onboard.

I<sup>2</sup>C (Inter-Integrated Circuit), pronounced I-squared-C, is a multi-master, multi-slave, packet switched, single-ended, serial computer bus invented by Philips Semiconductor (now NXP Semiconductors). It is typically used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication. Alternatively I<sup>2</sup>C is spelled I2C (pronounced I-two-C) or IIC (pronounced I-I-C).

### 6.2.4 – Calculations

One major calculation in the entire algorithm of the train system is the Distance Calculation between two GPS coordinates. The distance between two GPS coordinates is done using the Haversine formula shown below. Haversine formula is used to calculate the great-circle distance between two points – that is, the shortest distance over the earth's surface.

Haversine formula:

$$a = \sin^2(\Delta\phi/2) + \cos \phi_1 \cdot \cos \phi_2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a})$$

$$\text{distance} = \text{Radius of the Earth in meters} \cdot c$$

Where  $\phi$  is bearing of latitude,  $\lambda$  is bearing longitude

## 6.3 – PCB Design

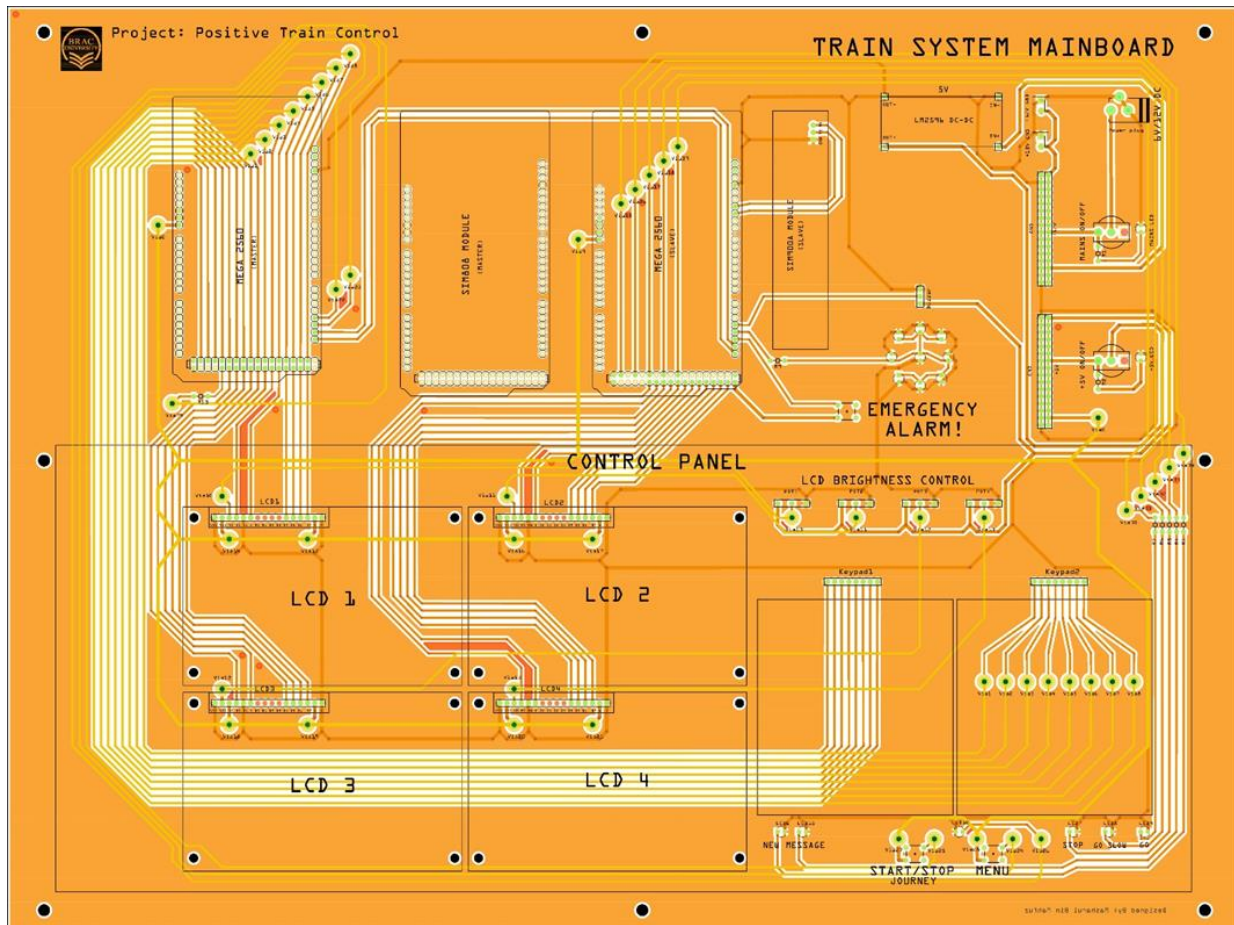


Figure 6.10 PCB Design of Train System Mainboard

## 6.4 – Working Principle

### 6.4.1 – Obtaining GPS Data

At the start of the system, it first initializes the setup which takes 15 to 20 seconds to complete. After the initialization is complete GPS data is obtained using the 1575.42 MHz GPS antenna attached with the SIM808 Module. The GPS data is then extracted into separate variables to make use of. The system is programmed to obtain GPS data every 2 seconds.

### 6.4.2 – Uploading data to Web Server

Information of the train along with the GPS data is then uploaded to the web server of the Central Control System using HTTP request. GPS data is uploaded every 10 to 15 seconds to the server

and saved as a .txt file in the server's database. The time delay is dependent on the mobile operator's cellular network and internet speed.

#### **6.4.3 – Obtaining Data from Web Server**

Data from the web server is obtained from the response that the website sends to the HTTP request sent by the SIM808 module.

#### **6.4.4 – Check for new SMS**

SMS check is done by the slave core and since it has a dedicated SIM808 module connected to it, check for new SMS can be performed every second. Whenever there is a new SMS it is displayed on LCD 3 and the 'New Message' LED glows.

#### **6.4.5 – Functions of the 'Start/Stop Journey' Button**

All trains that has their engines on, does not go on a journey. Some goes for a wash or some for maintenance. For these purposes the trains does not need to be connected with the Central Control System. This is where the 'Start/Stop Journey' button comes into play. This button connects the train with the Central Control System and sends a request if the train can start its journey towards its destination. The Central Control System replies back with a message for the train.

Upon the press of this button all information of the train is sent to the Level Crossing. The information that are sent the Central Control System are listed below:

- Train Name
- Train ID
- SIM Number on train
- Current Location (Latitude & Longitude)
- Current Speed
- Destination Station
- Departing Station

The Central Control System and the train are now connected to each other. All activities of the train can now be monitored. The train is now bound to listen to the commands from the Central Control System. Automatic communications will now take place between the Central Control System and the train.

### 6.4.6 – Functions of the ‘Menu’ Button

The ‘Menu’ button was added for the ease of using the GUI of the train system mainboard. This button opens a small menu to send messages manually to the Central Control System or any nearby Level Crossings during an emergency case. The Menu appears on LCD 2 of the train system mainboard. The flow chart below shows the entire Menu Map of the train system.

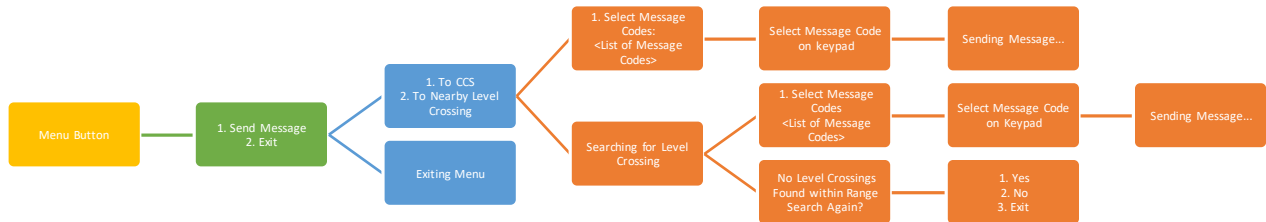


Figure 6.11 Menu Map of the Train System

### 6.4.7 – The User Interface

The user interface is made as simple as possible for the drivers to use since it requires fast communication during emergency situations. The User Interface comprises of four LCDs, 2 Keypads, buttons, LEDs and Potentiometers to control the brightness of the LCDs.

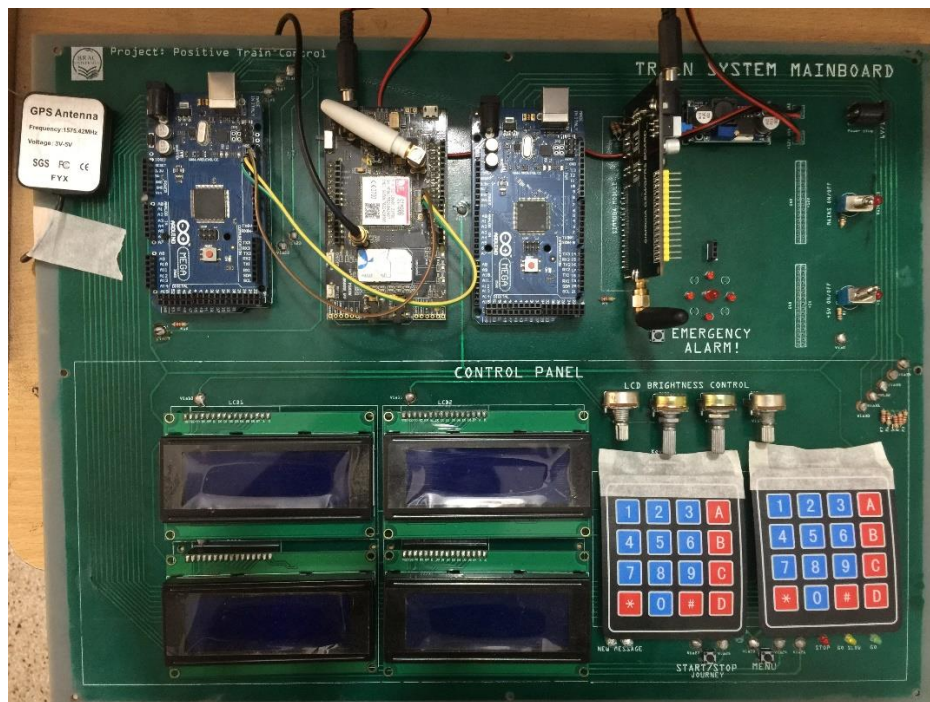


Figure 6.12 Train System Mainboard

LCD 1 shows the current Location of the train, its latitude and longitude and the Journey status.

LCD 2 shows system instructions for the driver.

LCD 3 shows Destination and Departure status, as well as the journey status.

LCD 4 shows any new messages from the Central Control System or any Level Crossings.

Keypads are installed for 'speed messaging' which means, upon pressing the 'Menu' button if 'Send Message' option is selected, the message code can be selected straight away from the keypad. This would really be hassle free and fast during emergency situations.

The press of 'Start/ Stop Journey' Button glows the 'On Route' Led indicating that the train is on a journey at that moment.

The press of 'Menu' button displays the Menu on LCD 2.

Signal LEDs (go, go slow and stop) glows whenever the train is instructed to alter its speed or stop completely.

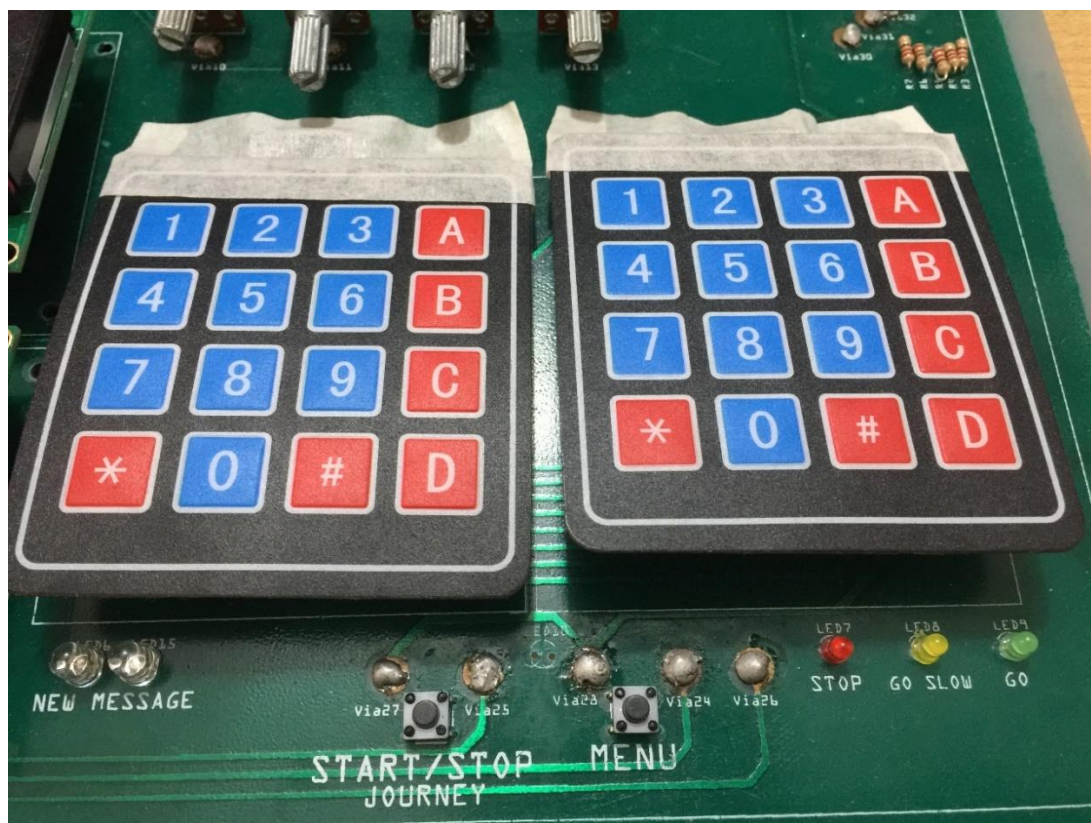


Figure 6.13 the Control Panel of Train System

Four potentiometers for four LCDs are installed to adjust the brightness of the LCDs.



*Figure 6.14 Potentiometers to Control LCD Brightness*



*Figure 6.15 Emergency Alarm Button and LEDs*

#### **6.4.8 – Distance Calculation**

Distance calculation is done using the Haversine formula as described earlier. Distance is calculated whenever a Level Crossing is found within the train's range.

#### **6.4.9 – Search for Nearest Level Crossing**

Searching for nearest level crossing is made easier since the GPS coordinates of all possible Level Crossings on the train's course is predefined in the system. The train system searches for level crossings that are within 1.5km range using the distance calculation and the GPS coordinates. However it only shows the level crossings that will be approached by the train on its radar and discards the others.

#### **6.4.10 – Obtain Nearest Level Crossing Data**

If a Level Crossing is found, the train system cross matches the GPS coordinates of the level crossing with its own database and finds the name of the location of the level crossing and the SIM number of that particular level crossing to start a communication between them.

#### **6.4.11 – Communicating with the nearest Level Crossing**

Once a level crossing is found that is to be approached by the train, the train system sends a message code indicating that the train is approaching that particular level crossing. The Level crossing replies back if there are any obstacles on the level crossing or not. After the train has left the level crossing and passed a safe distance, it then sends a message again to the level crossing with a message of departure.

#### **6.4.12 – Responses to Obstacle Detection**

If the Level Crossing replies the train with a message code indicating that there are obstacles on the level crossing, the train system determines the speed and distance of the train to the level crossing every second until it reaches the level crossing. By determining those speed and distance data the train decides what message needs to be instructed to the driver. The algorithm of obstacle detection responses are shown in the table next page.

<b>IF INCOMING MESSAGE = OBS-111</b>		
<b>Speed (Km/h)</b>	<b>Distance (Km)</b>	<b>Instructions</b>
60-80	0.0-1.2	WARNING! Obstacle Detected. Reduce Speed
50-60	0.8-1.2	WARNING! Obstacle Detected. Maintain Current Speed.
	0.0-0.8	WARNING! Obstacle Detected. Reduce Speed
30-50	0.4-1.2	WARNING! Obstacle Detected. Maintain Current Speed.
	0.0-0.4	WARNING! Obstacle Detected. Reduce Speed
10-30	0.0-1.2	WARNING! Obstacle Detected. Maintain Current Speed.
60-80	1.0-1.2	WARNING! Obstacle _____ m Ahead. Slow Down.
	0.8-1.0	WARNING! Obstacle _____ m Ahead. Apply Brake level 1.
	0.6-0.8	WARNING! Obstacle _____ m Ahead. Apply Brake level 2.
	0.4-0.6	WARNING! Obstacle _____ m Ahead. Prepare to Stop.
	0.2-0.4	WARNING! Obstacle _____ m Ahead. Apply Full Brakes & Stop
	0.0-0.2	WARNING! Obstacle _____ m Ahead. Apply Emergency Brakes & Stop Immediately
50-60	1.0-1.2	WARNING! Obstacle _____ m Ahead. Slow Down.
	0.8-1.0	WARNING! Obstacle _____ m Ahead. Apply Brake level 1.
	0.6-0.8	WARNING! Obstacle _____ m Ahead. Apply Brake level 2.
	0.4-0.6	WARNING! Obstacle _____ m Ahead. Prepare to Stop.
	0.2-0.4	WARNING! Obstacle _____ m Ahead. Apply Full Brakes & Stop
	0.0-0.2	WARNING! Obstacle _____ m Ahead. Apply Emergency Brakes and Stop Immediately.
30-50	1.0-1.2	WARNING! Obstacle _____ m Ahead. Maintain Current Speed.
	0.8-1.0	WARNING! Obstacle _____ m Ahead. Apply Brake level 2.
	0.6-0.8	
	0.4-0.6	WARNING! Obstacle _____ m Ahead. Prepare to Stop.
	0.2-0.4	WARNING! Obstacle _____ m Ahead. Apply Full Brakes & Stop



	0.0-0.2	WARNING! Obstacle ____ m Ahead. Apply Emergency Brakes and Stop Immediately.
10-30	1.0-1.2	WARNING! Obstacle ____ m Ahead. Maintain Current Speed.
	0.8-1.0	
	0.6-0.8	
	0.4-0.6	
	0.2-0.4	WARNING! Obstacle ____ m Ahead. Apply Brake level 1.
	0.0-0.2	WARNING! Obstacle ____ m Ahead. Apply Full Brakes and Stop.
<b>IF INCOMING MESSAGE = OBS-010, OBS-001</b>		
60-80	0.0-1.2	WARNING! Obstacle Detected. Reduce Speed
50-60	0.6-1.2	WARNING! Obstacle Detected. Maintain Current Speed.
	0.0-0.6	WARNING! Obstacle Detected. Reduce Speed
10-50	0.0-1.2	WARNING! Obstacle Detected. Maintain Current Speed.

# CHAPTER 7 –THE LEVEL CROSSING SYSTEM

## 7.1 – Hardware Configuration

### 7.1.1 – Single Core Technology

Unlike the train system, the Level Crossing System comprises of a single core design where one microcontroller unit acts as the core of the system. All other components are connected and controlled by this microcontroller. The reason of using single core technology in the Level Crossing system is that it doesn't need to get GPS data and upload it to the server continuously rather the GPS data of all level crossings are preliminary set in all Train Systems and the Central Control System. The level crossing has one major task to accomplish which is getting SMS messages from the trains and CCS and acting according to its instruction.

The block diagram of the level crossing system is shown below.

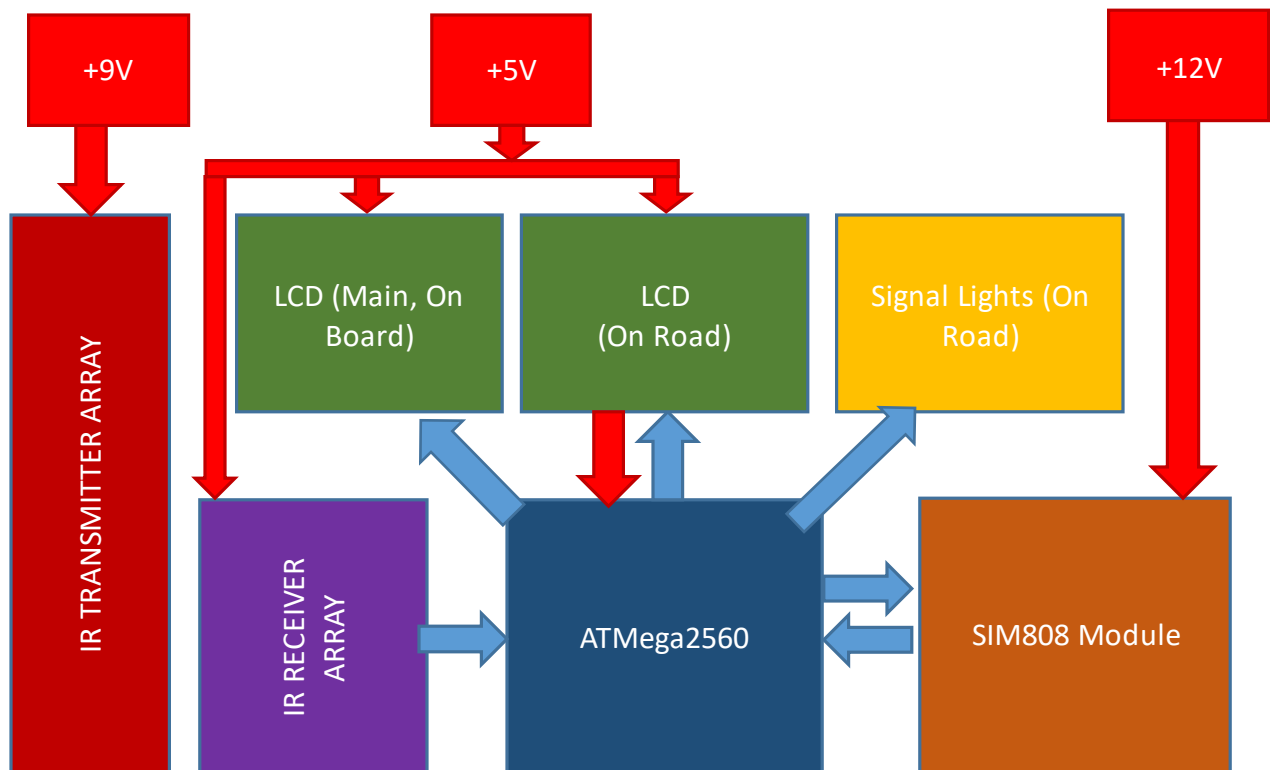


Figure 7.1 Block Diagram of the Level Crossing System

### 7.1.2 – IR Transmitter Circuit

Normally IR transmitters transmit IR to a very short range like 2 meters at maximum. We designed a circuit through extensive research work for the IR transmitters to transmit IR rays up to 10 meters range.

This Long range Infrared transmitter can emit pulsed IR rays up to 10 meters. This IR transmitter is ideal to use in Infrared receivers using Phototransistor or Photodiode as IR sensor.

The circuit is a simple Astable Multivibrator using IC NE555. Resistors R1, R2 and capacitor C1 fix the output frequency to 38 kHz. T1 is the Darlington high gain PNP transistor that drives two Infrared LEDs. Resistor R4 is the pull up resistor that keeps the base of T1 high for its proper working. By changing R2 with a 4.7K preset, output frequency can be changed. Red LED indicates whether the IR LED is working or not.

The circuit schematic is shown below:

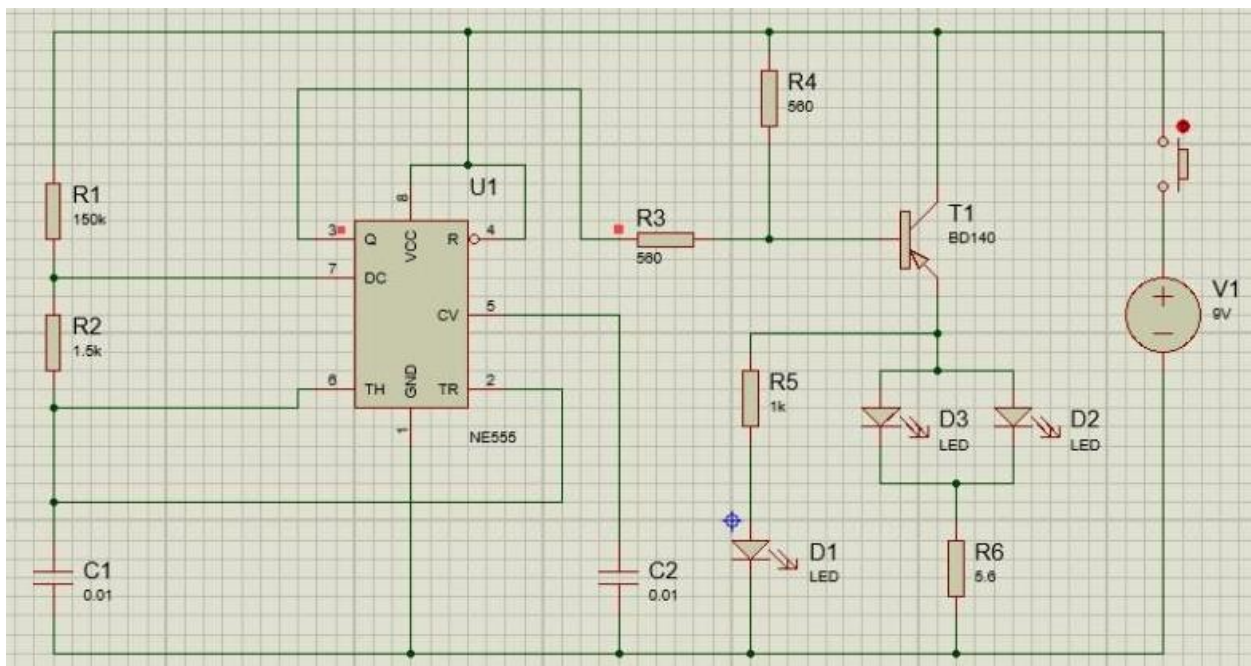


Figure 7.2 Circuit Schematic of Long Range IR Transmitter

An array of twelve copies of this circuit was created as we have twelve positions of obstacle detection (*explained later*). This circuit uses a 9V DC power to operate and is not connected with the microcontroller unit as it does not need to be controlled by the microcontroller.

### 7.1.3 – IR Receiver Circuit

We used TSOP1738 IR receivers in the receiver circuit so that it receives only IR rays of frequency 38 kHz and no other rays. This makes the system unique, safe and efficient as the obstacle detection process will not be affected by any external factors. The IR receiver circuits pulse pins are connected to the Analog pins of the microcontroller.

IR Receiver circuit is very simple. We just need to connect a LED to the output of the TSOP1738, to test the receiver. We have use BC557 PNP transistor here, to reverse the effect of TSOP, means whenever the output is HIGH LED will be OFF and whenever it detects IR and output is low, LED will be ON. PNP transistor behaves opposite to the NPN transistor, it acts as open switch when a voltage applied to its base and acts as closed switch when there is no voltage at its base. So normally TSOP output remains HIGH and Transistor behaves as open switch and LED will be OFF. As soon as TSOP detects Infrared, its output becomes low and transistor behaves as closed switch and LED will be ON. A 10k resistor is used for provide proper biasing to transistor and a 470ohm resistor is used at LED for limiting the current. So whenever we switch the IR transmitter ON, it is detected by TSOP1738 and LED will glow.

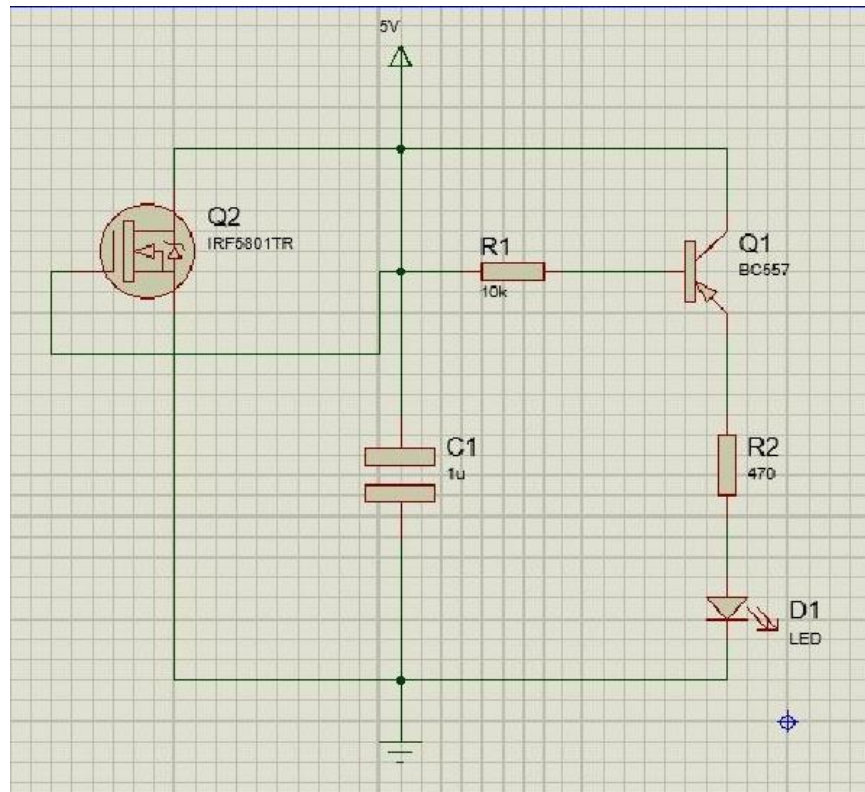


Figure 7.3 IR Receiver Circuit Schematic

### 7.1.4 – Peripheral Devices

The peripheral devices that are involved within the Level Crossing System are used to indicate whether a train is arriving to that level crossing or not.

LCD on board shows gate status of the level crossing and any new messages either from train or the CCS.

Two LCDs on the road indicates the traffic whether to ‘Go’ or to ‘Stop’. These LCDs normally displays ‘Go’. Whenever a train is approaching it changes to display ‘Stop, Train Approaching’.

All LCDs are connected to the microcontroller and has its own dedicated potentiometers to control the brightness.

The keypad is connected to the microcontroller for the ease of choosing and sending messages quickly from the Menu.

The ‘Menu’ button displays the Menu on LCD on board.

The most important components are the servos which are responsible for opening and closing of the gates at the level crossing.

Two sets of two LEDs were used as road signals and two sets of three LEDs were used as train signals. Four LEDs for four gates were connected to know the status of the level crossing gates.

A buzzer was connected as an alarm if a train is arriving that particular level crossing.

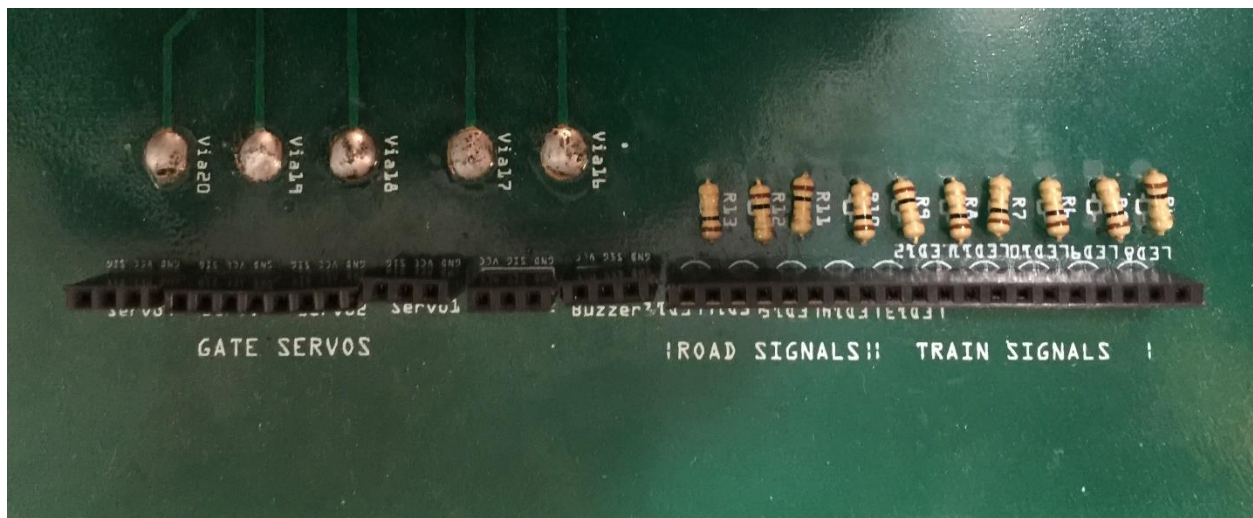


Figure 7.4 Peripheral Devices ports on Level Crossing Mainboard

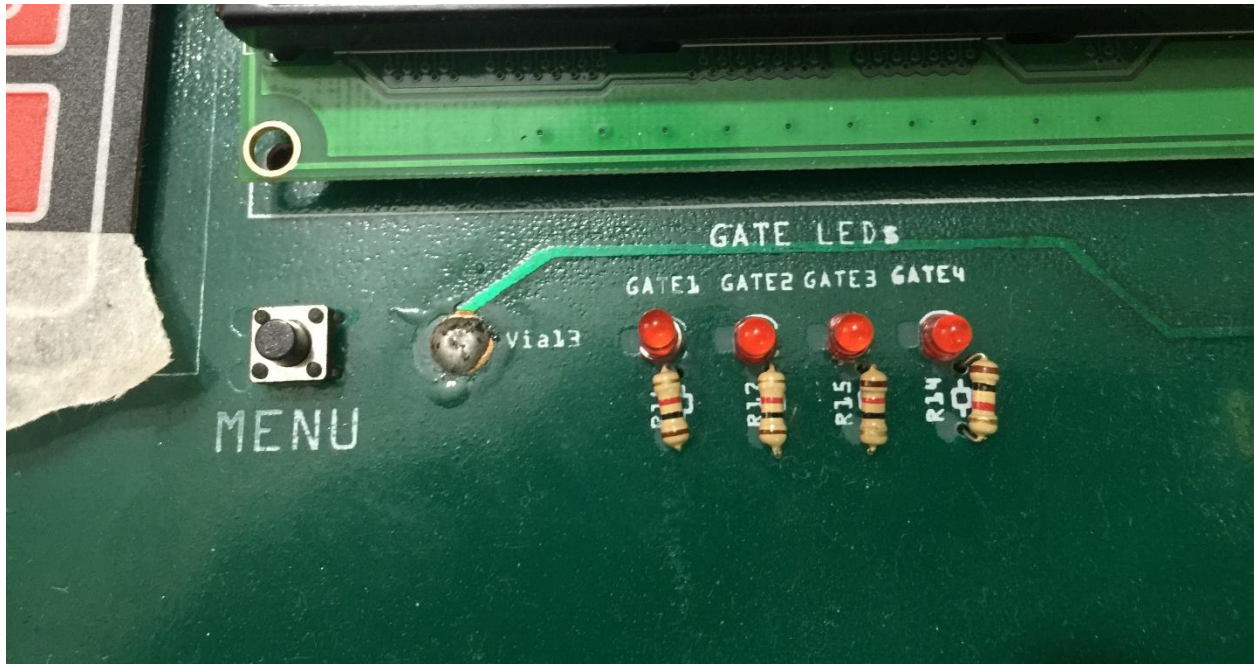


Figure 7.5 Menu button and Gate LEDs on the Level Crossing Mainboard

### 7.1.5 – Power Supply

As stated before, the Level Crossing System uses three types of power sources which are, 9V, 5V and 12V. 9V DC power source is used only by the IR Transmitter circuit, 12V source is used for the SIM808 module and 5V for all other devices and components. The Level Crossing mainboard is supplied with 12V, 3A DC Power adapter and then converted to 9V and 5V by using two LM2596 Step-down buck converter respectively. This was how three separate sources of power were supplied on the Level Crossing Mainboard.

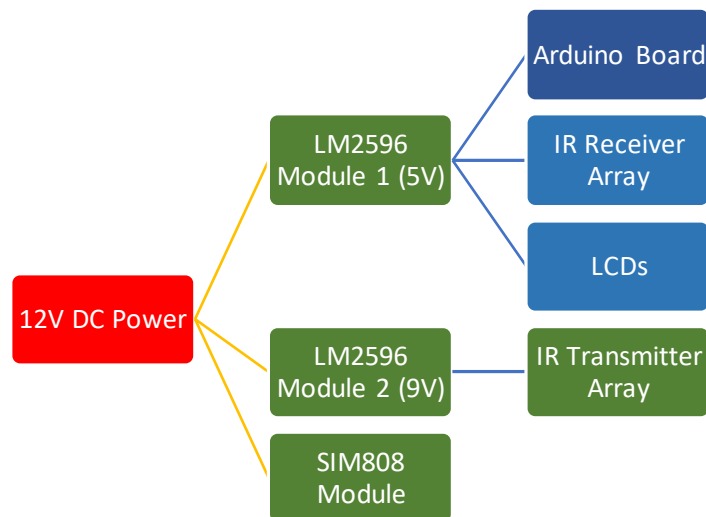


Figure 7.6 Power Distribution diagram in the Level Crossing Mainboard

## 7.2 – Code Algorithms

### 7.2.1 – Train Detection Algorithm

**Send or Receive SMS:** The primary task of the level crossing core is to send or receive messages from other systems. SMS that has been received, is deleted automatically after it has been read by the user. This is done so that the message memory of the SIM doesn't get full over time and block new SMS from incoming. This reduces the chances of miscommunication and the probability of any hazard. The AT Commands that were used to send and receive SMS messages are given in the following table.

Command	Description
AT+CMGD	Delete SMS message
AT+CMGF	Select SMS message format
AT+CMGL	List SMS messages from preferred store
AT+CMGR	Read SMS message
AT+CMGS	Send SMS message
AT+CMGW	Write SMS message to memory
AT+CMSS	Send SMS message from storage
AT+CNMI	New SMS message indications
AT+CPMS	Preferred SMS message storage
AT+CRES	Restore SMS settings
AT+CSAS	Save SMS settings
AT+CSCA	SMS service center address
AT+CSCB	Select cell broadcast SMS messages
AT+CSDH	Show SMS text mode parameters
AT+CSMP	Set SMS text mode parameters
AT+CSMS	Select message service

Table 7.1 AT Commands used to send or Receive SMS Messages

**Extraction of Received SMS Messages:** Upon sending the AT Command, AT+CNMI=<mode>[,<mt>[,<bm>[,<ds>[,<bfr>]]]] notification of any new SMS messages are received. SMS Messages that are received are displayed in the format below:

+CMT: <oa>, <scts>[,<toa>,<fo>,<pid>,<dcs>,<sca>,<tosca>,<length>]<CR><LF><data>

## Example Output:

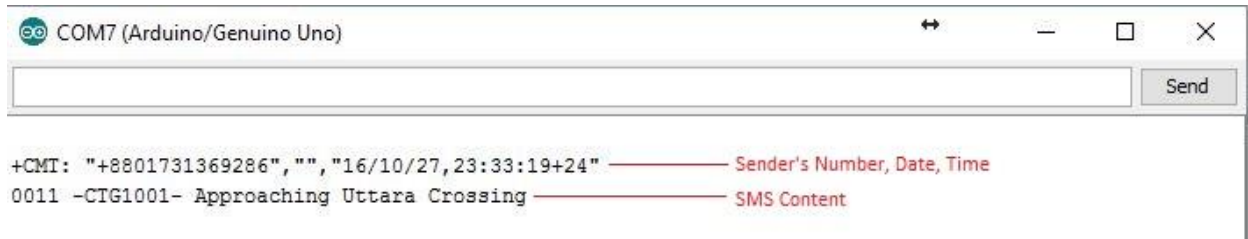


Figure 7.7 Screenshot of a Received SMS Raw Output

Extraction of SMS is done by the 'Comma Detection' Algorithm. This means every content between each comma are extracted and stored in a separate variable. The parameters that are extracted from a SMS raw output are:

- Sender's Number
- Date & Time at which the SMS was received
- SMS Message content

These variables are then used to perform different other tasks.

SMS Extraction Code is as follows:

```
boolean ExtractSMSdata(String value) {
    debugPrintln("Extracting SMS contents from the provided value:");
    int start = 0; //Start position of trim
    int counter = 0; //Number of commas detected

    for (int i = 0; i < value.length(); i++) { //Start searching for commas to split String

        if (value[i] == ',' || i == value.length() - 1) { //For detection of comma or for the last position of the String
            debugPrintln("Comma detected");
            String text = ""; //Text extracted from SMS response
            debugPrintln("Extracting the content:");
            for (int j = start; j <= i; j++) { //Split it from Starting point till current position
                if (value[j] != ',' && value[j] != "" && value[j] != '/' && value[j] != '\\' && value[j] != '\n') text = text + value[j]; //Add each character to
the result
            }
            else{
                if(value[j] == '/' || value[j] == '\\'){
                    text = text + '-';
                }
            }
        }
    }
}
```



```

    }
    debugPrint("Content Extracted: ");
    debugPrintln(text);

    if (counter == 2) {
        debugPrintln("Sender Number recieved");
        sim_number = text; //1st comma indicates Sender Number
    }
    else if (counter == 4) {
        debugPrintln("Date recieved");
        sms_date = text;
    }
    else if (counter == 5) {
        debugPrintln("Time recieved");
        sms_time = text;
    }
    else if (counter == 6) {
        debugPrintln("Message recieved");
        sms_message = text;
        counter++; //Increase counter
        start = i; //Set next starting point of trim
    }
}
debugPrint("Total Data recieved: ");
debugPrintln(counter);
if (counter >= 5) {
    debugPrintln("SMS contents extracted");
    return true;
}
else {
    debugPrintln("SMS contents not extracted");
    return false;
}
}
}

```

We know that the Level Crossing System is always searching for new SMS messages. Whenever a new message is received, it checks whether the message is from a train or the CCS from its own database where the Train Name, Train ID, and SIM Numbers are saved in three different arrays.

The following code shows the database of the Level Crossing System.

```

//Information about trains
String trainNameList[] ={"SUB-701", "EKO-705", "TIS-707"};
String trainNumberList[] = {"+8801780339963", "+8801731369286", "+8801676964718"};
int trainAmount = 3;

```

```

String CCS[] ={"CENTRAL CONTROL SYSTEM"};
String CCSNumber[] = {"+8801670669566"};
int CCSAmount = 1;

//Info About Level Crossing
String lcName = "Mohakhali"; //01777549538
String lcKey = "83856645554849";//Demo Key of SUB-701

//Message Codes
int ccsCodeArraySize = 16;
String CCSMessageCodes[] = {"HLT-001", "HLT-111", "HLT-000", "RES-111", "BRK-111", "ENG-111",
"PPP-111", "STA-111", "STO-111", "DUR-111", "MED-111", "ACC-111", "HJK-111", "FIR-111", "LAW-111",
"SER-111"};

int trainCodeArraySize = 16;
String trainCodes[] = {"HLT-001", "HLT-111", "HLT-000", "RES-111", "BRK-111", "ENG-111", "PPP-111",
"STA-111", "STO-111", "DUR-111", "MED-111", "ACC-111", "HJK-111", "FIR-111", "LAW-111", "SER-
111"};

```

### 7.2.2 – Obstacle Detection Algorithm

Obstacle is detection is done by reading each of the receiver's values that are connected to the analog pins of the microcontroller. Normally when no obstacle is present the reading stays at a value of 1020 to 1024. Upon placing an obstacle in between the transmitter and receiver, the value goes down and ranges from 500 to 800.

The sensor values were stored continuously in an array. The code for initialization is shown below:

```

//Sensor Values
int sensors[] = {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1};
boolean sensorObstacles[] = {false, false, false, false, false, false, false, false, false, false, false, false, false};
int sensorArraySize = 13;

```

The code for obstacle detection is as follows:

```
//Read Sensor Values
boolean getSensorValue(){
  Serial.println("Waiting to clear the obstacle");
  delay(5000);

  sensors[1] = analogRead(A11);
  sensors[2] = analogRead(A10);
  sensors[3] = analogRead(A9);
  sensors[4] = analogRead(A8);
  sensors[5] = analogRead(A7);
  sensors[6] = analogRead(A6);
  sensors[7] = analogRead(A5);
  sensors[8] = analogRead(A4);
  sensors[9] = analogRead(A3);
  sensors[10] = analogRead(A2);
  sensors[11] = analogRead(A1);
  sensors[12] = analogRead(A0);

  for(int i=1; i < sensorArraySize; i++){//Checking Obstacle
    Serial.print("Checking for obstacle in sensor - ");
    Serial.println(sensors[i]);
    if(sensors[i] < 10){
      Serial.print("Sensor Value: ");
      Serial.print(i);
      Serial.print(" : ");
      Serial.println(sensors[i]);
      delay(2000);
      sensorObstacles[i] = true;
    }
  }
  return true;
}
//Checks for Obstacle
void checkObstacle(){
  int obstacles = 0;
  for(int i=1; i < sensorArraySize; i++){
    if(sensorObstacles[i] == true){
      obstacles++;
    }
  }
  if(obstacles == 0){
    Serial.println("No Obstacles. Train can proceed now");
  }
  else{

    if(sensorObstacles[1] || sensorObstacles[3] || sensorObstacles[5] || sensorObstacles[7] || sensorObstacles[9] ||
    sensorObstacles[11]){//Left Side

      if(sensorObstacles[3] || sensorObstacles[5] || sensorObstacles[7] || sensorObstacles[9]){//Obstacle in Track
        if(trainTrackInSMS == "A"){//Train is in track A
          if(sensorObstacles[3] || sensorObstacles[5]){//Obstacle in track A
            Serial.println("Obstacle in track A and so are you. Stop, Idiot!");
          }
        }
      }
    }
  }
}
```

```

    }
    else{
        Serial.println("Obstacle in track B and you are in track A. Go Slow!");
    }
}
else{//Train is in track B
    if(sensorObstacles[3] || sensorObstacles[5]){//Obstacle in track A
        Serial.println("Obstacle in track A and you are in track B. Go Slow!");
    }
    else{
        Serial.println("Obstacle in track B and so are you. Stop, Idiot!");
    }
}

}
else{//Obstacle in Gate
    Serial.println("Obstacle is in left gate. Gate 1 and 3 are kept open. Go slow!");
}
}
else if(sensorObstacles[2] || sensorObstacles[4] || sensorObstacles[6] || sensorObstacles[8] || sensorObstacles[10]
|| sensorObstacles[12]){//Right Side

    if(sensorObstacles[4] || sensorObstacles[6] || sensorObstacles[8] || sensorObstacles[10]){//Obstacle in Track
        if(trainTrackInSMS == "A"){//Train is in track A
            if(sensorObstacles[4] || sensorObstacles[6]){//Obstacle in track A
                Serial.println("Obstacle in track A and so are you. Stop, Idiot!");
            }
            else{
                Serial.println("Obstacle in track B and you are in track A. Go Slow!");
            }
        }
        else{//Train is in track B
            if(sensorObstacles[8] || sensorObstacles[10]){//Obstacle in track A
                Serial.println("Obstacle in track A and you are in track B. Go Slow!");
            }
            else{
                Serial.println("Obstacle in track B and so are you. Stop, Idiot!");
            }
        }
    }
}
else{//Obstacle in Gate
    Serial.println("Obstacle is in Right gate. Gate 1 and 3 are kept open. Go slow!");
}
}
}
}
}

```

### 7.2.3 – Road Controlling Algorithm

Liquid Crystal library was used to control the LCDs. A separate method was written to print a text to the LCD. Example code:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd_main(50, 52, 53, 51, 49, 47); //Level Crossing LCD
void displayInLCDLine1(LiquidCrystal lcd, String text){

    lcd.setCursor(0, 0);
    lcd.print(text);
    delay(2000);

}
void displayInLCDLine2(LiquidCrystal lcd, String text){

    lcd.setCursor(0, 1);
    lcd.print(text);
    delay(2000);

}
void displayInLCDLine3(LiquidCrystal lcd, String text){

    lcd.setCursor(0, 2);
    lcd.print(text);
    delay(2000);

}
void displayInLCDLine4(LiquidCrystal lcd, String text){

    lcd.setCursor(0, 3);
    lcd.print(text);
    delay(2000);

}
```

LEDs and buzzers were controlled using the ‘digitalWrite() command’. The function of this command is to write a HIGH or a LOW value to a digital pin.

If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V for HIGH, 0V (ground) for LOW.

If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin. It is recommended to set the pinMode() to INPUT\_PULLUP to enable the internal pull-up resistor.

NOTE: If you do not set the pinMode() to OUTPUT, and connect an LED to a pin, when calling digitalWrite(HIGH), the LED may appear dim. Without explicitly setting pinMode(), digitalWrite() will have enabled the internal pull-up resistor, which acts like a large current-limiting resistor.

#### Syntax

digitalWrite(pin, value)

### Parameters

pin: the pin number

value: HIGH or LOW

**Interrupt:** The system is also responsible for controlling the action of the ‘Menu’ button. Input from the button is taken using the function ‘attachInterrupt ()’.

The first parameter to attachInterrupt is an interrupt number. Normally we should use digitalPinToInterrupt(pin) to translate the actual digital pin to the specific interrupt number. For example, if we connect to pin 3, use digitalPinToInterrupt(3) as the first parameter to attachInterrupt.

Digital Pins Usable For Interrupts: 2, 3, 18, 19, 20, and 21

Inside the attached function, delay() won't work and the value returned by millis() will not increment. Serial data received while in the function may be lost. Therefore we declared any variables as volatile that were modified within the attached function.

### Syntax

```
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);
```

```
attachInterrupt(interrupt, ISR, mode)
```

```
attachInterrupt(pin, ISR, mode) ;
```

### Parameters

interrupt: the number of the interrupt (int)

pin: the pin number

ISR: the ISR to call when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.

mode: defines when the interrupt should be triggered. Four constants are predefined as valid values:

LOW to trigger the interrupt whenever the pin is low,

CHANGE to trigger the interrupt whenever the pin changes value

RISING to trigger when the pin goes from low to high,

FALLING for when the pin goes from high to low.

## Code: [Note: Full code is not provided due to copyright reasons.]

level\_crossingSystem\_v3.1 | Arduino 1.8.1

File Edit Sketch Tools Help

