

## Chapter 1

# INTRODUCTION

### 1.1 Introduction

**Traffic Sign Recognition (TSR)** is a technology by which a vehicle is able to recognise the traffic signs put on the road e.g. "Speed Limit" or "No Parking" or "Turn Ahead" etc. This is part of the features collectively called **ADAS (Advanced Driver-Assistance Systems)**. ADAS, are systems to help the driver in the driving process. When designed with a safe human-machine interface, they should increase car safety and more generally road safety. Advanced driver-assistance systems (ADAS) are systems developed to automate/adapt/enhance vehicle systems for safety and better driving. Safety features are designed to avoid collisions and accidents by offering technologies that alert the driver to potential problems, or to avoid collisions by implementing safeguards and taking over control of the vehicle. Adaptive features may automate lighting, provide adaptive cruise control, automate braking, incorporate GPS/traffic warnings, connect to smartphones, alert driver to other cars or dangers, keep the driver in the correct lane, or show what is in blind spots. The technology is being developed by many automotive suppliers, including Continental and Delphi.

These first TSR systems which recognize speed limits were developed in cooperation by Mobileye and Continental AG. They first appeared in late 2008 on the redesigned BMW 7-Series, and the following year on the Mercedes-Benz S-Class. Currently these systems only detect the