```
level_crossingSystem_v3.1
1 #include <SoftwareSerial.h>
2 #include <LiquidCrystal.h>
3 //Initializing Global Variables - START
4 SoftwareSerial mySerial(10, 11); //RX, TX of GSM Module
5 #include <Keypad.h>
6 #include <Servo.h>
7 // initialize the library with the numbers of the interface pins
8 LiquidCrystal lcd_1(48, 43, 41, 39, 37, 35); //LCD-2
9 LiquidCrystal lcd_1(48, 43, 41, 39, 37, 35); //LCD-2
10 LiquidCrystal lcd_main(50, 52, 53, 51, 49, 47); //Level Crossing LCD
11
12 //Info About Level Crossing
13 String lcdName = "Mokashali"; //0177548588
14 String lcdKey = "83856455449"; //Demo Key of SUB-701
15 //Message Codes
16 int ccCodeArraySize = 16;
17 String CCMessageCodes[] = {"HLT-001", "HLT-111", "HLT-000", "RES-111", "BRK-111", "ENG-111", "PPP-111", "STA-111", "STO-111", "DUR-111", "MED-111", "ACC-111", "BRK-111", "FIR-111", "LAW-111", "SER-111"};
18
19 int trainCodeArraySize = 16;
20 String trainCodes[] = {"HLT-001", "HLT-111", "HLT-000", "RES-111", "BRK-111", "ENG-111", "PPP-111", "STA-111", "STO-111", "DUR-111", "MED-111", "ACC-111", "BRK-111", "FIR-111", "LAW-111", "SER-111"};
21
22 //Control for Servo
23 int pos = 0; // variable to store the servo position
24 Servo gate1; // create servo object to control a servo
25 Servo gate2; // create servo object to control a servo
26 Servo gate3; // create servo object to control a servo
27 Servo gate4; // create servo object to control a servo
28
29 boolean isGateOpen[] = {false, false, false, false};
30
31 //Controls for LED (LED No- 14-17)
32 const int redLED1 = 8;
33 const int greenLED1 = 1;
34 const int redLED2 = 2;
35 const int greenLED2 = 3;
36
37
38 int gate1Bar = 15;
39 int gate4Bar = 17;
40 int gate2Bar = 16;
41 int gate3Bar = 19;
42
43 //Keypad for Menu
44 const byte ROWS = 4; //four rows
45 const byte COLS = 4; //four columns
46 //define the symbols on the buttons of the keypad
47 char hexaKeys[ROWS][COLS] = {
48   {'1','4','7','*'},
49   {'2','5','8','0'},
50   {'3','6','9','#'},
51   {'A','B','C','D'}
52 };
53 byte rowPins[ROWS] = {46, 48, 50, 52}; //connect to the row pinouts of the keypad
54 byte colPins[COLS] = {38, 40, 42, 44}; //connect to the column pinouts of the keypad
55
56 //initialize an instance of class NewKeypad
57 Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
58
59
60 //Variables for SMS
61 String sim_number;
62 String sms_date;
63 String sms_time;
64 String sms_message;
65
66 //SMS upload Control
67 boolean newText=false;
68 String SMSmessage="";
69
70 //Menu Controls
71 boolean menuPressed = false; //True when Interrupt is received
72 const int menuButton = 20;
73
74 //Information about trains
75 String trainNameList[] = {"SUB-701", "ERO-705", "TIS-707"};
76 String trainNumberList[] = {"+8801780339943", "+8801731369286", "+8801676964718"};
77 int trainAmount = 3;
78
79 String CCS[] = {"CENTRAL CONTROL SYSTEM"};
80 String CCNumber[] = {"+8801670669564"};
81 int CCSAmount = 1;
82
83 String trainTrackInSMS=""; //Stores track number of the train
84 String msgCodeTxInSMS=""; //Stores message code sent by train
85 String msgCodeNumberInSMS=""; //Stores message code sent by train
86 String trainNameTxInSMS = ""; //Stores Train Name
87 String trainCodeTxInSMS = ""; //Stores Train Code
88 String trainNumberInSMS = "";
89
90 //Sensor Values
91 int sensors[] = {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1};
92
93 boolean sensorObstacles[] = {false, false, false, false, false, false, false, false, false, false, false};
94 int sensorArraySize = 13;
95
96 boolean obstacleCheck = false;
97
98 //Gate Sensors: 1-G1, 2-G2, 11-G3, 12-G4
99 //Train Track Sensors: 3,4,5,6-TA, 7,8,9,10-TB
100 boolean debugMode=false; //Controller of Debug Mode [Default: OFF]
101 void setup() {
102   // put your setup code here, to run once:
103   Serial.begin(9600); // Setting the baud rate of Serial Monitor (Arduino)
104   mySerial.begin(9600); // Setting the baud rate of GSM Module
105
106   //Starting LCD Monitor
107   lcd_main.begin(20, 4);
108   lcd_2.begin(20, 4);
109   turnDebugOff(); //Turning Debug Mode OFF: turnDebugOff() || Turning Debug Mode ON: turnDebugOn()
110   debugPrintln("Debug Mode is turned ON. Turn it off to stop viewing these messages.");
111
112   debugPrintln("Executing AT commands to set up the GSM Module");
113   debugPrintln("");
114   debugPrintln("Communicating with GSM Module");
115
116   mySerial.write("AT\r"); //Communicating with GSM
117   delay(1000);
118   ShowSerialData();
119
120   debugPrintln("Setting Default Configurations of GSM");
```

```

1117 //Get Specific Info from SMS
1118 boolean extractSMSData(String value) {
1119     debugPrintln("Extracting SMS contents from the provided value");
1120     int start = 0; //Start position of trim
1121     int counter = 0; //Number of commas detected
1122
1123     for (int i = 0; i < value.length(); i++) { //Start searching for commas to split String
1124
1125         if (value[i] == ',' || i == value.length() - 1) { //For detection of comma or for the last position of the String
1126             debugPrintln("Comma detected");
1127             String text = "" //Text extracted from GPS response
1128             debugPrintln("Extracting the contents");
1129             for (int j = start; j <= i; j++) { //Split it from starting point till current position
1130                 if (value[j] != ',' || value[j] != '\n') { //If '\n' || value[j] != '\n' text = text + value[j]; //Add each character to the result
1131                     else{
1132                         if(value[j] == '/' || value[j] == '\\'){
1133                             text = text + "-";
1134                         }
1135                     }
1136                 }
1137             }
1138             debugPrint("Content Extracted: ");
1139             debugPrintln(text);
1140
1141             if (counter == 2) {
1142                 debugPrintln("Sender Number received");
1143                 sim_number = text; //1st comma indicates Sender Number
1144             }
1145             else if (counter == 4) {
1146                 debugPrintln("Date received");
1147                 sms_date = text; //4th Comma comes after Latitude in GPS response
1148             }
1149             else if (counter == 5) {
1150                 debugPrintln("Time received");
1151                 sms_time = text; //5th Comma comes after Longitude in GPS response
1152             }
1153             else if (counter == 6) {
1154                 debugPrintln("Message received");
1155                 sms_message = text; //6th Comma comes after altitude in GPS response
1156             }
1157             counter++; //Increase counter
1158             start = i; //Set next starting point of trim
1159         }
1160     }
1161     debugPrint("Total Data received: ");
1162     debugPrintln(counter);
1163     if (counter >= 5) {
1164         debugPrintln("SMS contents extracted");
1165         return true;
1166     }
1167     else {
1168         debugPrintln("SMS contents not extracted");
1169         return false;
1170     }
1171 }
1172
1173 //Displays Menu
1174 void displayMenu() {
1175     Serial.println("Menu Switch is pressed. Please wait while the current task is being executed");
1176     lcd_main.clear();
1177     lcd_main.setCursor(0,0);
1178     lcd_main.print("PLEASE WAIT");
1179     menuPressed = true;
1180 }
1181
1182 //Extracts Train ID and contents from the train's SMS
1183 boolean extractMessageCode(String data) {
1184     int dashCounter = 0;
1185     int start = 0;
1186     for (int i=0; i < data.length(); i++) {
1187         if (data[i] == '-' || i == data.length() - 1) {
1188             String content = "";
1189             for (int j = start; j <= i; j++) {
1190                 if (data[j] != '-' || content != data[j]) {
1191                     content += data[j];
1192                 }
1193             }
1194             if (dashCounter == 0) {
1195                 Serial.print("Message Code Text Extracted: ");
1196                 msgCodeTxtInSMS = content;
1197             }
1198             else if (dashCounter == 1) {
1199                 Serial.print("Message Code Text Extracted: ");
1200                 trainNameTxtInSMS = content;
1201             }
1202             else if (dashCounter == 2) {
1203                 Serial.print("Message Code Text Extracted: ");
1204                 trainCodeTxtInSMS = content;
1205             }
1206             else if (dashCounter == 3) {
1207                 Serial.print("Message Code Text Extracted: ");
1208                 trainTrackInSMS = content;
1209             }
1210             else if (dashCounter == 4) {
1211                 Serial.print("Message Code Text Extracted: ");
1212                 msgCodeNumberInSMS = content;
1213             }
1214             dashCounter++;
1215             Serial.println(content);
1216             start = i;
1217         }
1218     }
1219     if (dashCounter == 5) {
1220         Serial.println("Contents extracted successfully");
1221         Serial.println("Train Name: " + trainNameTxtInSMS + "-" + trainCodeTxtInSMS);
1222         Serial.println("Train Track: " + trainTrackInSMS);
1223         Serial.println("Message Code: " + msgCodeTxtInSMS + "-" + msgCodeNumberInSMS);
1224         if (sim_number != "") {
1225             trainNumberInSMS = sim_number;
1226             Serial.println("Train's Number: " + trainNumberInSMS);
1227         }
1228         return true;
1229     }
1230     else {
1231         Serial.println("Contents could not be extracted successfully");
1232         Serial.println("Train Name: " + trainNameTxtInSMS + "-" + trainCodeTxtInSMS);
1233         Serial.println("Train Track: " + trainTrackInSMS);
1234         Serial.println("Message Code: " + msgCodeTxtInSMS + "-" + msgCodeNumberInSMS);
1235     }
1236     return false;
1237 }
1238
1239 //Checks if SMS is from train
1240 boolean isFromTrain() {
1241     if (sim_number.length() > 0) {
1242         for (int i=0; i < trainAmount; i++) {
1243             if (trainNumberList[i] == sim_number) {
1244                 return true;
1245             }
1246         }
1247     }
1248     return false;
1249 }
1250
1251 //Checks if SMS is from CCS
1252 boolean isFromCCS() {
1253     if (sim_number.length() > 0) {
1254         for (int i=0; i < CCSAmount; i++) {
1255             if (CCSNumber[i] == sim_number) {
1256                 return true;
1257             }
1258         }
1259     }
1260     return false;
1261 }

```



## 7.3 – PCB Design

### 7.3.1 – IR Transmitter Array

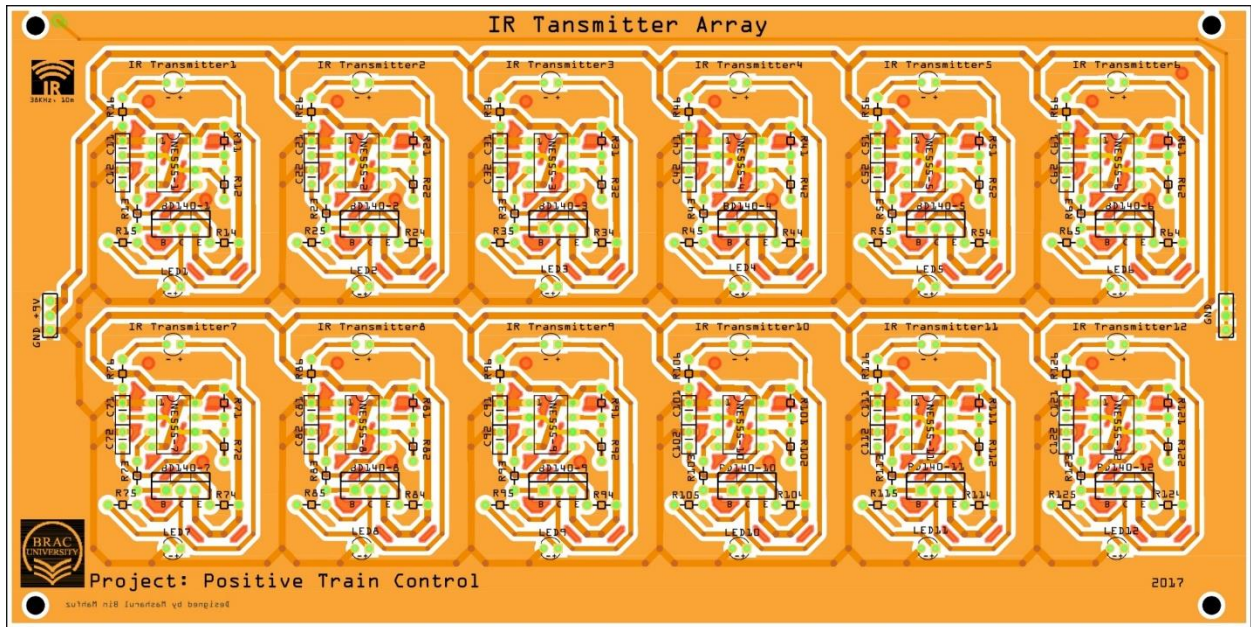


Figure 7.8 PCB Design of IR Transmitter Array

### 7.3.2 – IR Receiver Array

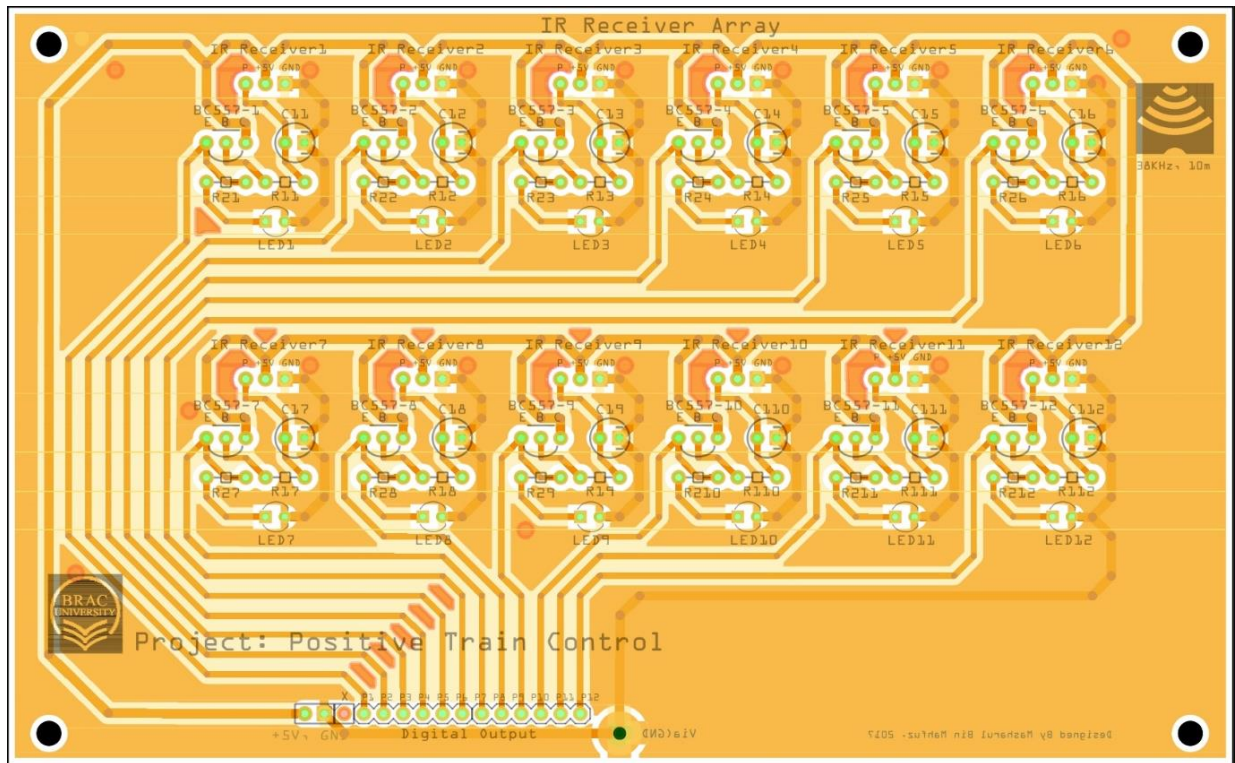


Figure 7.9 PCB Design of IR Receiver Array

### 7.3.3 – Level Crossing Mainboard

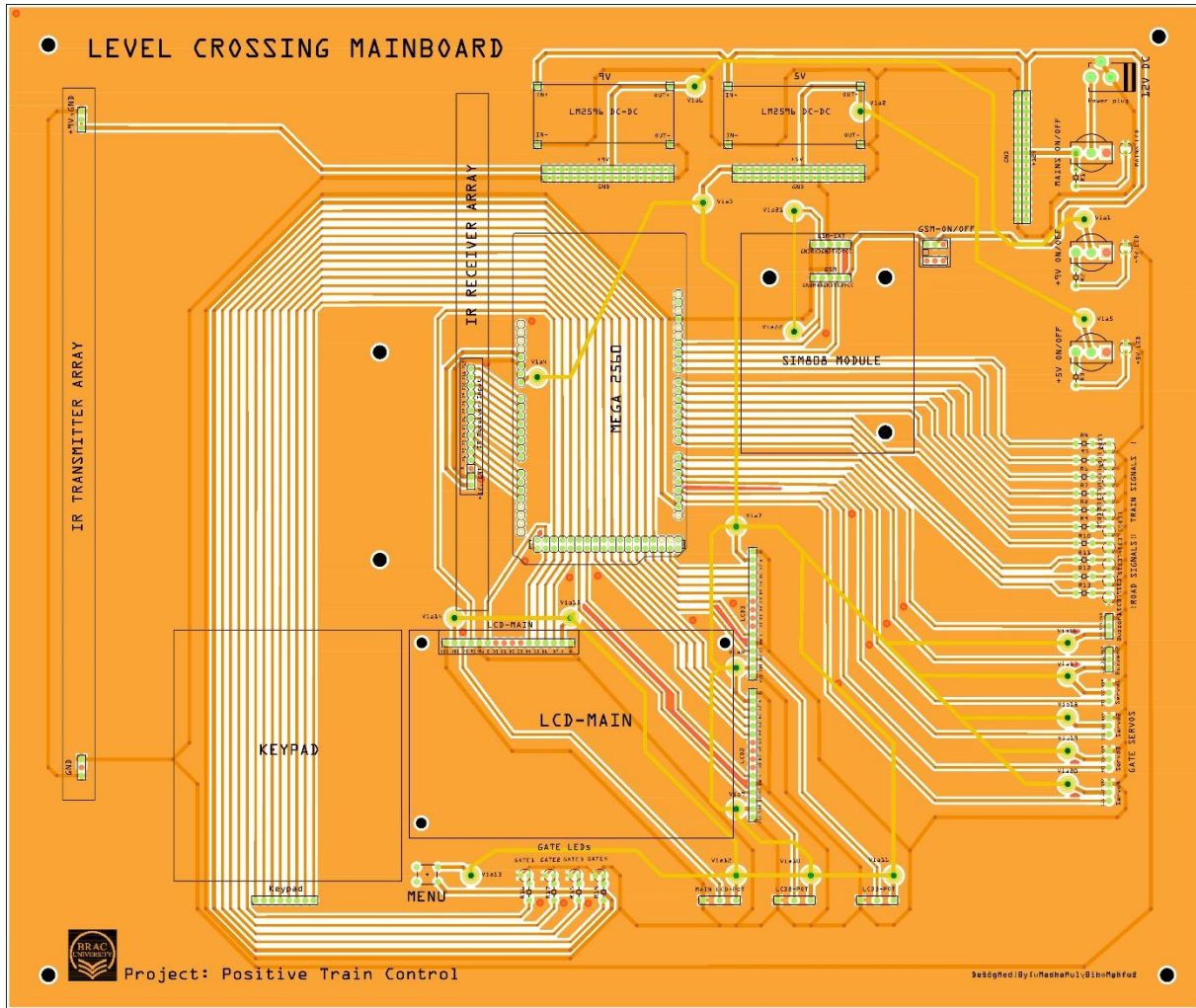


Figure 7.10 PCB Design of Level Crossing Mainboard

## **7.4 – Working Principle**

### **7.4.1 – The Train Detection Mechanism**

Whenever a train is approaching the level crossing and is within 1.5KM radius of the level crossing it sends a code message to the level crossing indicating its arrival and informing the level crossing with its ID, SIM number and track number. It is important for the level crossings to know the track number of the train to know the direction of approach, as in Dhaka two tracks are used for train transportation. This is how the level crossing knows that a train is approaching towards it.

### **7.4.2 – Obtaining and Storing Train Information**

The information of the train is obtained by two methods, i) cross matching the SIM number that the SMS was received from with the own database of the level crossing where the train SIM numbers and names are stored. ii) by the extraction of the message code sent by the train to the level crossing. The message code sent by the train is in the following format:

“APP-<TRAIN ID>-<TRACK NO>-0001”

The SIM number is saved in another variable to establish a communication between the train and the level crossing only until the period the train is within the level crossing region.

### **7.4.3 – Sensor Positions for Obstacle Detection**

As soon as the level crossing knows the information about an arriving train, it then checks for obstacles. Obstacles are checked at twelve positions on the level crossing. The positions are indicated on the diagram on the next page. The red arrows labelled S1 to S12 are the IR receivers. The IR Transmitters will be placed parallel to each receiver in the road dividers, facing to the receiver’s way. Sensors 1, 2, 11 and 12 are responsible for detecting obstacles under or parallel to the gates 1, 2, 3 and 4 respectively. Sensors 3, 4, 5 and 6 are responsible for detecting obstacles on Track A and sensors 7, 8, 9 and 10 are responsible for detecting obstacles on Track B.

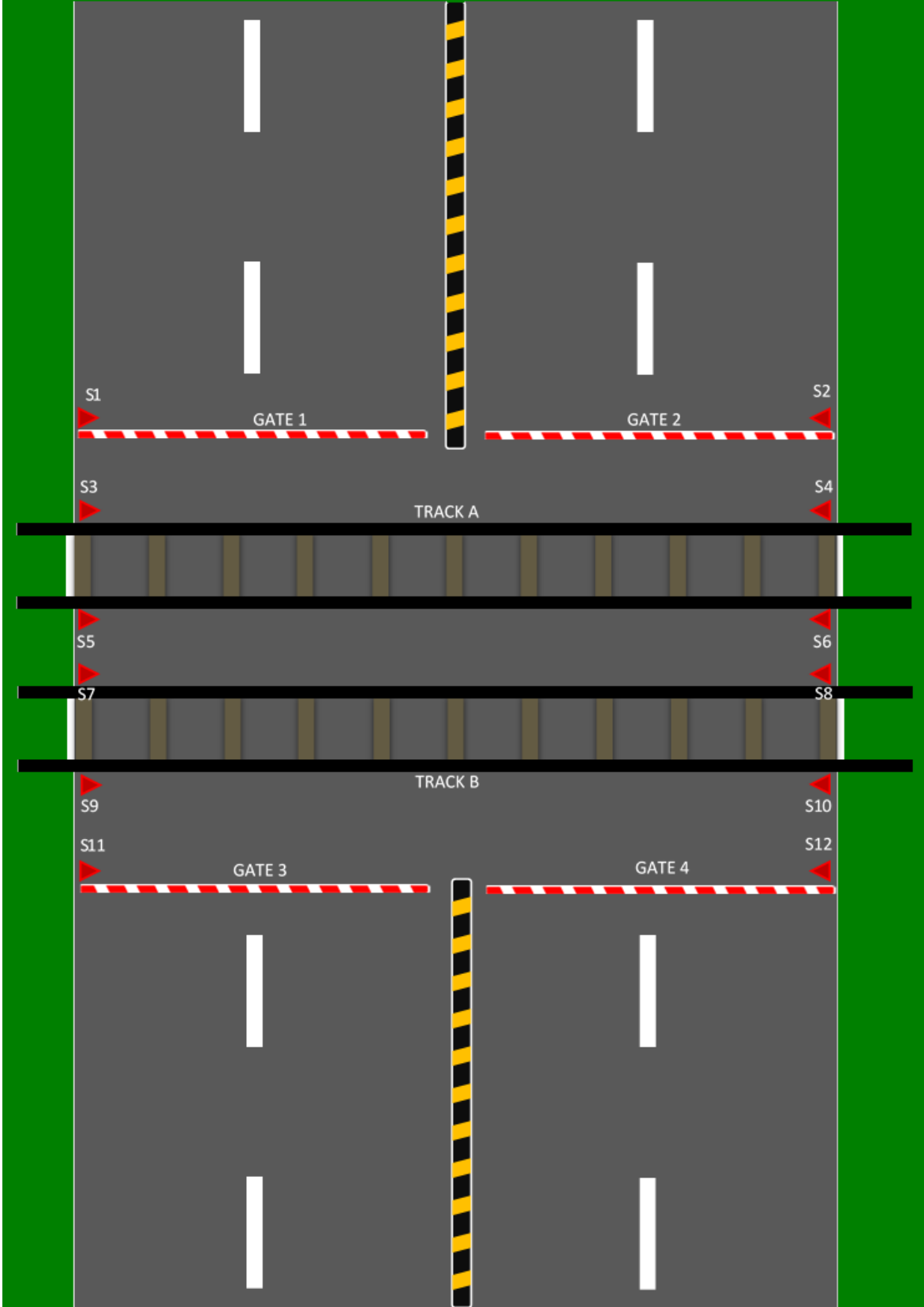


Figure 7.11 Sensor Positions for Obstacle Detection at a Level Crossing

#### **7.4.4 – Obstacle Detection Mechanism**

We know that obstacle detection process starts as soon as the Level Crossing System knows that a train is arriving. Obstacle is detected by the change of the sensor values. If any object is between any of the sensors and the sensor value is less than 800 and more than 300 for more than five seconds then that object is an obstacle and the train will be notified about it with its position. If the obstacle moves away from the sensors, the train will then be notified that the obstacle has moved. If there are any obstacles under any gates those gates and the other gate on the same side of the road will remain open. This is done to prevent the bar from falling onto any vehicles and damage them. The other gate on the same side is remained open so that the vehicle cross the rail track and go onto the other side and not get trapped within the rail tracks and become prone to a collision.

For example if an obstacle is present under Gate 1, both Gate 1 and Gate 3 would remain opened until the vehicle passes into a safe zone.

Similarly if the obstacle is on any of the tracks, the gates on the side at which the obstacle is, are remained open so that the vehicle can pass through to a safe area.

For Example of the obstacle is on Track A and being a barrier for sensors 3 and 5, both Gate 1 and Gate 3 would remain opened until the vehicle passes into a safe zone.

Different message codes are sent to the train for different positions of obstacles. The process is described in the table on the next page.

<b>TRAIN ON TRACK</b>	<b>SENSOR NUMBER</b>	<b>OBSTACLE POSITION</b>	<b>SMS CODE</b>	<b>LEVEL CROSSING ACTIVITY</b>
TRACK A	S1	GATE 1	OBS-100	GATE 1 & 3 OPEN
	S2	GATE 2	OBS-100	GATE 2 & 4 OPEN
	S3    S4    S5    S6	TRACK A	OBS-111	GATE 1 & 2 or GATE 3 & 4 OPEN
	S7    S8    S9    S10	TRACK B	OBS-100	GATE 1 & 2 or GATE 3 & 4 OPEN
	S11	GATE 3	OBS-100	GATE 3 & 1 OPEN
	S12	GATE 4	OBS-100	GATE 4 & 2 OPEN
	N/A	N/A	OBS-000	ALL GATES CLOSED
TRACK B	S1	GATE 1	OBS-100	GATE 1 & 3 OPEN
	S2	GATE 2	OBS-100	GATE 2 & 4 OPEN
	S3    S4    S5    S6	TRACK A	OBS-100	GATE 1 & 2 or GATE 3 & 4 OPEN
	S7    S8    S9    S10	TRACK B	OBS-111	GATE 1 & 2 or GATE 3 & 4 OPEN
	S11	GATE 3	OBS-100	GATE 3 & 1 OPEN
	S12	GATE 4	OBS-100	GATE 4 & 2 OPEN
	N/A	N/A	OBS-000	ALL GATES CLOSED

*Table 7.2 SMS Codes sent to Train for different obstacle positions and its corresponding level crossing activity*

Flow chart of obstacle detection:

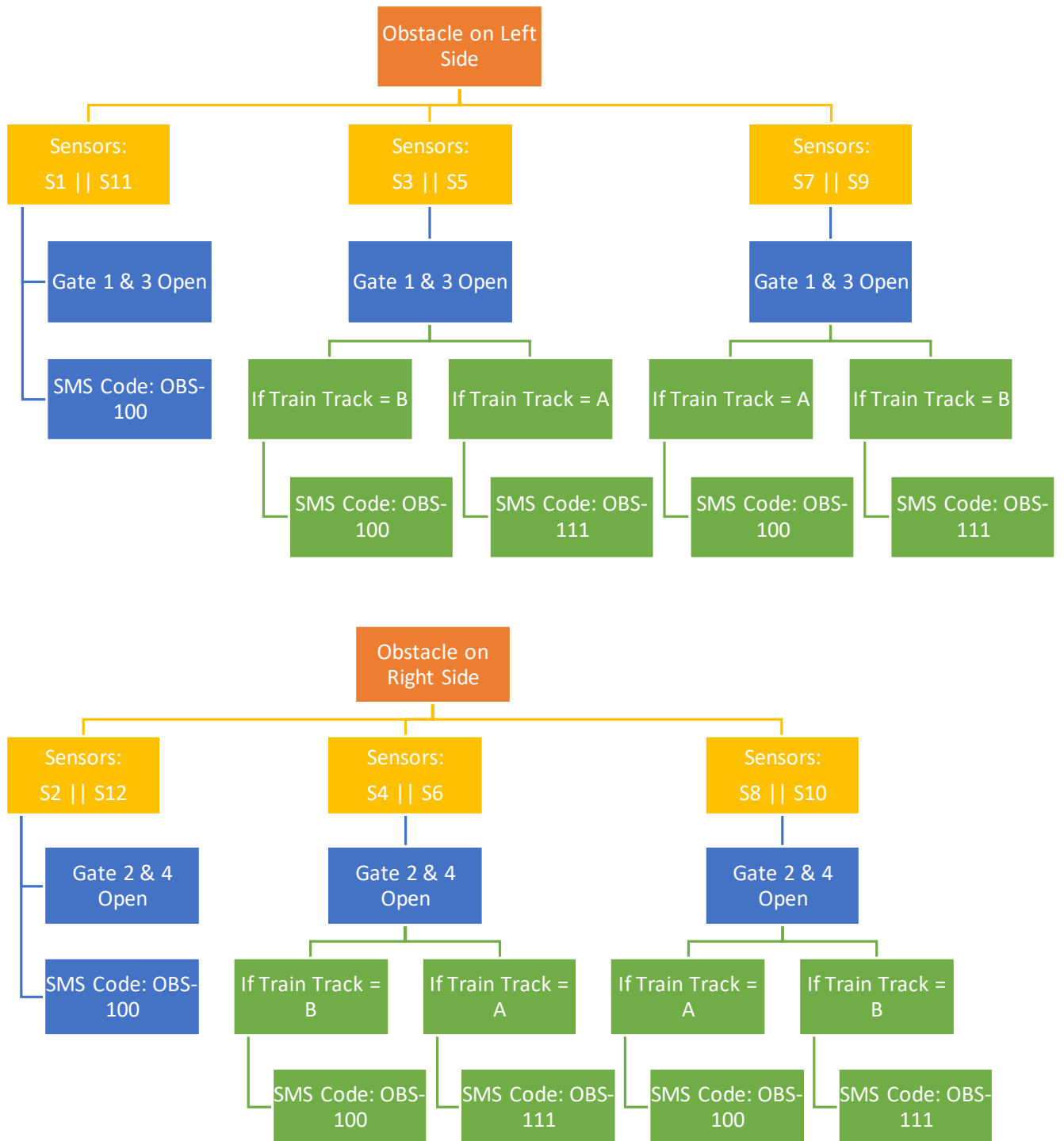


Figure 7.12 Flow Chart for Obstacle Detection at Level Crossings

#### **7.4.5 – Closing and Opening Gates**

Closing and opening of gates are done using the servo motors. Keeping the obstacle detection algorithm aside, level crossing gates are programmed to close as soon as it knows that a train is approaching towards it. The gates will again open when the train sends another message that it has left the level crossing region and has travelled to a safe distance. The gates are normally kept open.

#### **7.4.6 – Control of Road Signals and Alarm**

Two sets of two LEDs are used in the road signal for two sides of the roads and a buzzer is used as an alarm. As soon as the level crossing knows the arrival of a train towards it, it toggles the red LED ON, on both sides of the road also switching on the buzzer. The red LEDs and the Buzzer stays on until the train sends the message of leaving the level crossing region. The red LEDs and the buzzer turns OFF and the Green LEDs are switched on indicating that the road is now safe to use.

#### **7.4.7 – Control of Train Track Signals**

Two sets three LEDs, Green, Yellow and Red are used for signaling the trains on both the tracks. These LEDs are at work during the obstacle detection process and indicates whether the train should maintain its current speed, reduce its speed or stop completely through the green, yellow and red LEDs respectively.

#### **7.4.8 – The User Interface**

The user interface is made as simple as possible for the drivers to use since it requires fast communication during emergency situations. The User Interface comprises of one LCD, a Keypad, a 'Menu' button, LEDs and Potentiometers to control the brightness of the LCDs.

The LCD on board shows all events that are happening in the level crossing.

A keypad is installed for 'speed messaging' which means, upon pressing the 'Menu' button if 'Send Message' option is selected, the message code can be selected straight away from the keypad. This would really be hassle free and fast during emergency situations.

There is also a 'New Message' LED on the board to indicate if there is a new message or not.

Four gate control LEDs are installed to know if the level crossing gates are open or not. LEDs are normally off when the gates are open.



Potentiometers to control the brightness of the LCDs are also installed on the board for a very easy user experience.



Figure 7.13 Level Crossing Mainboard

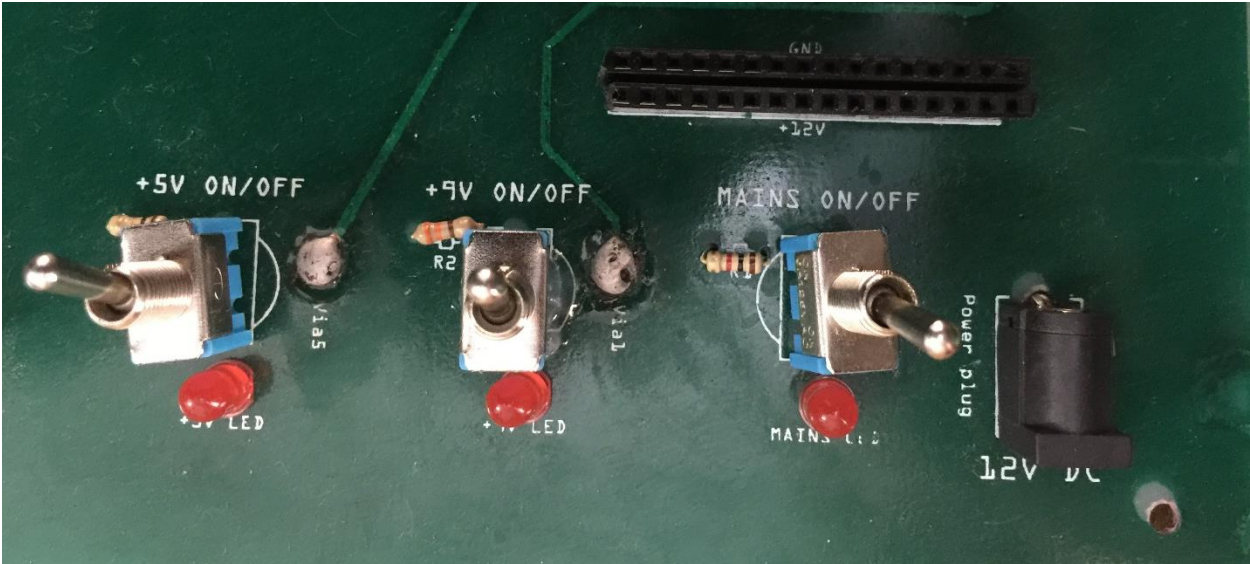


Figure 7.14 Power Distribution on the Level Crossing Mainboard

### 7.4.9 – Functions of the ‘Menu’ Button

The ‘Menu’ button was added for the ease of using the GUI of the Level Crossing System mainboard. This button opens a small menu to control the level crossing gates, train signals and send messages manually to the Central Control System or any nearby trains during an emergency case. The Menu appears on the LCD on board of the Level Crossing System mainboard. The flow chart below shows the entire Menu Map of the train system.

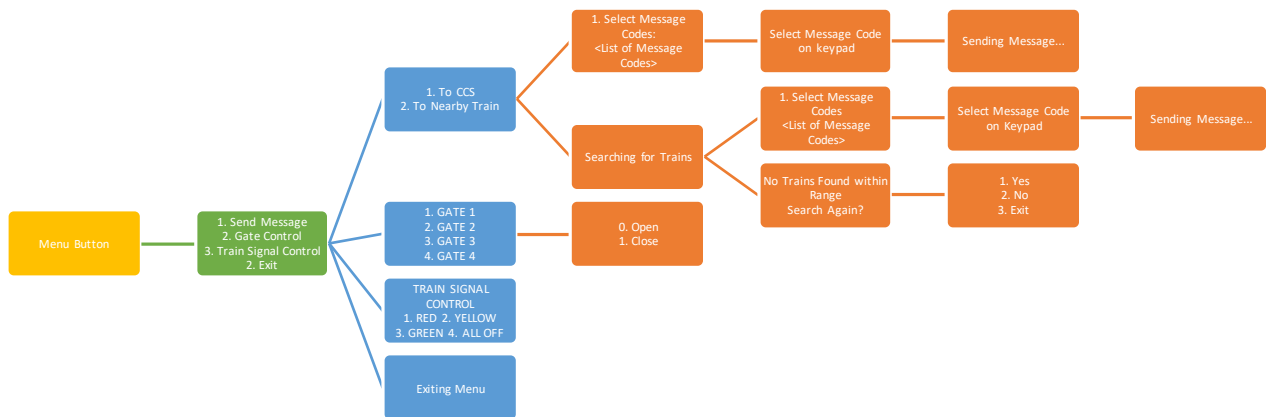
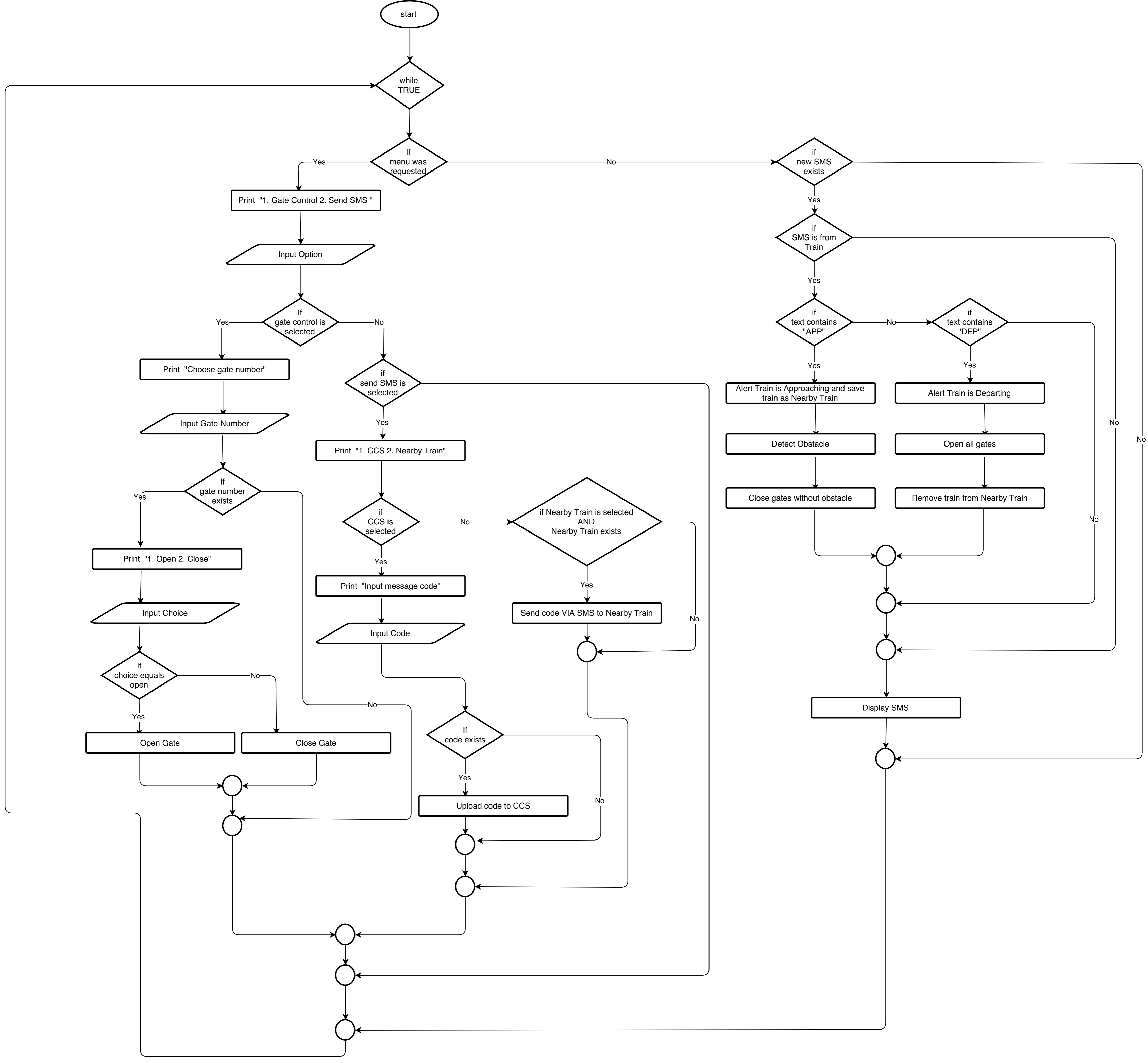


Figure 7.15 Menu Map of the Level Crossing System



## 7.5 – The Level Crossing Model

### 7.5.1 – Materials Used

- PVC Board
- Wood
- Spray Paint
- Jumper Wires
- Tape

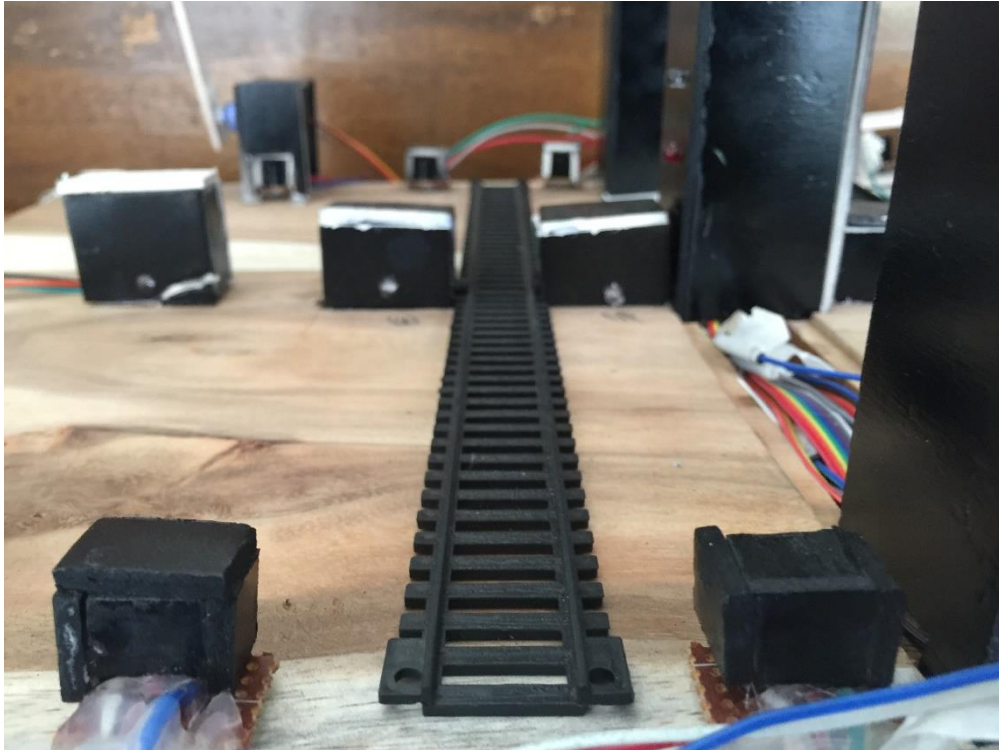
### 7.5.2 – Pictures of the Model



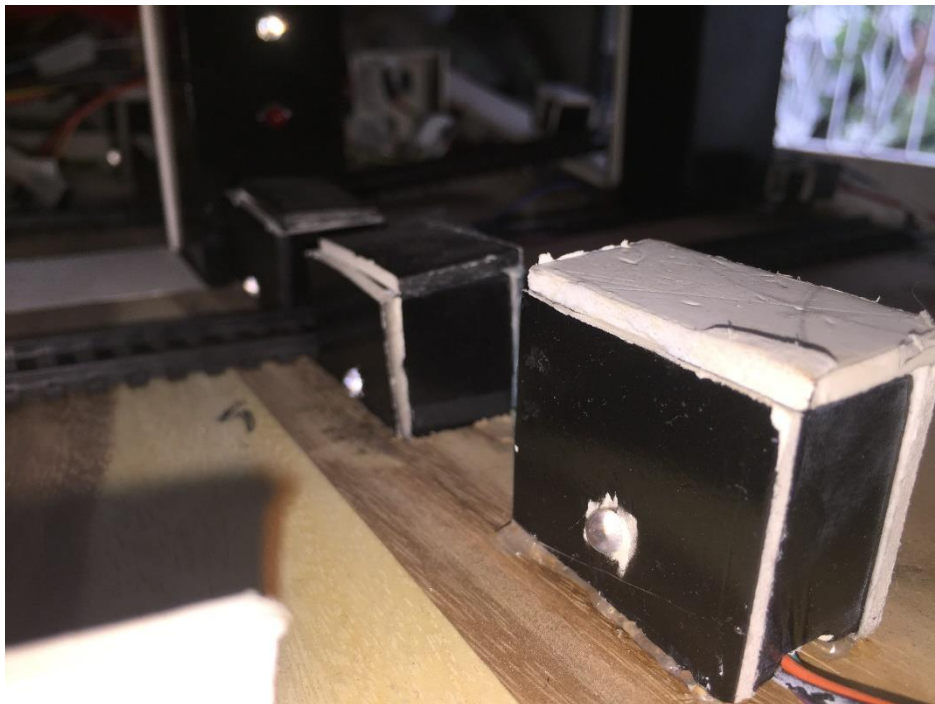
*Figure 7.17 Level Crossing Model Side View*



*Figure 7.18 Level Crossing Model Front View*



*Figure 7.19 IR Transmitter and receiver's position at the Level Crossing Model*



*Figure 7.20 IR Transmitters' position in the Level Crossing Model*

# CHAPTER 8 –THE CENTRAL CONTROL SYSTEM

## 8.1 – Introduction

The Central Control System is a website based on Model-View Controller (MVC) architecture. Only the employees who would be appointed as the operator of the Central Control System would have access to this system. The website has a registration page where the appointed employees would have to register themselves to the system by providing valid employee ID and other personal details. The Central Control System will have to be monitored by those registered operators very carefully and at all times.

The Central Control System has the following features:

- See live locations of all trains on a Google Map API
- See details (speed, departing station, destination station, current location, latitude and longitude) of trains which are on route
- See today's train schedules on the homepage
- See messages that has been received/ sent to or from the trains
- See detailed train schedules
- See the current weather conditions, date and time
- Notification icon for any unseen events (like Facebook)
- View, call or send message to all Maintenance Teams
- View, call or send message to all Hospitals, Police Stations or Fire Stations
- View current Railway News
- Chat Room for chatting live with other employees of the railway system

The address of the website is: [www.ptc-ccs.com](http://www.ptc-ccs.com)

Screenshots of the website:



Figure 8.1 Login Page

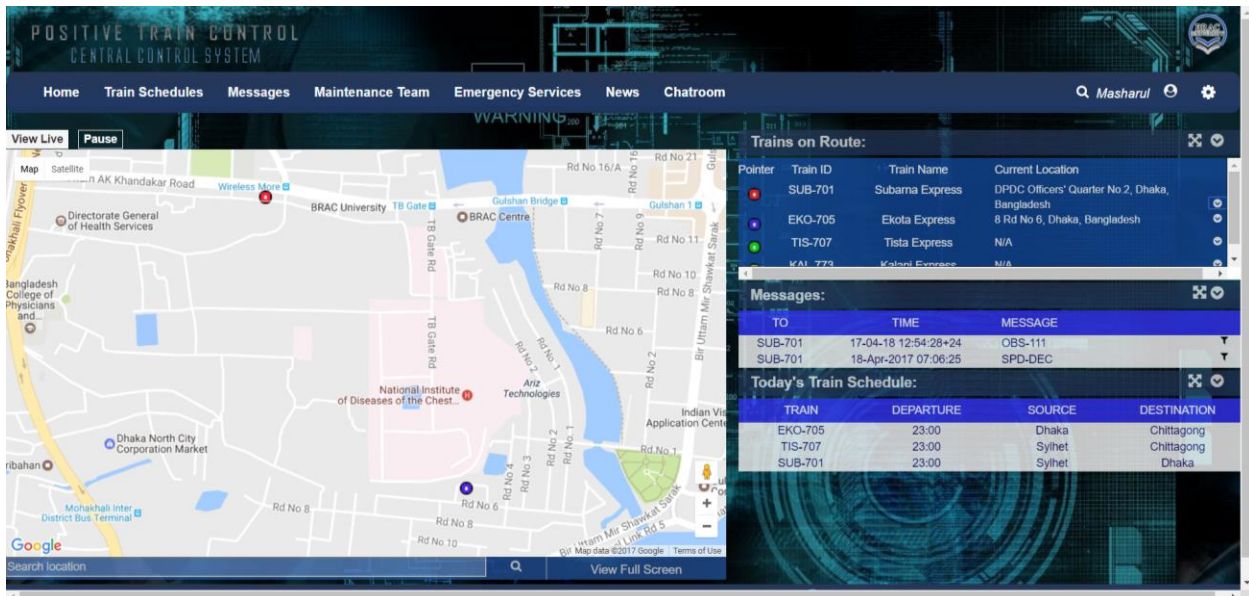


Figure 8.2 Home page

The screenshot displays the 'Trains on Route' section of the Positive Train Control Central Control System. It features a map on the left and a list of trains on the right. The train list includes columns for Pointer, Train ID, Train Name, and Current Location. Below the list, there are sections for 'Send a Command', 'Message History', and 'View Message Codes' for each train.

Pointer	Train ID	Train Name	Current Location
●	SUB-701	Subarna Express	DPDC Officers' Quarter No 2, Dhaka, Bangladesh
●	EKO-705	Ekota Express	8 Rd No 6, Dhaka, Bangladesh

Train ID	Train Name	Current Location
TIS-707	Tista Express	N/A
KAL-773	Kalani Express	N/A

Figure 8.3 Train Details and options to send message

The screenshot displays the 'Maintenance Team' section of the Positive Train Control Central Control System. It shows a list of teams organized by location (Dhaka, Chittagong, Sylhet, Rajshahi, and Khulna). Each team entry includes the team name, members, and a contact number.

Location	Team	Members	Contact Number
Dhaka	Team GAZ-01	MEMBERS: KHAIRUL, AKKAS, MOTALEB, SUJON, KABIR	+8801670669566
	Team GAZ-02	MEMBERS: MIZAN, KHALIL, HALIM, RAJU, BABU	+8801670669567
	TEAM TON-01	MEMBERS: KHAIRUL, AKKAS, MOTALEB, SUJON, KABIR	+8801670669566
	Team AIR-01	MEMBERS: KHAIRUL, AKKAS, MOTALEB	+8801670669566
Chittagong	Team AIR-02	MEMBERS: MIZAN, KHALIL, HALIM, RAJU, BABU	+8801670669567
	Team AIR-03	MEMBERS: KHAIRUL, AKKAS, MOTALEB, SUJON, KABIR	+8801670669567
	Team AIR-04	MEMBERS: MIZAN, KHALIL, HALIM, RAJU, BABU	+8801670669567
Kamalapur	Team KAM-01	MEMBERS: KHAIRUL, AKKAS, MOTALEB	+8801670669566
	Team KAM-02	MEMBERS: MIZAN, KHALIL, HALIM, RAJU, BABU	+8801670669567
	Team KAM-03	MEMBERS: KHAIRUL, AKKAS, MOTALEB, SUJON, KABIR	+8801670669567
	Team KAM-04	MEMBERS: MIZAN, KHALIL, HALIM, RAJU, BABU	+8801670669567

Figure 8.4 Maintenance Team List



## **8.2 – Specifications**

Architecture: Model View Controller: Symfony 2.8

Server: Apache Version 2.4.25

Cpanel Version: 62.0 (Build 17)

Operating System: Linux (x86\_64)

Programming Languages: HTML5, CSS3, PHP, SQL, Javascript, JQuery, Doctrine

Domain Name: URL: <http://www.ptc-ccs.com>

## **8.3 – Database Structure**

Database Name: samdadoc\_trainmvc

Database: MySQL

Database Schema:

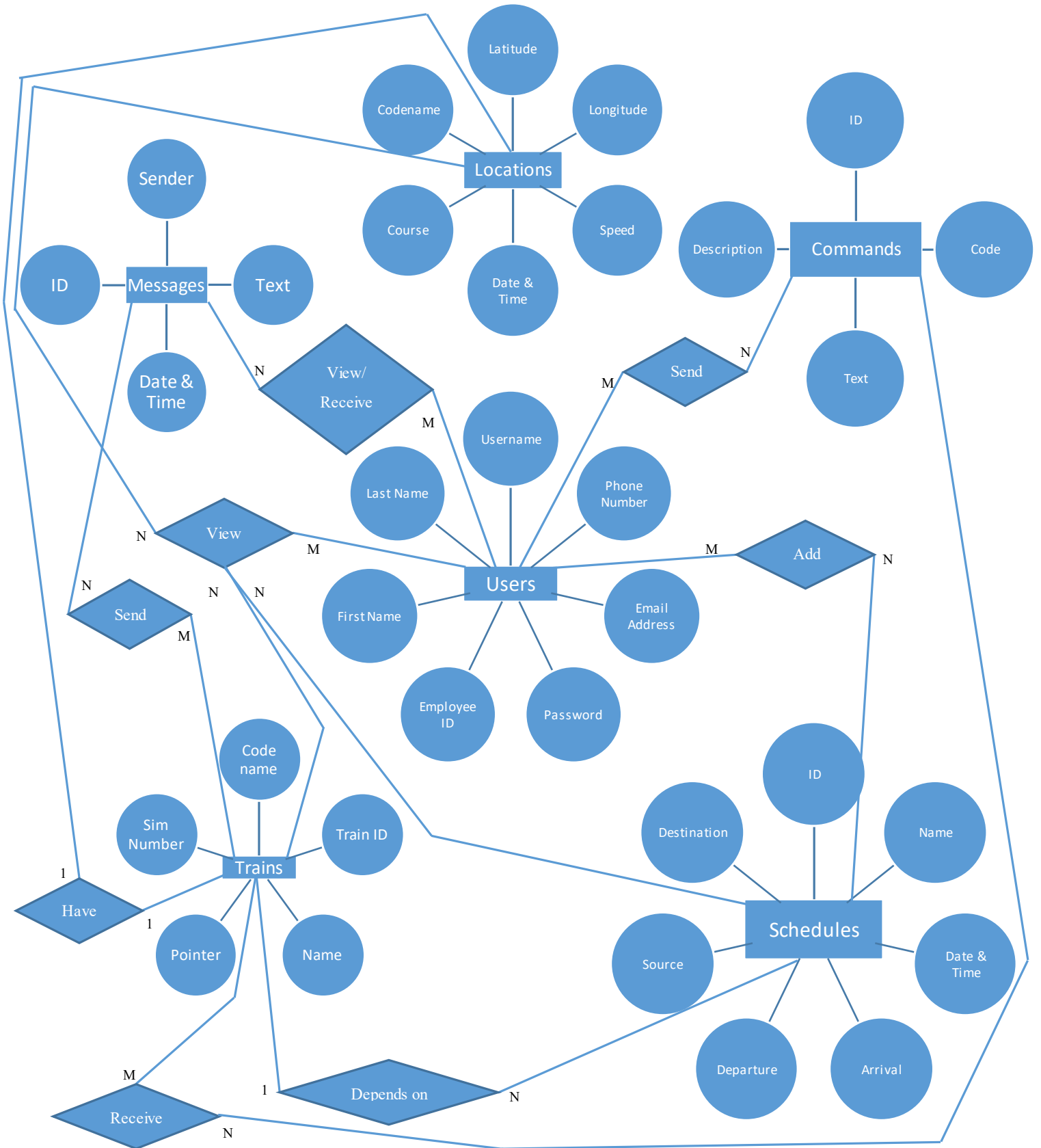
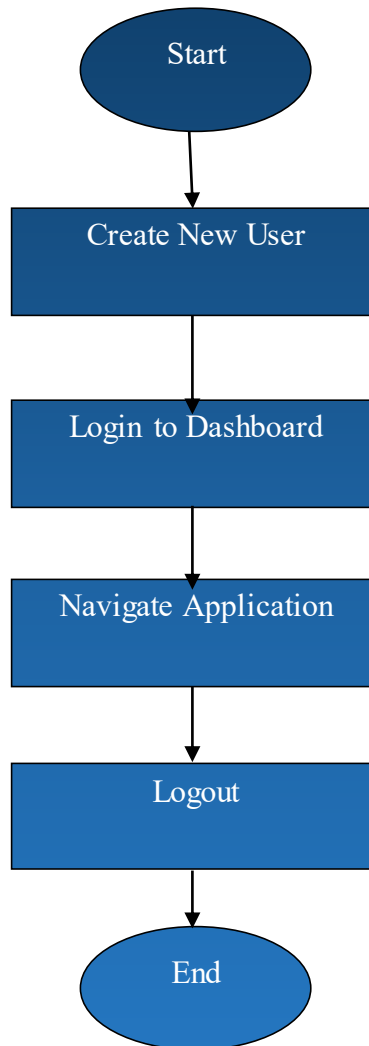


Figure 8.5 Database Schema: Entity Relationship Diagram

## 8.4 – Website Design

### 8.4.1 – User Navigation (Web Application)



*Figure 8.6 Flow Chart of User Navigation*

### 8.4.2 – Location Upload (Web Application)

Locations of each trains are uploaded to a file named 'add\_location.php'

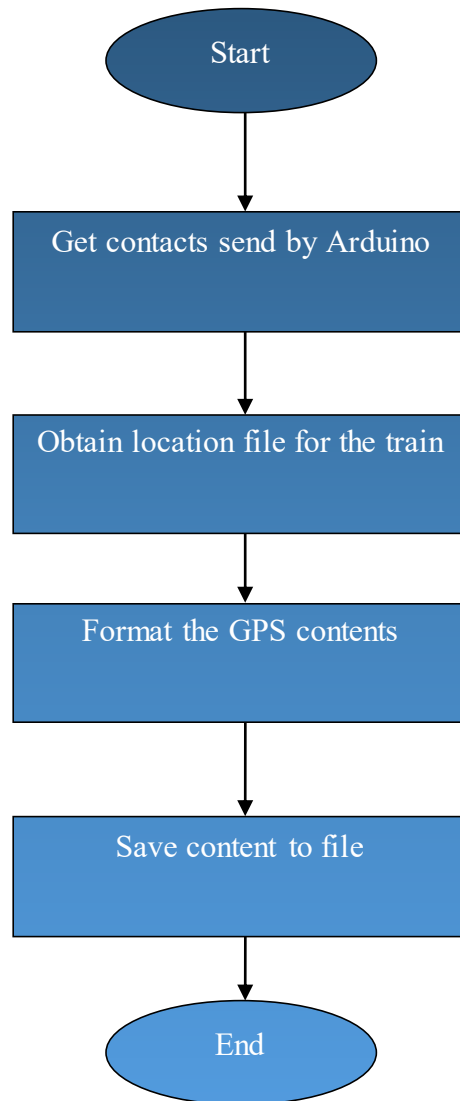


Figure 8.7 Flow chart for uploading location in the web site

### 8.4.3 – Location View (Web Application)

File: google\_map\_new.js

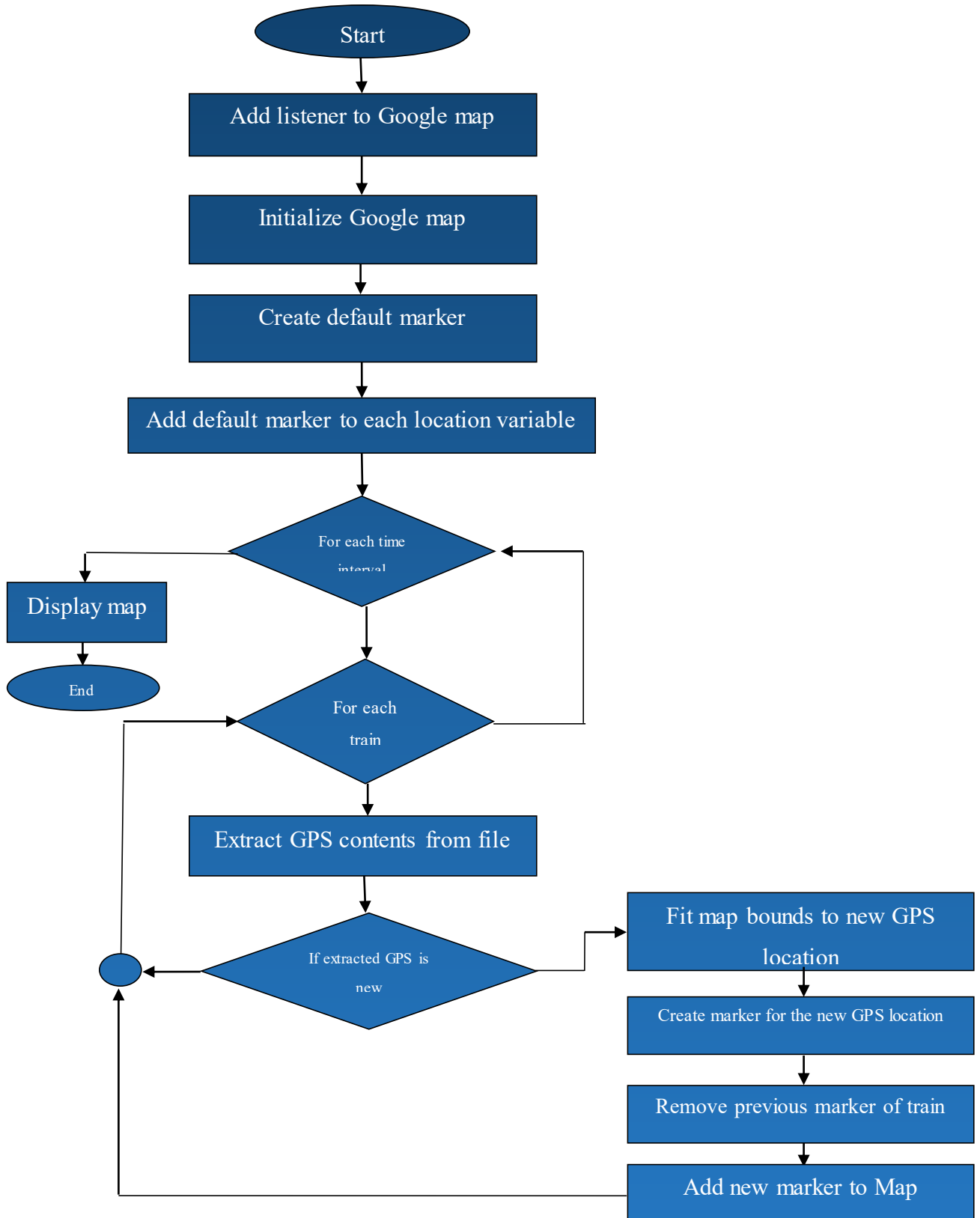


Figure 8.8 Flow Chart of viewing train location in the Web Site

#### 8.4.4 – View Train Information

URL: /home

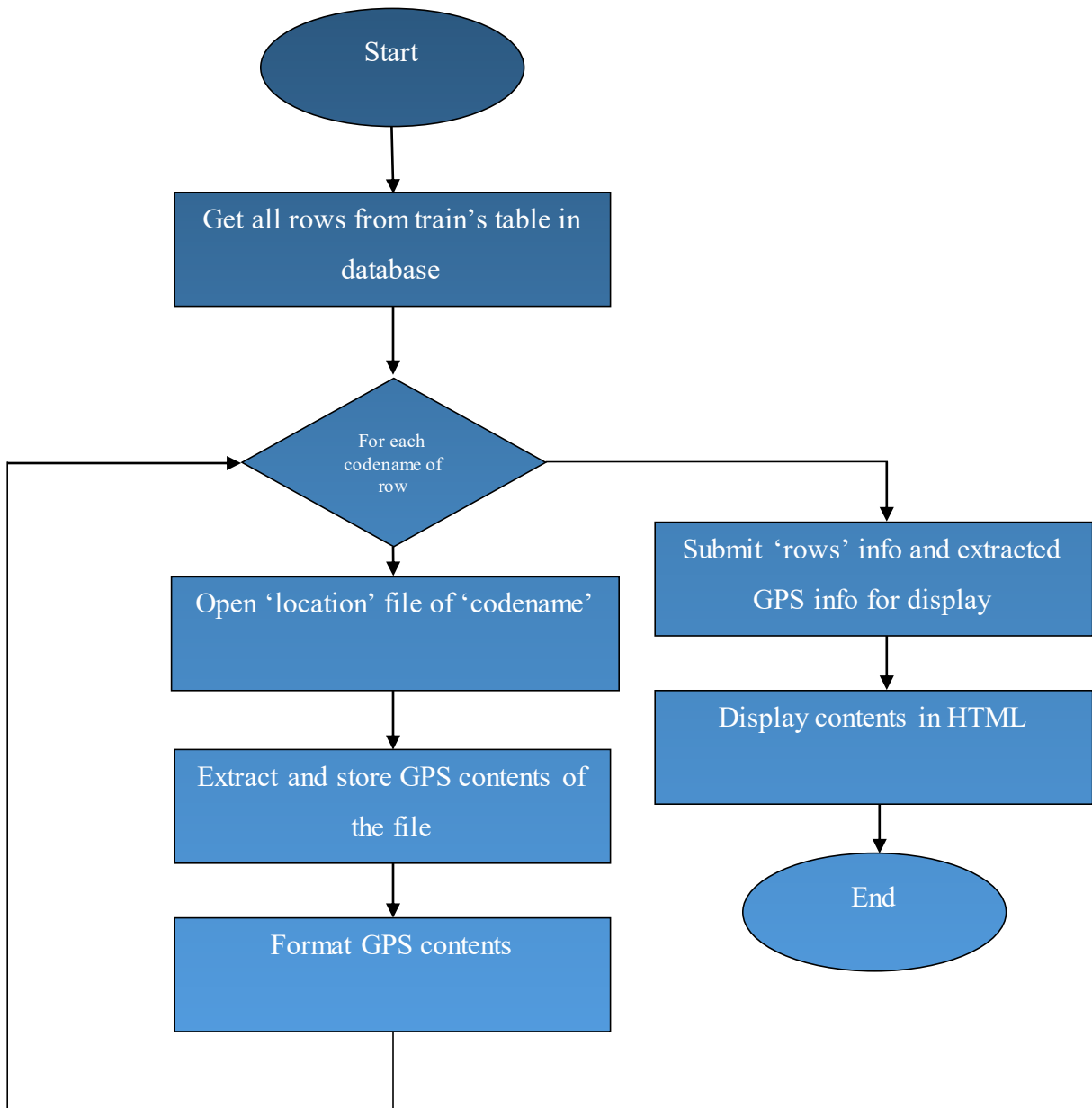


Figure 8.9 Flow chart of viewing train information in the web site

### 8.4.5 – Add Message

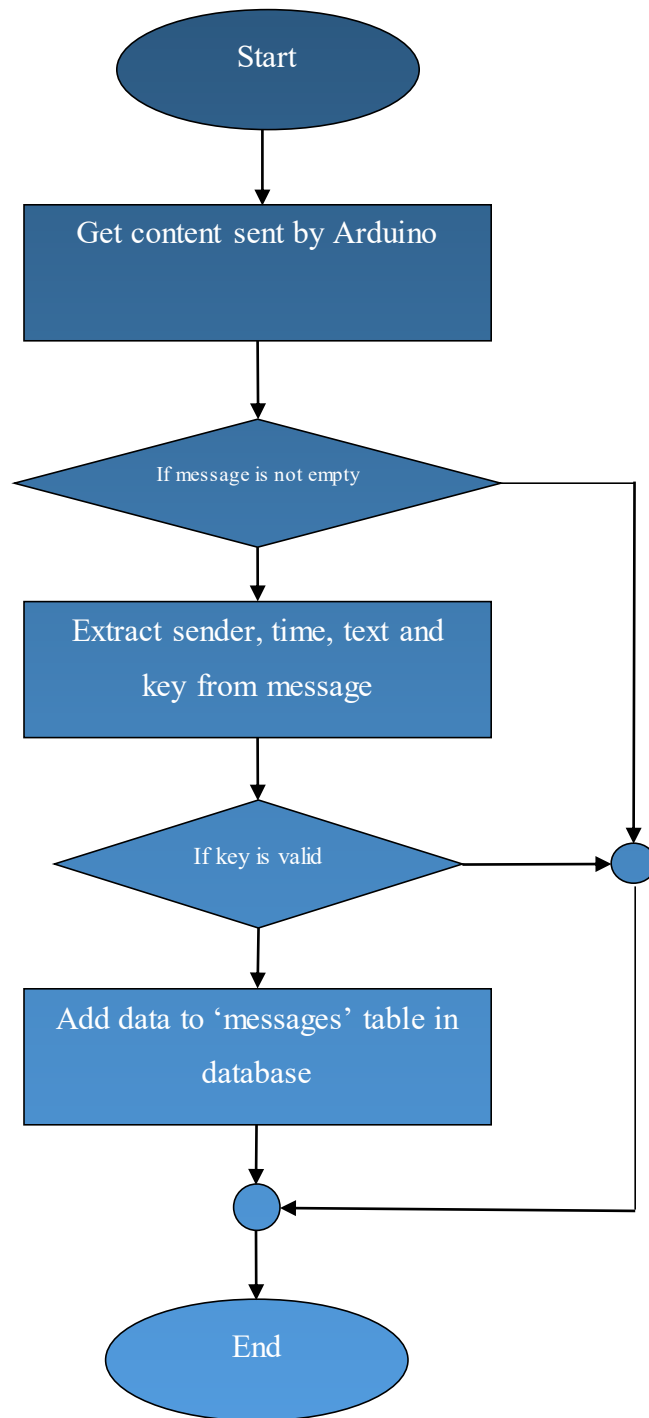


Figure 8.10 Flow chart for adding messages to the database

8.4.6 – Website Functions

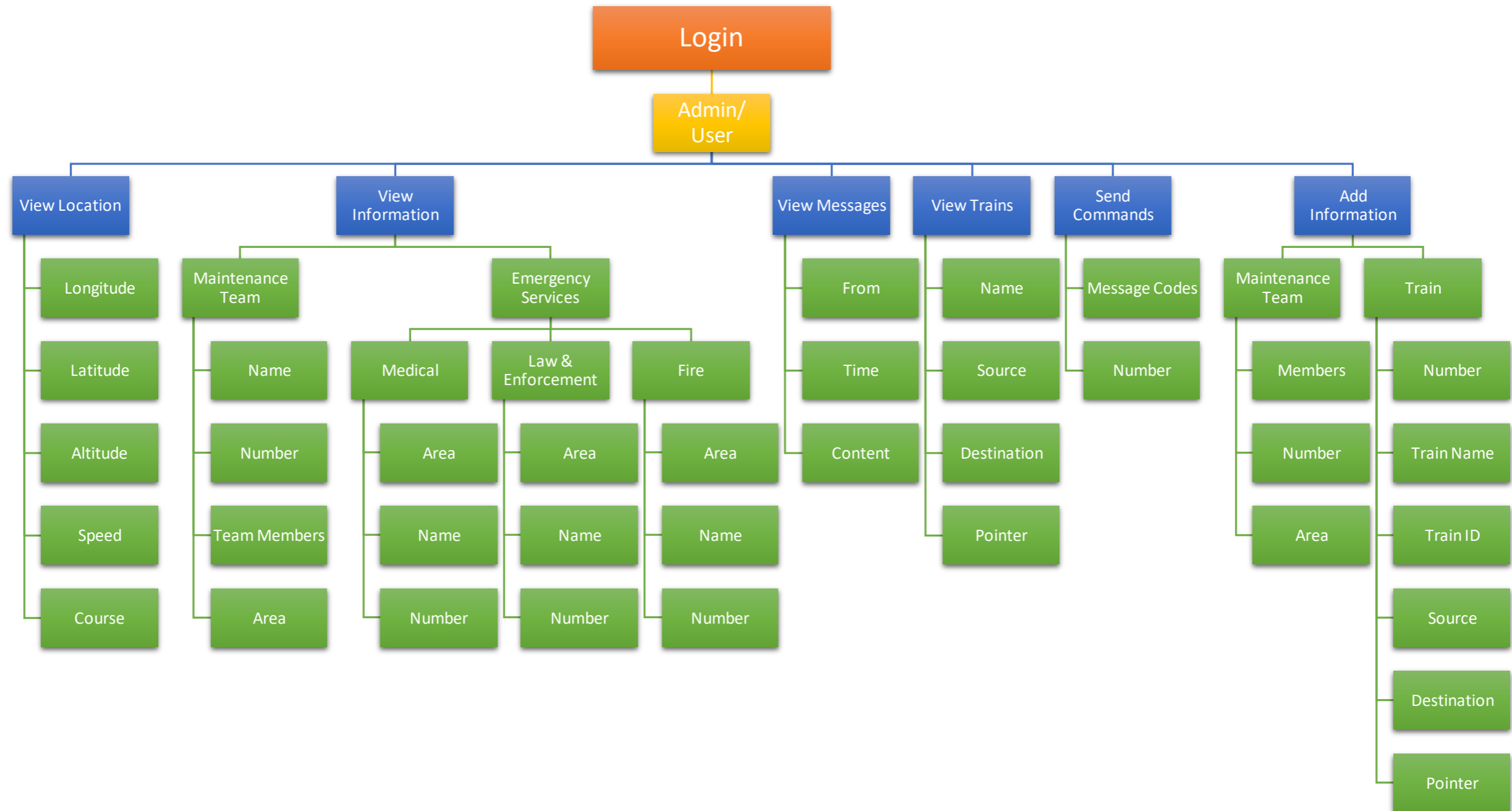


Figure 8.11 Flow Chart of Website Functions



## 8.4.7 – Site Map

### Application Structure:

1. Login Page (index.html)
2. Register User (/register\_user)
3. Add/ Update Location (/add\_location)
4. Dashboard:
  - a. Home (/home)
    - i. View Google Map
    - ii. View train's information
    - iii. View weather and date
    - iv. View Messages
    - v. View Today's Train Schedule
  - b. Train Schedules:
    - i. Add Schedules (/add\_schedule)
    - ii. View Schedules (/view\_schedule)
  - c. Messages:
    - i. View Message History (/messages/{filter})
    - ii. View Message Codes (/view\_message\_codes)
    - iii. Add Message Codes (/add\_message\_codes)
  - d. Maintenance Team
  - e. Emergency Services
    - i. Fire Service
    - ii. Law & Enforcement Service
    - iii. Medical Service
  - f. News
    - i. Railway News
  - g. Profile (/view\_profile)
  - h. Chat Room
  - i. Logout (/logout)

## **8.5 – Software Used**

### **8.5.1 – PHPStorm**

PHPStorm was used for the development of the Model View Controller (MVC) architecture of the web application. The front end view was developed in PHPStorm using JQuery, Javascript, HTML, CSS, Bootstrap framework and Google Map API. As MVC architecture was used in the development of the web application, the controller segment of the application was developed using PHP framework.

### **8.5.2 – Adobe Dreamweaver**

Adobe Dreamweaver was also used for the front end development of the website.

### **8.5.3 – Apache Server**

Apache Server was used for the server side scripting where MySQL and Doctrine was used to develop model portion of the application. Apache interacts with client side and database in order to ensure security and reduce vulnerability of the application. At first the application was tested in WAMP server which was later transferred to CPanel for production environment.

## CHAPTER 9 – TESTS & RESULTS

### 9.1 – SMS Messaging Test

#### 9.1.1 – Test Type 1

**Description:** SMS was sent from the microcontroller board to a mobile phone several times on different conditions. Eight tests out of many are shown in the table below.

Test no.	Condition	Time Required for SMS to reach Client	Result	Bugs	Solution
1	Closed Room	N/A	Unsuccessful	Wrong Pin Connection	Interchange TX and RX pins
2	Closed Room	N/A	Unsuccessful	No validity of SIM Card	Recharge SIM card
3	Closed Room	12 Seconds	Successful	N/A	N/A
4	Closed Room	10 Seconds	Successful	N/A	N/A
5	Open Air	8 Seconds	Successful	N/A	N/A
6	Rainy, Closed Room	N/A	Unsuccessful	No network Coverage	Go in open air
7	Rainy, Closed Room	14 Seconds	Successful	Low network coverage	Go in open air
8	Rainy, Open Air	10 Seconds	Successful	N/A	N/A

*Table 9.1 SMS Messaging Test Type-1*

### 9.1.2 – Test Type 2

**Description:** SMS was received to the microcontroller board from a mobile phone several times on different conditions. Eight tests out of many are shown in the table below.

Test no.	Condition	Time Required for SMS to reach Client	Result	Bugs	Solutions
1	Closed Room	N/A	Unsuccessful	Wrong Pin Connection	Interchange TX and RX pins
2	Closed Room	N/A	Unsuccessful	No credit in SIM Card	Recharge SIM card
3	Closed Room	N/A	Unsuccessful	Unknown	Unknown
4	Closed Room	8 Seconds	Successful	Only first 12 characters of the SMS content was received	Increase buffer size of Software Serial
5	Open Air	8 Seconds	Successful	N/A	
6	Rainy, Closed Room	N/A	Unsuccessful	No network Coverage	Go in open air
7	Rainy, Closed Room	13 Seconds	Successful	Low network coverage	Go in open air
8	Rainy, Open Air	10 Seconds	Successful	N/A	

*Table 9.2 SMS Messaging Test Type-2*

## 9.2 – GPS Test

**Description:** Take values of Latitude and Longitudes at 30 second intervals of two different fixed points on different conditions.

Set 1, Point A (Google Map Coordinates: Lat: 23.780736, Lon: 90.407600) Conditions: Open Air, Sunny					
Reading No.	Time	Latitude	Longitude	% Error in Latitude	% Error in Longitude
1	0:30	23.780740	90.407600	-0.000017	0.000000
2	1:00	23.780740	90.407600	-0.000017	0.000000
3	1:30	23.780740	90.407600	-0.000017	0.000000
4	2:00	23.780738	90.407600	-0.000008	0.000000
5	2:30	23.780738	90.407600	-0.000008	0.000000
6	3:00	23.780738	90.407600	-0.000008	0.000000
7	3:30	23.780738	90.407600	-0.000008	0.000000
8	4:00	23.780736	90.407600	0.000000	0.000000
9	4:30	23.780736	90.407600	0.000000	0.000000
10	5:00	23.780736	90.407600	0.000000	0.000000

Table 9.3 GPS test, Set 1. Point A

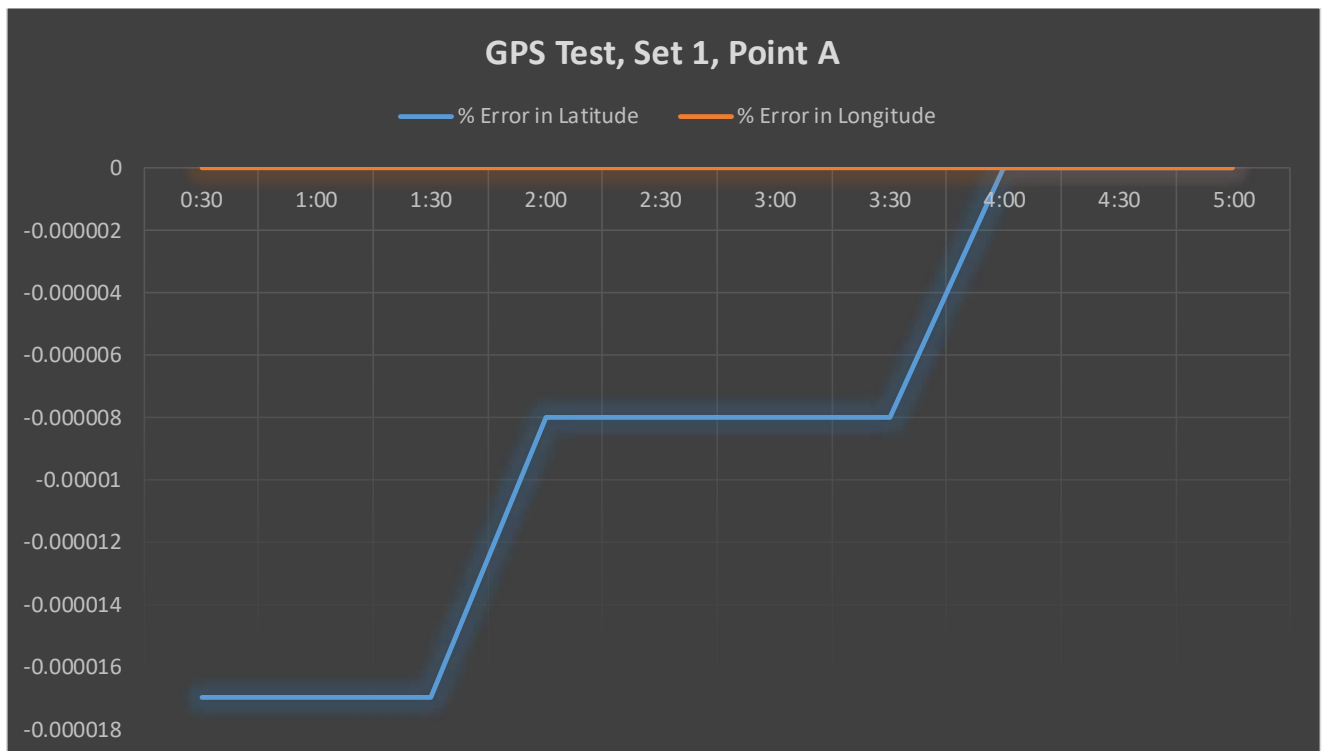


Figure 9.1 GPS Test, Set 1, Point A graph

Set 2, Point A (Google Map Coordinates: Lat: 23.780736, Lon: 90.407600)					
Conditions: Closed Room, Sunny					
Reading No.	Time	Latitude	Longitude	% Error in Latitude	% Error in Longitude
1	0:30	23.780695	90.407589	0.000172	0.000012
2	1:00	23.780721	90.407589	0.000063	0.000012
3	1:30	23.780721	90.407598	0.000063	0.000002
4	2:00	23.780730	90.407600	0.000025	0.000000
5	2:30	23.780731	90.407600	0.000021	0.000000
6	3:00	23.780733	90.407600	0.000013	0.000000
7	3:30	23.780736	90.407606	0.000000	-0.000007
8	4:00	23.780736	90.407600	0.000000	0.000000
9	4:30	23.780736	90.407600	0.000000	0.000000
10	5:00	23.780736	90.407600	0.000000	0.000000

Table 9.4 GPS test, Set 2. Point A

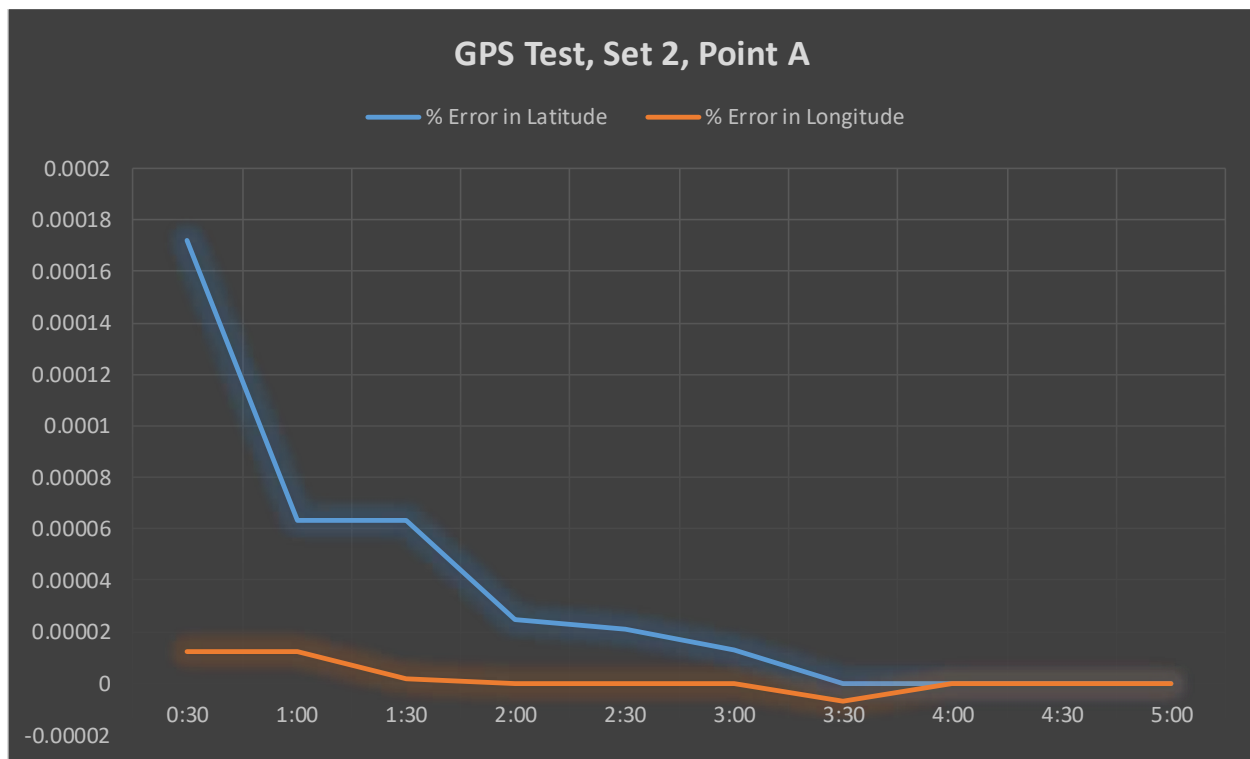


Figure 9.2 GPS Test, Set 2, Point A graph

Set 1, Point B (Google Map Coordinates: Lat: 23.780736, Lon: 90.407600)					
Conditions: Closed Room, Sunny					
Reading No.	Time	Latitude	Longitude	% Error in Latitude	% Error in Longitude
1	0:30	23.780798	90.407600	0.000345	0.000007
2	1:00	23.780798	90.407600	0.000345	0.000007
3	1:30	23.780798	90.407600	0.000345	0.000007
4	2:00	23.780824	90.407600	0.000235	0.000007
5	2:30	23.780824	90.407600	0.000235	0.000007
6	3:00	23.780856	90.407600	0.000101	0.000007
7	3:30	23.780856	90.407600	0.000101	0.000007
8	4:00	23.780872	90.407600	0.000034	0.000007
9	4:30	23.780872	90.407600	0.000034	0.000007
10	5:00	23.780872	90.407600	0.000034	0.000007

Table 9.5 GPS Test, Set 1, Point B

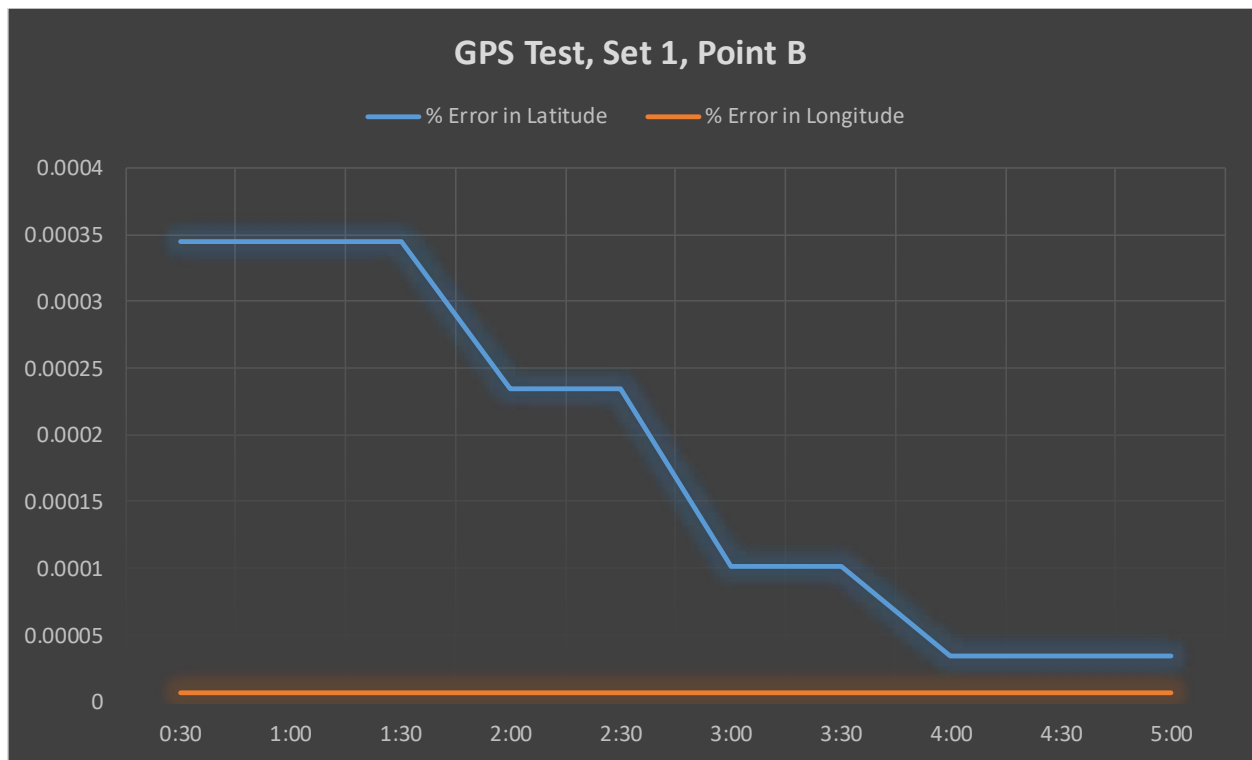


Figure 9.3 GPS Test, Set 1, Point B Graph

Set 2, Point B (Google Map Coordinates: Lat: 23.780880, Lon: 90.407606)					
Conditions: Open Air, Sunny					
Reading no.	Time	Latitude	Longitude	% Error in Latitude	% Error in Longitude
1	0:30	23.780768	90.407600	0.000471	0.000007
2	1:00	23.780785	90.407600	0.000399	0.000007
3	1:30	23.780829	90.407600	0.000214	0.000007
4	2:00	23.780858	90.407600	0.000093	0.000007
5	2:30	23.780878	90.407600	0.000008	0.000007
6	3:00	23.780878	90.407600	0.000008	0.000007
7	3:30	23.780878	90.407600	0.000008	0.000007
8	4:00	23.780880	90.407600	0.000000	0.000007
9	4:30	23.780880	90.407602	0.000000	0.000004
10	5:00	23.780880	90.407602	0.000000	0.000004

Table 9.6 GPS test, Set 2, Point B

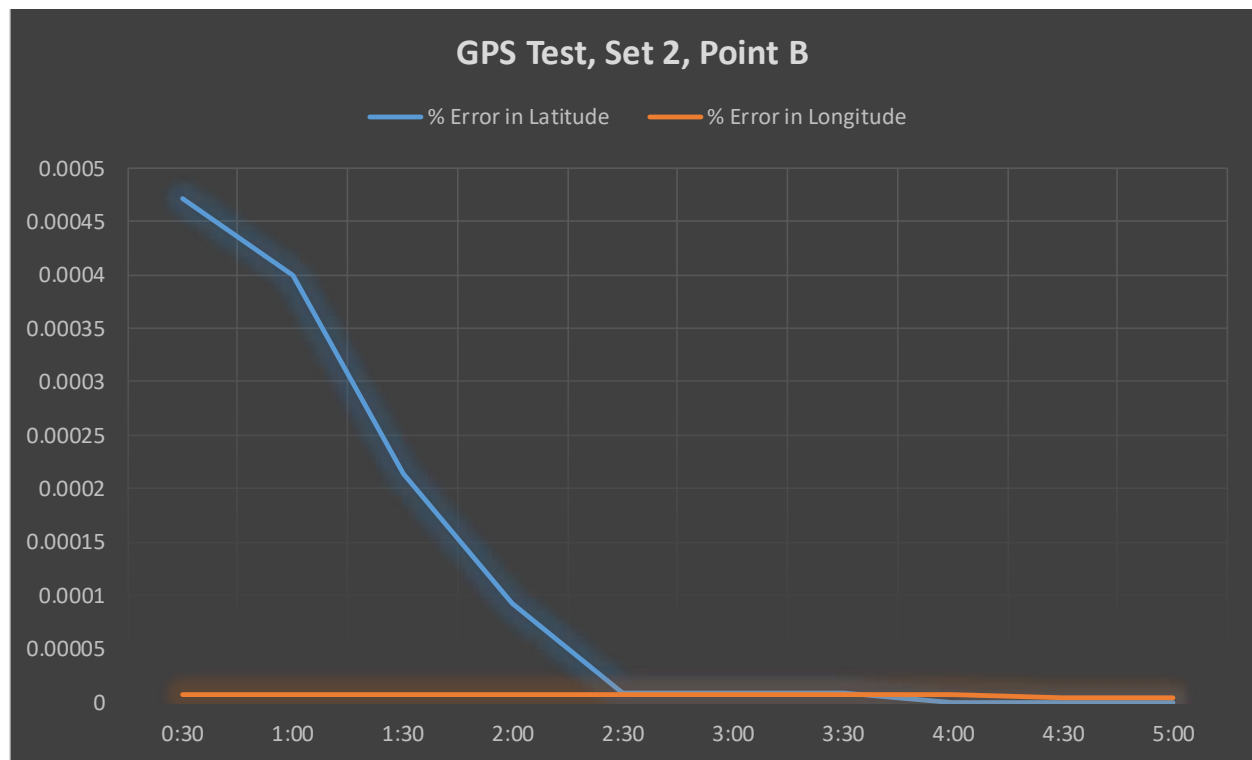


Figure 9.4 GPS test, Set 2, Point B



### 9.3 – Obstacle Detection Test

**Description:** The IR transmitters and receivers were placed approximately 10 meters apart and an obstacle of at least 1 inch were placed between them. Obstacle was moved slowly from the IR transmitter end to the receiver end and the readings were observed. Readings were taken from the microcontroller with which the IR receivers were connected.

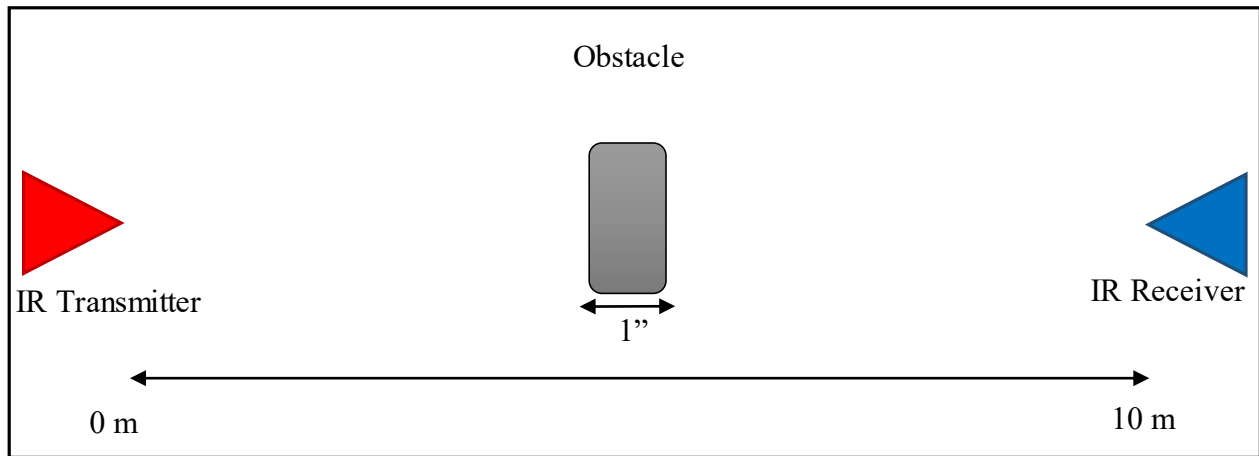
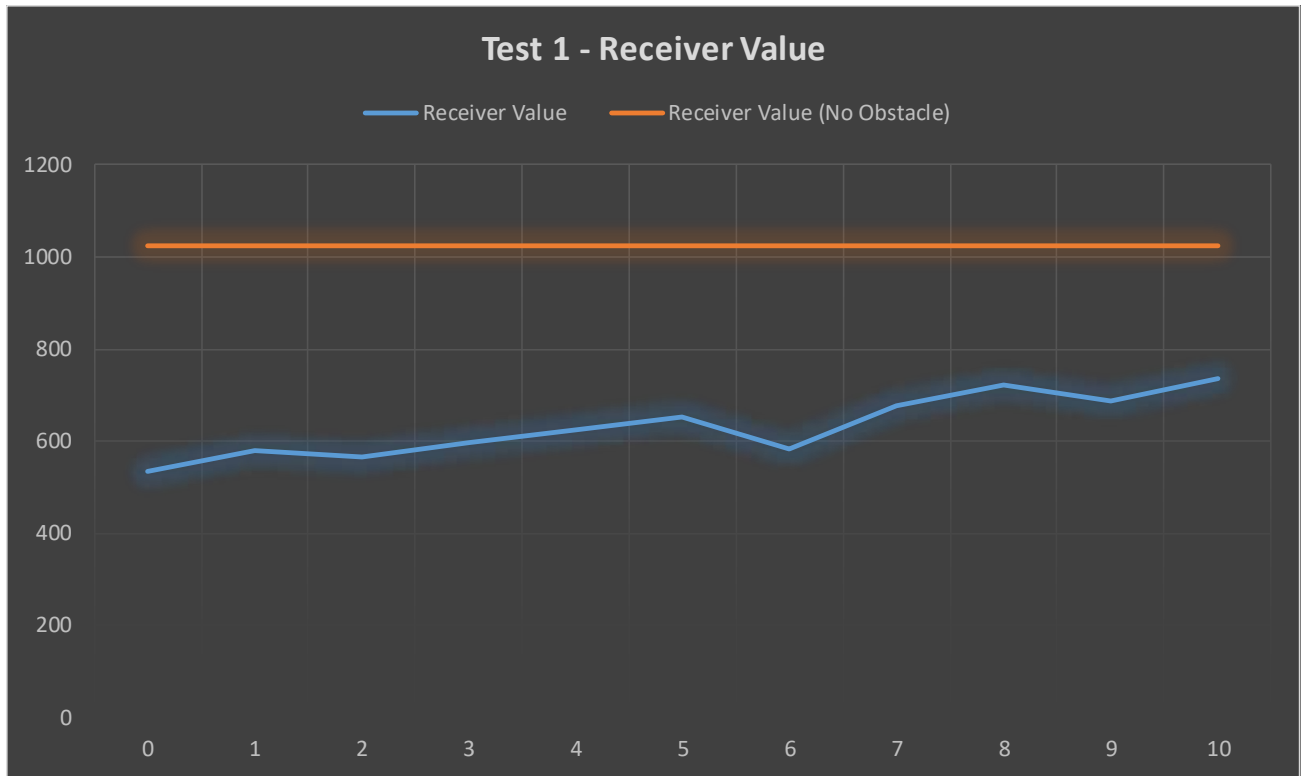


Figure 9.5 Obstacle Detection Test Setup Diagram

Test 1		
No obstacle reading = 1023		
Reading no.	Obstacle position (in meters)	Receiver Value
1	0	536
2	1	579
3	2	565
4	3	596
5	4	625
6	5	653
7	6	584
8	7	678
9	8	721
10	9	687
11	10	735

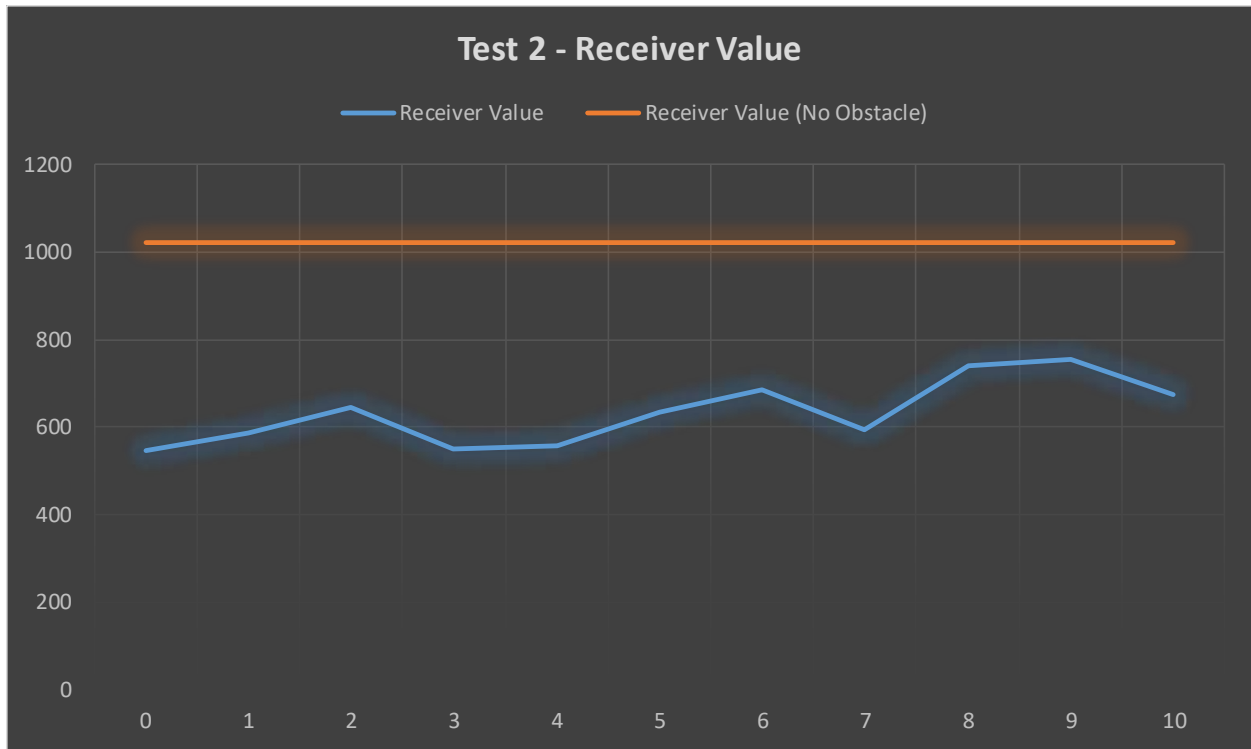
Table 9.7 Obstacle Detection Test 1



*Figure 9.6 Obstacle Detection Test 1 Graph*

Test 2		
No obstacle reading = 1022		
Reading no.	Obstacle position (in meters)	Receiver Value
1	0	547
2	1	587
3	2	645
4	3	552
5	4	558
6	5	633
7	6	686
8	7	594
9	8	741
10	9	756
11	10	676

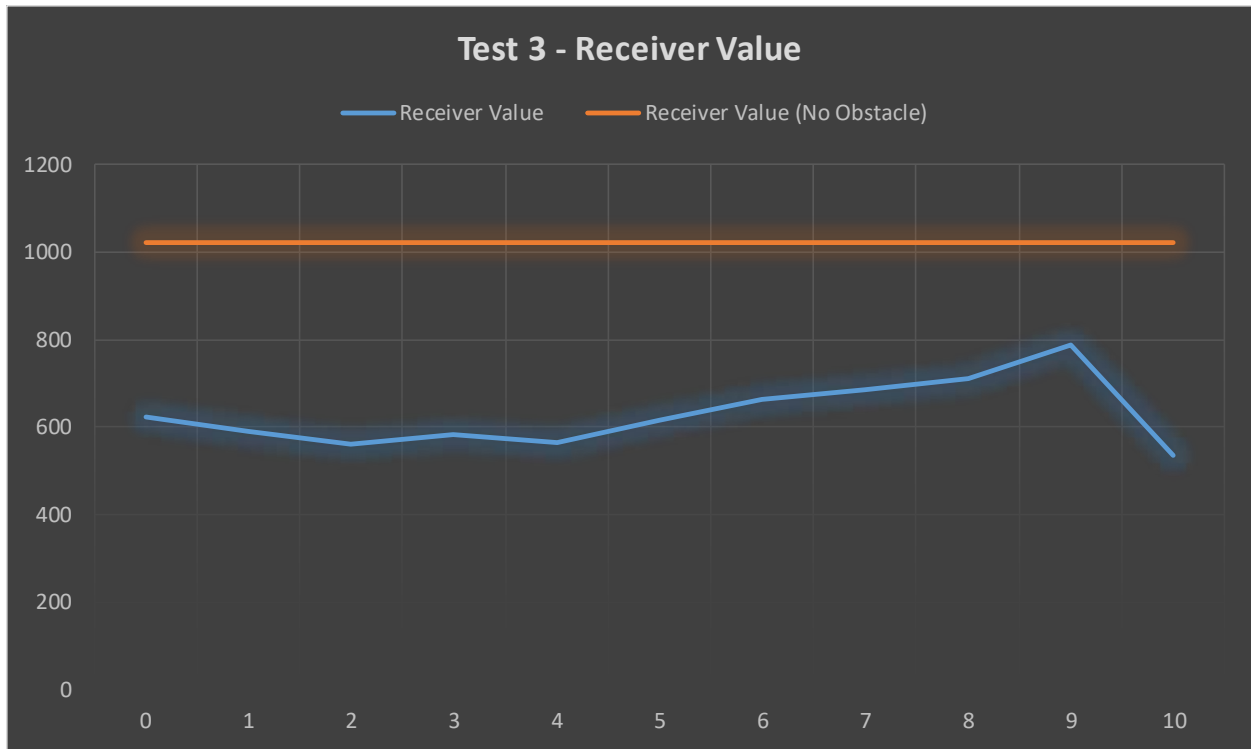
*Table 9.8 Obstacle Detection Test 2*



*Figure 9.7 Obstacle Detection Test 2 Graph*

Test 3		
No obstacle reading = 1023		
Reading no.	Obstacle position (in meters)	Receiver Value
1	0	623
2	1	591
3	2	563
4	3	584
5	4	566
6	5	617
7	6	665
8	7	687
9	8	712
10	9	787
11	10	537

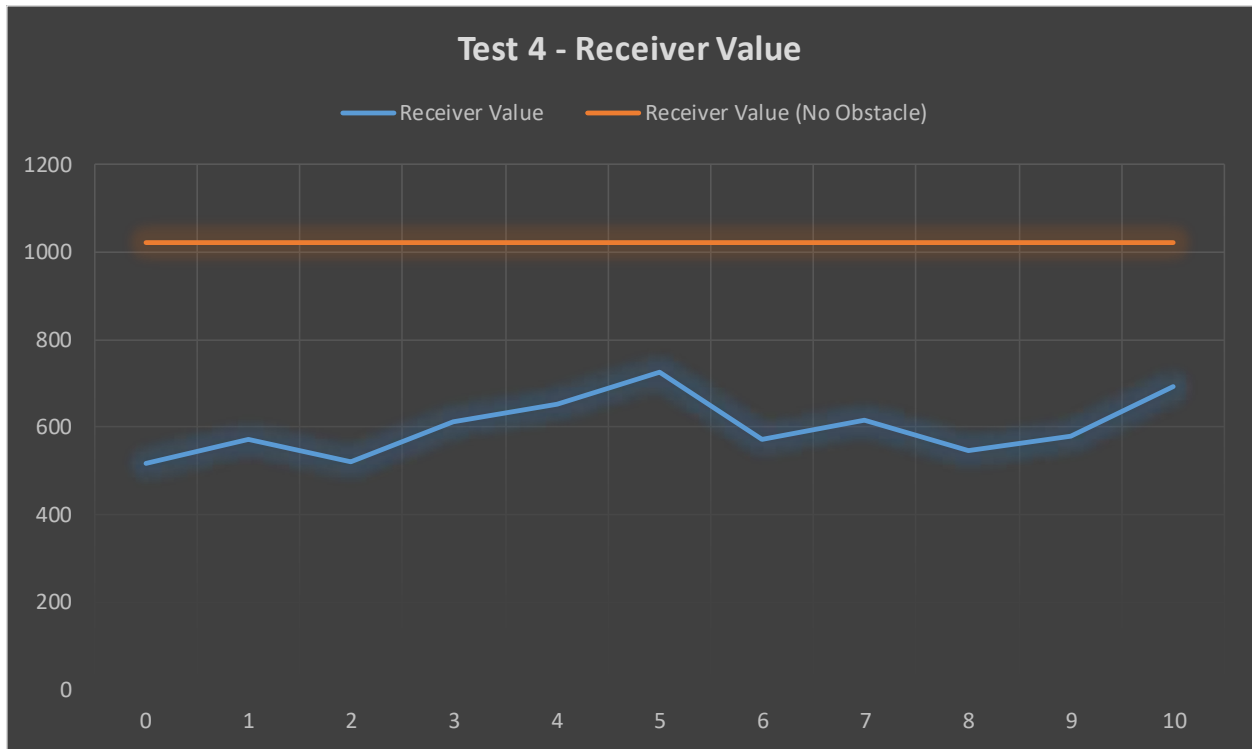
*Table 9.9 Obstacle Detection Test 3*



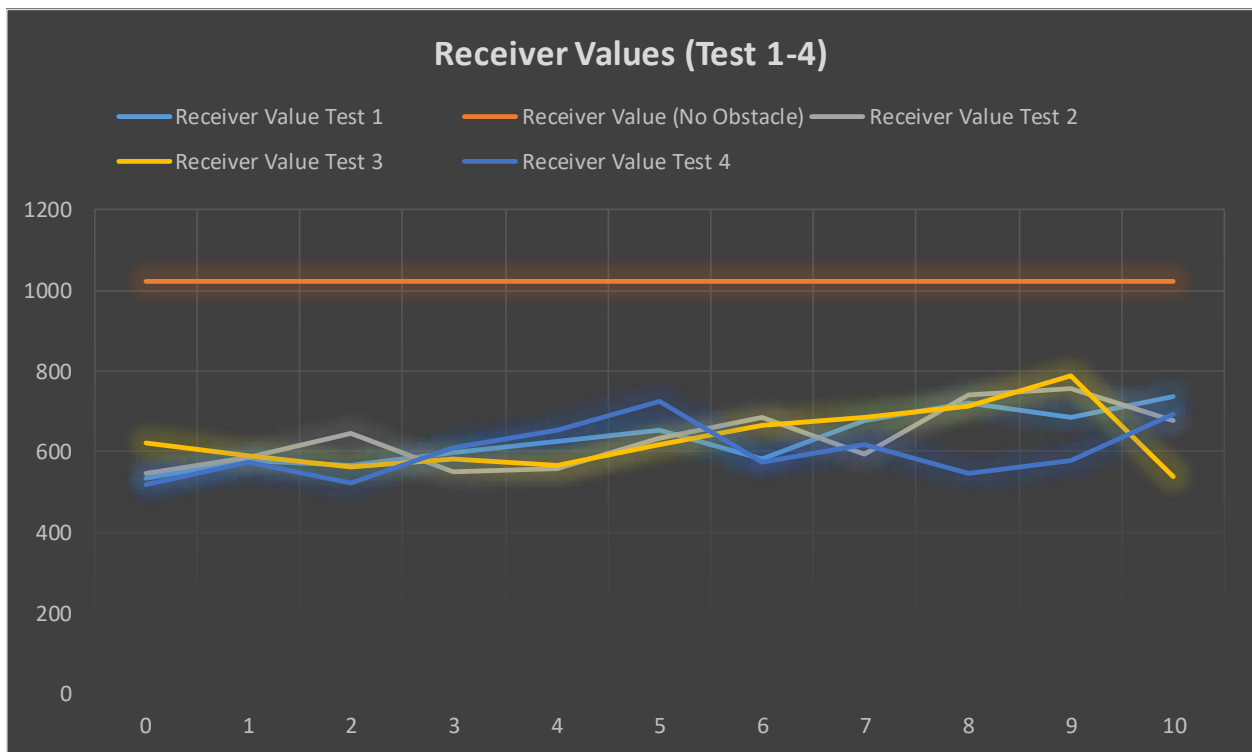
*Figure 9.8 Obstacle Detection Test 3 Graph*

Test 4		
No obstacle reading = 1024		
Reading no.	Obstacle position (in meters)	Receiver Value
1	0	519
2	1	574
3	2	522
4	3	611
5	4	652
6	5	725
7	6	574
8	7	618
9	8	546
10	9	578
11	10	694

*Table 9.10 Obstacle Detection Test 4*



*Figure 9.9 Obstacle Detection Test 4 Graph*



*Figure 9.9 Obstacle Detection Combined Graph*

## **9.4 – Website Test**

Website tests were carried out by changing the codes and executing the program over and over again.

### **9.4.1 - Location Upload Test**

An array of different latitudes and longitudes were declared in the train system master core and were uploaded from there. It was observed that the pointer on the map changes its position at approximately 10 seconds interval. This means that the new location of the trains can be seen at 10 seconds interval. Initially the time delay was 25 seconds but later by efficient coding, the delay has been reduced as much as possible.

### **9.4.2 – Test of sending SMS Command to trains**

SMS were sent to a train several times from the web site. It was observed that it took 11 seconds on an average for a SMS message to reach to the train. No SMS were missed.

### **9.4.3 – Server Test**

As it was mentioned earlier that the server used for the web site is Apache Server. Nevertheless, the web application was first tested in WAMP server and then later transferred to CPanel in the Apache for production environment when it was ensured that the client-server communication was working perfectly.

### **9.4.4 – Testing on Local Host**

All sorts of testing with the web site were done in a local host ftp server. When it was observed that the website was fully functional only then the domain name was bought for one year.

## CHAPTER 10 – COSTS

### 10.1 – Train System

Sl.no	Item	Qty	Unit Price in BDT	Total Price in BDT
1	Arduino MEGA2560	2	1000	2000
2	SIM808 Module	2	1500	3000
3	20X4 LCD	4	330	1320
4	4X4 Matrix Keypad	2	80	160
5	GPS Antenna	1	100	100
6	GSM Antenna	2	50	100
7	LM2596 Module	1	250	250
8	Potentiometer	4	20	80
9	DC Socket	1	10	10
10	DC Plug	2	10	20
11	Pushbutton switch	3	5	15
12	Toggle switch	2	15	30
13	LED	17	5	85
14	Female Conectors	25	20	500
15	Arduino Stackable Header Pins	20	20	400
16	Arduino Stackable Header Pins	8	33	264
17	Arduino Stackable Header Pins	5	25	125
18	Buzzer	1	20	20
19	PCB	1	4620	4620
20	12V, 3A Power Adapter	1	500	500
21	Grameenphone SIM card	2	100	200
			<b>Sub-Total</b>	<b>13799</b>

Cost of one unit of Train System = BDT 13799.00

Cost of 1GB Internet and 500SMS for 1 Month for one unit of train system= BDT 286.45

*Table 10.1 Table of Costs of Train System*

## 10.2 – Level Crossing System

Sl.no	Item	Qty	Unit Price in	Total Price in
			BDT	BDT
1	Arduino MEGA2560	1	1000	1000
2	SIM808 Module	1	1500	1500
3	20X4 LCD	1	330	330
4	4X4 Matrix Keypad	1	80	80
5	GPS Antenna	1	100	100
6	GSM Antenna	1	50	50
7	LM2596 Module	2	250	500
8	Potentiometer	1	20	20
9	DC Socket	1	10	10
10	Pushbutton switch	3	5	15
11	Toggle switch	3	15	45
12	LED	41	5	205
13	Connecors	20	20	400
14	Arduino Stackable Header Pins	5	20	100
15	Arduino Stackable Header Pins	2	33	66
16	Arduino Stackable Header Pins	2	25	50
17	Buzzer	2	20	40
18	IR Transmitters	12	5	60
19	IR Receivers	12	20	240
20	NE555	12	10	120
21	BD140	12	3	36
22	BC557	12	5	60
23	Resistors	100	0.5	50
24	Capacitors	36	5	180
25	12V, 3A Power Adapter	1	500	500
26	Grameenphone SIM card	1	100	100
27	Mainboard PCB	1	3665	3665



<b>28</b>	IR Transmitter PCB	1	645	645
<b>29</b>	IR Receiver PCB	1	940	940
			<b>Sub-Total</b>	<b>11107</b>

Cost of one unit of Train System = BDT 11107.00

Cost of 1GB Internet and 500SMS for 1 Month for one unit of train system= BDT 286.45

*Table 10.2 Table of Costs of Level Crossing System*

### 10.3 – Central Control System

<b>Sl.no</b>	<b>Item</b>	<b>Qty</b>	<b>Unit Price</b>	<b>Total Price</b>
<b>1</b>	Desktop/ Laptop	1	40000	40000
			<b>Sub-Total</b>	<b>40000</b>

Domain fee per month = BDT 900.00

Bulk SMS per month = BDT 500.00

Internet Connection per month = BDT 2000.00

*Table 10.3 Table of Costs for Central Control System*

### 10.4 – Full Package

<b>One Time Costs:</b>				
<b>Sl.no</b>	<b>Item</b>	<b>Qty</b>	<b>Unit Price</b>	<b>Total Price</b>
<b>1</b>	Train System	1	13799	13799
<b>2</b>	Level Crossing System	1	11107	11107
<b>3</b>	Central Control System	1	40000	40000
			<b>Sub-Total</b>	<b>64906</b>

*Table 10.4 One Time Costs of entire System*

**One Time Cost of Full Package = BDT 64906.00**

<b>Monthly Costs:</b>				
<b>Sl.no</b>	<b>Item</b>	<b>Qty</b>	<b>Unit Price</b>	<b>Total Price</b>
<b>1</b>	Train System	1	286.45	286.45
<b>2</b>	Level Crossing System	1	286.45	286.45
<b>3</b>	Central Control System	1	3400	3400
			<b>Sub-Total</b>	<b>3972.9</b>

*Table 10.5 Monthly Cost of entire system*

# CHAPTER 11 – CONCLUSION

## 11.1 – Limitations

The only limitation of this system is the time delay. This is due to using single core microcontrollers with single threads and less number of bits in the architecture. Due to these low end microcontrollers our system is confined within certain boundaries and becomes bottlenecked having so much to do. This results into a lot of time delays of almost 15 to 20 seconds between each processes.

## 11.2 – Challenges Faced

A countable number of challenges were faced while working on this project. The most substantial challenges were designing the functions of the whole system and choosing the right components for each of the functions. These are the two major aspects where ample of time were consumed due to extensive research. Additionally, challenges were also faced in learning the use of the SIM808 modules, creating the IR transmitter and receiver circuits, coding, developing the website and designing the PCBs.

## 11.3 – Future Works & Improvements

We planned to add some more important features to this system which are as follows:

- Developing electronic brakes for the trains
- The ability of trains to contact directly to emergency services rather via the Central Control System
- Detect obstacles at least 1Km ahead of the train all along the track rather detecting only at level crossings. Obstacles would also be detected along a curved path.
- The ability of trains to adjust to an optimum speed automatically in order to prevent derailment due to over speeding.
- Maintenance Teams would have the ability to block a portion of the train track in order to resolve an issue. The blockade would be seen in the Google Map API and trains on that track would be notified automatically.
- An option to have voice communications with operators at the Central Control System and the train drivers.

- Reduce time delays as much as possible
- Use more powerful microprocessor
- Reducing size of PCBs

## **11.4 – Summary**

In a nutshell, it was a very helpful experience throughout the whole thesis project. Bangladesh Railway is rapidly growing and is well invested into by the government in the recent years. This thesis project gave us a unique experience of amalgamating both the computer science field with the electronics, communication and networking fields of engineering and bring out a significance in the development of a system that can be well implemented in the real-life scenario. Our team is looking forward to learn the concept of both electronic engineering and computer science more extensively and hopefully work with bigger organizations and bigger projects in different field of works and incorporate our ideas and expertise to produce high-end results and quality systems for the upcoming future.

## REFERENCES

- [1] Positive Train Control. (n.d.). Retrieved April 20, 2017, from [http://www.up.com/media/media\\_kit/ptc/about-ptc/](http://www.up.com/media/media_kit/ptc/about-ptc/)
- [2] Bangladesh Railway. (2017, April 19). Retrieved April 20, 2017, from [https://en.wikipedia.org/wiki/Bangladesh\\_Railway](https://en.wikipedia.org/wiki/Bangladesh_Railway)  
[https://en.wikipedia.org/wiki/Bangladesh\\_Railway](https://en.wikipedia.org/wiki/Bangladesh_Railway)
- [3] Parveen, S. (2010, December 09). 2010 sees 590 train accidents. Retrieved April 20, 2017, from <http://www.thedailystar.net/news-detail-165506>
- [4] Railway. (n.d.). Retrieved April 20, 2017, from <http://en.banglapedia.org/index.php?title=Railway>
- [5] Analysis of Problems |Bangladesh Railway-Government of the People of Republic Bangladesh | বাংলাদেশ রেলওয়ে-গণপ্রজাতন্ত্রী বাংলাদেশ সরকার. (n.d.). Retrieved April 20, 2017, from <http://www.railway.gov.bd/site/page/f4460242-15eb-47b6-ba91-dc8f2897cea9/-Analysis-of-Problems>
- [6] Positive Train Control and GNSS Railway Enhancement. (n.d.). Retrieved April 21, 2017, from <https://scpnt.stanford.edu/research/current-research/positive-train-control>
- [7] Positive Train Control. (n.d.). Retrieved April 21, 2017, from <https://www.fra.dot.gov/ptc>
- [8] van Roon, Fig 3 & related text.
- [9] Scherz, Paul (2000) "Practical Electronics for Inventors", p. 589. McGraw-Hill/TAB Electronics. ISBN 978-0-07-058078-7. Retrieved 2010-04-05.
- [10] Jung, Walter G. (1983) "IC Timer Cookbook, Second Edition", pp. 40–41. Sams Technical Publishing; 2nd ed. ISBN 978-0-672-21932-0. Retrieved 2010-04-05.
- [11] Ward, Jack (2004). The 555 Timer IC – An Interview with Hans Camenzind. The Semiconductor Museum. Retrieved 2010-04-05
- [12] SIM808. (n.d.). Retrieved April 22, 2017, from <http://simcom.ee/modules/gsm-gprs-gnss/sim808/>

- [13] Brühlmann, Thomas (2010). Arduino: Praxiseinstieg. Hüthig Jehle Rehm. p. 270. ISBN 978-3-8266-5605-7.
- [14] Gläser, Thomas; Markus Jaritz; Philipp Sackl (13 September 2009). "Hardware-Hacking: So baut man einen Tentakel-Roboter für 100 Euro". Der Spiegel. Retrieved 14 April 2011
- [15] <http://blog.fritzing.org/2014/12/02/its-fritzmas-new-fritzing-code-view-release-and-a-little-present>.
- [16] IEEEExplore White Paper (May 2011) "Application of Proteus VSM in modelling brushless DC motor drives."
- [17] IEEEExplore White Paper (Dec 2006)."An efficient approach for implementing Space Vector Modulation for controlling induction motor."
- [18] IEEEExplore White Paper (Aug 2011)."The simulation of temperature and humidity control system based on Proteus."
- [19] IEEEExplore White Paper (Aug 2011)."Design of thermostat system based on Proteus simulation software."
- [20] IEEEExplore White Paper (Dec. 2010)."LED Display Screen Design and Proteus Simulation Based on Single-Chip Microcomputer."
- [21] Circuits Gallery (October 2014). "Arduino and Proteus VSM".
- [22] Elecnote Hobby Projects."Electronic circuits based on PIC microcontrollers and Arduino boards".
- [23] Online Training with Microchip and Proteus VSM "Get Started with MPLAB® X IDE and Microchip Tools".
- [24] Future Engineers Proteus VSM projects."Online Training Projects".
- [25] IEEEExplore White Paper (June 2011)."The application of Proteus in teaching of microcomputer principles"
- [26] IEEEExplore White Paper (April 2010)."Application of Proteus virtual system modelling (VSM) in teaching of microcontroller".

- [27] Gruending, N. (11 December 2012). "PCB Software Comparison".
- [28] "Dreamweaver system requirements".. Retrieved on 2013-07-21.
- [29] "Adobe Completes Acquisition of Macromedia" (PDF). Press Releases. Adobe, Inc. Retrieved 15 November 2011.
- [30] (n.d.). Retrieved April 22, 2017, from <http://kb2.adobe.com/cps/402/kb402489.html>
- [31] David Feugey (2014-09-22). "Avec PhpStorm 8, JetBrains renforce sa présence sur le marché PHP professionne". Silicon.fr. Retrieved 2015-10-12.
- [32] Darryl K. Taft (2012-09-13). "JetBrains PhpStorm 5.0 Provides New PHP Framework Support". Eweek. Retrieved 2013-02-19.
- [33] Adrian Bridgwater (2012-09-18). "JetBrains PhpStorm 5.0 Aligns To Symfony2 and Yii". Dr.Dobbs. Retrieved 2013-02-19.
- [34] "PhpStorm vs WebStorm".
- [35] "PhpStorm". Retrieved 2013-04-21. PhpStorm includes all the functionality of WebStorm (HTML/CSS Editor, JavaScript Editor) and adds full-fledged support for PHP and Databases/SQL.
- [36] "Which IDE do I need?". PhpStorm FAQ. Retrieved 2013-04-21.
- [37] Bruno Skvorc (2012-08-13). "PhpStorm – Review and Give Away". PHP Master. Retrieved 2013-02-19.
- [38] Yulia Tsuba (2015-02-12). "PHPStorm Live Templates for Drupal". X-Team. Retrieved 2015-10-12.
- [39] Chris Cornutt (2015-05-07). "Gary Hockin: Debugging PHP Command Line (with PHPStorm and XDebug)". PHPDeveloper.org. Retrieved 2015-10-12.
- [40] Netcraft Market Share for Top Servers Across All Domains August 1995 - today (monthly updated)
- [41] "February 2009 Web Server Survey". Netcraft. Archived from the original on 26 February 2009. Retrieved 2009-03-29.

[42] "What is Apache Web Server? Webopedia". webopedia.com.

[43] E. (2017, March 18). IC TSOP1738. Retrieved April 25, 2017, from <https://www.engineersgarage.com/electronic-components/tsop1738-datasheet>



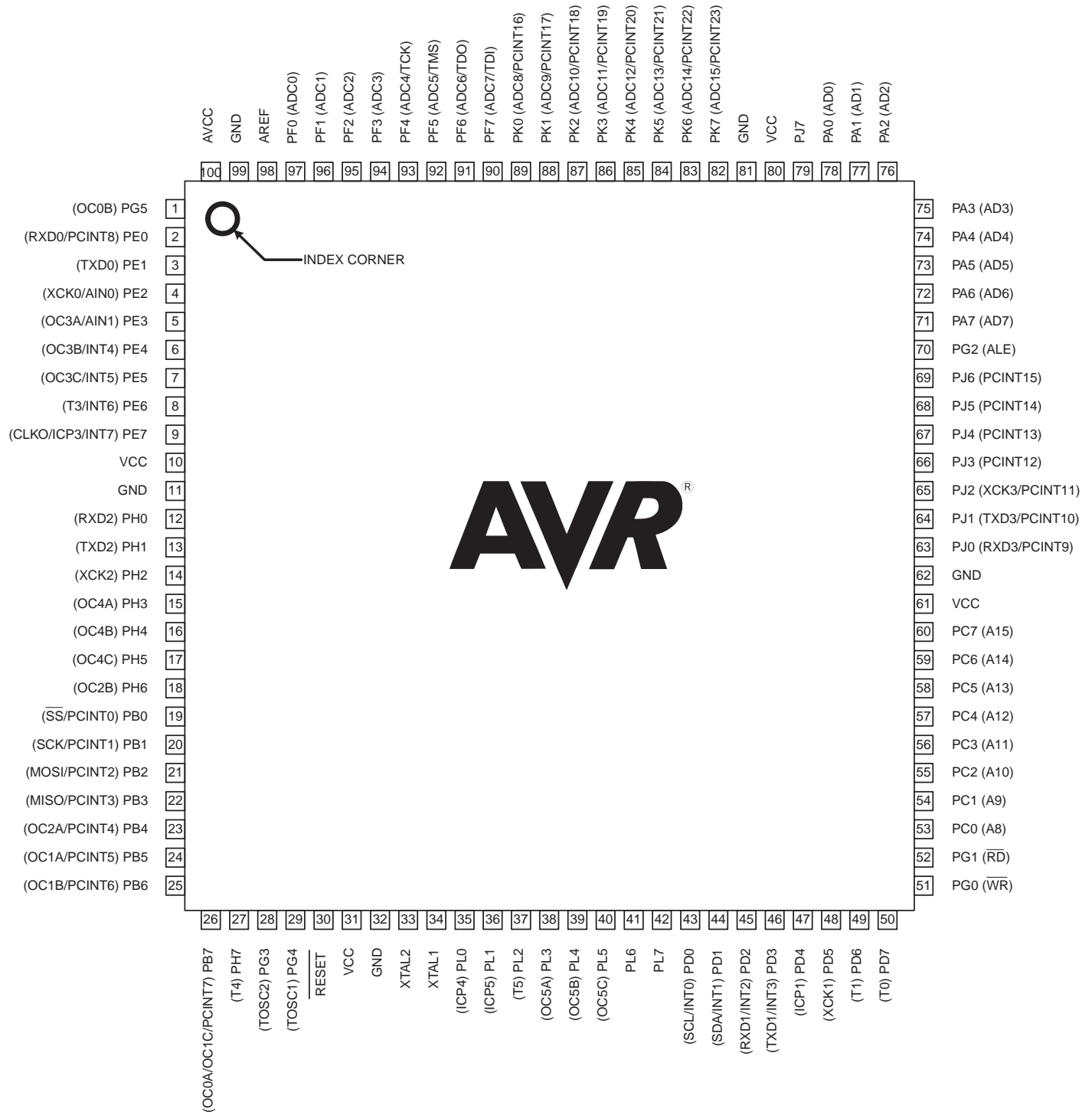
## **APPENDIX**

## Features

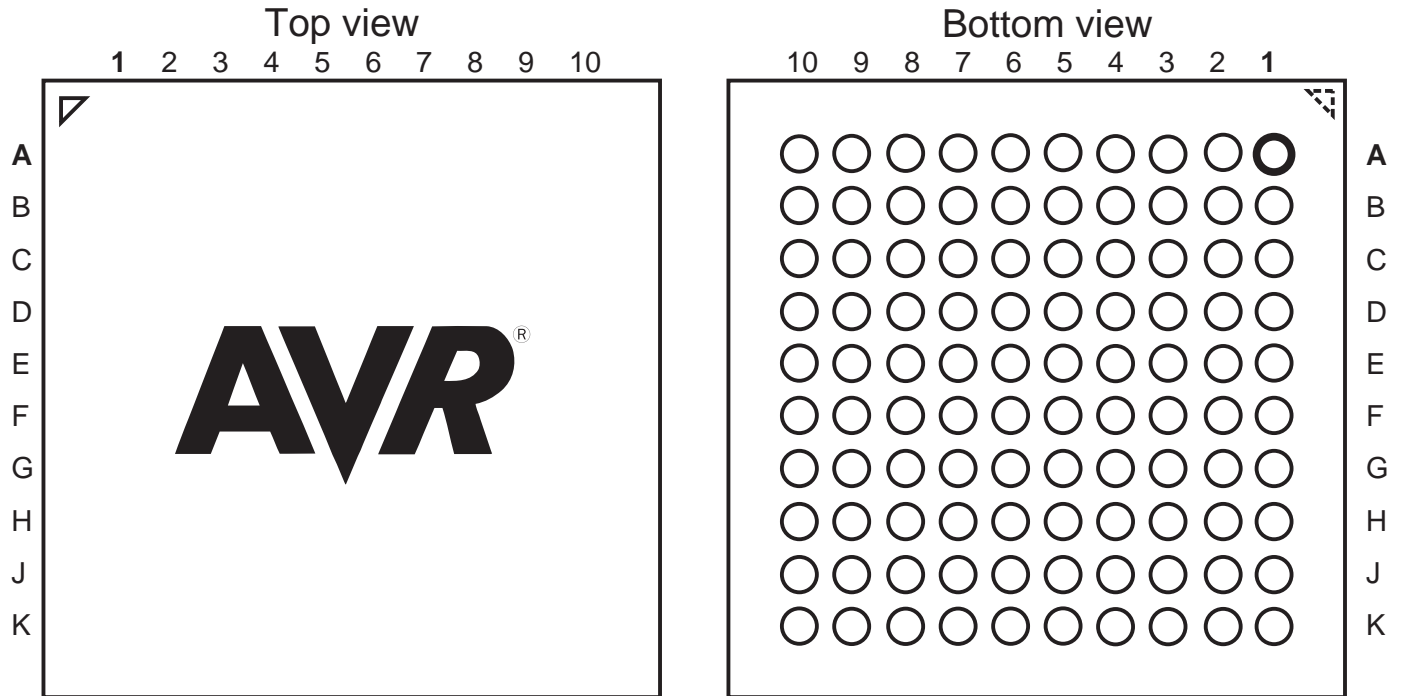
- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 64K/128K/256KBytes of In-System Self-Programmable Flash
  - 4Kbytes EEPROM
  - 8Kbytes Internal SRAM
  - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
    - Endurance: Up to 64Kbytes Optional External Memory Space
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix acquisition
  - Up to 64 sense channels
- JTAG (IEEE® std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
  - RoHS/Fully Green
- Temperature Range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode: 1MHz, 1.8V: 500µA
  - Power-down Mode: 0.1µA at 1.8V
- Speed Grade:
  - ATmega640V/ATmega1280V/ATmega1281V:
    - 0 - 4MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega2560V/ATmega2561V:
    - 0 - 2MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega640/ATmega1280/ATmega1281:
    - 0 - 8MHz @ 2.7V - 5.5V, 0 - 16MHz @ 4.5V - 5.5V
  - ATmega2560/ATmega2561:
    - 0 - 16MHz @ 4.5V - 5.5V

# 1. Pin Configurations

Figure 1-1. TQFP-pinout ATmega640/1280/2560



**Figure 1-2.** CBGA-pinout ATmega640/1280/2560

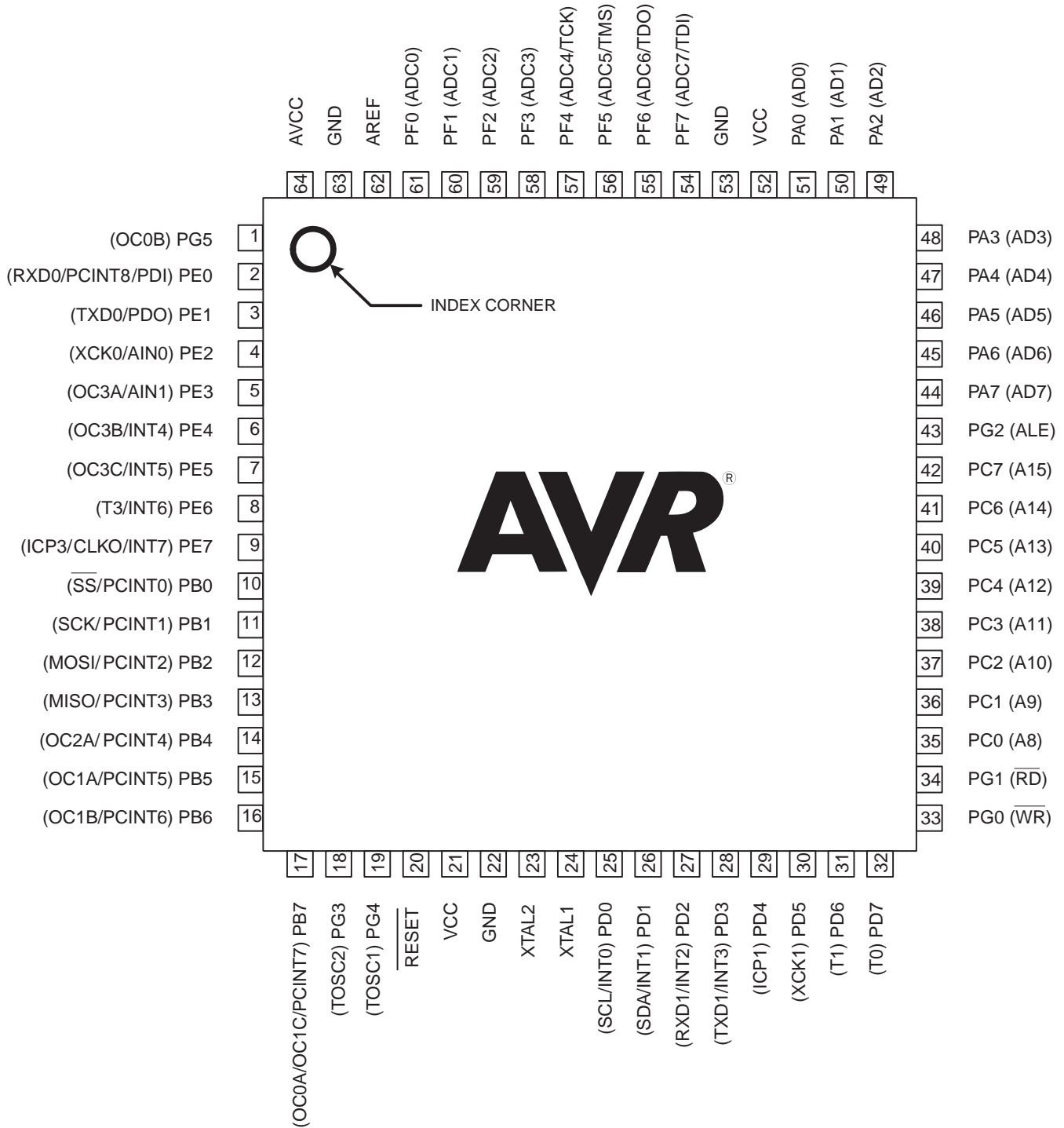


**Table 1-1.** CBGA-pinout ATmega640/1280/2560

	1	2	3	4	5	6	7	8	9	10
A	GND	AREF	PF0	PF2	PF5	PK0	PK3	PK6	GND	VCC
B	AVCC	PG5	PF1	PF3	PF6	PK1	PK4	PK7	PA0	PA2
C	PE2	PE0	PE1	PF4	PF7	PK2	PK5	PJ7	PA1	PA3
D	PE3	PE4	PE5	PE6	PH2	PA4	PA5	PA6	PA7	PG2
E	PE7	PH0	PH1	PH3	PH5	PJ6	PJ5	PJ4	PJ3	PJ2
F	VCC	PH4	PH6	PB0	PL4	PD1	PJ1	PJ0	PC7	GND
G	GND	PB1	PB2	PB5	PL2	PD0	PD5	PC5	PC6	VCC
H	PB3	PB4	RESET	PL1	PL3	PL7	PD4	PC4	PC3	PC2
J	PH7	PG3	PB6	PL0	XTAL2	PL6	PD3	PC1	PC0	PG1
K	PB7	PG4	VCC	GND	XTAL1	PL5	PD2	PD6	PD7	PG0

Note: The functions for each pin is the same as for the 100 pin packages shown in [Figure 1-1 on page 2](#).

**Figure 1-3.** Pinout ATmega1281/2561



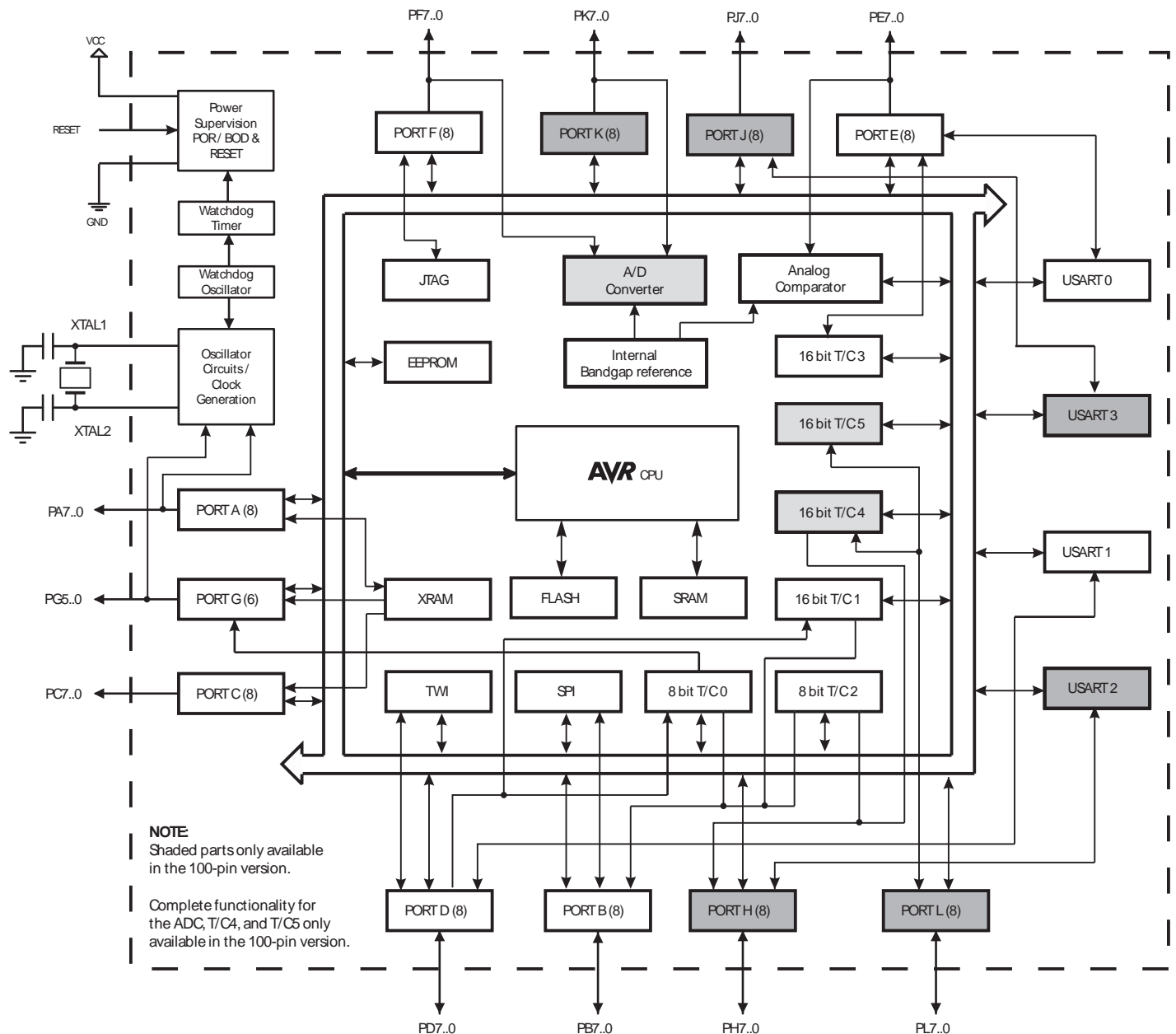
Note: The large center pad underneath the QFN/MLF package is made of metal and internally connected to GND. It should be soldered or glued to the board to ensure good mechanical stability. If the center pad is left unconnected, the package might loosen from the board.

## 2. Overview

The ATmega640/1280/1281/2560/2561 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega640/1280/1281/2560/2561 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

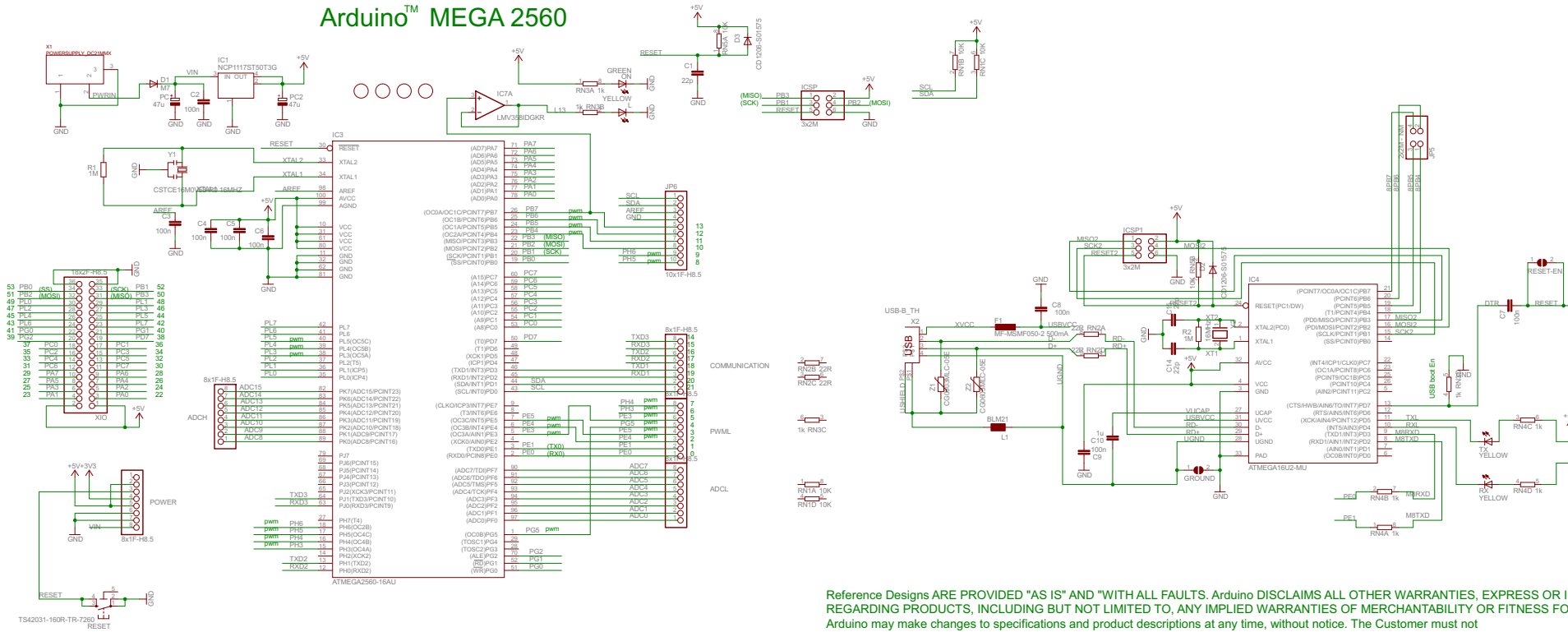
### 2.1 Block Diagram

Figure 2-1. Block Diagram



The Atmel® AVR® core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

# Arduino™ MEGA 2560

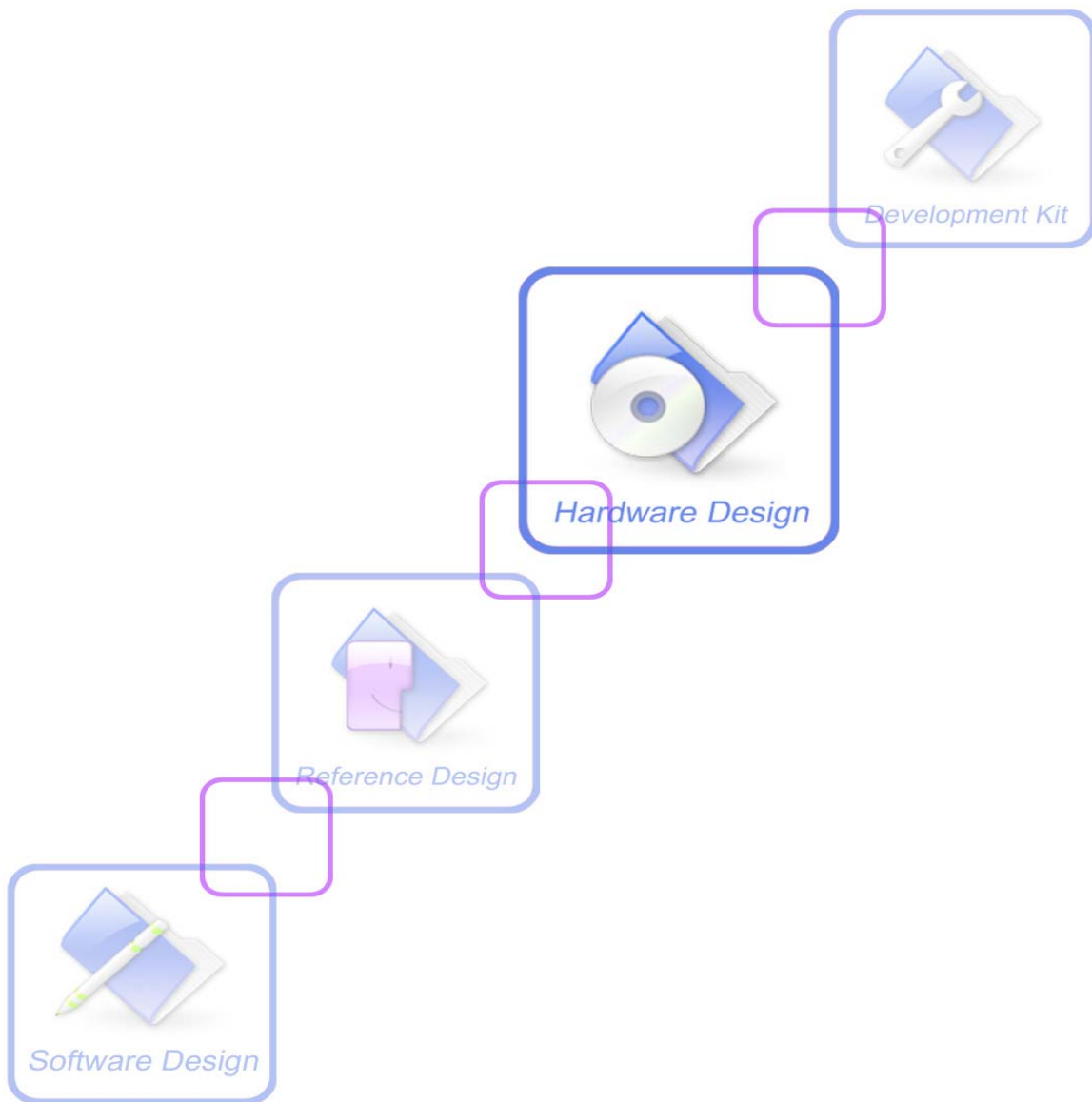


Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS. Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information. ARDUINO is a registered trademark.



A company of SIM Tech

# SIM808\_Hardware Design\_V1.00





# 1 Introduction

This document describes SIM808 hardware interface in great detail. This document can help user to quickly understand SIM808 interface specifications, electrical and mechanical details. With the help of this document and other SIM808 application notes, user guide, users can use SIM808 to design various applications quickly.

## 2 SIM808 Overview

Designed for global market, SIM808 is integrated with a high performance GSM/GPRS engine, a GPS engine and a BT engine. The GSM/GPRS engine is a quad-band GSM/GPRS module that works on frequencies GSM 850MHz, EGSM 900MHz, DCS 1800MHz and PCS 1900MHz. SIM808 features GPRS multi-slot class 12/ class 10 (optional) and supports the GPRS coding schemes CS-1, CS-2, CS-3 and CS-4. The GPS solution offers best-in-class acquisition and tracing sensitivity, Time-To-First-Fix (TTFF) and accuracy.

With a tiny configuration of 24\*24\*2.6mm, SIM808 can meet almost all the space requirements in user applications, such as M2M, smart phone, PDA, tracker and other mobile devices.

SIM808 has 68 SMT pads, and provides all hardware interfaces between the module and customers' boards.

- Support 4\*4\*2 keypads.
- One full modem serial port.
- One USB, the USB interfaces can debug, download software.
- Audio channels which include a microphone input and a receiver output.
- One SIM card interface.
- Charging interface.
- Programmable general purpose input and output.
- Support Bluetooth function.
- Support PWM and ADC.
- PCM/SPI/SD card interface, only one function can be accessed synchronously. (Default function is PCM).

SIM808 is designed with power saving technique so that the current consumption is as low as 1mA in sleep mode (GPS engine is powered down).

SIM808 integrates TCP/IP protocol and extended TCP/IP AT commands which are very useful for data transfer applications. For details about TCP/IP applications, please refer to *document [2]*.

### 2.1 SIM808 Key Features

**Table 1: SIM808 GSM/GPRS engine key features**

Feature	Implementation
Power supply	3.4V ~ 4.4V
Power saving	Typical power consumption in sleep mode is 1mA ( BS-PA-MFRMS=9, GPS engine is powered down )
Charging	Supports charging control for Li-Ion battery
Frequency bands	● SIM808 Quad-band: GSM 850, EGSM 900, DCS 1800, PCS 1900. SIM808

	<p>can search the 4 frequency bands automatically. The frequency bands also can be set by AT command “AT+CBAND”. For details, please refer to <i>document [1]</i>.</p> <ul style="list-style-type: none"> <li>● Compliant to GSM Phase 2/2+</li> </ul>
Transmitting power	<ul style="list-style-type: none"> <li>● Class 4 (2W) at GSM 850 and EGSM 900</li> <li>● Class 1 (1W) at DCS 1800 and PCS 1900</li> </ul>
GPRS connectivity	<ul style="list-style-type: none"> <li>● GPRS multi-slot class 12 ( default )</li> <li>● GPRS multi-slot class 1~12 (optional)</li> </ul>
Temperature range	<ul style="list-style-type: none"> <li>● Normal operation: -40°C ~ +85°C</li> <li>● Storage temperature -45°C~ +90°C</li> </ul>
Data GPRS	<ul style="list-style-type: none"> <li>● GPRS data downlink transfer: max. 85.6 kbps</li> <li>● GPRS data uplink transfer: max. 85.6 kbps</li> <li>● Coding scheme: CS-1, CS-2, CS-3 and CS-4</li> <li>● PAP protocol for PPP connect</li> <li>● Integrate the TCP/IP protocol.</li> <li>● Support Packet Broadcast Control Channel (PBCCH)</li> <li>● CSD transmission rates: 2.4, 4.8, 9.6, 14.4 kbps</li> </ul>
CSD	<ul style="list-style-type: none"> <li>● Support CSD transmission</li> </ul>
USSD	<ul style="list-style-type: none"> <li>● Unstructured Supplementary Services Data (USSD) support</li> </ul>
SMS	<ul style="list-style-type: none"> <li>● MT, MO, CB, Text and PDU mode</li> <li>● SMS storage: SIM card</li> </ul>
SIM interface	Support SIM card: 1.8V, 3V
External antenna	Antenna pad
Audio features	<p>Speech codec modes:</p> <ul style="list-style-type: none"> <li>● Half Rate (ETS 06.20)</li> <li>● Full Rate (ETS 06.10)</li> <li>● Enhanced Full Rate (ETS 06.50 / 06.60 / 06.80)</li> <li>● Adaptive multi rate (AMR)</li> <li>● Echo Cancellation</li> <li>● Noise Suppression</li> </ul>
Serial port and USB interface	<p><b>Serial port:</b></p> <ul style="list-style-type: none"> <li>● Full modem interface with status and control lines, unbalanced, asynchronous.</li> <li>● 1200bps to 115200bps.</li> <li>● Can be used for AT commands or data stream.</li> <li>● Support RTS/CTS hardware handshake and software ON/OFF flow control.</li> <li>● Multiplex ability according to GSM 07.10 Multiplexer Protocol.</li> <li>● Autobauding supports baud rate from 1200 bps to 115200bps.</li> </ul> <p><b>USB interface:</b></p> <ul style="list-style-type: none"> <li>● Can be used as debugging and firmware upgrading.</li> </ul>
Phonebook management	Support phonebook types: SM, FD, LD, RC, ON, MC.
SIM application toolkit	GSM 11.14 Release 99
Real time clock	Support RTC
Alarm function	Can be set by AT command
Physical characteristics	<p>Size: 24*24*2.6mm</p> <p>Weight: 3.5g</p>

Firmware upgrade

Firmware upgrading by USB interface.

**Table 2: GPS engine Performance**

Parameter	Description	Performance			
		Min	Type	Max	Unit
Horizontal Position Accuracy(1)	Autonomous		<2.5		m
Velocity Accuracy(2)	Without Aid		0.1		m/s
	DGPS		0.05		m/s
Acceleration Accuracy	Without Aid		0.1		m/s <sup>2</sup>
	DGPS		0.05		m/s <sup>2</sup>
Timing Accuracy			10		nS
Dynamic Performance	Maximum Altitude			18000	m
	Maximum Velocity			515	m/s
	Maximum Acceleration			4	G
Time To First Fix <sup>(3)</sup>	Hot start		1		s
	Warm start		28		s
	Cold start		30		s
Sensitivity	Autonomous acquisition(cold start)		-147		dBm
	Re-acquisition		-159		dBm
	Tracking		-165		dBm
Receiver	Channels		22/66		
	Update rate			5	Hz
	Tracking L1, CA Code				
	Protocol support NMEA				
Power consumption <sup>(4)</sup>	Acquisition		42		mA
	Continuous tracking		24		mA

(1) 50% 24hr static, -130dBm

(2) 50% at 30m/s

(3) GPS signal level: -130dBm

(4) Single Power supply 3.8V@-130dBm,GSM IDLE

**Table 3: Coding schemes and maximum net data rates over air interface**

Coding scheme	1 timeslot	2 timeslot	4 timeslot
CS-1	9.05kbps	18.1kbps	36.2kbps
CS-2	13.4kbps	26.8kbps	53.6kbps
CS-3	15.6kbps	31.2kbps	62.4kbps
CS-4	21.4kbps	42.8kbps	85.6kbps

## 2.2 Operating Modes

The table below summarizes the various operating modes of SIM808.

**Table 4: Overview of operating modes**

Mode	Function
Normal operation	GSM/GPRS SLEEP Module will automatically go into sleep mode if the conditions of sleep mode are enabling and there is no on air and no hardware interrupt (such as GPIO interrupt or data on serial port). In this case, the current consumption of module will reduce to the minimal level. In sleep mode, the module can still receive paging message and SMS.
	GSM IDLE Software is active. Module registered to the GSM network, and the module is ready to communicate.
	GSM TALK Connection between two subscribers is in progress. In this case, the power consumption depends on network settings such as DTX off/on, FR/EFR/HR, hopping sequences, antenna.
	GPRS STANDBY Module is ready for GPRS data transfer, but no data is currently sent or received. In this case, power consumption depends on network settings and GPRS configuration.
	GPRS DATA There is GPRS data transfer (PPP or TCP or UDP) in progress. In this case, power consumption is related with network settings (e.g. power control level); uplink/downlink data rates and GPRS configuration (e.g. used multi-slot settings).
	Charge The mode support charge function (Default is not support).
Power down	Normal power down by sending the AT command “AT+CPOWD=1” or using the PWRKEY. The power management unit shuts down the power supply for the baseband part of the module, and only the power supply for the RTC is remained. Software is not active. The serial port is not accessible. Power supply (connected to VBAT) remains applied.
Minimum functionality mode	AT command “AT+CFUN” can be used to set the module to a minimum functionality mode without removing the power supply. In this mode, the RF part of the module will not work or the SIM card will not be accessible, or both RF part and SIM card will be closed, and the serial port is still accessible. The power consumption in this mode is lower than normal mode.

### 2.3 SIM808 Functional Diagram

The following figure shows a functional diagram of SIM808:

- The GSM baseband engine
- The GPS engine
- Flash
- The GSM radio frequency part
- The antenna interface
- The other interfaces

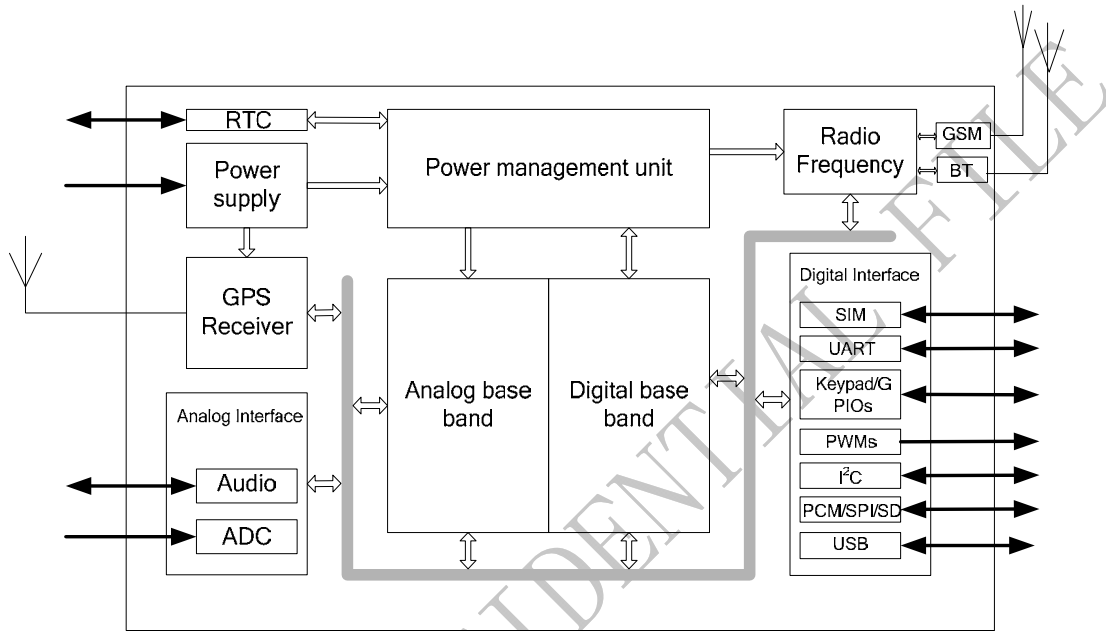


Figure 1: SIM808 functional diagram

### 3 Package Information

#### 3.1 Pin out Diagram

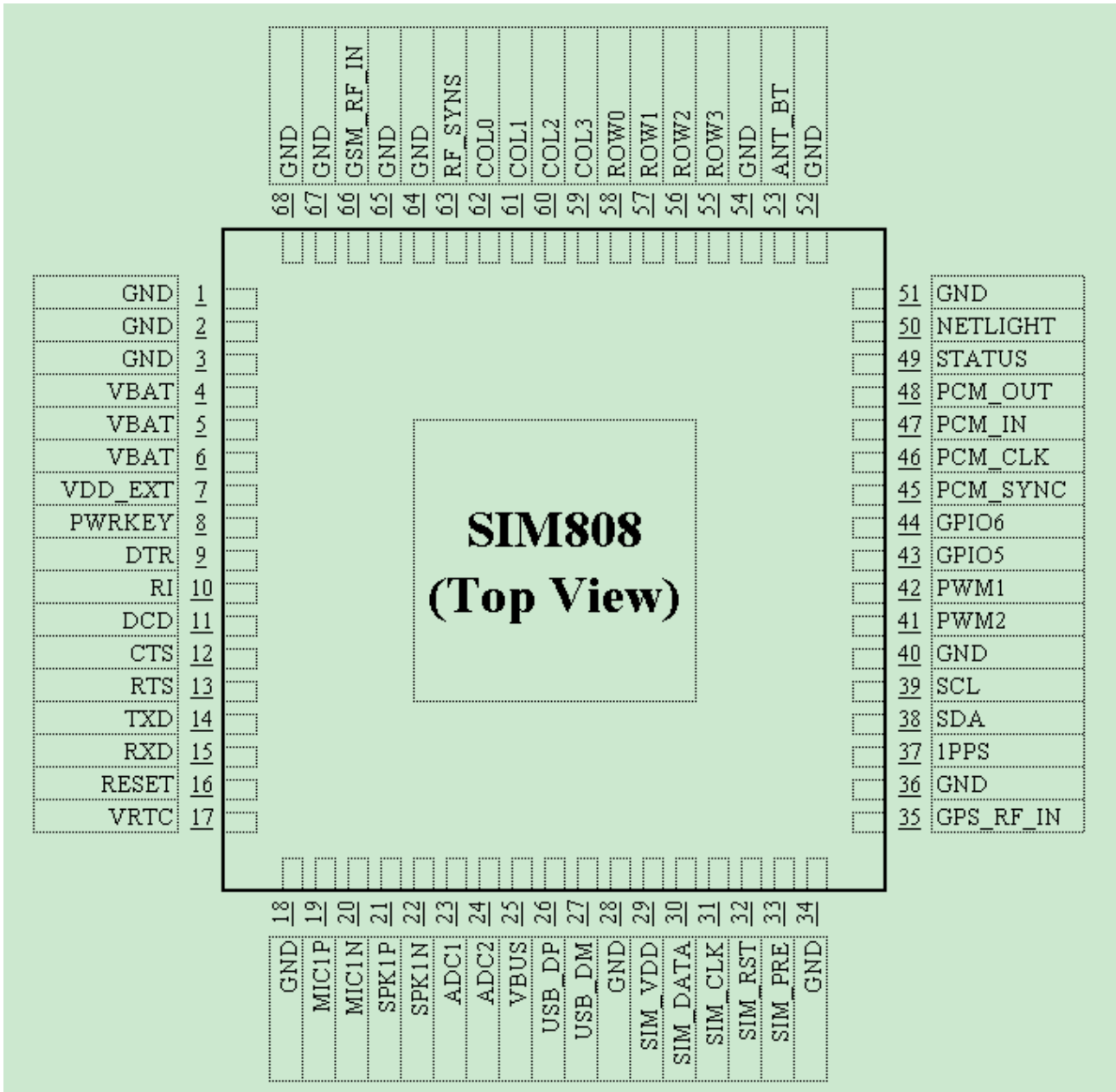
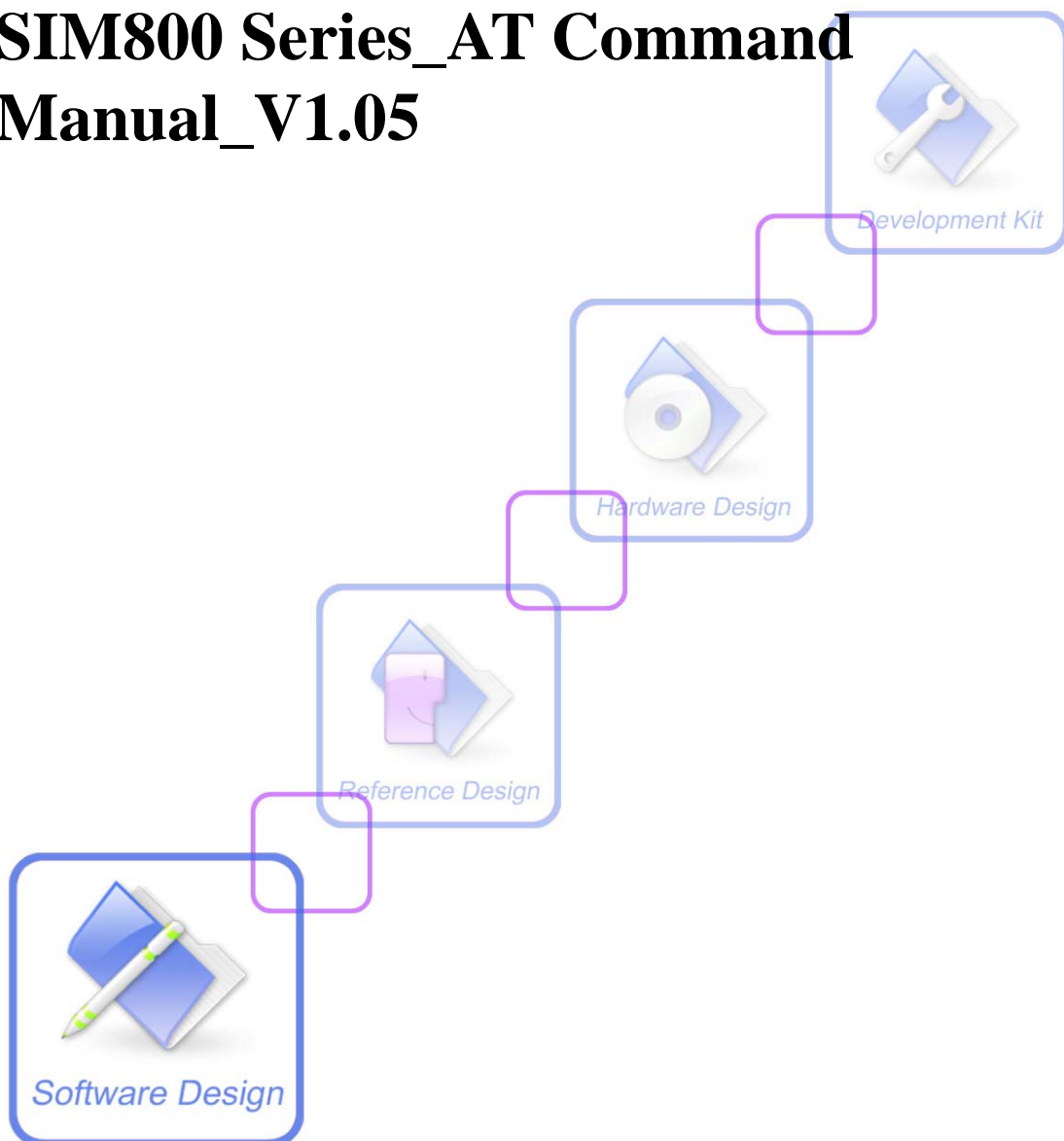


Figure 2: SIM808 pin out diagram (Top view)



A company of SIM Tech

# SIM800 Series\_AT Command Manual\_V1.05



## 4 AT Commands According to 3GPP TS 27.005

The 3GPP TS 27.005 commands are for performing SMS and CBS related operations. SIM800 Series supports both Text and PDU modes.

### 4.1 Overview of AT Commands According to 3GPP TS 27.005

Command	Description
AT+CMGD	Delete SMS message
AT+CMGF	Select SMS message format
AT+CMGL	List SMS messages from preferred store
AT+CMGR	Read SMS message
AT+CMGS	Send SMS message
AT+CMGW	Write SMS message to memory
AT+CMSS	Send SMS message from storage
AT+CNMI	New SMS message indications
AT+CPMS	Preferred SMS message storage
AT+CRES	Restore SMS settings
AT+CSAS	Save SMS settings
AT+CSCA	SMS service center address
AT+CSCB	Select cell broadcast SMS messages
AT+CSDH	Show SMS text mode parameters
AT+CSMP	Set SMS text mode parameters
AT+CSMS	Select message service

### 4.2 Detailed Descriptions of AT Commands According to 3GPP TS 27.005

#### 4.2.1 AT+CMGD Delete SMS Message

AT+CMGD Delete SMS Message	
Test Command AT+CMGD=?	Response +CMGD: (list of supported <index>s),(list of supported <delflag>s)  OK
Write Command AT+CMGD=<index>[,<delflag>]	Parameters See Write Command
Write Command AT+CMGD=<index>[,<delflag>]	Response TA deletes message from preferred message storage <mem1> location <index>. OK



	<p><b>ERROR</b></p> <p>If error is related to ME functionality: <b>+CMS ERROR:&lt;err&gt;</b></p> <p>Parameters</p> <p><b>&lt;index&gt;</b> Integer type; value in the range of location numbers supported by the associated memory</p> <p><b>&lt;delflag&gt;</b></p> <ul style="list-style-type: none"> <li>0 Delete the message specified in &lt;index&gt;</li> <li>1 Delete all read messages from preferred message storage, leaving unread messages and stored mobile originated messages (whether sent or not) untouched</li> <li>2 Delete all read messages from preferred message storage and sent mobile originated messages, leaving unread messages and unsent mobile originated messages untouched</li> <li>3 Delete all read messages from preferred message storage, sent and unsent mobile originated messages leaving unread messages untouched</li> <li>4 Delete all messages from preferred message storage including unread messages</li> </ul>
Parameter Saving Mode	NO_SAVE
Max Response Time	5s (delete 1 message) 25s (delete 50 messages) 25s (delete 150 messages)
Reference	Note
3GPP TS 27.005	

#### 4.2.2 AT+CMGF Select SMS Message Format

<b>AT+CMGF Select SMS Message Format</b>	
Test Command <b>AT+CMGF=?</b>	<p>Response</p> <p><b>+CMGF:</b> (list of supported <b>&lt;mode&gt;</b>s)</p> <p><b>OK</b></p> <p>Parameter See Write Command</p>
Read Command <b>AT+CMGF?</b>	<p>Response</p> <p><b>+CMGF:</b> <b>&lt;mode&gt;</b></p> <p><b>OK</b></p> <p>Parameter See Write Command</p>
Write Command	Response

<b>AT+CMGF=[&lt;mode&gt;]</b>	TA sets parameter to denote which input and output format of messages to use. <b>OK</b>
	Parameter <b>&lt;mode&gt;</b> <u>0</u> PDU mode 1 Text mode
Parameter Saving Mode	AT&W_SAVE
Max Response Time	-
Reference 3GPP TS 27.005	Note

#### 4.2.3 AT+CMGL List SMS Messages from Preferred Store

<b>AT+CMGL List SMS Messages from Preferred Store</b>	
Test Command <b>AT+CMGL=?</b>	Response <b>+CMGL:</b> (list of supported <stat>s)  <b>OK</b>
	Parameter See Write Command
Write Command <b>AT+CMGL=&lt;stat&gt;[,&lt;mode&gt;]</b>	Parameters 1) If text mode: <b>&lt;stat&gt;</b> <u>"REC UNREAD"</u> Received unread messages "REC READ"    Received read messages "STO UNSENT"    Stored unsent messages "STO SENT"      Stored sent messages "ALL"            All messages  <b>&lt;mode&gt;</b> <u>0</u> Normal 1 Not change status of the specified SMS record 2) If PDU mode: <b>&lt;stat&gt;</b> <u>0</u> Received unread messages 1 Received read messages 2 Stored unsent messages 3 Stored sent messages 4 All messages  <b>&lt;mode&gt;</b> <u>0</u> Normal 1 Not change status of the specified SMS record
	Response TA returns messages with status value <stat> from message storage <mem1> to the TE. If status of the message is 'received unread', status in the storage changes to 'received read'.

	<p><b>&lt;tooa&gt;</b> GSM 04.11 TP-Originating-Address Type-of-Address octet in integer format (default refer &lt;toda&gt;)</p> <p><b>&lt;tosca&gt;</b> GSM 04.11 RP SC address Type-of-Address octet in integer format (default refer &lt;toda&gt;)</p> <p><b>&lt;vp&gt;</b> Depending on SMS-SUBMIT &lt;fo&gt; setting: GSM 03.40 TP-Validity-Period either in integer format (default 167) or in time-string format (refer &lt;dt&gt;)</p>
Parameter Saving Mode	NO_SAVE
Max Response Time	5s
Reference 3GPP TS 27.005	Note

#### 4.2.5 AT+CMGS Send SMS Message

AT+CMGS Send SMS Message	
Test Command	Response
<b>AT+CMGS=?</b>	<b>OK</b>
Write Command	Parameters
1) If text mode (+CMGF=1): <b>+CMGS=&lt;da&gt;[, &lt;toda&gt;]</b> <b>&lt;CR&gt;text</b> is entered <b>&lt;ctrl-Z/ESC&gt;</b> ESC quits without sending	<p><b>&lt;da&gt;</b> GSM 03.40 TP-Destination-Address Address-Value field in string format(string should be included in quotation marks); BCD numbers (or GSM default alphabet characters) are converted to characters of the currently selected TE character set (specified by +CSCS in 3GPP TS 27.007); type of address given by &lt;toda&gt;</p> <p><b>&lt;toda&gt;</b> GSM 04.11 TP-Destination-Address Type-of-Address octet in integer format (when first character of &lt;da&gt; is + (IRA 43) default is 145, otherwise default is 129)</p> <p><b>&lt;length&gt;</b> Integer type value (not exceed 160 bytes) indicating in the text mode (+CMGF=1) the length of the message body &lt;data&gt; (or &lt;cdata&gt;) in characters; or in PDU mode (+CMGF=0), the length of the actual TP data unit in octets (i.e. the RP layer SMSC address octets are not counted in the length)</p>
2) If PDU mode (+CMGF=0): <b>+CMGS=&lt;length&gt;</b> <b>&gt;</b> <b>&lt;CR&gt;PDU</b> is given <b>&lt;ctrl-Z/ESC&gt;</b>	<p>Response</p> <p>TA sends message from a TE to the network (SMS-SUBMIT). Message reference value &lt;mr&gt; is returned to the TE on successful message delivery. Optionally (when +CSMS &lt;service&gt; value is 1 and network supports) &lt;scts&gt; is returned. Values can be used to identify message upon unsolicited delivery status report result code.</p> <p>1) If text mode(+CMGF=1) and sending successful:  <b>+CMGS: &lt;mr&gt;</b></p> <p><b>OK</b></p> <p>2) If PDU mode(+CMGF=0) and sending successful:</p>

#### 4.2.8 AT+CNMI New SMS Message Indications

AT+CNMI New SMS Message Indications	
Test Command <b>AT+CNMI=?</b>	Response <b>+CNMI:</b> (list of supported <mode>s),(list of supported <mt>s),(list of supported <bm>s),(list of supported <ds>s),(list of supported <bfr>s)  <b>OK</b>  Parameters See Write Command
Read Command <b>AT+CNMI?</b>	Response <b>+CNMI:</b> <mode>,<mt>,<bm>,<ds>,<bfr>  <b>OK</b>  Parameters See Write Command
Write Command <b>AT+CNMI=&lt;mode&gt;[,&lt;mt&gt;[,&lt;bm&gt;[,&lt;ds&gt;[,&lt;bfr&gt;]]]]</b>	Response TA selects the procedure for how the receiving of new messages from the network is indicated to the TE when TE is active, e.g. DTR signal is ON. If TE is inactive (e.g. DTR signal is OFF), message receiving should be done as specified in GSM 03.38.  <b>OK</b> <b>ERROR</b>  Parameters <b>&lt;mode&gt;</b> 0     Buffer unsolicited result codes in the TA. If TA result code buffer is full, indications can be buffered in some other place or the oldest indications may be discarded and replaced with the new received indications. 1     Discard indication and reject new received message unsolicited result codes when TA-TE link is reserved (e.g. in on-line data mode). Otherwise forward them directly to the TE. 2     Buffer unsolicited result codes in the TA when TA-TE link is reserved (e.g. in on-line data mode) and flush them to the TE after reservation. Otherwise forward them directly to the TE. 3     Forward unsolicited result codes directly to the TE. TA-TE link specific inband technique used to embed result codes and data when TA is in on-line data mode. <b>&lt;mt&gt;</b> (the rules for storing received SMS depend on its data coding scheme (refer GSM 03.38 [2]), preferred memory storage (+CPMS) setting and this value): 0     No SMS-DELIVER indications are routed to the TE.

## 7 AT Commands for GPRS Support

### 7.1 Overview of AT Commands for GPRS Support

Command	Description
AT+CGATT	Attach or detach from GPRS service
AT+CGDCONT	Define PDP context
AT+CGQMIN	Quality of service profile (minimum acceptable)
AT+CGQREQ	Quality of service profile (requested)
AT+CGACT	PDP context activate or deactivate
AT+CGDATA	Enter data state
AT+CGPADDR	Show PDP address
AT+CGCLASS	GPRS mobile station class
AT+CGEREP	Control unsolicited GPRS event reporting
AT+CGREG	Network registration status
AT+CGSMS	Select service for MO SMS messages

### 7.2 Detailed Descriptions of AT Commands for GPRS Support

#### 7.2.1 AT+CGATT Attach or Detach from GPRS Service

AT+CGATT Attach or Detach from GPRS Service	
Test Command AT+CGATT=?	<p>Response</p> <p>+CGATT: (list of supported &lt;state&gt;s)</p> <p><b>OK</b></p> <p>Parameters</p> <p>See Write Command</p>
Read Command AT+CGATT?	<p>Response</p> <p>+CGATT: &lt;state&gt;</p> <p><b>OK</b></p> <p>Parameters</p> <p>See Write Command</p>
Write Command AT+CGATT=<state>	<p>Response</p> <p><b>OK</b></p> <p>If error is related to ME functionality: +CME ERROR: &lt;err&gt;</p> <p>Parameters</p> <p>&lt;state&gt;            Indicates the state of GPRS attachment</p>

## 8 AT Commands for TCPIP Application Toolkit

### 8.1 Overview

Command	Description
AT+CIPMUX	Start up multi-IP connection
AT+CIPSTART	Start up TCP or UDP connection
AT+CIPSEND	Send data through TCP or UDP connection
AT+CIPQSEND	Select data transmitting mode
AT+CIPACK	Query previous connection data transmitting state
AT+CIPCLOSE	Close TCP or UDP connection
AT+CIPSHUT	Deactivate GPRS PDP context
AT+CLPORT	Set local port
AT+CSST	Start task and set APN, user name, password
AT+CIICR	Bring up wireless connection with GPRS or CSD
AT+CIFSR	Get local IP address
AT+CIPSTATUS	Query current connection status
AT+CDNSCFG	Configure domain name server
AT+CDNSGIP	Query the IP address of given domain name
AT+CIPHEAD	Add an IP head at the beginning of a package received
AT+CIPATS	Set auto sending timer
AT+CIPSPRT	Set prompt of '>' when module sends data
AT+CIPSERVER	Configure module as server
AT+CIPCSGP	Set CSD or GPRS for connection mode
AT+CIPSRIP	Show remote IP address and port when received data
AT+CIPDPDP	Set whether to check state of GPRS network timing
AT+CIPMODE	Select TCPIP application mode
AT+CIPCCFG	Configure transparent transfer mode
AT+CIPSHOWTP	Display transfer protocol in IP head when received data
AT+CIPUDPMODE	UDP extended mode
AT+CIPRXGET	Get data from network manually
AT+CIPSCONT	Save TCPIP application context
AT+CIPRDTIMER	Set remode delay timer
AT+CIPSGTXT	Select GPRS PDP context
AT+CIPTKA	Set TCP keepalive parameters

## 8.2 Detailed Descriptions of Commands

### 8.2.1 AT+CIPMUX Start Up Multi-IP Connection

<b>AT+CIPMUX Start Up Multi-IP Connection</b>	
Test Command <b>AT+CIPMUX=?</b>	Response <b>+CIPMUX: (0,1)</b>  <b>OK</b>  Parameters See Write Command
Read Command <b>AT+CIPMUX?</b>	Response <b>+CIPMUX: &lt;n&gt;</b>  <b>OK</b>  Parameters See Write Command
Write Command <b>AT+CIPMUX=&lt;n&gt;</b>	Response <b>OK</b>  Parameters <b>&lt;n&gt;</b> 0    Single IP connection 1    Multi IP connection
Parameter Saving Mode	NO_SAVE
Max Response Time	-
Reference	Note Only in IP initial state, AT+CIPMUX=1 is effective; Only when multi IP connection and GPRS application are both shut down, AT+CIPMUX=0 is effective.

### 8.2.2 AT+CIPSTART Start Up TCP or UDP Connection

<b>AT+CIPSTART Start Up TCP or UDP Connection</b>	
Test Command <b>AT+CIPSTART=?</b>	Response 1) If AT+CIPMUX=0 <b>+CIPSTART: (list of supported &lt;mode&gt;),( &lt;IP address&gt;),( &lt;port&gt;)</b> <b>+CIPSTART: (list of supported &lt;mode&gt;),( &lt;domain name&gt;),( &lt;port&gt;)</b>  <b>OK</b> 2) If AT+CIPMUX=1 <b>+CIPSTART: (list of supported &lt;n&gt;),(list of supported &lt;mode&gt;),( &lt;IP address&gt;),( &lt;port&gt;)</b> <b>+CIPSTART: (list of supported &lt;n&gt;),(list of supported &lt;mode&gt;),( &lt;domain</b>

	<p><b>name&gt;),( &lt;port&gt;)</b></p> <p><b>OK</b></p> <p>Parameters See Write Command</p>
<p>Write Command</p> <p>1)If single IP connection (+CIPMUX=0) <b>AT+CIPSTART=&lt;mode&gt;,&lt;IP address&gt;,&lt;port&gt;</b></p> <p>Or</p> <p><b>AT+CIPSTART=&lt;mode&gt;,&lt;domain name&gt;,&lt;port&gt;</b></p> <p>2)If multi-IP connection (+CIPMUX=1) <b>AT+CIPSTART=&lt;n&gt;,&lt;mode&gt;,&lt;address&gt;,&lt;port&gt;</b></p> <p><b>AT+CIPSTART=&lt;n&gt;,&lt;mode&gt;,&lt;domain name&gt;,&lt;port&gt;</b></p>	<p>Response</p> <p>1)If single IP connection (+CIPMUX=0) If format is right response <b>OK</b> otherwise response If error is related to ME functionality: <b>+CME ERROR &lt;err&gt;</b></p> <p>Response when connection exists <b>ALREADY CONNECT</b></p> <p>Response when connection is successful <b>CONNECT OK</b></p> <p>Otherwise <b>STATE: &lt;state&gt;</b></p> <p><b>CONNECT FAIL</b></p> <p>2)If multi-IP connection (+CIPMUX=1) If format is right <b>OK,</b> otherwise response If error is related to ME functionality: <b>+CME ERROR &lt;err&gt;</b></p> <p>Response when connection exists <b>&lt;n&gt;,ALREADY CONNECT</b></p> <p>If connection is successful <b>&lt;n&gt;,CONNECT OK</b></p> <p>Otherwise <b>&lt;n&gt;,CONNECT FAIL</b></p> <p>Parameters</p> <p><b>&lt;n&gt;</b>            0..5    A numeric parameter which indicates the connection number</p> <p><b>&lt;mode&gt;</b>            A string parameter which indicates the connection type "TCP"    Establish a TCP connection "UDP"    Establish a UDP connection</p> <p><b>&lt;IP address&gt;</b>    A string parameter which indicates remote server IP address</p> <p><b>&lt;port&gt;</b>            Remote server port</p> <p><b>&lt;domain name&gt;</b> A string parameter which indicates remote server domain name</p>



	<p><b>&lt;state&gt;</b> A string parameter which indicates the progress of connecting</p> <ul style="list-style-type: none"> <li>0 IP INITIAL</li> <li>1 IP START</li> <li>2 IP CONFIG</li> <li>3 IP GPRSACT</li> <li>4 IP STATUS</li> <li>5 TCP CONNECTING/UDP CONNECTING/ SERVER LISTENING</li> <li>6 CONNECT OK</li> <li>7 TCP CLOSING/UDP CLOSING</li> <li>8 TCP CLOSED/UDP CLOSED</li> <li>9 PDP DEACT</li> </ul> <p>In Multi-IP state:</p> <ul style="list-style-type: none"> <li>0 IP INITIAL</li> <li>1 IP START</li> <li>2 IP CONFIG</li> <li>3 IP GPRSACT</li> <li>4 IP STATUS</li> <li>5 IP PROCESSING</li> <li>9 PDP DEACT</li> </ul>
Parameter Saving Mode	NO_SAVE
Max Response Time	When mode is multi-IP state, the max response time 75 seconds. When mode is single state, and the state is IP INITIAL, the max response time is 160 seconds.
Reference	<p>Note</p> <p>This command allows establishment of a TCP/UDP connection only when the state is IP INITIAL or IP STATUS when it is in single state. In multi-IP state, the state is in IP STATUS only. So it is necessary to process "AT+CIPSHUT" before user establishes a TCP/UDP connection with this command when the state is not IP INITIAL or IP STATUS.</p> <p>When module is in multi-IP state, before this command is executed, it is necessary to process "AT+CSTT, AT+CIICR, AT+CIFSR".</p>

### 8.2.3 AT+CIPSEND Send Data Through TCP or UDP Connection

<b>AT+CIPSEND Send Data Through TCP or UDP Connection</b>	
Test Command	Response
AT+CIPSEND=?	<p>1) For single IP connection (+CIPMUX=0) <b>+CIPSEND: &lt;length&gt;</b></p> <p><b>OK</b></p> <p>2) For multi IP connection (+CIPMUX=1)</p>

	<p><b>&lt;user name&gt;</b> A string parameter which indicates the GPRS user name</p> <p><b>&lt;password&gt;</b> A string parameter which indicates the GPRS password</p>
Parameter Saving Mode	NO_SAVE
Max Response Time	-
Execution Command <b>AT+CSTT</b>	<p>Response</p> <p><b>OK</b></p> <p><b>ERROR</b></p>
Reference	<p>Note</p> <p>The write command and execution command of this command is valid only at the state of IP INITIAL. After this command is executed, the state will be changed to IP START.</p>

### 8.2.10 AT+CIICR Bring Up Wireless Connection with GPRS or CSD

<b>AT+CIICR Bring Up Wireless Connection with GPRS or CSD</b>	
Test Command <b>AT+CIICR=?</b>	<p>Response</p> <p><b>OK</b></p>
Execution Command <b>AT+CIICR</b>	<p>Response</p> <p><b>OK</b></p> <p><b>ERROR</b></p>
Parameter Saving Mode	NO_SAVE
Max Response Time	85 seconds
Reference	<p>Note</p> <p>AT+CIICR only activates moving scene at the status of IP START, after operating this Command is executed, the state will be changed to IP CONFIG.</p> <p>After module accepts the activated operation, if it is activated successfully, module state will be changed to IP GPRSACT, and it responds OK, otherwise it will respond ERROR.</p>

### 8.2.11 AT+CIFSR Get Local IP Address

<b>AT+CIFSR Get Local IP Address</b>	
Test Command <b>AT+CIFSR=?</b>	<p>Response</p> <p><b>OK</b></p>
Execution Command <b>AT+CIFSR</b>	<p>Response</p> <p><b>&lt;IP address&gt;</b></p> <p><b>ERROR</b></p>
	Parameter

	<IP address> a string parameter which indicates the IP address assigned from GPRS or CSD.
Parameter Saving Mode	NO_SAVE
Max Response Time	-
Reference	Note Only after PDP context is activated, local IP Address can be obtained by AT+CIFSR, otherwise it will respond ERROR. The active status are IP GPRSACT, TCP/UDP CONNECTING, CONNECT OK, IP CLOSE.

### 8.2.12 AT+CIPSTATUS Query Current Connection Status

AT+CIPSTATUS Query Current Connection Status	
Test Command <b>AT+CIPSTATUS=?</b>	Response <b>OK</b>
Write Command If multi IP connection mode (+CIPMUX=1) <b>AT+CIPSTATUS=&lt;n&gt;</b>	Response <b>+CIPSTATUS: &lt;n&gt;,&lt;bearer&gt;, &lt;TCP/UDP&gt;, &lt;IP address&gt;, &lt;port&gt;, &lt;client state&gt;</b> <b>OK</b>
	Parameters See Execution Command
Execution Command <b>AT+CIPSTATUS</b>	Response 1) If in single connection mode (+CIPMUX=0) <b>OK</b>  <b>STATE: &lt;state&gt;</b> 2) If in multi-connection mode (+CIPMUX=1) <b>OK</b>  <b>STATE: &lt;state&gt;</b> If the module is set as server <b>S: 0, &lt;bearer&gt;, &lt;port&gt;, &lt;server state&gt;</b> <b>C: &lt;n&gt;,&lt;bearer&gt;, &lt;TCP/UDP&gt;, &lt;IP address&gt;, &lt;port&gt;, &lt;client state&gt;</b>
	Parameters <b>&lt;n&gt;</b> 0-5 A numeric parameter which indicates the connection number <b>&lt;bearer&gt;</b> 0-1 GPRS bearer, default is 0 <b>&lt;server state&gt;</b> OPENING LISTENING CLOSING

Test Command <b>AT+CIPATS=?</b>	Response <b>+CIPATS:</b> (list of supported <mode>s),(list of supported <time>)  <b>OK</b>
	Parameters See Write Command
Read Command <b>AT+CIPATS?</b>	Response <b>+CIPATS:</b> <mode>,<time>  <b>OK</b>
	Parameters See Write Command
Write Command <b>AT+CIPATS=&lt;mode&gt;[,&lt;time&gt;]</b>	Response <b>OK</b> <b>ERROR</b>
	Parameters <b>&lt;mode&gt;</b> A numeric parameter which indicates whether set timer when module is sending data 0 Not set timer when module is sending data 1 Set timer when module is sending data <b>&lt;time&gt;</b> 1..100 A numeric parameter which indicates the seconds after which the data will be sent
Parameter Saving Mode	NO_SAVE
Max Response Time	-
Reference	Note

### 8.2.17 AT+CIPSPRT Set Prompt of '>' When Module Sends Data

<b>AT+CIPSPRT Set Prompt of '&gt;' When Module Sends Data</b>	
Test Command <b>AT+CIPSPRT=?</b>	Response <b>+CIPSPRT:</b> (list of supported <send prompt>s)  <b>OK</b>
	Parameters See Write Command
Read Command <b>AT+CIPSPRT?</b>	Response <b>+CIPSPRT:</b> <send prompt>  <b>OK</b>
	Parameters

	See Write Command
Write Command <b>AT+CIPSPRT=&lt;send prompt&gt;</b>	<p>Response</p> <p><b>OK</b></p> <p><b>ERROR</b></p>
	<p>Parameters</p> <p><b>&lt;send prompt&gt;</b> A numeric parameter which indicates whether to echo prompt ‘&gt;’ after module issues AT+CIPSEND command.</p> <p>0 It shows "send ok" but does not prompt echo ‘&gt;’ when sending is successful.</p> <p>1 It prompts echo ‘&gt;’ and shows "send ok" when sending is successful.</p> <p>2 It neither prompts echo ‘&gt;’ nor shows "send ok" when sending is successful.</p>
Parameter Saving Mode	NO_SAVE
Max Response Time	-
Reference	Note

### 8.2.18 AT+CIPSERVER Configure Module as Server

<b>AT+CIPSERVER Configure Module as Server</b>	
Test Command <b>AT+CIPSERVE R=?</b>	<p>Response</p> <p><b>+CIPSERVER: (0-CLOSE SERVER, 1-OPEN SERVER),(1-65535)</b></p> <p><b>OK</b></p>
	<p>Parameters</p> <p>See Write Command</p>
Read Command <b>AT+CIPSERVE R?</b>	<p>Response</p> <p><b>+CIPSERVER: &lt;mode&gt;[,&lt;port&gt;,&lt;channel id&gt;,&lt;bearer&gt;]</b></p> <p><b>OK</b></p>
	<p>Parameters</p> <p>See Write Command</p>
Write Command <b>AT+CIPSERVE R=&lt;mode&gt;[,&lt;port&gt;]</b>	<p>Response</p> <p><b>OK</b></p> <p><b>ERROR</b></p>
	<p>Parameters</p> <p><b>&lt;mode&gt;</b>      0    Close server</p> <p>                 1    Open server</p> <p><b>&lt;port&gt;</b>        1..65535    Listening port</p> <p><b>&lt;channel id&gt;</b> Channel id</p>

## 9 AT Commands for IP Application

### 9.1 Overview

Command	Description
AT+SAPBR	Bearer settings for applications based on IP

### 9.2 Detailed Descriptions of Commands

#### 9.2.1 AT+SAPBR Bearer Settings for Applications Based on IP

AT+SAPBR Bearer Settings for Applications Based on IP	
Test Command <b>AT+SAPBR=?</b>	Response <b>+SAPBR: (0-4),(1-3), "ConParamTag","ConParamValue"</b>  <b>OK</b>  Parameters See Write Command
Write Command <b>AT+SAPBR=&lt;cmd_type&gt;,&lt;cid&gt;[,&lt;ConParamTag&gt;,&lt;ConParamValue&gt;]</b>	Response <b>OK</b>  <b>If &lt;cmd_type&gt; = 2</b> <b>+SAPBR: &lt;cid&gt;,&lt;Status&gt;,&lt;IP_Addr&gt;</b> <b>OK</b> <b>If &lt;cmd_type&gt;=4</b> <b>+SAPBR:</b> <b>&lt;ConParamTag&gt;,&lt;ConParamValue&gt;</b> <b>OK</b>  Unsolicited Result Code <b>+SAPBR &lt;cid&gt;: DEACT</b>
	Parameters <b>&lt;cmd_type&gt;</b> 0 Close bearer 1 Open bearer 2 Query bearer 3 Set bearer parameters 4 Get bearer parameters  <b>&lt;cid&gt;</b> Bearer profile identifier <b>&lt;Status&gt;</b> 0 Bearer is connecting 1 Bearer is connected

## 11 AT Commands for HTTP Application

SIM800 series has an embedded TCP/IP stack that is driven by AT commands and enables the host application to easily access the Internet HTTP service. This chapter is a reference guide to all the AT commands and responses defined to use with the TCP/IP stack in HTTP Service.

### 11.1 Overview

Command	Description
AT+HTTPIPINIT	Initialize HTTP service
AT+HTTPIPTERM	Terminate HTTP service
AT+HTTPIP PARA	Set HTTP parameters value
AT+HTTPIP DATA	Input HTTP data
AT+HTTPIP ACTION	HTTP method action
AT+HTTPIP READ	Read the HTTP server response
AT+HTTPIP SCONT	Save HTTP application context
AT+HTTPIP STATUS	Read HTTP status

### 11.2 Detailed Descriptions of Commands

#### 11.2.1 AT+HTTPIPINIT Initialize HTTP Service

AT+HTTPIPINIT Initialize HTTP Service	
Test Command AT+HTTPIPINIT=?	Response <b>OK</b>
Execution Command AT+HTTPIPINIT	Response <b>OK</b>  If error is related to ME functionality: <b>+CME ERROR: &lt;err&gt;</b>
Parameter Saving Mode	NO_SAVE
Max Response Time	-
Reference	Note HTTPIPINIT should first be executed to initialize the HTTP service.

### 11.2.2 AT+HTTPTERM Terminate HTTP Service

AT+HTTPTERM Terminate HTTP Service	
Test Command <b>AT+HTTPTERM</b> <b>M=?</b>	Response <b>OK</b>
Execution command <b>AT+HTTPTERM</b> <b>M</b>	Response <b>OK</b> If error is related to ME functionality: <b>+CME ERROR: &lt;err&gt;</b>
Parameter Saving Mode	NO_SAVE
Max Response Time	-
Reference	Note

### 11.2.3 AT+HTTPPARA Set HTTP Parameters Value

AT+HTTPPARA Set HTTP Parameters Value	
Test Command <b>AT+HTTPPARA</b> <b>=?</b>	Response <b>+HTTPPARA: "HTTPParamTag","HTTPParamValue"</b>  <b>OK</b>
	Parameters See Write Command
Read Command <b>AT+HTTPPARA</b> <b>?</b>	Response <b>+HTTPPARA:</b> <b>&lt;HTTPParamTag&gt;,&lt;HTTPParamValue&gt;</b>  <b>OK</b>
	Parameters See Write Command
Write Command <b>AT+HTTPPARA</b> <b>=&lt;HTTPParamTag&gt;,&lt;HTTPParamValue&gt;</b>	Response <b>OK</b> If error is related to ME functionality: <b>+CME ERROR: &lt;err&gt;</b>
	Parameters
	<b>&lt;HTTPParamTag&gt;</b> HTTP Parameter
	"CID" (Mandatory Parameter) Bearer profile identifier



	<p>411 Length Required</p> <p>412 Precondition Failed</p> <p>413 Request Entity Too Large</p> <p>414 Request-URI Too Large</p> <p>415 Unsupported Media Type</p> <p>416 Requested range not satisfiable</p> <p>417 Expectation Failed</p> <p>500 Internal Server Error</p> <p>501 Not Implemented</p> <p>502 Bad Gateway</p> <p>503 Service Unavailable</p> <p>504 Gateway Time-out</p> <p>505 HTTP Version not supported</p> <p>600 Not HTTP PDU</p> <p>601 Network Error</p> <p>602 No memory</p> <p>603 DNS Error</p> <p>604 Stack Busy</p> <p><b>&lt;DataLen&gt;</b> the length of data got</p>
Parameter Saving Mode	NO_SAVE
Max Response Time	About 5 seconds in test, dependence on network status and the size of request website
Reference	Note

### 11.2.6 AT+HTTPREAD Read the HTTP Server Response

AT+HTTPREAD Read the HTTP Server Response	
<p>Test Command</p> <p><b>AT+HTTPREAD=?</b></p>	<p>Response</p> <p><b>+HTTPREAD:</b> (list of supported <b>&lt;start_address&gt;</b>s),(list of supported<b>&lt;byte_size&gt;</b>s)</p> <p><b>OK</b></p> <p>Parameters</p> <p>See Write Command</p>
<p>Write Command</p> <p><b>AT+HTTPREAD=&lt;start_addresses&gt;,&lt;byte_size&gt;</b></p>	<p>Response</p> <p><b>+HTTPREAD: &lt;data_len&gt;</b></p> <p><b>&lt;data&gt;</b></p> <p><b>OK</b></p> <p>Read data when AT+HTTPACTION=0 or AT+HTTPDATA is executed.</p>



# **SIM800 Series\_GNSS\_Application Note\_V1.00**



<b>Document Title:</b>	SIM800 Series_GNSS_Application Note
<b>Version:</b>	1.00
<b>Date:</b>	2015-04-10
<b>Status:</b>	Released
<b>Document Control ID:</b>	SIM800 Seires_GNSS_Application Note_V1.00

### General Notes

SIMCom offers this information as a service to its customers, to support application and engineering efforts that use the products designed by SIMCom. The information provided is based upon requirements specifically provided to SIMCom by the customers. SIMCom has not undertaken any independent search for additional relevant information, including any information that may be in the customer's possession. Furthermore, system validation of this product designed by SIMCom within a larger electronic system remains the responsibility of the customer or the customer's system integrator. All specifications supplied herein are subject to change.

### Copyright

This document contains proprietary technical information which is the property of SIMCom Limited., copying of this document and giving it to others and the using or communication of the contents thereof, are forbidden without express authority. Offenders are liable to the payment of damages. All rights reserved in the event of grant of a patent or the registration of a utility model or design. All specification supplied herein are subject to change without notice at any time.

*Copyright © Shanghai SIMCom Wireless Solutions Ltd. 2015*

## Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
<b>2</b>	<b>AT Command</b>	<b>8</b>
2.1	AT+CGNSPWR GNSS power control	8
2.2	AT+CGNSSEQ Define the last NMEA sentence that parsed	9
2.3	AT+CGNSINF GNSS navigation information parsed from NMEA sentences	10
2.4	AT+CGNSURC GNSS navigation, GEO-fences and speed alarm URC report	13
2.5	AT+CGNSCMD Send command to GNSS	13
2.6	AT+CGNSTST Send data received from GNSS to AT UART	14
<b>3</b>	<b>CME Error Code</b>	<b>16</b>
<b>4</b>	<b>AT Commands Examples</b>	<b>17</b>
	<b>Appendix</b>	<b>18</b>
A	Related documents	18
B	Terms and Abbreviations	18

**Tables**

TABLE 2-1: PARSED NMEA MESSAGE..... 9  
TABLE 2-2: PARSED GNSS NAVIGATION PARAMETERS..... 9  
TABLE 2-3: AT+CGNSINF RETURN PARAMETERS ..... 11

**Figures**

FIGURE 1-1 SIM808 SYSTEM CONNECTION 7

**VERSION HISTORY**

Date	Version	Description of change	Author
2015-04-10	1.00	New version	Zhongyu.gou

**Scope**

This document presents the AT command of GNSS function and application examples. The document can apply to SIM800 series modules, including SIM808 with hardware release version is V2.01 and above and the software release version is 1418B01SIM808M32 and above.

## 1 Introduction

SIM808 module combines GNSS technology for satellite navigation. Featuring an industry-standard interface and GNSS function, it allows variable assets to be tracked seamlessly at any location and anytime with signal coverage.

GNSS application provides a method to interact with a GNSS module.

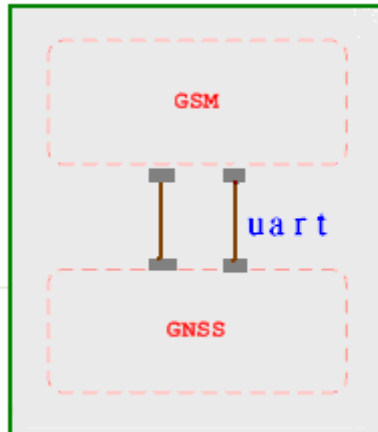


Figure 1-1 SIM808 System connection

## 2 AT Command

SIM800 series modules provide GNSS AT command is as follows:

Commands	Description
AT+CGNSPWR	GNSS power control
AT+CGNSSEQ	Define the last NMEA sentence that parsed
AT+CGNSINF	GNSS navigation information parsed from NMEA sentences
AT+CGNSURC	GNSS navigation, GEO-fence and speed alarm URC report control
AT+CGNSCMD	Send command to GNSS
AT+CGNSTST	Send data received from GNSS to AT UART

### 2.1 AT+CGNSPWR GNSS power control

AT+CGNSPWR GNSS power control	
Test Command <b>AT+CGNSPWR=?</b>	Response <b>+CGNSPWR:</b> (list of supported <mode>s )  <b>OK</b>
	Parameters See Write Command
Read Command <b>AT+CGNSPWR?</b>	Response TA returns the current status of GNSS Power supply <b>+CGNSPWR:</b> <mode>  <b>OK</b>
	Parameters See Write Command
Write Command <b>AT+CGNSPWR=&lt;mode&gt;</b>	Response GNSS POWER CONTROL ON/OFF <b>OK</b> <b>ERROR</b>
	Parameters <mode>    0    Turn off GNSS power supply 1    Turn on GNSS power supply
Reference	



## 2.2 AT+CGNSSEQ Define the last NMEA sentence that parsed

AT+CGNSSEQ Define the last NMEA sentence that parsed	
<b>Test Command</b> <b>AT+CGNSSEQ=?</b>	<b>Response</b> <b>+CGNSSEQ: (GGA,GSA,RMC,GSV)</b>  <b>OK</b>
	<b>Parameter</b> See Write Command
<b>Read Command</b> <b>AT+CGNSSEQ?</b>	<b>Response</b> TA returns the current setting of last sentence parsed: <b>+CGNSSEQ: &lt;last sentence&gt;</b>  <b>OK</b>
	<b>Parameter</b> See Write Command
<b>Write Command</b> <b>AT+CGNSSEQ=&lt;last sentence&gt;</b>	<b>Response</b> <b>OK</b> <b>ERROR</b>
	<b>Parameters</b> <last sentence> is a string type parameter: "GGA" refer to "GPGGA" or "GLGGA" or "GNGGA" "GSA" refer to "GPGSA" or "GLGSA" or "GNGSA" "GSV" refer to "GPGSV" or "GLGSV" or "GNGSV" "RMC" refer to "GPRMC" or "GLRMC" or "GNRMC"
<b>Reference</b>	<b>Note</b> Factory setting is: <b>AT+CGNSSEQ="RMC"</b> .

Table 2-1: parsed NMEA message

Message	Description	Possible Talker Identifiers
GGA	Time, position and fix type data	GP
GSA	GNSS receiver operating mode, satellites used in the position solution, and DOP values	GP, GN
GSV	Number of GNSS satellites in view satellite ID numbers, elevation, azimuth, & SNR values	GP, GL, GN
RMC	Time, date, position, course and speed data	GP, GN

Table 2-2: parsed GNSS navigation parameters

Parameters	Description
UTC Time	Parsed from "\$-RMC" NMEA sentence
fix status	Parsed from "\$-RMC" NMEA sentence

Latitude	Parsed from "\$--RMC" NMEA sentence
N/S Indicator	Parsed from "\$--RMC" NMEA sentence
Longitude	Parsed from "\$--RMC" NMEA sentence
E/W Indicator	Parsed from "\$--RMC" NMEA sentence
Speed Over Ground	Parsed from "\$--RMC" NMEA sentence
Course Over Ground	Parsed from "\$--RMC" NMEA sentence
Date	Parsed from "\$--RMC" NMEA sentence
Magnetic Variation	Reserved
East/West Indicator	Reserved
RMC mode	Parsed from "\$--GGA" NMEA sentence
HDOP	Parsed from "\$--GGA" NMEA sentence
MSL Altitude	Parsed from "\$--GGA" NMEA sentence
Units	Parsed from "\$--GGA" NMEA sentence
Geoid Separation	Reserved
Units	Reserved
Age of Diff. Corr.	Reserved
Diff. Ref. Station ID	Reserved
Satellites Used	Parsed from "\$--GGA" NMEA sentence
PDOP	Parsed from "\$--GGA" NMEA sentence
VDOP	Parsed from "\$--GGA" NMEA sentence
Satellites in View	Parsed from "\$--GSV" NMEA sentence
HPA	Reserved
VPA	Reserved

### 2.3 AT+CGNSINF GNSS navigation information parsed from NMEA sentences

AT+CGNSINF GNSS navigation information parsed from NMEA sentences	
Execution Command <b>AT+CGNSINF</b>	Response <b>+CGNSINF: &lt;GNSS run status&gt;,&lt;Fix status&gt;,&lt;UTC date &amp; Time&gt;,&lt;Latitude&gt;,&lt;Longitude&gt;,&lt;Speed Over Ground&gt;,&lt;Course Over Ground&gt;,&lt;Date&gt;,&lt;Magnetic Variation&gt;,&lt;East/West Indicator&gt;,&lt;RMC mode&gt;,&lt;HDOP&gt;,&lt;MSL Altitude&gt;,&lt;Units&gt;,&lt;Geoid Separation&gt;,&lt;Units&gt;,&lt;Age of Diff. Corr.&gt;,&lt;Diff. Ref. Station ID&gt;,&lt;Satellites Used&gt;,&lt;PDOP&gt;,&lt;VDOP&gt;,&lt;Satellites in View&gt;,&lt;HPA&gt;,&lt;VPA&gt;</b>

	<p>&lt;MSL Altitude&gt;,&lt;Speed Over Ground&gt;, &lt;Course Over Ground&gt;, &lt;Fix Mode&gt;,&lt;Reserved1&gt;,&lt;HDOP&gt;,&lt;PDOP&gt;, &lt;VDOP&gt;,&lt;Reserved2&gt;,&lt;GNSS Satellites in View&gt;, &lt;GNSS Satellites Used&gt;,&lt;GLONASS Satellites Used&gt;,&lt;Reserved3&gt;,&lt;C/N0 max&gt;,&lt;HPA&gt;,&lt;VPA&gt;</p> <p><b>OK</b></p>
	<p>Parameters See below table 2-3.</p>
Reference	

Table 2-3: AT+CGNSINF return Parameters

Index	Parameter	Unit	Range	Length
1	GPS run status	--	0-1	1
2	Fix status	--	0-1	1
3	UTC date & Time	yyyyMMddhh mmss.sss	yyyy: [1980,2039] MM : [1,12] dd: [1,31] hh: [0,23] mm: [0,59] ss.sss:[0.000,60.999]	18
4	Latitude	±dd.dddddd	[-90.000000,90.000000]	10
5	Longitude	±ddd.dddddd	[-180.000000,180.000000]	11
6	MSL Altitude	meters		8
7	Speed Over Ground	Km/hour	[0,999.99]	6
8	Course Over Ground	degrees	[0,360.00]	6
9	Fix Mode	--	0,1,2 <sup>[1]</sup>	1
10	Reserved1			0
11	HDOP	--	[0,99.9]	4
12	PDOP	--	[0,99.9]	4
13	VDOP	--	[0,99.9]	4
14	Reserved2			0
15	GPS Satellites in View	--	[0,99]	2
16	GNSS Satellites Used	--	[0,99]	2
17	GLONASS Satellites in View	--	[0,99]	2
18	Reserved3			0
19	C/N0 max	dBHz	[0,55]	2
20	HPA <sup>[2]</sup>	meters	[0,9999.9]	6

21	VPA <sup>[2]</sup>	meters	[0,9999.9]	6
				Total: (94) chars

**Note:**

1. *The range of <Fix Mode> depends on the GNSS chip used.*
2. *Reserved.*

## 2.4 AT+CGNSURC GNSS navigation, GEO-fences and speed alarm URC report

AT+CGNSURC GNSS navigation, GEO-fences and speed alarm URC report	
Test Command <b>AT+CGNSURC=?</b>	Response <b>+CGNSURC: (0-255)</b>  <b>OK</b>
	Parameters See Write Command
Read Command <b>AT+CGNSURC?</b>	Response TA returns the current URC setting <b>+CGNSURC: &lt;Navigation mode&gt;</b>  <b>OK</b>
	Parameters See Write Command
	Unsolicited Result Code <b>+UGNSINF: &lt;GNSS run status&gt;,&lt;Fix status&gt;,&lt;UTC date &amp; Time&gt;,&lt;Latitude&gt;,&lt;Longitude&gt;,&lt;MSL Altitude&gt;,&lt;Speed Over Ground&gt;,&lt;Course Over Ground&gt;,&lt;Fix Mode&gt;,&lt;Reserved1&gt;,&lt;HDOP&gt;,&lt;PDOP&gt;,&lt;VDOP&gt;,&lt;Reserved2&gt;,&lt;Satellites in View&gt;,&lt;Satellites Used&gt;,&lt;Reserved3&gt;,&lt;C/N0 max&gt;,&lt;HPA&gt;,&lt;VPA&gt;</b>
Write Command <b>AT+CGNSURC=&lt;Navigation mode&gt;</b>	Parameters <b>&lt;Navigation mode&gt;</b> : 0 Turn off navigation data URC report 1 Turn on navigation data URC report, and report every GNSS FIX 2 Turn on navigation data URC report, and report every 2 GNSS FIX ... 255 Turn on navigation data URC report, and report every 255 GNSS FIX
Reference	Note 1. Factory setting is "AT+CGNSURC=0". 2. URC "+UGNSINF:" parameters are the same as "+CGNSINF:" return.

## 2.5 AT+CGNSCMD Send command to GNSS

AT+CGNSCMD Send command to GNSS	
Test Command <b>AT+CGNSCMD=?</b>	Response <b>+CGNSCMD: (0-1),"CmdString"</b>

	<p><b>OK</b></p> <p>Parameters See Write Command</p>
<p>Write Command <b>AT+CGNSCMD=&lt;Cmdtype&gt;,&lt;CmdString&gt;</b></p>	<p>Response</p> <p>If send ok: <b>OK</b></p> <p>If send false: <b>ERROR</b></p> <p>Parameters <b>&lt;CmdType&gt;</b></p> <ul style="list-style-type: none"> <li>0 NMEA style command</li> <li>1 HEX style command</li> </ul> <p><b>&lt;CmdString&gt;</b> command string</p> <p>For example, if you want to send "\$PMTK000*32&lt;CR&gt;&lt;LF&gt;" command to GNSS: You can use: AT+CGNSCMD=0,"\$PMTK000*32" Or: AT+CGNSCMD=1,"24504D544B3030302A33320D0A"</p>
Reference	<p>Note</p> <p>Max length of &lt;CmdString&gt; is 258.</p>

## 2.6 AT+CGNSTST Send data received from GNSS to AT UART

<b>AT+CGNSTST Send data received from GNSS to AT UART</b>	
<p>Test Command <b>AT+CGNSTST=?</b></p>	<p>Response</p> <p><b>+CGNSTST: (0-1)</b></p> <p><b>OK</b></p> <p>Parameters See Write Command</p>
<p>Read Command <b>AT+CGNSTST?</b></p>	<p>Response</p> <p>GNSS test mode on/off <b>+CGNSTST: &lt;mode&gt;</b></p> <p><b>OK</b></p> <p>Parameters See Write Command</p>
<p>Write Command <b>AT+CGNSTST=&lt;mode&gt;</b></p>	<p>Response</p> <p><b>OK</b></p> <p><b>ERROR</b></p>

	<p>Parameters</p> <p><b>&lt;mode&gt;</b></p> <p><u>0</u> Turn off GNSS test mode</p> <p>1 Turn on GNSS test mode</p>
<p>Reference</p>	<p>Note</p> <p>This command is used for test.</p>

### 3 CME Error Code

The following errors are related to GPS. The format is like this: +CME ERROR: <err>. The detail error code and description is list in the following table.

Code	Description
895	GNSS baud rate selected by HW
891	GNSS data check sum err



## 4 AT Commands Examples

Demonstration	Syntax	Expect Result
Turn on GNSS power	AT+CGNSPWR=1	OK
Turn off GNSS power	AT+CGNSPWR=0	OK
Define the last NMEA sentence that parsed	AT+CGNSSEQ="RMC"	OK
Read GNSS navigation information	AT+CGNSINF	+CGNSINF: 1,1,20150327014838.000,31.2 21783,121.354528,114.600,0. 28,0.0,1,,1.9,2.2,1.0,,8,4,,42,,  OK
Set URC reporting every 2(1-255) GNSS fix	AT+CGNSURC=2	OK
Turn off URC reporting	AT+CGNSURC=0	OK
Send Command to GNSS	AT+CGNSCMD=0,"\$PMTK000*32"	OK
Send NMEA data to AT UART	AT+CGNSTST=1	OK

## Appendix

### A Related documents

SN	Document name	Remark
[1]		

### B Terms and Abbreviations

Abbreviation	Definition
APN	Access Point Name
URC	Unsolicited Result Code
FTP	File Transfer Protocol
GGA	Global Positioning System Fixed Data
GLL	Geographic Position - Latitude/Longitude
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
AGPS	Assisted GPS
DGPS	Differential Global Positioning System
GPRS	General Packet Radio Service
GSA	GNSS DOP and Active Satellites
GSV	GNSS Satellites in View
HPA	Horizontal Position Accuracy
VPA	Vertical Position Accuracy
GEO-Fence	A geographic area
HTTP	The Hypertext Transfer Protocol
HDOP	Horizontal Dilution of Precision
HTTP	Hypertext Transfer Protocol
NMEA	National Marine Electronics Association
NMEA	National Marine Electronics Association
PDOP	Position Dilution of Precision
PDP	Packet Data Protocol
RMC	Recommended Minimum Specific GNSS Data
VDOP	Vertical Dilution of Precision
VTG	Course Over Ground and Ground Speed
ZDA	Time & Date

**Contact us:**

**Shanghai SIMCom Wireless Solutions Ltd.**

Add: Building A, SIM Technology Building, No.633 Jinzhong Road, Changning District, Shanghai,  
P. R. China 200335

Tel: +86 21 3252 3300

Fax: +86 21 3252 3020

URL: [www.sim.com/wm](http://www.sim.com/wm)

# LM2596

## 3.0 A, Step-Down Switching Regulator

The LM2596 regulator is monolithic integrated circuit ideally suited for easy and convenient design of a step-down switching regulator (buck converter). It is capable of driving a 3.0 A load with excellent line and load regulation. This device is available in adjustable output version and it is internally compensated to minimize the number of external components to simplify the power supply design.

Since LM2596 converter is a switch-mode power supply, its efficiency is significantly higher in comparison with popular three-terminal linear regulators, especially with higher input voltages.

The LM2596 operates at a switching frequency of 150 kHz thus allowing smaller sized filter components than what would be needed with lower frequency switching regulators. Available in a standard 5-lead TO-220 package with several different lead bend options, and D<sup>2</sup>PAK surface mount package.

The other features include a guaranteed  $\pm 4\%$  tolerance on output voltage within specified input voltages and output load conditions, and  $\pm 15\%$  on the oscillator frequency. External shutdown is included, featuring 80  $\mu\text{A}$  (typical) standby current. Self protection features include switch cycle-by-cycle current limit for the output switch, as well as thermal shutdown for complete protection under fault conditions.

### Features

- Adjustable Output Voltage Range 1.23 V – 37 V
- Guaranteed 3.0 A Output Load Current
- Wide Input Voltage Range up to 40 V
- 150 kHz Fixed Frequency Internal Oscillator
- TTL Shutdown Capability
- Low Power Standby Mode, typ 80  $\mu\text{A}$
- Thermal Shutdown and Current Limit Protection
- Internal Loop Compensation
- Moisture Sensitivity Level (MSL) Equals 1
- Pb-Free Packages are Available

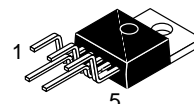
### Applications

- Simple High-Efficiency Step-Down (Buck) Regulator
- Efficient Pre-Regulator for Linear Regulators
- On-Card Switching Regulators
- Positive to Negative Converter (Buck-Boost)
- Negative Step-Up Converters
- Power Supply for Battery Chargers



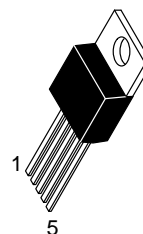
ON Semiconductor®

<http://onsemi.com>



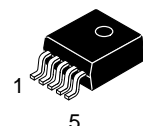
TO-220  
TV SUFFIX  
CASE 314B

Heatsink surface connected to Pin 3



TO-220  
T SUFFIX  
CASE 314D

Pin 1.  $V_{in}$   
2. Output  
3. Ground  
4. Feedback  
5. ON/OFF



D<sup>2</sup>PAK  
D2T SUFFIX  
CASE 936A

Heatsink surface (shown as terminal 6 in case outline drawing) is connected to Pin 3

### ORDERING INFORMATION

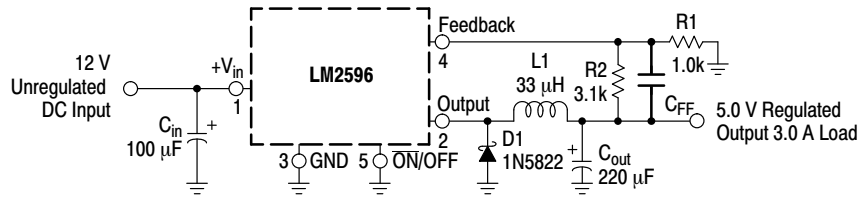
See detailed ordering and shipping information in the package dimensions section on page 23 of this data sheet.

### DEVICE MARKING INFORMATION

See general marking information in the device marking section on page 23 of this data sheet.

# LM2596

## Typical Application (Adjustable Output Voltage Version)



## Block Diagram

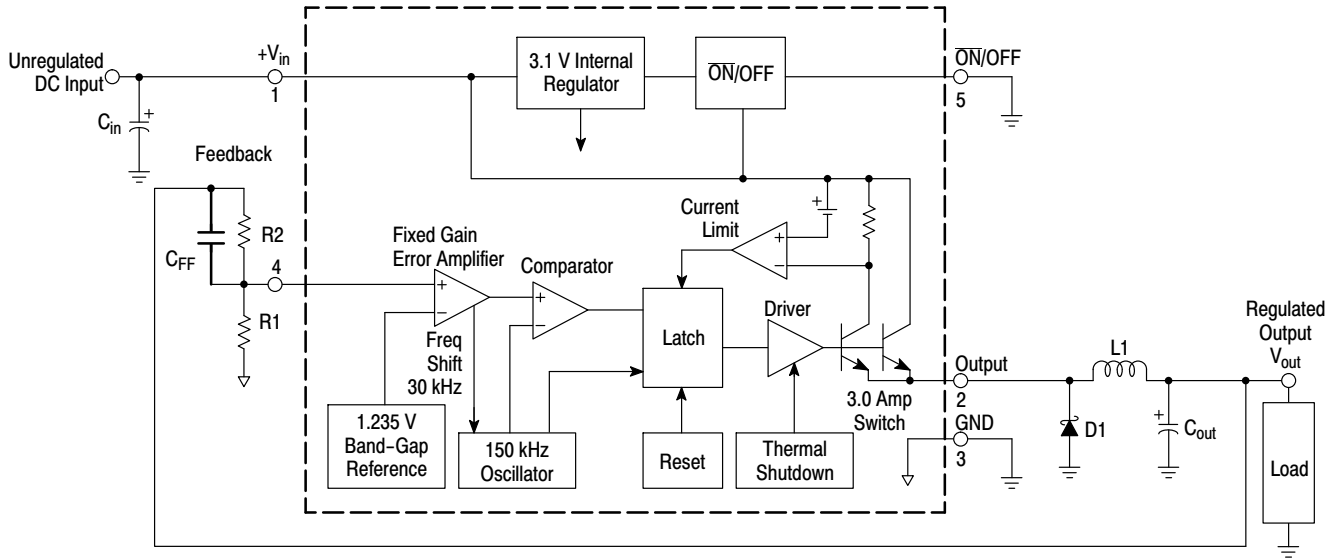


Figure 1. Typical Application and Internal Block Diagram

## MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Maximum Supply Voltage	$V_{in}$	45	V
ON/OFF Pin Input Voltage	-	$-0.3\text{ V} \leq V \leq +V_{in}$	V
Output Voltage to Ground (Steady-State)	-	-1.0	V
Power Dissipation			
Case 314B and 314D (TO-220, 5-Lead)	$P_D$	Internally Limited	W
Thermal Resistance, Junction-to-Ambient	$R_{\theta JA}$	65	$^{\circ}\text{C/W}$
Thermal Resistance, Junction-to-Case	$R_{\theta JC}$	5.0	$^{\circ}\text{C/W}$
Case 936A (D <sup>2</sup> PAK)	$P_D$	Internally Limited	W
Thermal Resistance, Junction-to-Ambient	$R_{\theta JA}$	70	$^{\circ}\text{C/W}$
Thermal Resistance, Junction-to-Case	$R_{\theta JC}$	5.0	$^{\circ}\text{C/W}$
Storage Temperature Range	$T_{stg}$	-65 to +150	$^{\circ}\text{C}$
Minimum ESD Rating (Human Body Model: C = 100 pF, R = 1.5 k $\Omega$ )	-	2.0	kV
Lead Temperature (Soldering, 10 seconds)	-	260	$^{\circ}\text{C}$
Maximum Junction Temperature	$T_J$	150	$^{\circ}\text{C}$

Stresses exceeding Maximum Ratings may damage the device. Maximum Ratings are stress ratings only. Functional operation above the Recommended Operating Conditions is not implied. Extended exposure to stresses above the Recommended Operating Conditions may affect device reliability.

# LM2596

## PIN FUNCTION DESCRIPTION

Pin	Symbol	Description (Refer to Figure 1)
1	$V_{in}$	This pin is the positive input supply for the LM2596 step-down switching regulator. In order to minimize voltage transients and to supply the switching currents needed by the regulator, a suitable input bypass capacitor must be present ( $C_{in}$ in Figure 1).
2	Output	This is the emitter of the internal switch. The saturation voltage $V_{sat}$ of this output switch is typically 1.5 V. It should be kept in mind that the PCB area connected to this pin should be kept to a minimum in order to minimize coupling to sensitive circuitry.
3	GND	Circuit ground pin. See the information about the printed circuit board layout.
4	Feedback	This pin is the direct input of the error amplifier and the resistor network R2, R1 is connected externally to allow programming of the output voltage.
5	$\overline{ON/OFF}$	It allows the switching regulator circuit to be shut down using logic level signals, thus dropping the total input supply current to approximately 80 $\mu$ A. The threshold voltage is typically 1.6 V. Applying a voltage above this value (up to $+V_{in}$ ) shuts the regulator off. If the voltage applied to this pin is lower than 1.6 V or if this pin is left open, the regulator will be in the "on" condition.

**OPERATING RATINGS** (Operating Ratings indicate conditions for which the device is intended to be functional, but do not guarantee specific performance limits. For guaranteed specifications and test conditions, see the Electrical Characteristics.)

Rating	Symbol	Value	Unit
Operating Junction Temperature Range	$T_J$	-40 to +125	$^{\circ}$ C
Supply Voltage	$V_{in}$	4.5 to 40	V

# LM2596

## SYSTEM PARAMETERS

**ELECTRICAL CHARACTERISTICS** Specifications with standard type face are for  $T_J = 25^\circ\text{C}$ , and those with boldface type apply over full Operating Temperature Range  $-40^\circ\text{C}$  to  $+125^\circ\text{C}$

Characteristics	Symbol	Min	Typ	Max	Unit
<b>LM2596</b> (Note 1, Test Circuit Figure 15)					
Feedback Voltage ( $V_{in} = 12\text{ V}$ , $I_{Load} = 0.5\text{ A}$ , $V_{out} = 5.0\text{ V}$ , )	$V_{FB\_nom}$		1.23		V
Feedback Voltage ( $8.5\text{ V} \leq V_{in} \leq 40\text{ V}$ , $0.5\text{ A} \leq I_{Load} \leq 3.0\text{ A}$ , $V_{out} = 5.0\text{ V}$ )	$V_{FB}$	1.193 <b>1.18</b>		1.267 <b>1.28</b>	V
Efficiency ( $V_{in} = 12\text{ V}$ , $I_{Load} = 3.0\text{ A}$ , $V_{out} = 5.0\text{ V}$ )	$\eta$	-	73	-	%
Characteristics	Symbol	Min	Typ	Max	Unit
Feedback Bias Current ( $V_{out} = 5.0\text{ V}$ )	$I_b$		25	100 <b>200</b>	nA
Oscillator Frequency (Note 2)	$f_{osc}$	135 <b>120</b>	150	165 <b>180</b>	kHz
Saturation Voltage ( $I_{out} = 3.0\text{ A}$ , Notes 3 and 4)	$V_{sat}$		1.5	1.8 <b>2.0</b>	V
Max Duty Cycle "ON" (Note 4)	DC		95		%
Current Limit (Peak Current, Notes 2 and 3)	$I_{CL}$	4.2 <b>3.5</b>	5.6	6.9 <b>7.5</b>	A
Output Leakage Current (Notes 5 and 6) Output = 0 V Output = -1.0 V	$I_L$		0.5 6.0	2.0 20	mA
Quiescent Current (Note 5)	$I_Q$		5.0	10	mA
Standby Quiescent Current ( $\overline{ON}/OFF$ Pin = 5.0 V ("OFF")) (Note 6)	$I_{stby}$		80	200 <b>250</b>	$\mu\text{A}$

### $\overline{ON}/OFF$ PIN LOGIC INPUT

Threshold Voltage			1.6		V
$V_{out} = 0\text{ V}$ (Regulator OFF)	$V_{IH}$	2.2 <b>2.4</b>			V
$V_{out} = \text{Nominal Output Voltage}$ (Regulator ON)	$V_{IL}$			1.0 <b>0.8</b>	V

### $\overline{ON}/OFF$ Pin Input Current

$\overline{ON}/OFF$ Pin = 5.0 V (Regulator OFF)	$I_{IH}$	-	15	30	$\mu\text{A}$
$\overline{ON}/OFF$ Pin = 0 V (regulator ON)	$I_{IL}$	-	0.01	5.0	$\mu\text{A}$

- External components such as the catch diode, inductor, input and output capacitors can affect switching regulator system performance. When the LM2596 is used as shown in the Figure 15 test circuit, system performance will be as shown in system parameters section.
- The oscillator frequency reduces to approximately 30 kHz in the event of an output short or an overload which causes the regulated output voltage to drop approximately 40% from the nominal output voltage. This self protection feature lowers the average dissipation of the IC by lowering the minimum duty cycle from 5% down to approximately 2%.
- No diode, inductor or capacitor connected to output (Pin 2) sourcing the current.
- Feedback (Pin 4) removed from output and connected to 0 V.
- Feedback (Pin 4) removed from output and connected to +12 V to force the output transistor "off".
- $V_{in} = 40\text{ V}$ .

TYPICAL PERFORMANCE CHARACTERISTICS (Circuit of Figure 15)

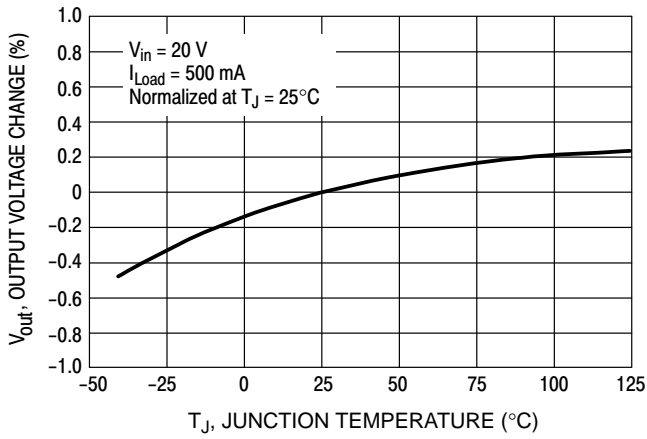


Figure 2. Normalized Output Voltage

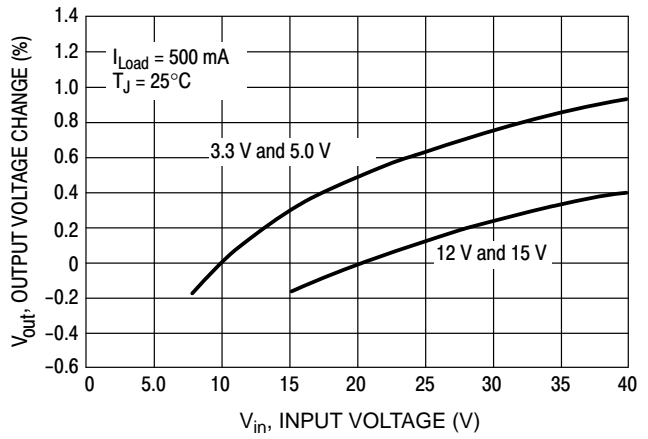


Figure 3. Line Regulation

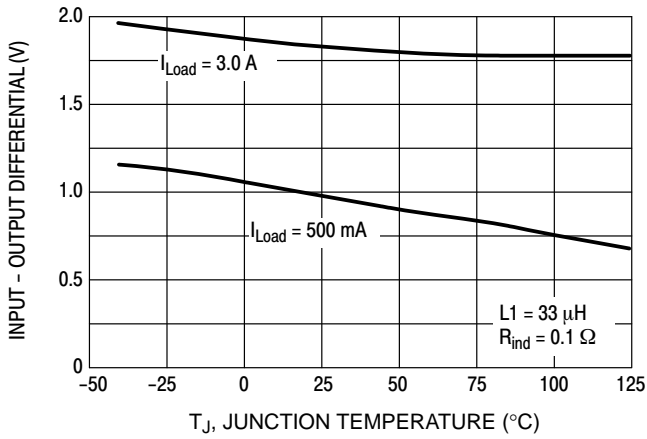


Figure 4. Dropout Voltage

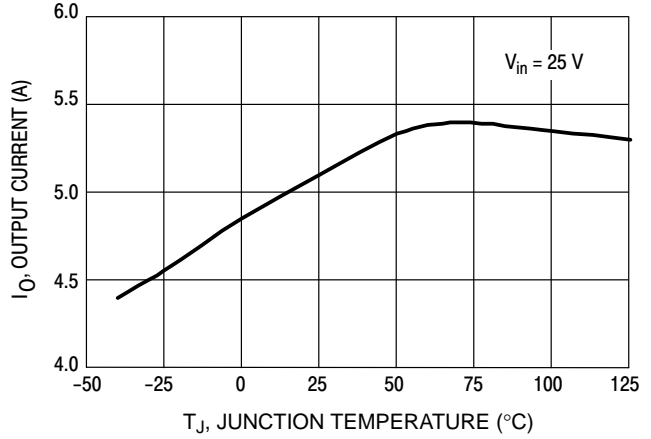


Figure 5. Current Limit

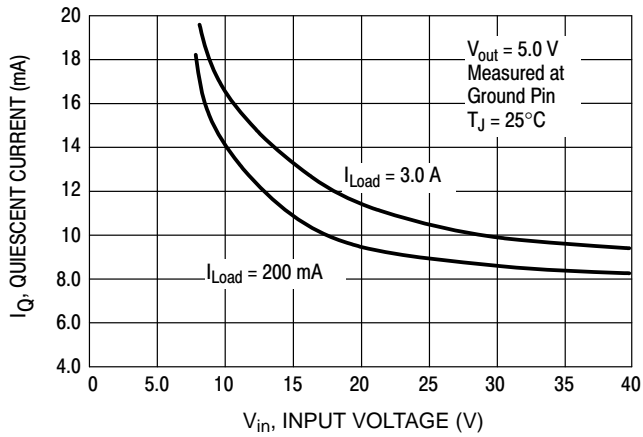


Figure 6. Quiescent Current

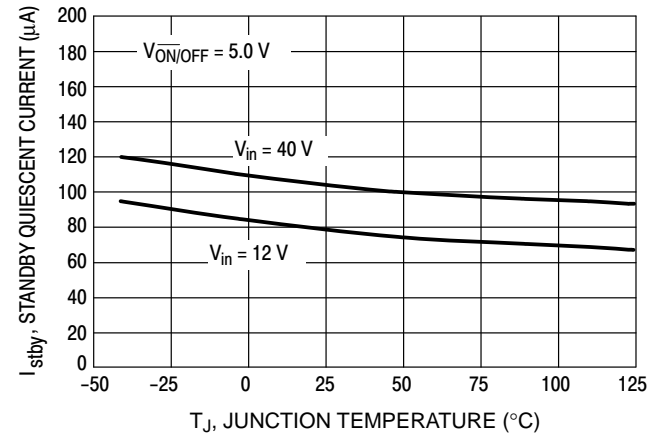


Figure 7. Standby Quiescent Current



# LM2596

## TYPICAL PERFORMANCE CHARACTERISTICS (Circuit of Figure 15)

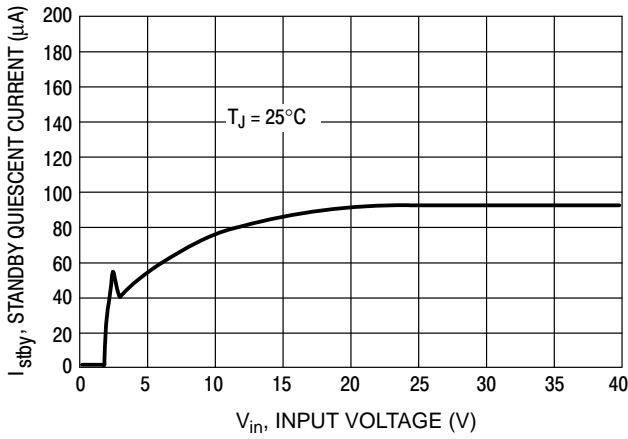


Figure 8. Standby Quiescent Current

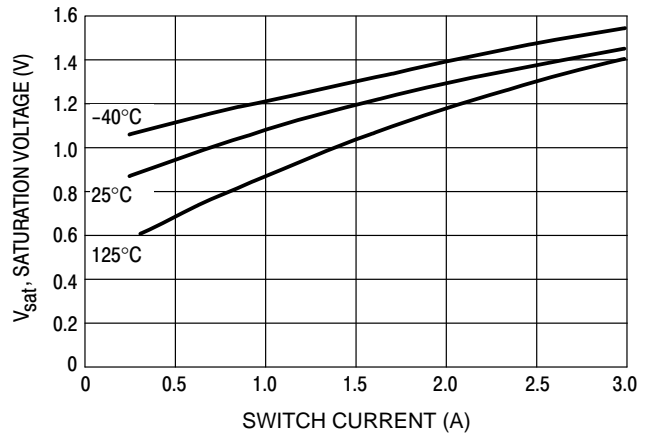


Figure 9. Switch Saturation Voltage

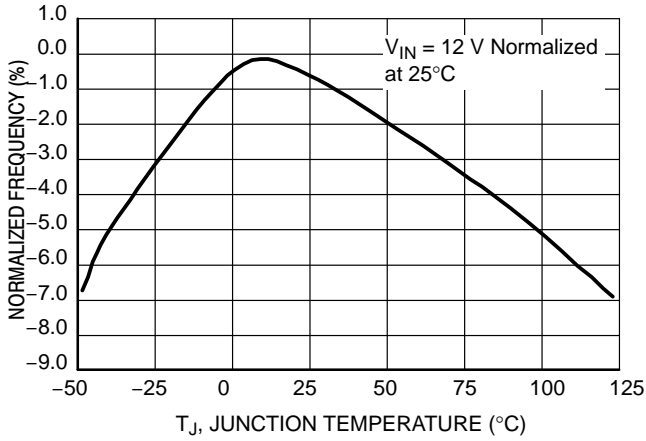


Figure 10. Switching Frequency

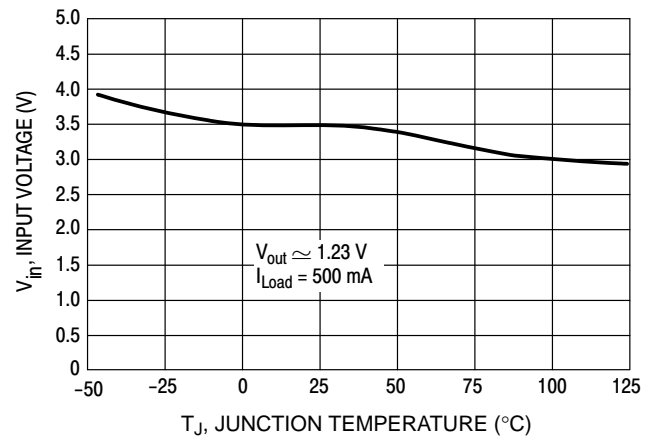


Figure 11. Minimum Supply Operating Voltage

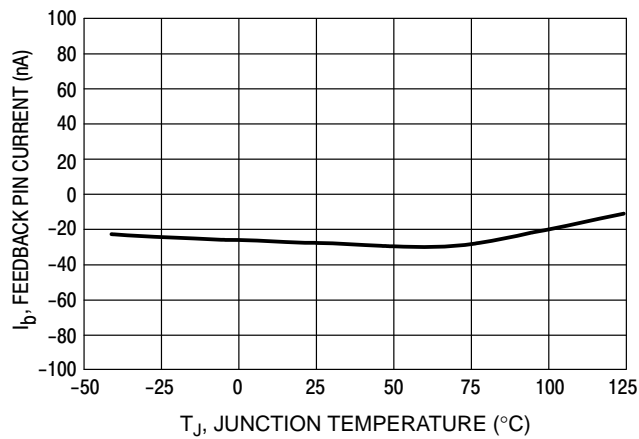


Figure 12. Feedback Pin Current

# LM2596

## TYPICAL PERFORMANCE CHARACTERISTICS (Circuit of Figure 15)

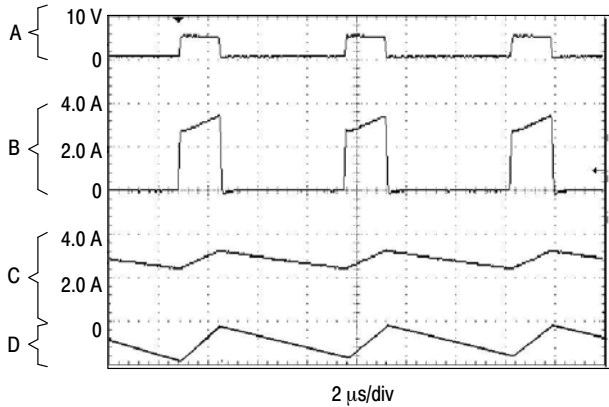


Figure 13. Switching Waveforms

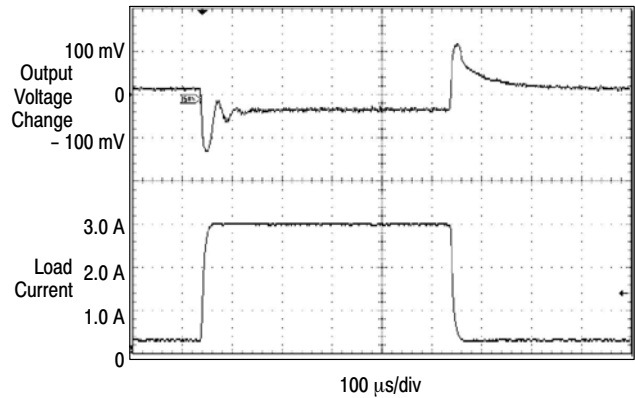
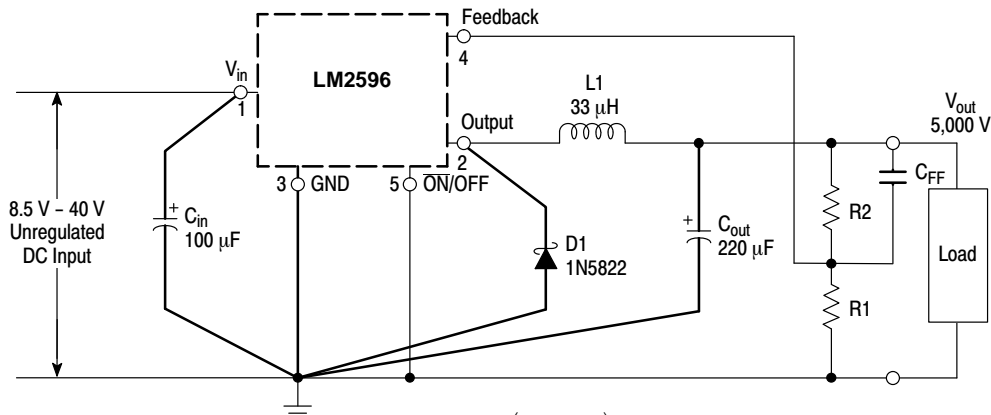


Figure 14. Load Transient Response

$V_{out} = 5\text{ V}$   
 A: Output Pin Voltage, 10 V/div  
 B: Switch Current, 2.0 A/div  
 C: Inductor Current, 2.0 A/div, AC-Coupled  
 D: Output Ripple Voltage, 50 mV/div, AC-Coupled  
 Horizontal Time Base: 5.0 μs/div

### Adjustable Output Voltage Versions



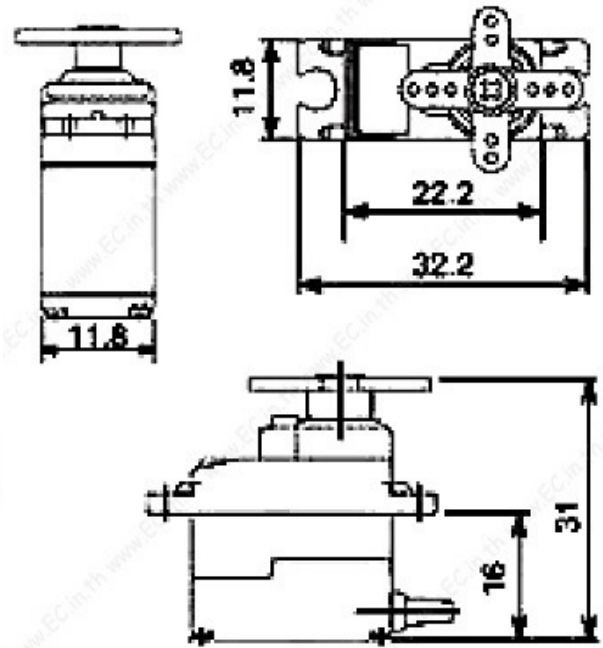
$$V_{out} = V_{ref} \left( 1.0 + \frac{R2}{R1} \right)$$

$$R2 = R1 \left( \frac{V_{out}}{V_{ref}} - 1.0 \right)$$

Where  $V_{ref} = 1.23\text{ V}$ ,  $R1$  between 1.0 k and 5.0 k

Figure 15. Typical Test Circuit

# SG90 9 g Micro Servo

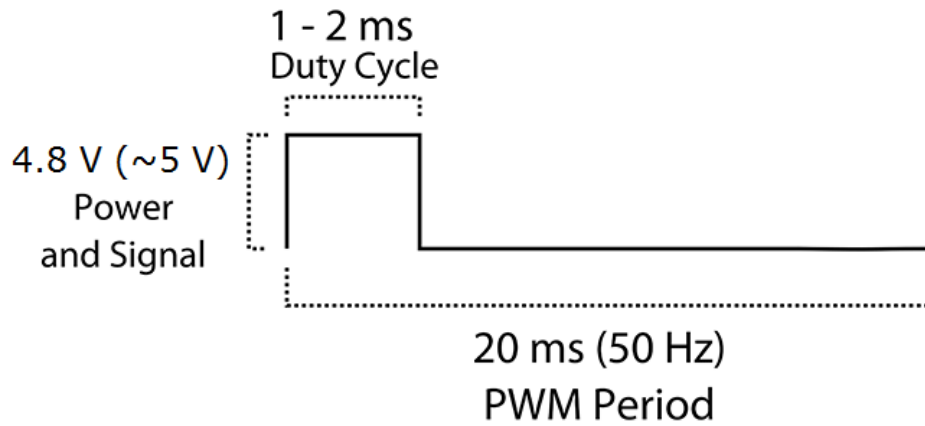
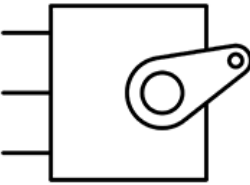


Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but *smaller*. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

## Specifications

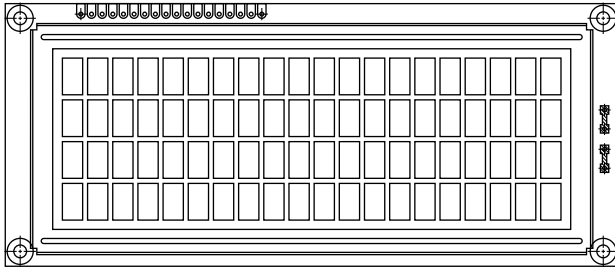
- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10  $\mu$ s
- Temperature range: 0 °C – 55 °C

PWM=Orange (  $\square$  )  
Vcc = Red ( + )  
Ground=Brown ( - )



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.

## 20 x 4 Character LCD



### FEATURES

- Type: Character
- Display format: 20 x 4 characters
- Built-in controller: ST 7066 (or equivalent)
- Duty cycle: 1/16
- 5 x 8 dots includes cursor
- + 5 V power supply (also available for + 3 V)
- LED can be driven by pin 1, pin 2, pin 15, pin 16 or A and K
- N.V. optional for + 3 V power supply
- Material categorization: For definitions of compliance please see [www.vishay.com/doc?99912](http://www.vishay.com/doc?99912)


**RoHS**  
COMPLIANT

MECHANICAL DATA		
ITEM	STANDARD VALUE	UNIT
Module Dimension	146.0 x 62.5	mm
Viewing Area	123.5 x 43.0	
Dot Size	0.92 x 1.10	
Dot Pitch	0.98 x 1.16	
Mounting Hole	139.0 x 55.5	
Character Size	4.84 x 9.22	

ABSOLUTE MAXIMUM RATINGS					
ITEM	SYMBOL	STANDARD VALUE			UNIT
		MIN.	TYP.	MAX.	
Power Supply	$V_{DD}$ to $V_{SS}$	- 0.3	-	7.0	V
Input Voltage	$V_I$	- 0.3	-	$V_{DD}$	

### Note

- $V_{SS} = 0$  V,  $V_{DD} = 5.0$  V

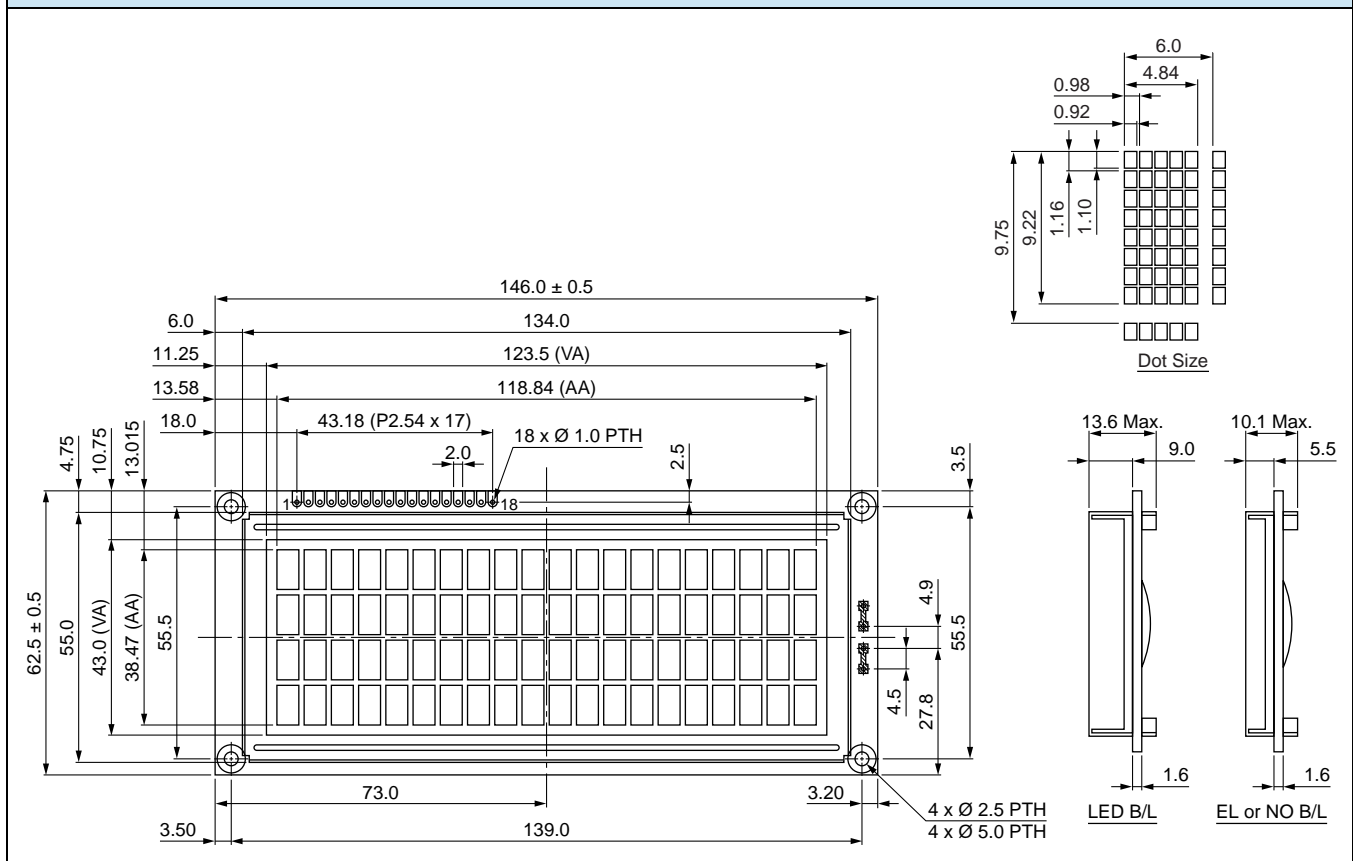
ELECTRICAL CHARACTERISTICS						
ITEM	SYMBOL	CONDITION	STANDARD VALUE			UNIT
			MIN.	TYP.	MAX.	
Input Voltage	$V_{DD}$	$V_{DD} = + 5$ V	4.7	5.0	5.3	V
		$V_{DD} = + 3$ V	2.7	3.0	5.3	
Supply Current	$I_{DD}$	$V_{DD} = + 5$ V	-	8.0	10.0	mA
Recommended LC Driving Voltage for Normal Temperature Version Module	$V_{DD}$ to $V_0$	- 20 °C	5.0	5.1	5.7	V
		0 °C	4.6	4.8	5.2	
		25 °C	4.1	4.5	4.7	
		50 °C	3.9	4.2	4.5	
		70 °C	3.7	3.9	4.3	
LED Forward Voltage	$V_F$	25 °C	-	4.2	4.6	V
LED Forward Current	$I_F$	25 °C	-	540	1080	mA
EL Power Supply Current	$I_{EL}$	$V_{EL} = 110$ V <sub>AC</sub> , 400 Hz	-	-	5.0	mA

OPTIONS									
PROCESS COLOR						BACKLIGHT			
TN	STN Gray	STN Yellow	STN Blue	FSTN B&W	STN Color	None	LED	EL	CCFL
x	x	x	x	x		x	x	x	

For detailed information, please see the "Product Numbering System" document.

DISPLAY CHARACTER ADDRESS CODE																				
Display Position																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
DD RAM Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
DD RAM Address	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53
DD RAM Address	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
DD RAM Address	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

INTERFACE PIN FUNCTION		
PIN NO.	SYMBOL	FUNCTION
1	$V_{SS}$	Ground
2	$V_{DD}$	+ 3 V or + 5 V
3	$V_0$	Contrast adjustment
4	RS	H/L register select signal
5	$R/\bar{W}$	H/L read/write signal
6	E	H → L enable signal
7	DB0	H/L data bus line
8	DB1	H/L data bus line
9	DB2	H/L data bus line
10	DB3	H/L data bus line
11	DB4	H/L data bus line
12	DB5	H/L data bus line
13	DB6	H/L data bus line
14	DB7	H/L data bus line
15	A	Power supply for LED (4.2 V)
16	K	Power supply for B/L (0 V)
17	NC/ $V_{EE}$	NC or negative voltage output
18	NC	NC connection

**DIMENSIONS** in millimeters


## 4x4 Matrix Membrane Keypad (#27899)

This 16-button keypad provides a useful human interface component for microcontroller projects. Convenient adhesive backing provides a simple way to mount the keypad in a variety of applications.

### Features

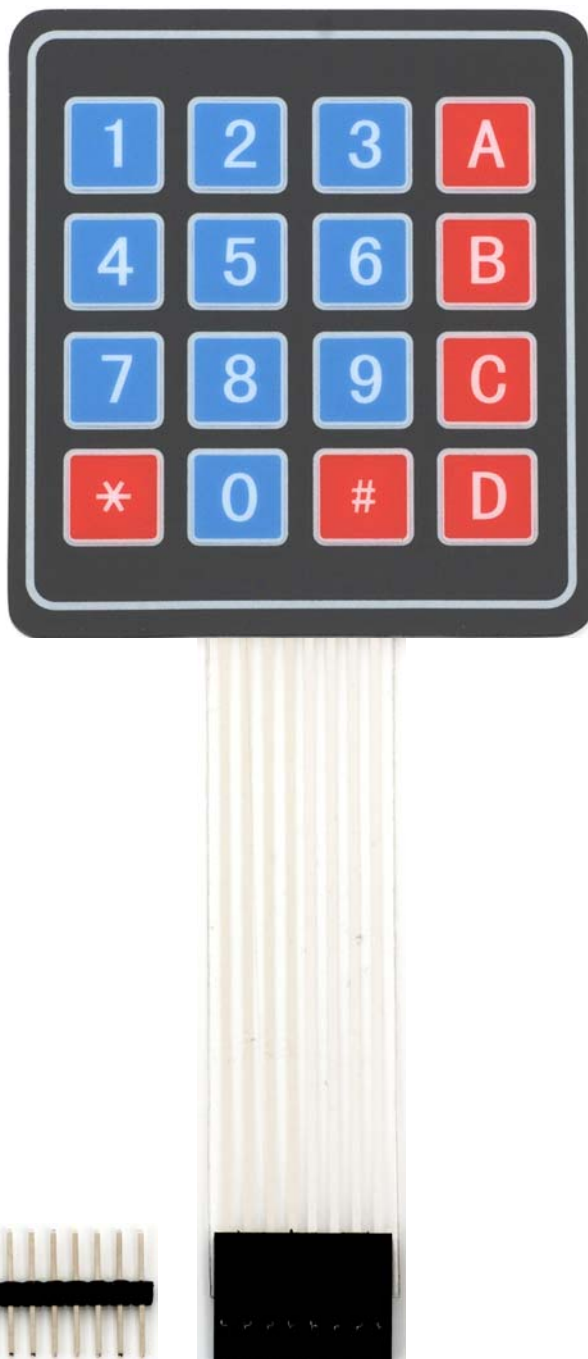
- Ultra-thin design
- Adhesive backing
- Excellent price/performance ratio
- Easy interface to any microcontroller
- Example programs provided for the BASIC Stamp 2 and Propeller P8X32A microcontrollers

### Key Specifications

- Maximum Rating: 24 VDC, 30 mA
- Interface: 8-pin access to 4x4 matrix
- Operating temperature: 32 to 122 °F (0 to 50°C)
- Dimensions:  
Keypad, 2.7 x 3.0 in (6.9 x 7.6 cm)  
Cable, 0.78 x 3.5 in (2.0 x 8.8 cm)

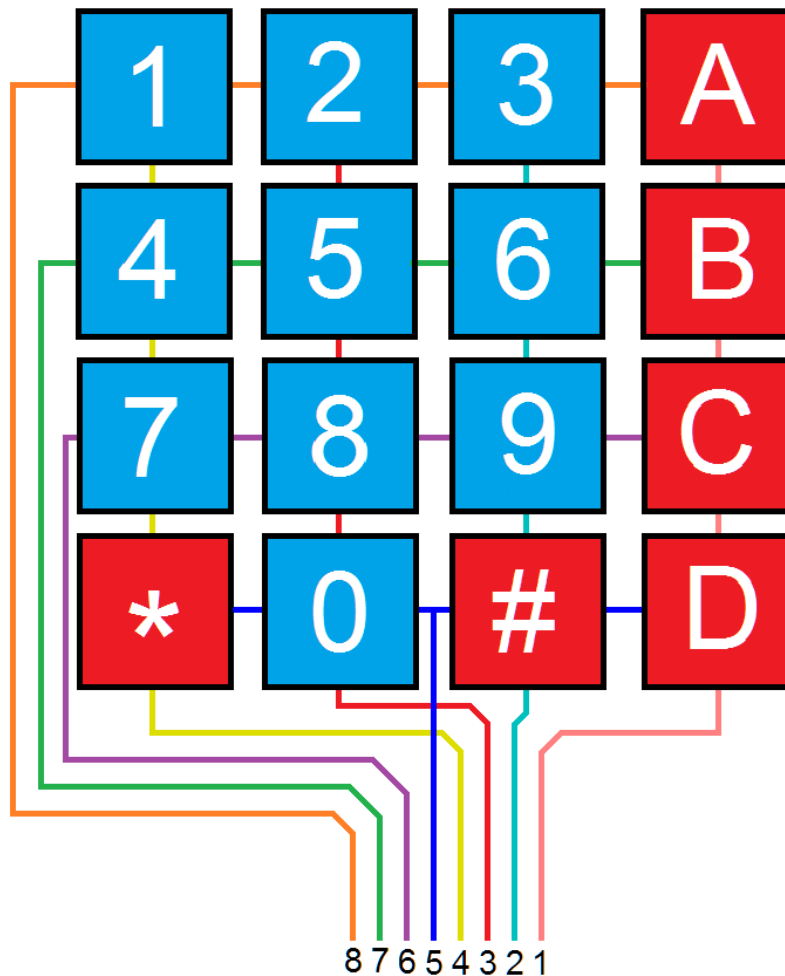
### Application Ideas

- Security systems
- Menu selection
- Data entry for embedded systems



## How it Works

Matrix keypads use a combination of four rows and four columns to provide button states to the host device, typically a microcontroller. Underneath each key is a pushbutton, with one end connected to one row, and the other end connected to one column. These connections are shown in Figure 1.



**Figure 1: Matrix Keypad Connections**

In order for the microcontroller to determine which button is pressed, it first needs to pull each of the four columns (pins 1-4) either low or high one at a time, and then poll the states of the four rows (pins 5-8). Depending on the states of the columns, the microcontroller can tell which button is pressed.

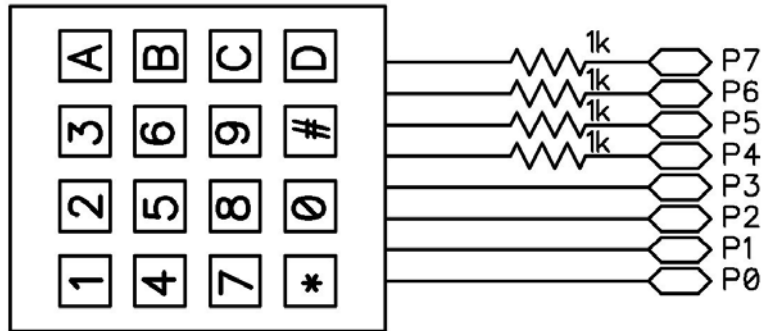
For example, say your program pulls all four columns low and then pulls the first row high. It then reads the input states of each column, and reads pin 1 high. This means that a contact has been made between column 4 and row 1, so button 'A' has been pressed.



## Connection Diagrams

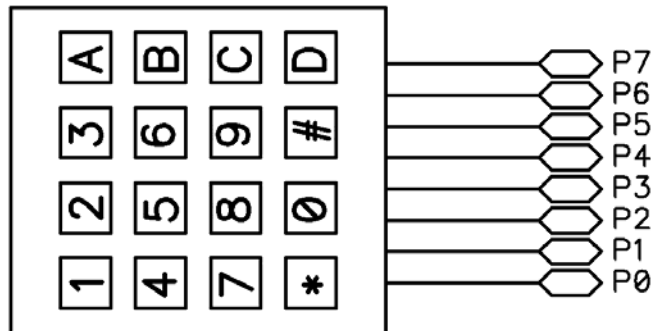
**Figure 2**

For use with the BASIC Stamp example program listed below.



**Figure 3**

For use with the Propeller P8X32A example program listed below.



## BASIC Stamp® Example Code

The example code below displays the button states of the 4x4 Matrix Membrane Keypad. It uses the Debug Terminal, which is built into the BASIC Stamp Editor software. The software is a free download from [www.parallax.com/basicstampsoftware](http://www.parallax.com/basicstampsoftware).

```
' 4x4MatrixKeypad_Demo.bs2
' Display buttons pressed on the 4x4 Matrix Membrane Keypad
' Author: Parallax HK Engineering

' {$STAMP BS2}
' {$PBASIC 2.5}

row          VAR Nib          ' Variable space for row counting
column      VAR Nib          ' Variable space for column counting
keypad      VAR Word         ' Variable space to store keypad output
keypadOld   VAR Word         ' Variable space to store old keypad output
temp        VAR Nib          ' Variable space for polling column states

DEBUG CLS                    ' Clear Debug Terminal
GOSUB Update                  ' Display keypad graphic

DO
  GOSUB ReadKeypad           ' Read keypad button states
  DEBUG HOME, BIN16 keypad, CR, CR, ' Display 16-bit keypad value
  BIN4 keypad >> 12, CR, ' Display 1st row 4-bit keypad value
  BIN4 keypad >> 8, CR, ' Display 2nd row 4-bit keypad value
  BIN4 keypad >> 4, CR, ' Display 3rd row 4-bit keypad value
  BIN4 keypad ' Display 4th row 4-bit keypad value
```

## General-purpose single bipolar timers

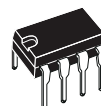
### Features

- Low turn-off time
- Maximum operating frequency greater than 500 kHz
- Timing from microseconds to hours
- Operates in both astable and monostable modes
- Output can source or sink up to 200 mA
- Adjustable duty cycle
- TTL compatible
- Temperature stability of 0.005% per °C

### Description

The NE555, SA555, and SE555 monolithic timing circuits are highly stable controllers capable of producing accurate time delays or oscillation. In the time delay mode of operation, the time is precisely controlled by one external resistor and capacitor. For a stable operation as an oscillator, the free running frequency and the duty cycle are both accurately controlled with two external resistors and one capacitor.

The circuit may be triggered and reset on falling waveforms, and the output structure can source or sink up to 200 mA.

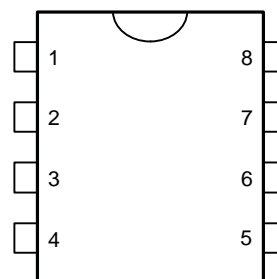


**N**  
**DIP8**  
(Plastic package)



**D**  
**SO8**  
(Plastic micropackage)

#### Pin connections (top view)



- |             |                     |
|-------------|---------------------|
| 1 - GND     | 5 - Control voltage |
| 2 - Trigger | 6 - Threshold       |
| 3 - Output  | 7 - Discharge       |
| 4 - Reset   | 8 - V <sub>CC</sub> |

# 1 Schematic diagrams

Figure 1. Block diagram

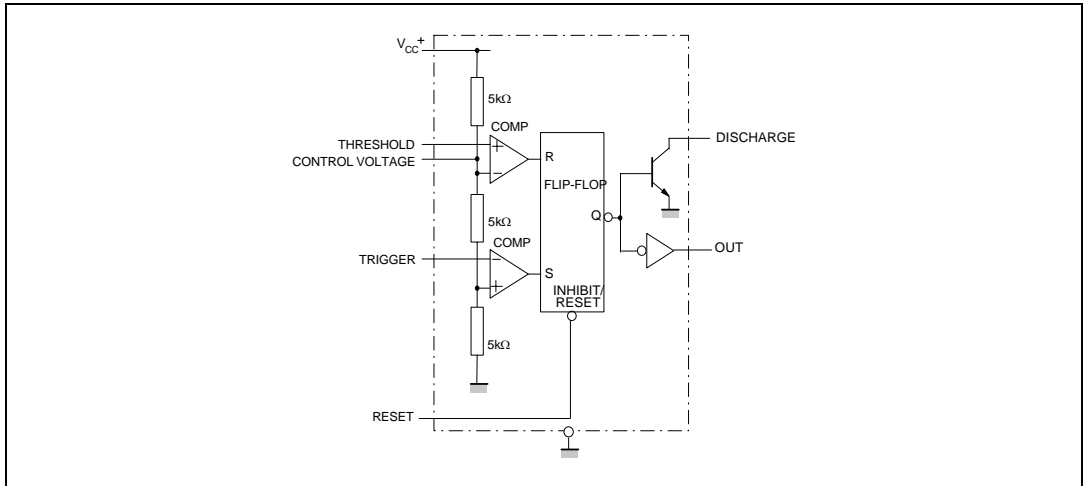
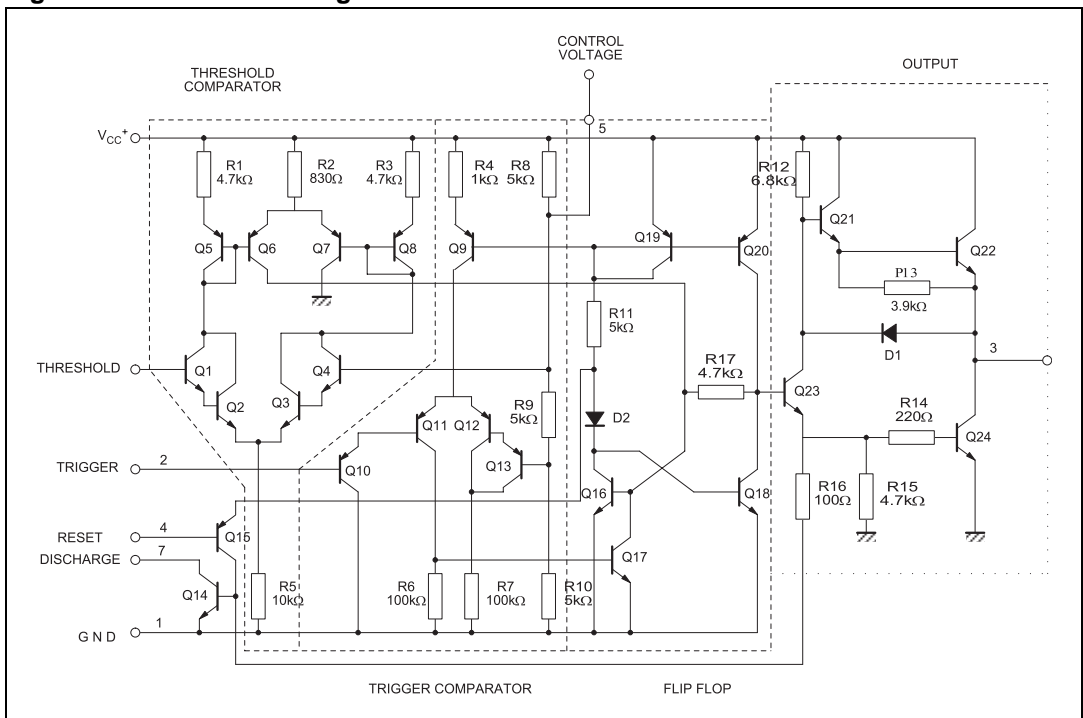


Figure 2. Schematic diagram



## 2 Absolute maximum ratings and operating conditions

**Table 1. Absolute maximum ratings**

Symbol	Parameter	Value	Unit
$V_{CC}$	Supply voltage	18	V
$I_{OUT}$	Output current (sink & source)	±225	mA
$R_{thja}$	Thermal resistance junction to ambient <sup>(1)</sup>		°C/W
	DIP8 SO-8	85 125	
$R_{thjc}$	Thermal resistance junction to case <sup>(1)</sup>		°C/W
	DIP8 SO-8	41 40	
ESD	Human body model (HBM) <sup>(2)</sup>	1000	V
	Machine model (MM) <sup>(3)</sup>	100	
	Charged device model (CDM) <sup>(4)</sup>	1500	
	Latch-up immunity	200	mA
$T_{LEAD}$	Lead temperature (soldering 10 seconds)	260	°C
$T_j$	Junction temperature	150	°C
$T_{stg}$	Storage temperature range	-65 to 150	°C

- Short-circuits can cause excessive heating. These values are typical.
- Human body model: a 100 pF capacitor is charged to the specified voltage, then discharged through a 1.5 kΩ resistor between two pins of the device. This is done for all couples of connected pin combinations while the other pins are floating.
- Machine model: a 200 pF capacitor is charged to the specified voltage, then discharged directly between two pins of the device with no external series resistor (internal resistor < 5 Ω). This is done for all couples of connected pin combinations while the other pins are floating.
- Charged device model: all pins and the package are charged together to the specified voltage and then discharged directly to the ground through only one pin. This is done for all pins.

**Table 2. Operating conditions**

Symbol	Parameter	Value	Unit
$V_{CC}$	Supply voltage		V
	NE555	4.5 to 16	
	SA555 SE555	4.5 to 16 4.5 to 18	
$V_{th}$ , $V_{trig}$ , $V_{cl}$ , $V_{reset}$	Maximum input voltage	$V_{CC}$	V
$I_{OUT}$	Output current (sink and source)	±200	mA
$T_{oper}$	Operating free air temperature range		°C
	NE555	0 to 70	
	SA555 SE555	-40 to 105 -55 to 125	

### 3 Electrical characteristics

Table 3.  $T_{amb} = +25^{\circ}C$ ,  $V_{CC} = +5V$  to  $+15V$  (unless otherwise specified)

Symbol	Parameter	SE555			NE555 - SA555			Unit
		Min.	Typ.	Max.	Min.	Typ.	Max.	
$I_{CC}$	Supply current ( $R_L = \infty$ )							mA
	Low state $V_{CC} = +5V$		3	5		3	6	
	$V_{CC} = +15V$		10	12		10	15	
	High state $V_{CC} = +5V$		2			2		
	Timing error (monostable) ( $R_A = 2k\Omega$ to $100k\Omega$ , $C = 0.1\mu F$ )							% ppm/ $^{\circ}C$ %/V
	Initial accuracy <sup>(1)</sup>		0.5	2		1	3	
	Drift with temperature		30	100		50		
	Drift with supply voltage		0.05	0.2		0.1	0.5	
	Timing error (astable) ( $R_A, R_B = 1k\Omega$ to $100k\Omega$ , $C = 0.1\mu F$ , $V_{CC} = +15V$ )							% ppm/ $^{\circ}C$ %/V
	Initial accuracy <sup>(1)</sup>		1.5			2.25		
	Drift with temperature		90			150		
	Drift with supply voltage		0.15			0.3		
$V_{CL}$	Control voltage level							V
	$V_{CC} = +15V$	9.6	10	10.4	9	10	11	
	$V_{CC} = +5V$	2.9	3.33	3.8	2.6	3.33	4	
$V_{th}$	Threshold voltage							V
	$V_{CC} = +15V$	9.4	10	10.6	8.8	10	11.2	
	$V_{CC} = +5V$	2.7	3.33	4	2.4	3.33	4.2	
$I_{th}$	Threshold current <sup>(2)</sup>		0.1	0.25		0.1	0.25	$\mu A$
$V_{trig}$	Trigger voltage							V
	$V_{CC} = +15V$	4.8	5	5.2	4.5	5	5.6	
	$V_{CC} = +5V$	1.45	1.67	1.9	1.1	1.67	2.2	
$I_{trig}$	Trigger current ( $V_{trig} = 0V$ )		0.5	0.9		0.5	2.0	$\mu A$
$V_{reset}$	Reset voltage <sup>(3)</sup>	0.4	0.7	1	0.4	0.7	1	V
$I_{reset}$	Reset current							mA
	$V_{reset} = +0.4V$		0.1	0.4		0.1	0.4	
	$V_{reset} = 0V$		0.4	1		0.4	1.5	
$V_{OL}$	Low level output voltage							V
	$V_{CC} = +15V$ $I_{O(sink)} = 10mA$		0.1	0.15		0.1	0.25	
	$I_{O(sink)} = 50mA$		0.4	0.5		0.4	0.75	
	$I_{O(sink)} = 100mA$		2	2.2		2	2.5	
	$I_{O(sink)} = 200mA$		2.5			2.5		
	$V_{CC} = +5V$ $I_{O(sink)} = 8mA$		0.1	0.25		0.3	0.4	
	$I_{O(sink)} = 5mA$		0.05	0.2		0.25	0.35	

Table 3.  $T_{amb} = +25^{\circ}C$ ,  $V_{CC} = +5V$  to  $+15V$  (unless otherwise specified) (continued)

Symbol	Parameter	SE555			NE555 - SA555			Unit
		Min.	Typ.	Max.	Min.	Typ.	Max.	
$V_{OH}$	High level output voltage $V_{CC} = +15V$ $I_{O(sink)} = 200mA$ $I_{O(sink)} = 100mA$ $V_{CC} = +5V$ $I_{O(sink)} = 100mA$	13 3	12.5 13.3 3.3		12.7 5 2.75	12.5 13.3 3.3		V
$I_{dis(off)}$	Discharge pin leakage current (output high) $V_{dis} = 10V$		20	100		20	100	nA
$V_{dis(sat)}$	Discharge pin saturation voltage (output low) <sup>(4)</sup> $V_{CC} = +15V$ , $I_{dis} = 15mA$ $V_{CC} = +5V$ , $I_{dis} = 4.5mA$		180 80	480 200		180 80	480 200	mV
$t_r$ $t_f$	Output rise time Output fall time		100 100	200 200		100 100	300 300	ns
$t_{off}$	Turn off time <sup>(5)</sup> ( $V_{reset} = V_{CC}$ )		0.5			0.5		$\mu s$

1. Tested at  $V_{CC} = +5V$  and  $V_{CC} = +15V$ .
2. This will determine the maximum value of  $R_A + R_B$  for 15 V operation. The maximum total ( $R_A + R_B$ ) is 20 M $\Omega$  for +15 V operation and 3.5 M $\Omega$  for +5 V operation.
3. Specified with trigger input high.
4. No protection against excessive pin 7 current is necessary, providing the package dissipation rating is not exceeded.
5. Time measured from a positive pulse (from 0 V to  $0.8 \times V_{CC}$ ) on the threshold pin to the transition from high to low on the output pin. Trigger is tied to threshold.

Figure 3. Minimum pulse width required for triggering

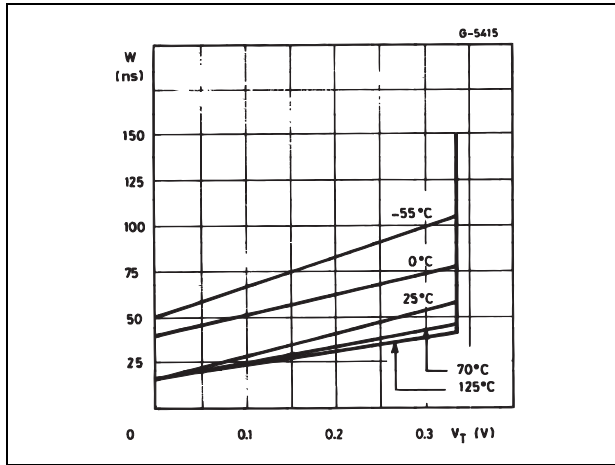


Figure 4. Supply current versus supply voltage

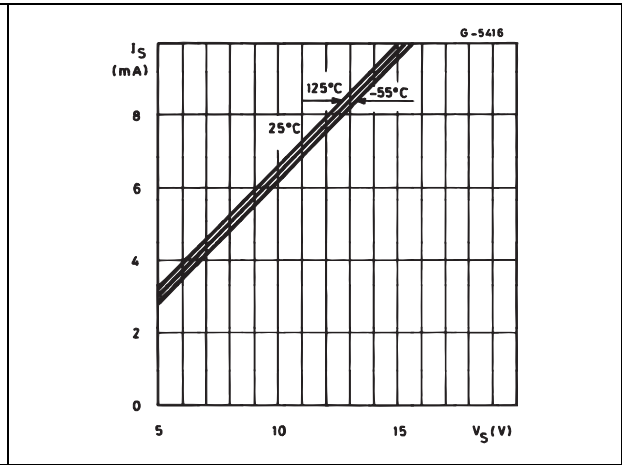


Figure 5. Delay time versus temperature

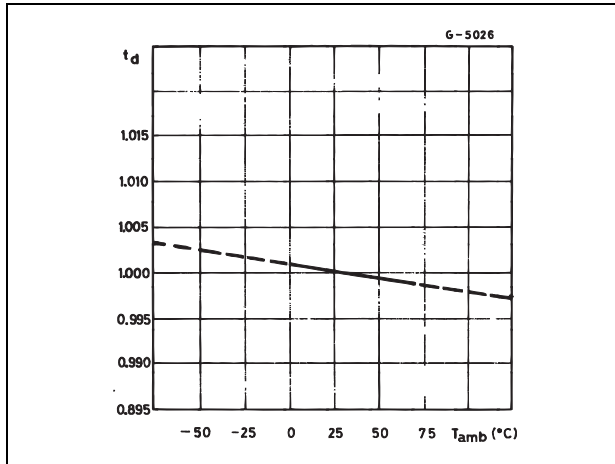


Figure 6. Low output voltage versus output sink current

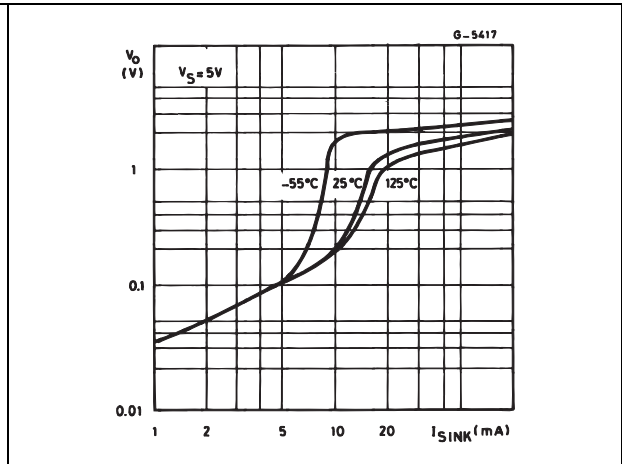


Figure 7. Low output voltage versus output sink current

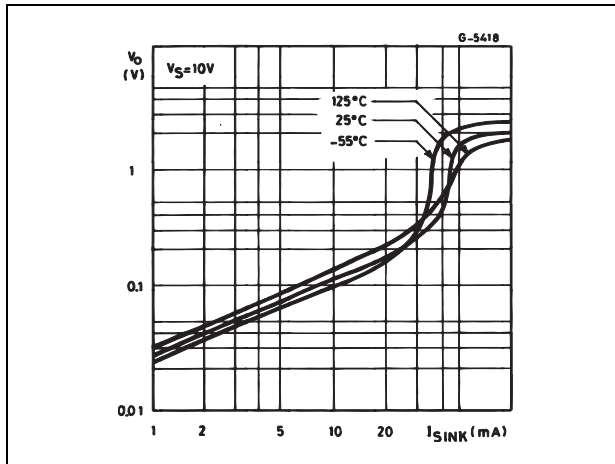


Figure 8. Low output voltage versus output sink current

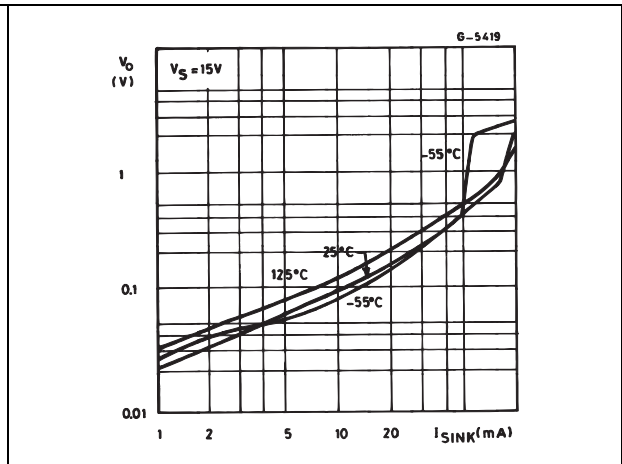


Figure 9. High output voltage drop versus output

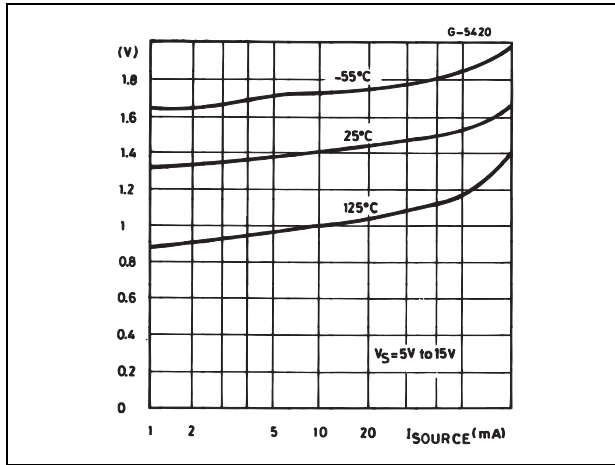


Figure 10. Delay time versus supply voltage

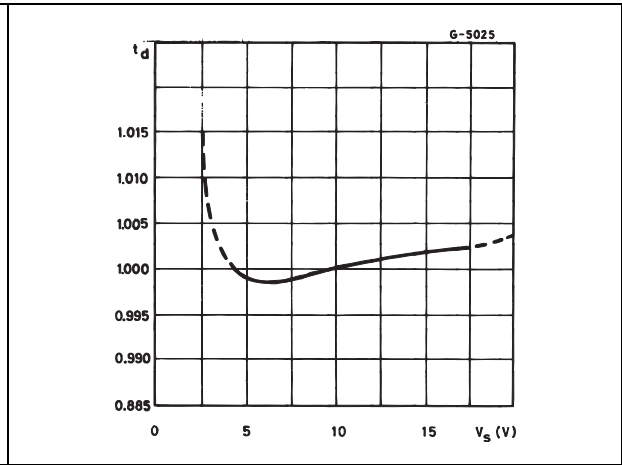
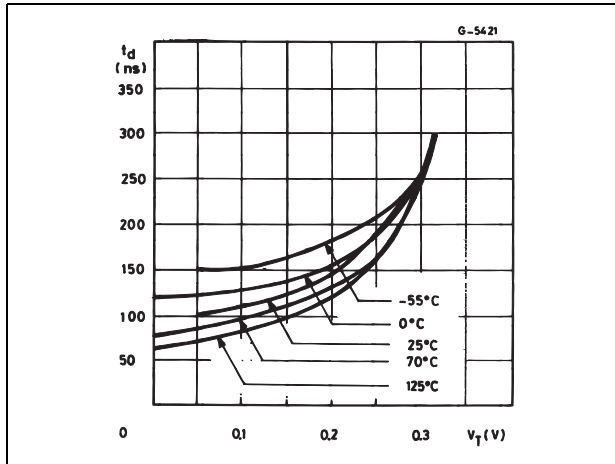


Figure 11. Propagation delay versus voltage level of trigger value



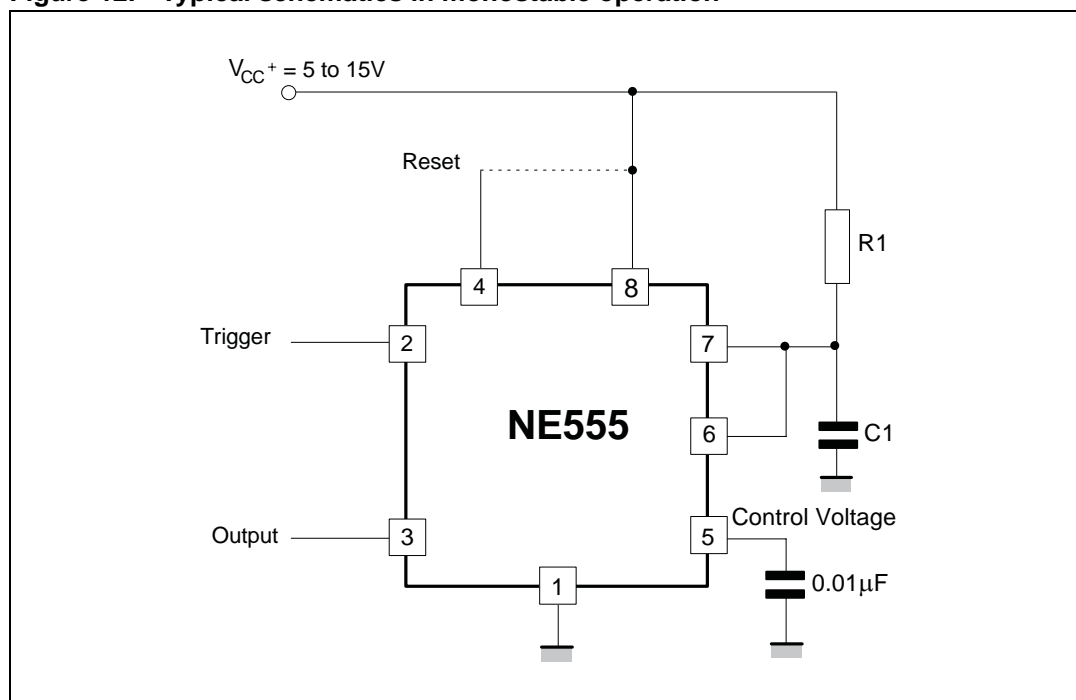


## 4 Application information

### 4.1 Monostable operation

In the monostable mode, the timer generates a single pulse. As shown in [Figure 12](#), the external capacitor is initially held discharged by a transistor inside the timer.

**Figure 12. Typical schematics in monostable operation**



The circuit triggers on a negative-going input signal when the level reaches  $1/3 V_{CC}$ . Once triggered, the circuit remains in this state until the set time has elapsed, even if it is triggered again during this interval. The duration of the output HIGH state is given by  $t = 1.1 R_1 C_1$  and is easily determined by [Figure 14](#).

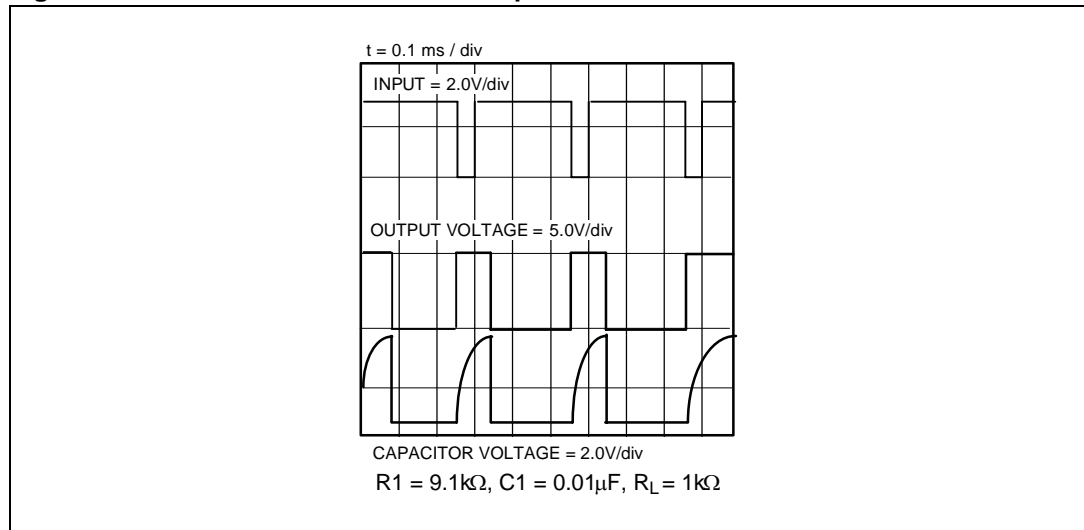
Note that because the charge rate and the threshold level of the comparator are both directly proportional to supply voltage, the timing interval is independent of supply. Applying a negative pulse simultaneously to the reset terminal (pin 4) and the trigger terminal (pin 2) during the timing cycle discharges the external capacitor and causes the cycle to start over. The timing cycle now starts on the positive edge of the reset pulse. During the time the reset pulse is applied, the output is driven to its LOW state.

When a negative trigger pulse is applied to pin 2, the flip-flop is set, releasing the short-circuit across the external capacitor and driving the output HIGH. The voltage across the capacitor increases exponentially with the time constant  $t = R_1 C_1$ . When the voltage across the capacitor equals  $2/3 V_{CC}$ , the comparator resets the flip-flop which then discharges the capacitor rapidly and drives the output to its LOW state.

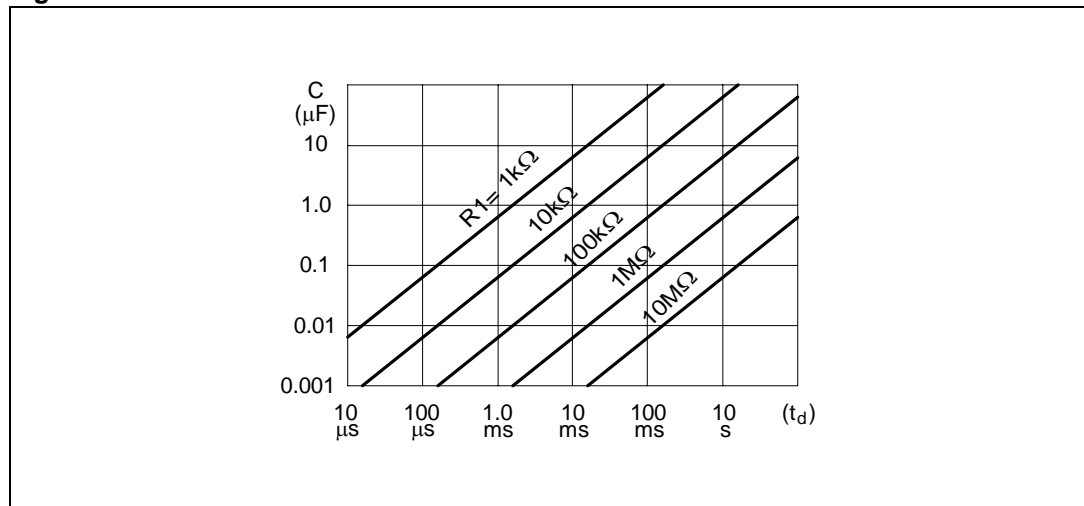
[Figure 13](#) shows the actual waveforms generated in this mode of operation.

When Reset is not used, it should be tied high to avoid any possibility of unwanted triggering.

**Figure 13. Waveforms in monostable operation**



**Figure 14. Pulse duration versus R1C1**



## 4.2 Astable operation

When the circuit is connected as shown in [Figure 15](#) (pins 2 and 6 connected) it triggers itself and free runs as a multi-vibrator. The external capacitor charges through  $R_1$  and  $R_2$  and discharges through  $R_2$  only. Thus the duty cycle can be set accurately by adjusting the ratio of these two resistors.

In the astable mode of operation,  $C_1$  charges and discharges between  $1/3 V_{CC}$  and  $2/3 V_{CC}$ . As in the triggered mode, the charge and discharge times and, therefore, frequency are independent of the supply voltage.

Figure 15. Typical schematics in astable operation

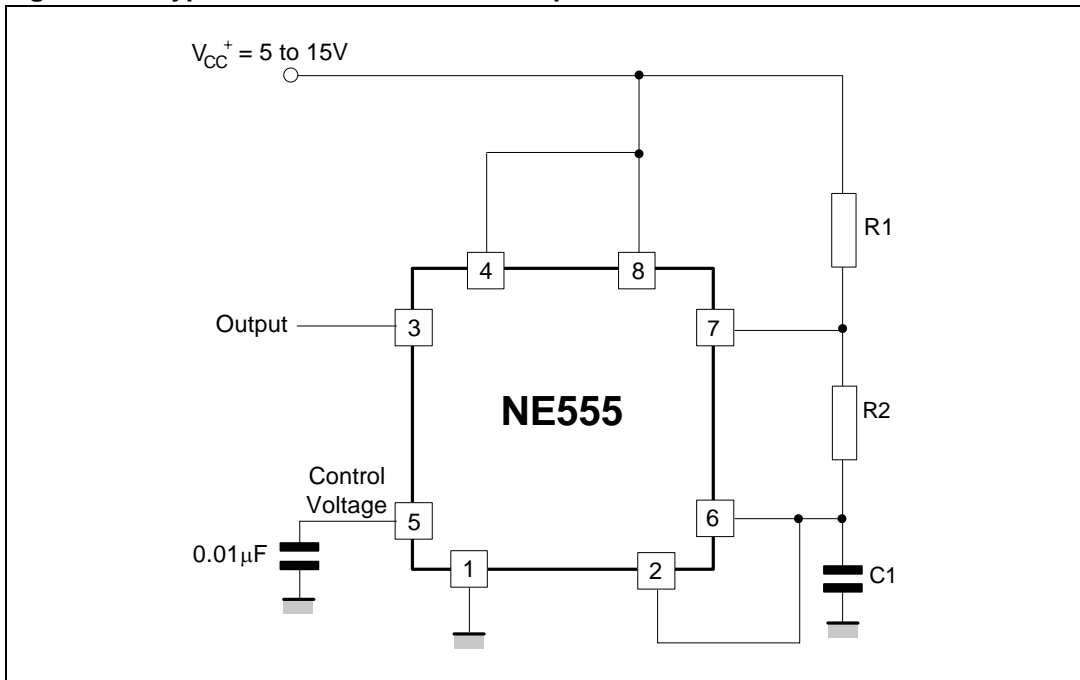


Figure 16 shows the actual waveforms generated in this mode of operation.

The charge time (output HIGH) is given by:

$$t1 = 0.693 (R1 + R2) C1$$

and the discharge time (output LOW) by:

$$t2 = 0.693 (R2) C1$$

Thus the total period T is given by:

$$T = t1 + t2 = 0.693 (R1 + 2R2) C1$$

The frequency of oscillation is then:

$$f = \frac{1}{T} = \frac{1.44}{(R1 + 2R2)C1}$$

It can easily be found from Figure 17.

The duty cycle is given by:

$$\frac{t1}{(t1 + t2)} = \frac{(R1 + R2)}{(R1 + 2 \cdot R2)} = 1 - \left[ \frac{R2}{(R1 + R2)} \right]$$



Is Now Part of



**ON Semiconductor®**

To learn more about ON Semiconductor, please visit our website at  
[www.onsemi.com](http://www.onsemi.com)

ON Semiconductor and the ON Semiconductor logo are trademarks of Semiconductor Components Industries, LLC dba ON Semiconductor or its subsidiaries in the United States and/or other countries. ON Semiconductor owns the rights to a number of patents, trademarks, copyrights, trade secrets, and other intellectual property. A listing of ON Semiconductor's product/patent coverage may be accessed at [www.onsemi.com/site/pdf/Patent-Marking.pdf](http://www.onsemi.com/site/pdf/Patent-Marking.pdf). ON Semiconductor reserves the right to make changes without further notice to any products herein. ON Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does ON Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages. Buyer is responsible for its products and applications using ON Semiconductor products, including compliance with all laws, regulations and safety requirements or standards, regardless of any support or applications information provided by ON Semiconductor. "Typical" parameters which may be provided in ON Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. ON Semiconductor does not convey any license under its patent rights nor the rights of others. ON Semiconductor products are not designed, intended, or authorized for use as a critical component in life support systems or any FDA Class 3 medical devices or medical devices with a same or similar classification in a foreign jurisdiction or any devices intended for implantation in the human body. Should Buyer purchase or use ON Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold ON Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that ON Semiconductor was negligent regarding the design or manufacture of the part. ON Semiconductor is an Equal Opportunity/Affirmative Action Employer. This literature is subject to all applicable copyright laws and is not to be reprinted in any manner.



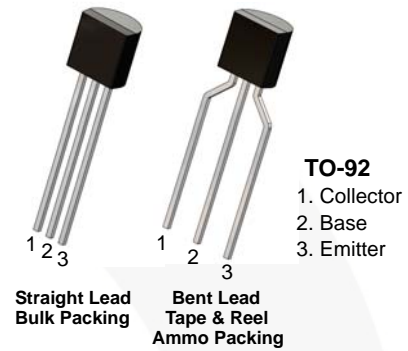
January 2016

# BC556 / BC557 / BC558 / BC559 / BC560

## PNP Epitaxial Silicon Transistor

### Features

- Switching and Amplifier
- High-Voltage: BC556,  $V_{CEO} = -65\text{ V}$
- Low-Noise: BC559, BC560
- Complement to BC546, BC547, BC548, BC549, and BC550



### Ordering Information

Part Number	Marking	Package	Packing Method
BC556ABU	BC556A	TO-92 3L	Bulk
BC556ATA	BC556A	TO-92 3L	Ammo
BC556BTA	BC556B	TO-92 3L	Ammo
BC556BTF	BC556B	TO-92 3L	Tape and Reel
BC556BTFR	BC556B	TO-92 3L	Tape and Reel
BC557ATA	BC557A	TO-92 3L	Ammo
BC557BTA	BC557B	TO-92 3L	Ammo
BC557BTF	BC557B	TO-92 3L	Tape and Reel
BC558BTA	BC558B	TO-92 3L	Ammo
BC559BTA	BC559B	TO-92 3L	Ammo
BC559CTA	BC559C	TO-92 3L	Ammo
BC560CTA	BC560C	TO-92 3L	Ammo

BC556 / BC557 / BC558 / BC559 / BC560 — PNP Epitaxial Silicon Transistor

## Absolute Maximum Ratings

Stresses exceeding the absolute maximum ratings may damage the device. The device may not function or be operable above the recommended operating conditions and stressing the parts to these levels is not recommended. In addition, extended exposure to stresses above the recommended operating conditions may affect device reliability. The absolute maximum ratings are stress ratings only. Values are at  $T_A = 25^\circ\text{C}$  unless otherwise noted.

Symbol	Parameter	Value	Unit	
$V_{CBO}$	Collector-Base Voltage	BC556	-80	V
		BC557 / BC560	-50	
		BC558 / BC559	-30	
$V_{CEO}$	Collector-Emitter Voltage	BC556	-65	V
		BC557 / BC560	-45	
		BC558 / BC559	-30	
$V_{EBO}$	Emitter-Base Voltage	-5	V	
$I_C$	Collector Current (DC)	-100	mA	
$I_{CP}$	Peak Collector Current (Pulse)	-200	mA	
$I_{BP}$	Peak Base Current (Pulse)	-200	mA	
$T_J$	Junction Temperature	150	$^\circ\text{C}$	
$T_{STG}$	Storage Temperature Range	-65 to +150	$^\circ\text{C}$	

## Thermal Characteristics<sup>(1)</sup>

Values are at  $T_A = 25^\circ\text{C}$  unless otherwise noted.

Symbol	Parameter	Max.	Unit
$P_D$	Total Power Dissipation	500	mW
	Derate Above $25^\circ\text{C}$	4.0	mW/ $^\circ\text{C}$
$R_{\theta JA}$	Thermal Resistance, Junction-to-Ambient	250	$^\circ\text{C}/\text{W}$

### Note:

1. PCB size: FR-4, 76 mm x 114 mm x 1.57 mm (3.0 inch x 4.5 inch x 0.062 inch) with minimum land pattern size.

## Electrical Characteristics

Values are at  $T_A = 25^\circ\text{C}$  unless otherwise noted.

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
$I_{CBO}$	Collector Cut-Off Current	$V_{CB} = -30\text{ V}, I_E = 0$			-15	nA
$h_{FE}$	DC Current Gain	$V_{CE} = -5\text{ V}, I_C = -2\text{ mA}$	110		800	
$V_{CE(sat)}$	Collector-Emitter Saturation Voltage	$I_C = -10\text{ mA}, I_B = -0.5\text{ mA}$		-90	-300	mV
		$I_C = -100\text{ mA}, I_B = -5\text{ mA}$		-250	-650	
$V_{BE(sat)}$	Collector-Base Saturation Voltage	$I_C = -10\text{ mA}, I_B = -0.5\text{ mA}$		-700		mV
		$I_C = -100\text{ mA}, I_B = -5\text{ mA}$		-900		
$V_{BE(on)}$	Base-Emitter On Voltage	$V_{CE} = -5\text{ V}, I_C = -2\text{ mA}$	-600	-660	-750	mV
		$V_{CE} = -5\text{ V}, I_C = -10\text{ mA}$			-800	
$f_T$	Current Gain Bandwidth Product	$V_{CE} = -5\text{ V}, I_C = -10\text{ mA}, f = 10\text{ MHz}$		150		MHz
$C_{ob}$	Output Capacitance	$V_{CB} = -10\text{ V}, I_E = 0, f = 1\text{ MHz}$			6	pF
NF	Noise Figure	BC556 / BC557 / BC558	$V_{CE} = -5\text{ V}, I_C = -200\text{ }\mu\text{A}, f = 1\text{ kHz}, R_G = 2\text{ k}\Omega$	2	10	dB
		BC559 / BC560		1	4	
		BC559	$V_{CE} = -5\text{ V}, I_C = -200\text{ }\mu\text{A}, R_G = 2\text{ k}\Omega, f = 30\text{ to }15000\text{ MHz}$	1.2	4.0	
		BC560		1.2	2.0	

## $h_{FE}$ Classification

Classification	A	B	C
$h_{FE}$	110 ~ 220	200 ~ 450	420 ~ 800

## Typical Performance Characteristics

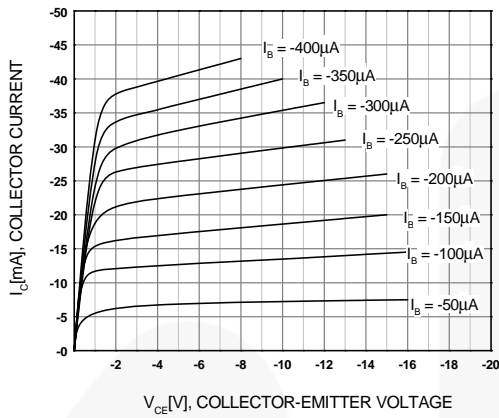


Figure 1. Static Characteristic

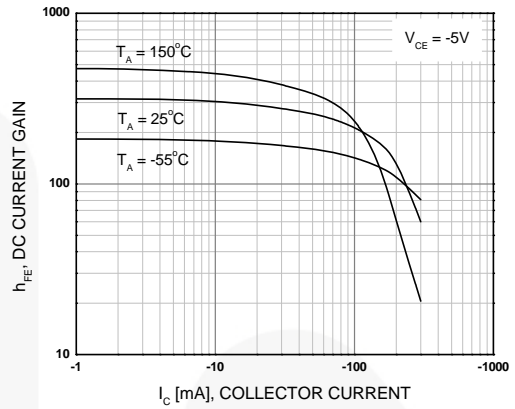


Figure 2. DC Current Gain

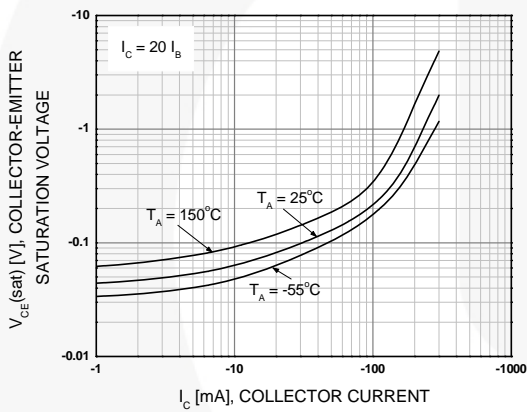


Figure 3. Collector-Emitter Saturation Voltage

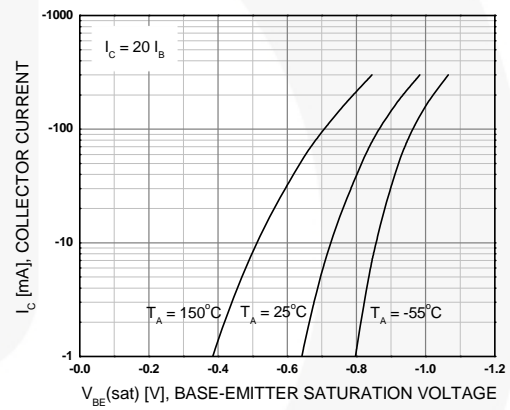


Figure 4. Base-Emitter Saturation Voltage

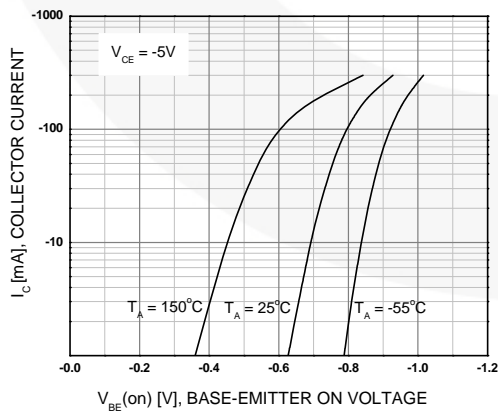


Figure 5. Base-Emitter On Voltage

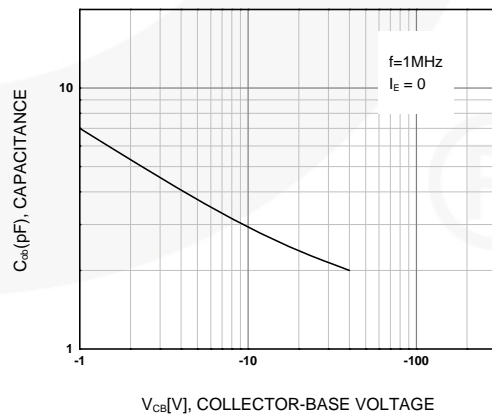


Figure 6. Collector Output Capacitance



Typical Performance Characteristics (Continued)

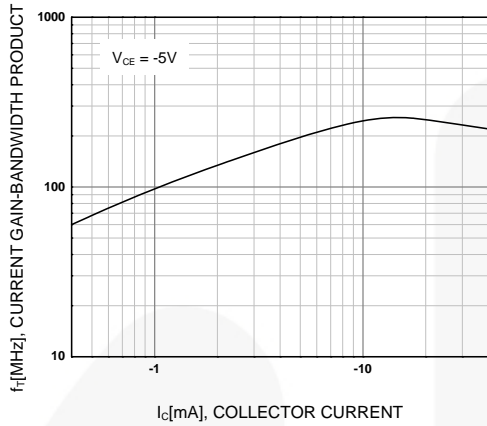


Figure 7. Current Gain Bandwidth Product

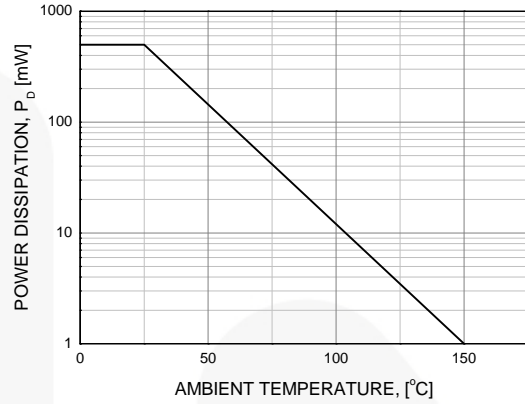


Figure 8. Power Deration

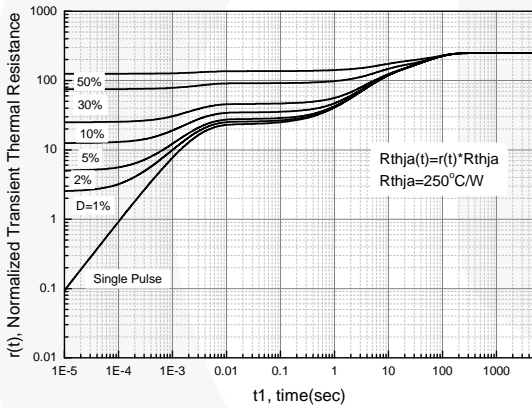
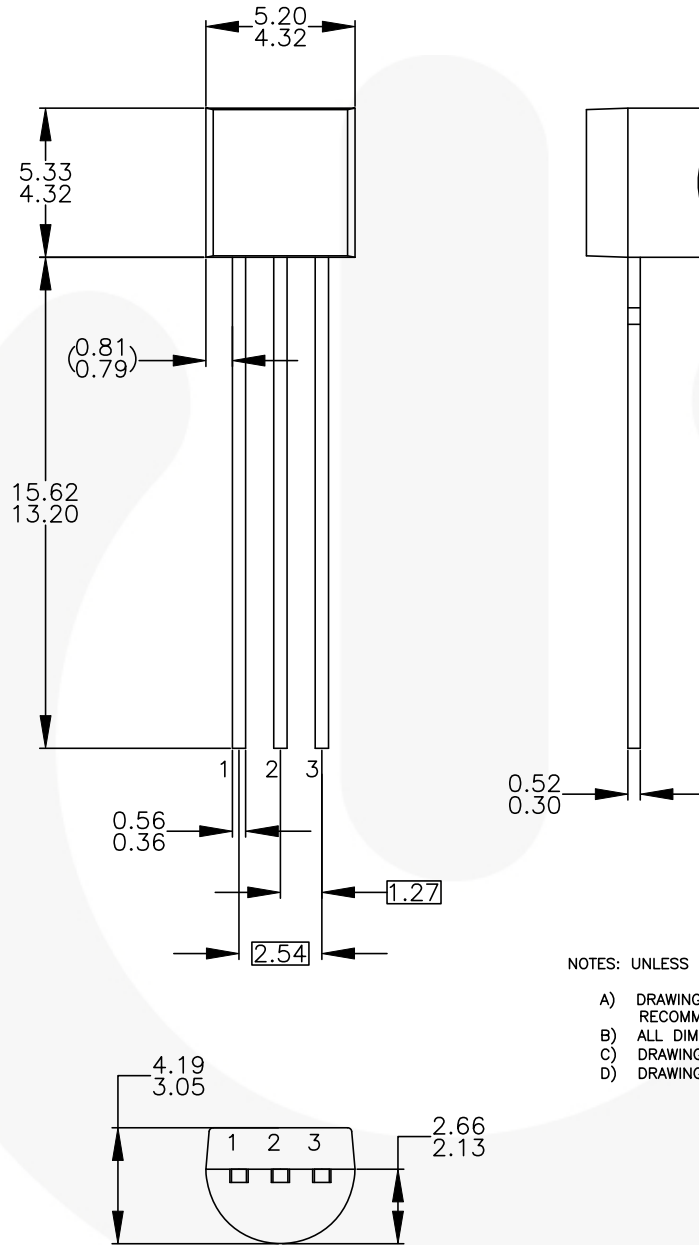


Figure 9. Normalized Transient Thermal Resistance

Physical Dimensions



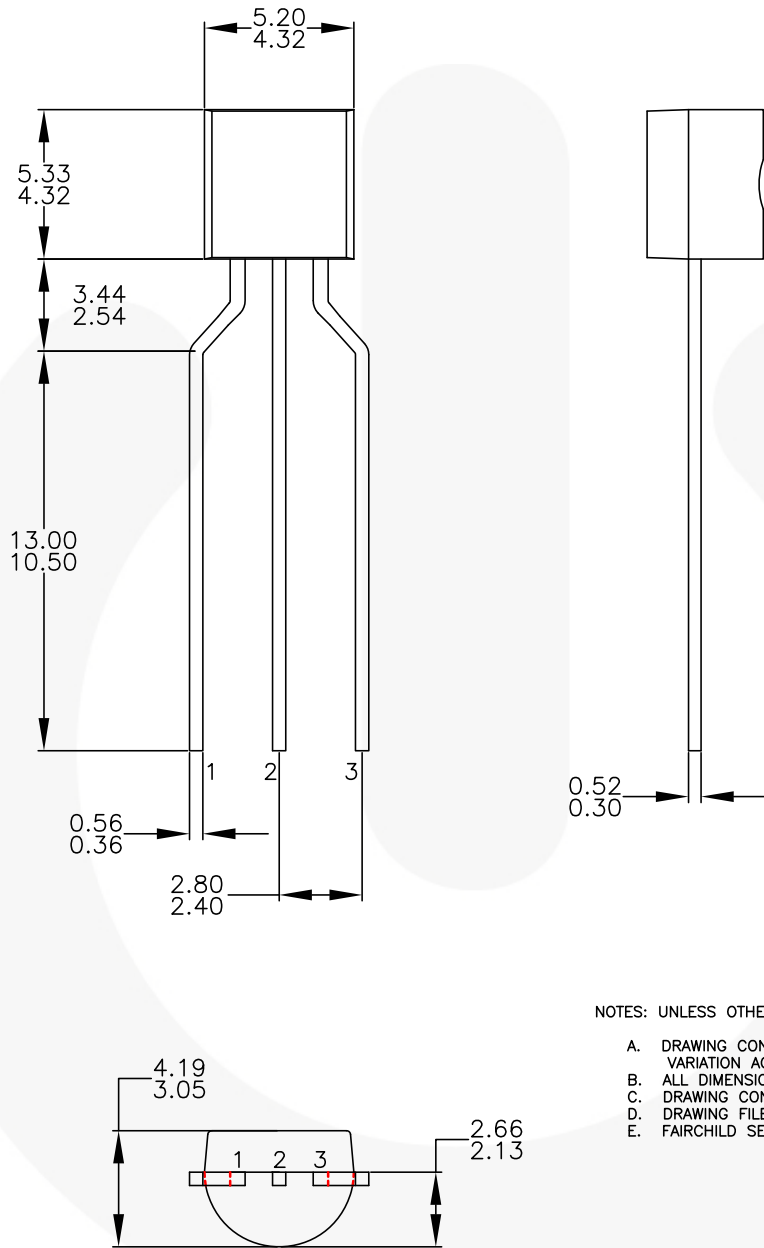
NOTES: UNLESS OTHERWISE SPECIFIED

- A) DRAWING WITH REFERENCE TO JEDEC TO-92 RECOMMENDATIONS.
- B) ALL DIMENSIONS ARE IN MILLIMETERS.
- C) DRAWING CONFORMS TO ASME Y14.5M-2009.
- D) DRAWING FILENAME: MKT-ZA03DREV4.



Figure 10. 3-LEAD, TO92, JEDEC TO-92 COMPLIANT STRAIGHT LEAD CONFIGURATION, BULK

Physical Dimensions (Continued)

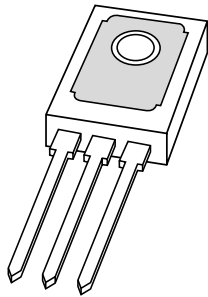


NOTES: UNLESS OTHERWISE SPECIFIED

- A. DRAWING CONFORMS TO JEDEC MS-013, VARIATION AC.
- B. ALL DIMENSIONS ARE IN MILLIMETERS.
- C. DRAWING CONFORMS TO ASME Y14.5M-2009.
- D. DRAWING FILENAME: MKT-ZA03FREV3.
- E. FAIRCHILD SEMICONDUCTOR.

Figure 11. 3-LEAD, TO92, MOLDED 0.200 IN LINE SPACING LEAD FORM, AMMO, TAPE AND REEL

# DATA SHEET



## **BD136; BD138; BD140** PNP power transistors

Product specification  
Supersedes data of 1997 Mar 26

1999 Apr 12

# PNP power transistors

# BD136; BD138; BD140

### FEATURES

- High current (max. 1.5 A)
- Low voltage (max. 80 V).

### APPLICATIONS

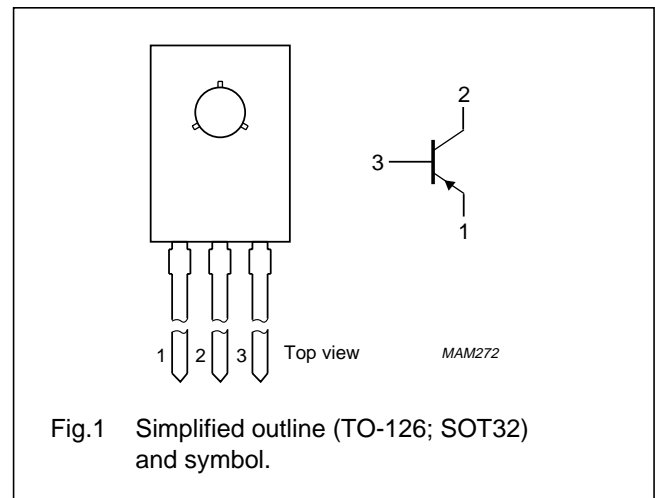
- General purpose power applications, e.g. driver stages in hi-fi amplifiers and television circuits.

### DESCRIPTION

PNP power transistor in a TO-126; SOT32 plastic package. NPN complements: BD135, BD137 and BD139.

### PINNING

PIN	DESCRIPTION
1	emitter
2	collector, connected to metal part of mounting surface
3	base



### LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134).

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V <sub>CBO</sub>	collector-base voltage	open emitter			
	BD136		–	–45	V
	BD138		–	–60	V
V <sub>CEO</sub>	collector-emitter voltage	open base			
	BD136		–	–45	V
	BD138		–	–60	V
	BD140		–	–80	V
V <sub>EBO</sub>	emitter-base voltage	open collector	–	–5	V
I <sub>C</sub>	collector current (DC)		–	–1.5	A
I <sub>CM</sub>	peak collector current		–	–2	A
I <sub>BM</sub>	peak base current		–	–1	A
P <sub>tot</sub>	total power dissipation	T <sub>mb</sub> ≤ 70 °C	–	8	W
T <sub>stg</sub>	storage temperature		–65	+150	°C
T <sub>j</sub>	junction temperature		–	150	°C
T <sub>amb</sub>	operating ambient temperature		–65	+150	°C

## PNP power transistors

## BD136; BD138; BD140

## THERMAL CHARACTERISTICS

SYMBOL	PARAMETER	CONDITIONS	VALUE	UNIT
$R_{th\ j-a}$	thermal resistance from junction to ambient	note 1	100	K/W
$R_{th\ j-mb}$	thermal resistance from junction to mounting base		10	K/W

## Note

1. Refer to TO-126 (SOT32) standard mounting conditions.

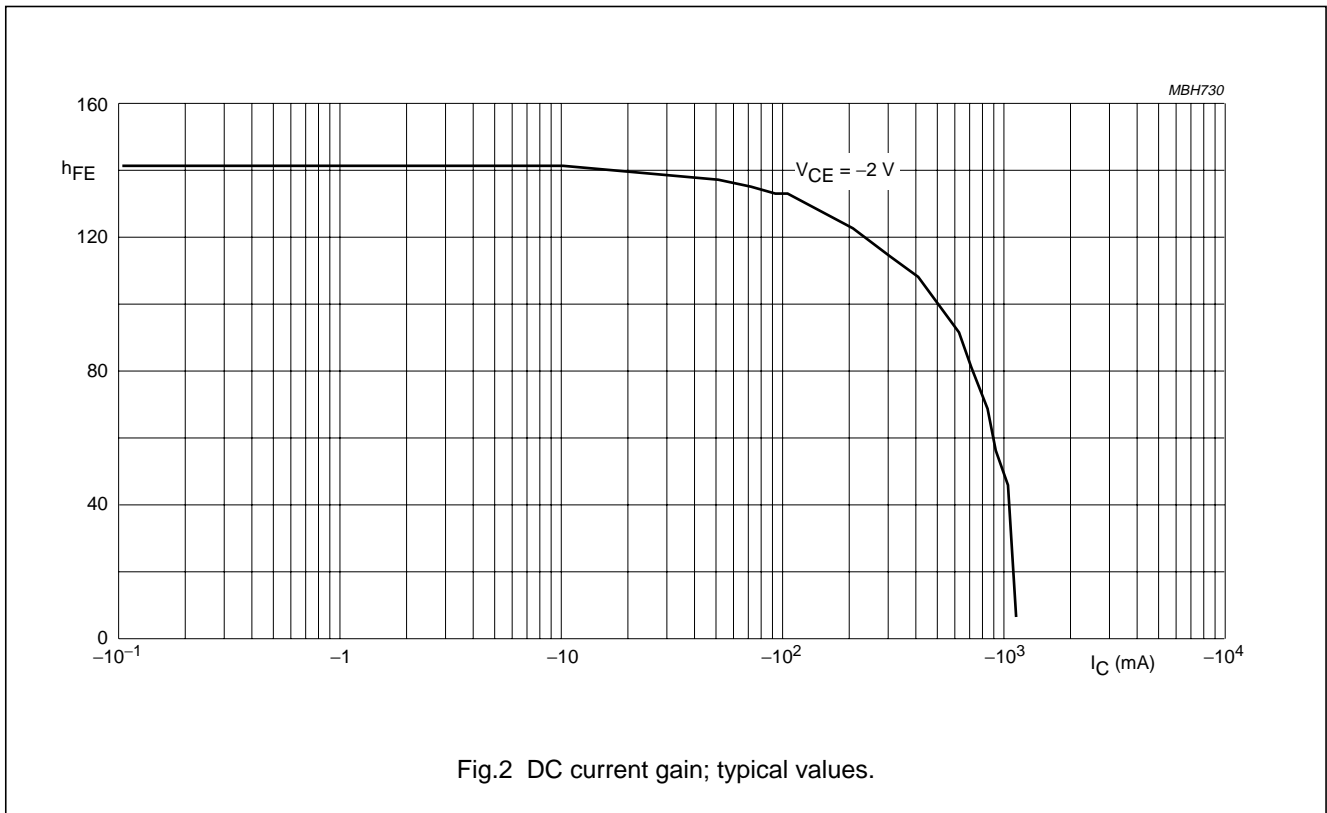
## CHARACTERISTICS

$T_j = 25\text{ °C}$  unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
$I_{CBO}$	collector cut-off current	$I_E = 0; V_{CB} = -30\text{ V}$	–	–	–100	nA
		$I_E = 0; V_{CB} = -30\text{ V}; T_j = 125\text{ °C}$	–	–	–10	$\mu\text{A}$
$I_{EBO}$	emitter cut-off current	$I_C = 0; V_{EB} = -5\text{ V}$	–	–	–100	nA
$h_{FE}$	DC current gain	$V_{CE} = -2\text{ V}$ ; (see Fig.2) $I_C = -5\text{ mA}$	40	–	–	
		$I_C = -150\text{ mA}$ $I_C = -500\text{ mA}$	63 25	–	250 –	
	DC current gain BD136-10; BD138-10; BD140-10 BD136-16; BD138-16; BD140-16	$I_C = -150\text{ mA}; V_{CE} = -2\text{ V}$ ; (see Fig.2)	63 100	–	160 250	
$V_{CEsat}$	collector-emitter saturation voltage	$I_C = -500\text{ mA}; I_B = -50\text{ mA}$	–	–	–0.5	V
$V_{BE}$	base-emitter voltage	$I_C = -500\text{ mA}; V_{CE} = -2\text{ V}$	–	–	–1	V
$f_T$	transition frequency	$I_C = -50\text{ mA}; V_{CE} = -5\text{ V}$ ; $f = 100\text{ MHz}$	–	160	–	MHz
$\frac{h_{FE1}}{h_{FE2}}$	DC current gain ratio of the complementary pairs	$ I_C  = 150\text{ mA};  V_{CE}  = 2\text{ V}$	–	1.3	1.6	

PNP power transistors

BD136; BD138; BD140

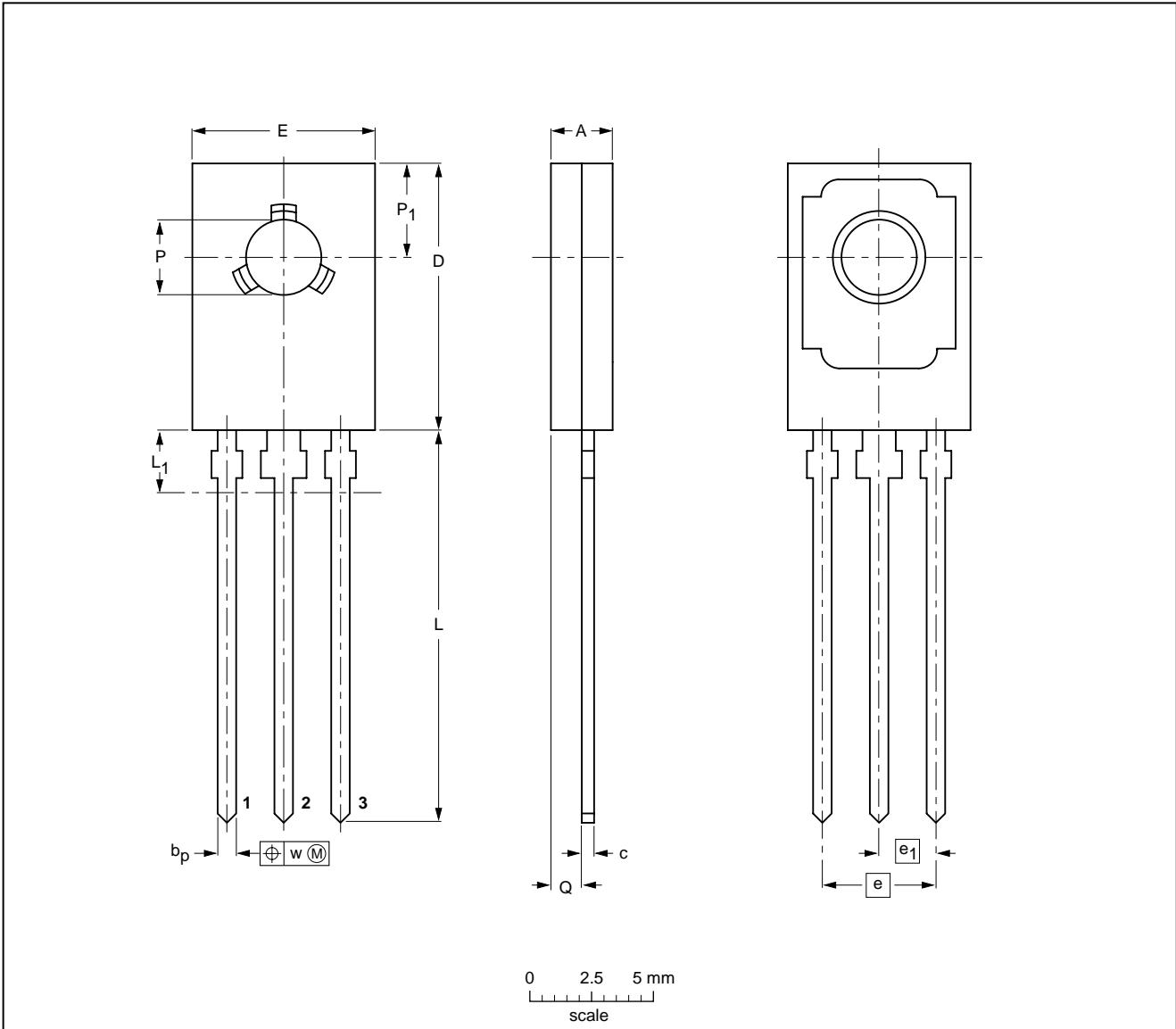


PNP power transistors

BD136; BD138; BD140

PACKAGE OUTLINE

Plastic single-ended leaded (through hole) package; mountable to heatsink, 1 mounting hole; 3 leads SOT32



DIMENSIONS (mm are the original dimensions)

UNIT	A	$b_p$	c	D	E	e	$e_1$	L	$L_1^{(1)}$ max	Q	P	$P_1$	w
mm	2.7 2.3	0.88 0.65	0.60 0.45	11.1 10.5	7.8 7.2	4.58	2.29	16.5 15.3	2.54	1.5 0.9	3.2 3.0	3.9 3.6	0.254

Note

1. Terminal dimensions within this zone are uncontrolled to allow for flow of plastic and terminal irregularities.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT32		TO-126				97-03-04



## PNP power transistors

## BD136; BD138; BD140

**DEFINITIONS**

<b>Data Sheet Status</b>	
Objective specification	This data sheet contains target or goal specifications for product development.
Preliminary specification	This data sheet contains preliminary data; supplementary data may be published later.
Product specification	This data sheet contains final product specifications.
<b>Limiting values</b>	
Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.	
<b>Application information</b>	
Where application information is given, it is advisory and does not form part of the specification.	

**LIFE SUPPORT APPLICATIONS**

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips for any damages resulting from such improper use or sale.

# Philips Semiconductors – a worldwide company

**Argentina:** see South America

**Australia:** 34 Waterloo Road, NORTH RYDE, NSW 2113,  
Tel. +61 2 9805 4455, Fax. +61 2 9805 4466

**Austria:** Computerstr. 6, A-1101 WIEN, P.O. Box 213,  
Tel. +43 1 60 101 1248, Fax. +43 1 60 101 1210

**Belarus:** Hotel Minsk Business Center, Bld. 3, r. 1211, Volodarski Str. 6,  
220050 MINSK, Tel. +375 172 20 0733, Fax. +375 172 20 0773

**Belgium:** see The Netherlands

**Brazil:** see South America

**Bulgaria:** Philips Bulgaria Ltd., Energoproject, 15th floor,  
51 James Bourchier Blvd., 1407 SOFIA,  
Tel. +359 2 68 9211, Fax. +359 2 68 9102

**Canada:** PHILIPS SEMICONDUCTORS/COMPONENTS,  
Tel. +1 800 234 7381, Fax. +1 800 943 0087

**China/Hong Kong:** 501 Hong Kong Industrial Technology Centre,  
72 Tat Chee Avenue, Kowloon Tong, HONG KONG,  
Tel. +852 2319 7888, Fax. +852 2319 7700

**Colombia:** see South America

**Czech Republic:** see Austria

**Denmark:** Sydhavnsgade 23, 1780 COPENHAGEN V,  
Tel. +45 33 29 3333, Fax. +45 33 29 3905

**Finland:** Sinikalliontie 3, FIN-02630 ESPOO,  
Tel. +358 9 615 800, Fax. +358 9 6158 0920

**France:** 51 Rue Carnot, BP317, 92156 SURESNES Cedex,  
Tel. +33 1 4099 6161, Fax. +33 1 4099 6427

**Germany:** Hammerbrookstraße 69, D-20097 HAMBURG,  
Tel. +49 40 2353 60, Fax. +49 40 2353 6300

**Hungary:** see Austria

**India:** Philips INDIA Ltd, Band Box Building, 2nd floor,  
254-D, Dr. Annie Besant Road, Worli, MUMBAI 400 025,  
Tel. +91 22 493 8541, Fax. +91 22 493 0966

**Indonesia:** PT Philips Development Corporation, Semiconductors Division,  
Gedung Philips, Jl. Buncit Raya Kav.99-100, JAKARTA 12510,  
Tel. +62 21 794 0040 ext. 2501, Fax. +62 21 794 0080

**Ireland:** Newstead, Clonskeagh, DUBLIN 14,  
Tel. +353 1 7640 000, Fax. +353 1 7640 200

**Israel:** RAPAC Electronics, 7 Kehilat Saloniki St, PO Box 18053,  
TEL AVIV 61180, Tel. +972 3 645 0444, Fax. +972 3 649 1007

**Italy:** PHILIPS SEMICONDUCTORS, Piazza IV Novembre 3,  
20124 MILANO, Tel. +39 2 6752 2531, Fax. +39 2 6752 2557

**Japan:** Philips Bldg 13-37, Kohnan 2-chome, Minato-ku,  
TOKYO 108-8507, Tel. +81 3 3740 5130, Fax. +81 3 3740 5077

**Korea:** Philips House, 260-199 Itaewon-dong, Yongsan-ku, SEOUL,  
Tel. +82 2 709 1412, Fax. +82 2 709 1415

**Malaysia:** No. 76 Jalan Universiti, 46200 PETALING JAYA, SELANGOR,  
Tel. +60 3 750 5214, Fax. +60 3 757 4880

**Mexico:** 5900 Gateway East, Suite 200, EL PASO, TEXAS 79905,  
Tel. +9-5 800 234 7381, Fax +9-5 800 943 0087

**Middle East:** see Italy

**Netherlands:** Postbus 90050, 5600 PB EINDHOVEN, Bldg. VB,  
Tel. +31 40 27 82785, Fax. +31 40 27 88399

**New Zealand:** 2 Wagener Place, C.P.O. Box 1041, AUCKLAND,  
Tel. +64 9 849 4160, Fax. +64 9 849 7811

**Norway:** Box 1, Manglerud 0612, OSLO,  
Tel. +47 22 74 8000, Fax. +47 22 74 8341

**Pakistan:** see Singapore

**Philippines:** Philips Semiconductors Philippines Inc.,  
106 Valero St. Salcedo Village, P.O. Box 2108 MCC, MAKATI,  
Metro MANILA, Tel. +63 2 816 6380, Fax. +63 2 817 3474

**Poland:** Ul. Lukiska 10, PL 04-123 WARSZAWA,  
Tel. +48 22 612 2831, Fax. +48 22 612 2327

**Portugal:** see Spain

**Romania:** see Italy

**Russia:** Philips Russia, Ul. Usatcheva 35A, 119048 MOSCOW,  
Tel. +7 095 755 6918, Fax. +7 095 755 6919

**Singapore:** Lorong 1, Toa Payoh, SINGAPORE 319762,  
Tel. +65 350 2538, Fax. +65 251 6500

**Slovakia:** see Austria

**Slovenia:** see Italy

**South Africa:** S.A. PHILIPS Pty Ltd., 195-215 Main Road Martindale,  
2092 JOHANNESBURG, P.O. Box 7430 Johannesburg 2000,  
Tel. +27 11 470 5911, Fax. +27 11 470 5494

**South America:** Al. Vicente Pinzon, 173, 6th floor,  
04547-130 SÃO PAULO, SP, Brazil,  
Tel. +55 11 821 2333, Fax. +55 11 821 2382

**Spain:** Balmes 22, 08007 BARCELONA,  
Tel. +34 93 301 6312, Fax. +34 93 301 4107

**Sweden:** Kottbygatan 7, Akalla, S-16485 STOCKHOLM,  
Tel. +46 8 5985 2000, Fax. +46 8 5985 2745

**Switzerland:** Allmendstrasse 140, CH-8027 ZÜRICH,  
Tel. +41 1 488 2741 Fax. +41 1 488 3263

**Taiwan:** Philips Semiconductors, 6F, No. 96, Chien Kuo N. Rd., Sec. 1,  
TAIPEI, Taiwan Tel. +886 2 2134 2886, Fax. +886 2 2134 2874

**Thailand:** PHILIPS ELECTRONICS (THAILAND) Ltd.,  
209/2 Sanpavuth-Bangna Road Prakanong, BANGKOK 10260,  
Tel. +66 2 745 4090, Fax. +66 2 398 0793

**Turkey:** Talatpasa Cad. No. 5, 80640 GÜLTEPE/ISTANBUL,  
Tel. +90 212 279 2770, Fax. +90 212 282 6707

**Ukraine:** PHILIPS UKRAINE, 4 Patrice Lumumba str., Building B, Floor 7,  
252042 KIEV, Tel. +380 44 264 2776, Fax. +380 44 268 0461

**United Kingdom:** Philips Semiconductors Ltd., 276 Bath Road, Hayes,  
MIDDLESEX UB3 5BX, Tel. +44 181 730 5000, Fax. +44 181 754 8421

**United States:** 811 East Arques Avenue, SUNNYVALE, CA 94088-3409,  
Tel. +1 800 234 7381, Fax. +1 800 943 0087

**Uruguay:** see South America

**Vietnam:** see Singapore

**Yugoslavia:** PHILIPS, Trg N. Pasica 5/v, 11000 BEOGRAD,  
Tel. +381 11 62 5344, Fax. +381 11 63 5777

**For all other countries apply to:** Philips Semiconductors,  
International Marketing & Sales Communications, Building BE-p, P.O. Box 218,  
5600 MD EINDHOVEN, The Netherlands, Fax. +31 40 27 24825

**Internet:** <http://www.semiconductors.philips.com>

© Philips Electronics N.V. 1999

SCA63

All rights are reserved. Reproduction in whole or in part is prohibited without the prior written consent of the copyright owner.

The information presented in this document does not form part of any quotation or contract, is believed to be accurate and reliable and may be changed without notice. No liability will be accepted by the publisher for any consequence of its use. Publication thereof does not convey nor imply any license under patent- or other industrial or intellectual property rights.

Printed in The Netherlands

115002/00/03/pp8

Date of release: 1999 Apr 12

Document order number: 9397 750 05575

## Photo Modules for PCM Remote Control Systems

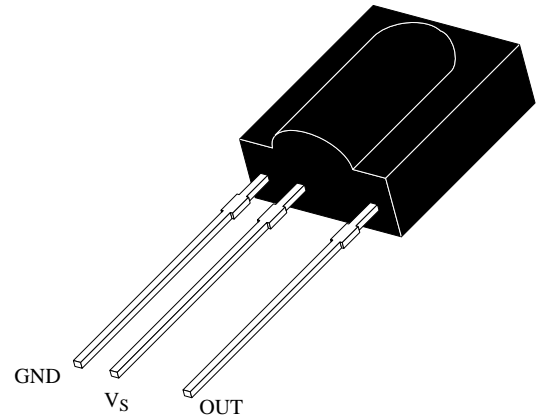
### Available types for different carrier frequencies

Type	fo	Type	fo
TSOP1730	30 kHz	TSOP1733	33 kHz
TSOP1736	36 kHz	TSOP1737	36.7 kHz
TSOP1738	38 kHz	TSOP1740	40 kHz
TSOP1756	56 kHz		

### Description

The TSOP17.. – series are miniaturized receivers for infrared remote control systems. PIN diode and preamplifier are assembled on lead frame, the epoxy package is designed as IR filter.

The demodulated output signal can directly be decoded by a microprocessor. TSOP17.. is the standard IR remote control receiver series, supporting all major transmission codes.

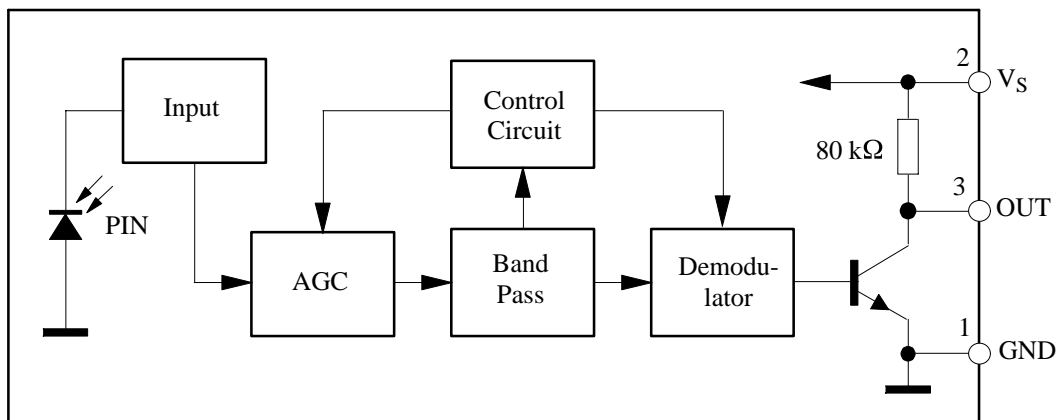


94 8691

### Features

- Photo detector and preamplifier in one package
- Internal filter for PCM frequency
- Improved shielding against electrical field disturbance
- TTL and CMOS compatibility
- Output active low
- Low power consumption
- High immunity against ambient light
- Continuous data transmission possible (up to 2400 bps)
- Suitable burst length . 10 cycles/burst

### Block Diagram



94 8136

### Absolute Maximum Ratings

$T_{amb} = 25^{\circ}\text{C}$

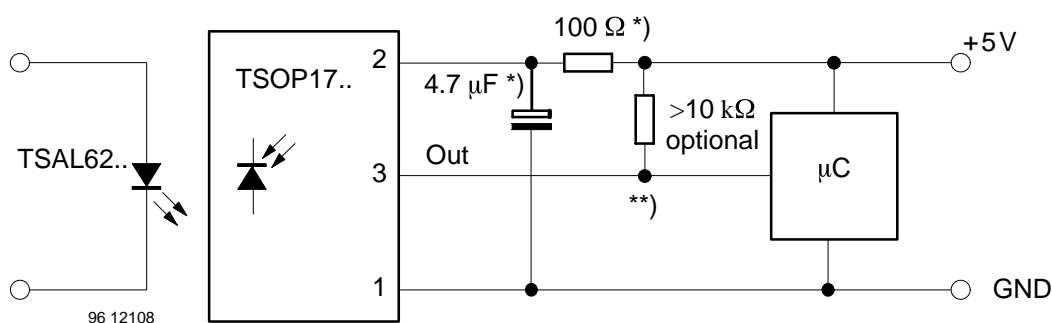
Parameter	Test Conditions	Symbol	Value	Unit
Supply Voltage	(Pin 2)	$V_S$	-0.3...6.0	V
Supply Current	(Pin 2)	$I_S$	5	mA
Output Voltage	(Pin 3)	$V_O$	-0.3...6.0	V
Output Current	(Pin 3)	$I_O$	5	mA
Junction Temperature		$T_j$	100	$^{\circ}\text{C}$
Storage Temperature Range		$T_{stg}$	-25...+85	$^{\circ}\text{C}$
Operating Temperature Range		$T_{amb}$	-25...+85	$^{\circ}\text{C}$
Power Consumption	( $T_{amb} \leq 85^{\circ}\text{C}$ )	$P_{tot}$	50	mW
Soldering Temperature	$t \leq 10\text{ s}$ , 1 mm from case	$T_{sd}$	260	$^{\circ}\text{C}$

### Basic Characteristics

$T_{amb} = 25^{\circ}\text{C}$

Parameter	Test Conditions	Symbol	Min	Typ	Max	Unit
Supply Current (Pin 2)	$V_S = 5\text{ V}$ , $E_V = 0$	$I_{SD}$	0.4	0.6	1.5	mA
	$V_S = 5\text{ V}$ , $E_V = 40\text{ klx}$ , sunlight	$I_{SH}$		1.0		mA
Supply Voltage (Pin 2)		$V_S$	4.5		5.5	V
Transmission Distance	$E_V = 0$ , test signal see fig.7, IR diode TSAL6200, $I_F = 400\text{ mA}$	$d$		35		m
Output Voltage Low (Pin 3)	$I_{OSL} = 0.5\text{ mA}$ , $E_e = 0.7\text{ mW/m}^2$ , $f = f_o$ , $t_p/T = 0.4$	$V_{OSL}$			250	mV
Irradiance (30 – 40 kHz)	Pulse width tolerance: $t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$ , test signal (see fig.7)	$E_{e\ min}$		0.35	0.5	$\text{mW/m}^2$
Irradiance (56 kHz)	Pulse width tolerance: $t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$ , test signal (see fig.7)	$E_{e\ min}$		0.4	0.6	$\text{mW/m}^2$
Irradiance	$t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$	$E_{e\ max}$	30			$\text{W/m}^2$
Directivity	Angle of half transmission distance	$\phi_{1/2}$		$\pm 45$		deg

### Application Circuit



\*) recommended to suppress power supply disturbances

\*\*\*) The output voltage should not be hold continuously at a voltage below 3.3V by the external circuit.

## Suitable Data Format

The circuit of the TSOP17.. is designed in that way that unexpected output pulses due to noise or disturbance signals are avoided. A bandpassfilter, an integrator stage and an automatic gain control are used to suppress such disturbances.

The distinguishing mark between data signal and disturbance signal are carrier frequency, burst length and duty cycle.

The data signal should fulfill the following condition:

- Carrier frequency should be close to center frequency of the bandpass (e.g. 38kHz).
- Burst length should be 10 cycles/burst or longer.
- After each burst which is between 10 cycles and 70 cycles a gap time of at least 14 cycles is necessary.
- For each burst which is longer than 1.8ms a corresponding gap time is necessary at some time in the data stream. This gap time should have at least same length as the burst.
- Up to 1400 short bursts per second can be received continuously.

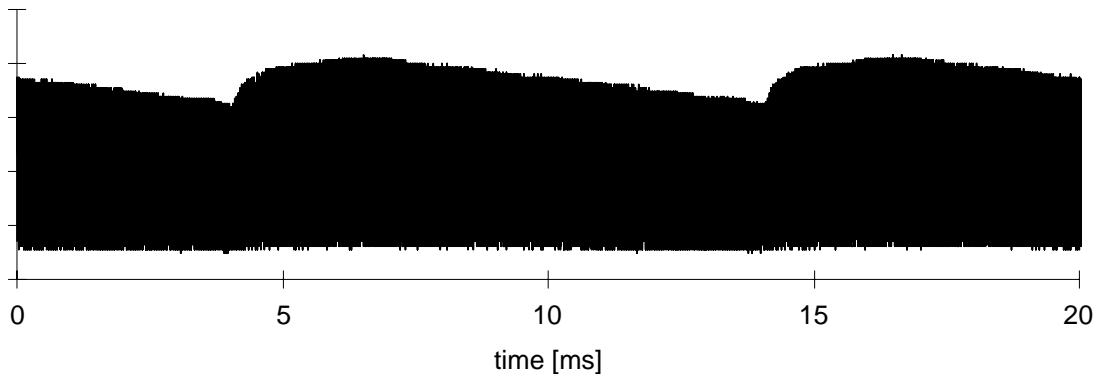
Some examples for suitable data format are:

NEC Code, Toshiba Micom Format, Sharp Code, RC5 Code, RC6 Code, R-2000 Code, Sony Format (SIRCS).

When a disturbance signal is applied to the TSOP17.. it can still receive the data signal. However the sensitivity is reduced to that level that no unexpected pulses will occur.

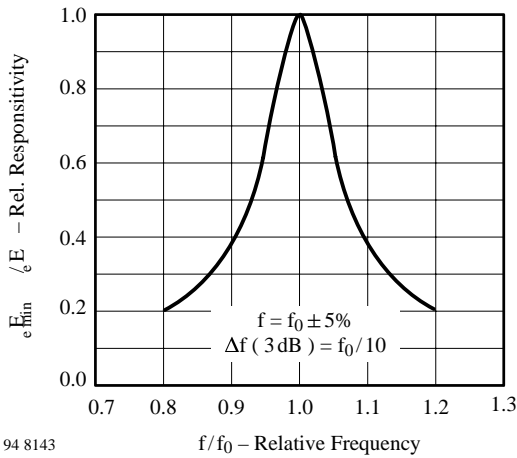
Some examples for such disturbance signals which are suppressed by the TSOP17.. are:

- DC light (e.g. from tungsten bulb or sunlight)
- Continuous signal at 38kHz or at any other frequency
- Signals from fluorescent lamps with electronic ballast (an example of the signal modulation is in the figure below).



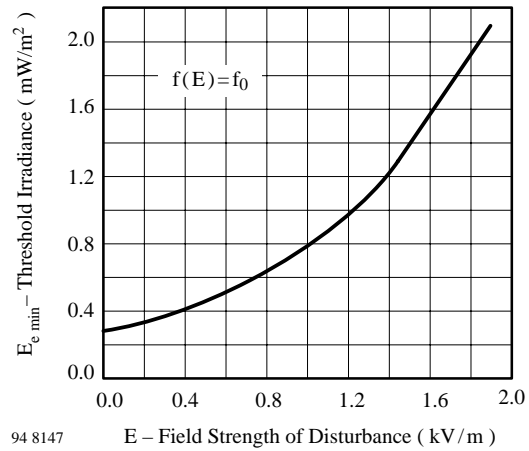
IR Signal from Fluorescent Lamp with low Modulation

Typical Characteristics ( $T_{amb} = 25^{\circ}\text{C}$  unless otherwise specified)



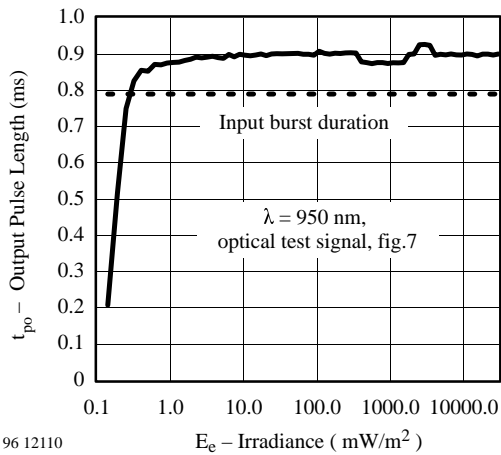
94 8143

Figure 1. Frequency Dependence of Responsivity



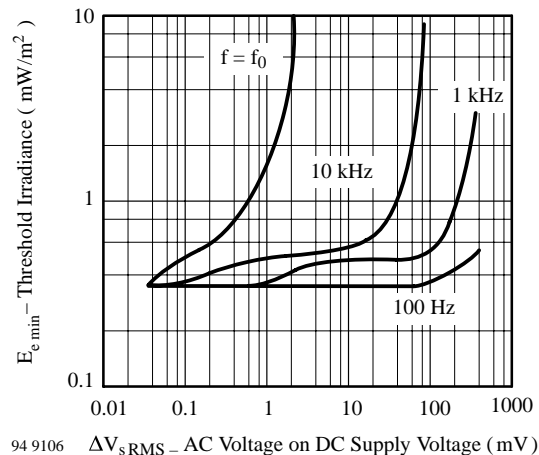
94 8147

Figure 4. Sensitivity vs. Electric Field Disturbances



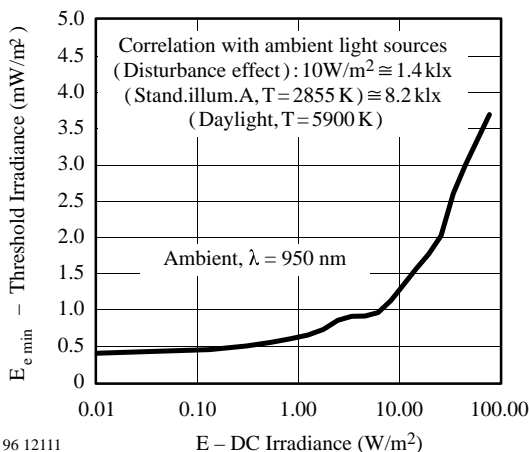
96 12110

Figure 2. Sensitivity in Dark Ambient



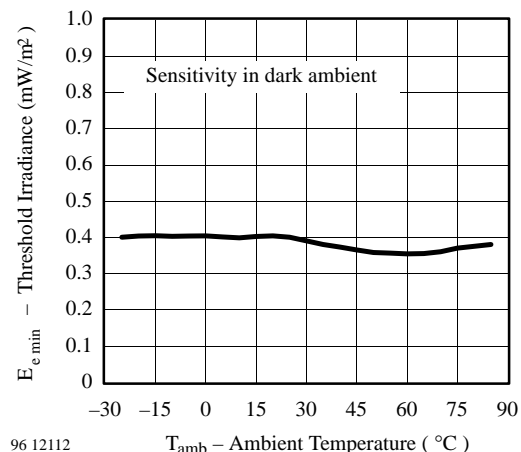
94 9106

Figure 5. Sensitivity vs. Supply Voltage Disturbances



96 12111

Figure 3. Sensitivity in Bright Ambient



96 12112

Figure 6. Sensitivity vs. Ambient Temperature

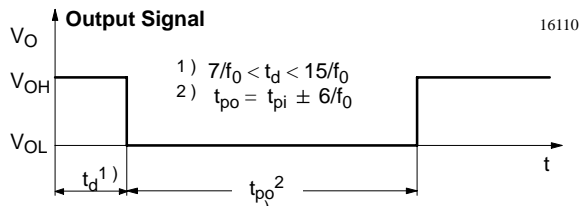
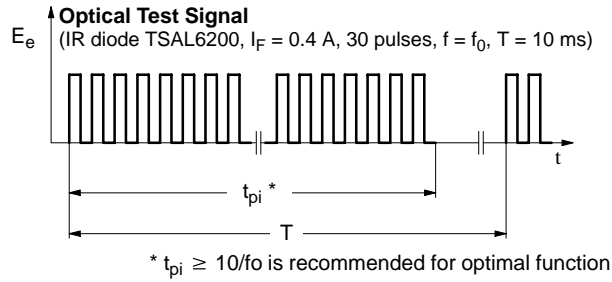


Figure 7. Output Function

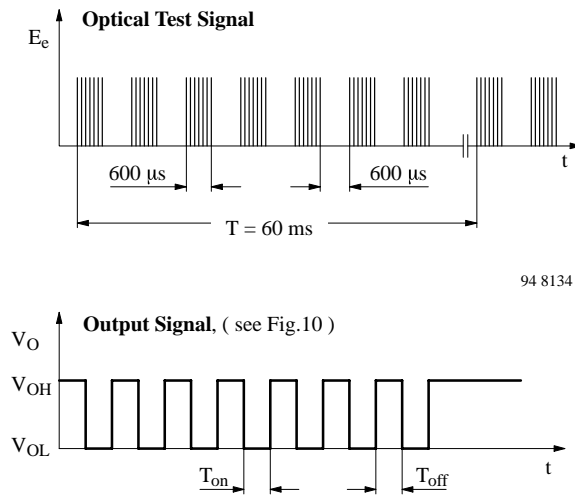
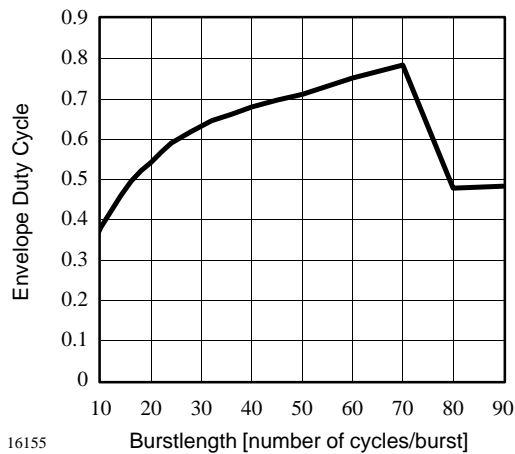
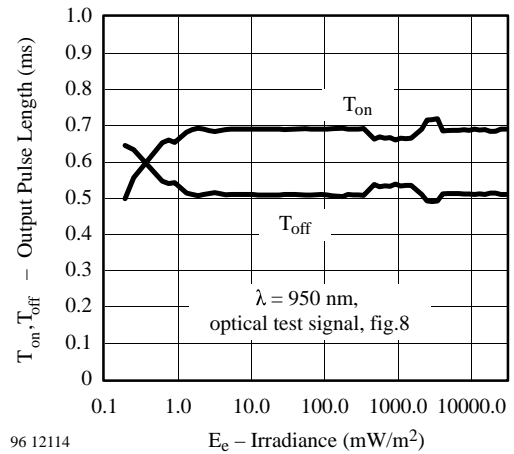


Figure 8. Output Function



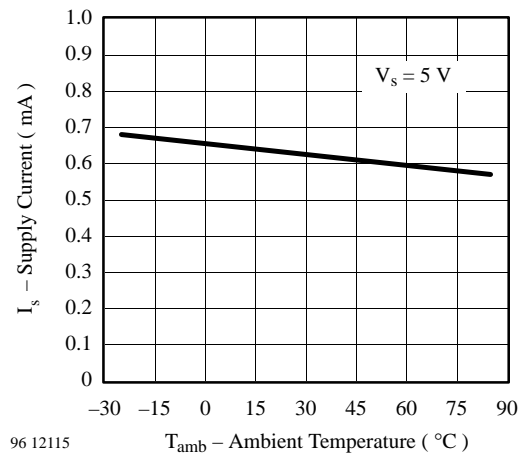
16155

Figure 9. Max. Envelope Duty Cycle vs. Burstlength



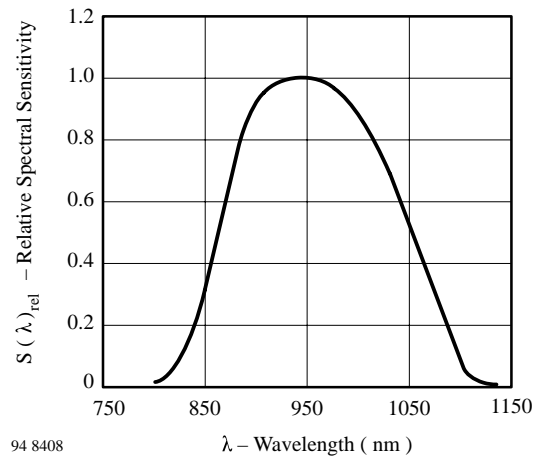
96 12114

Figure 10. Output Pulse Diagram



96 12115

Figure 11. Supply Current vs. Ambient Temperature



94 8408

Figure 12. Relative Spectral Sensitivity vs. Wavelength

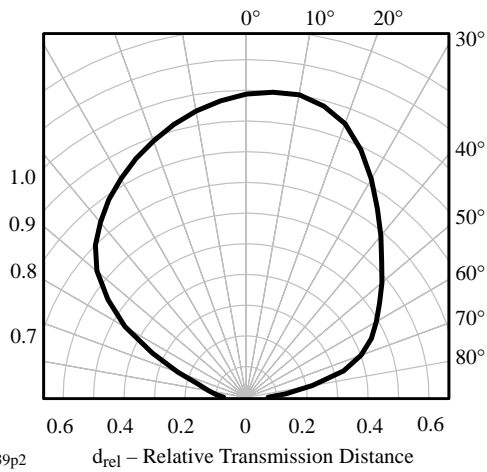


Figure 13. Vertical Directivity  $\phi_y$

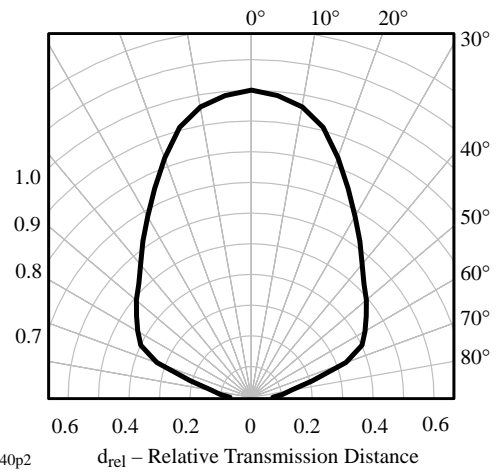
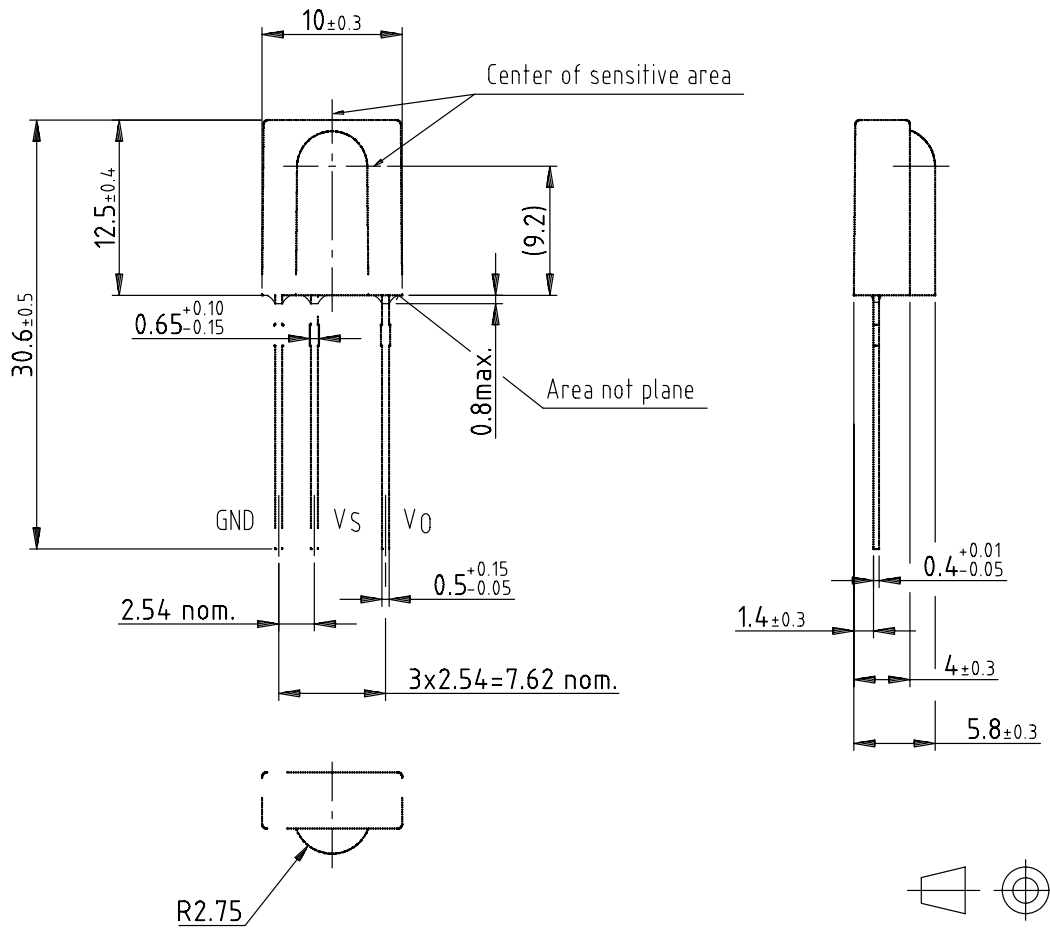


Figure 14. Horizontal Directivity  $\phi_x$

Dimensions in mm



96 12116

technical drawings according to DIN specifications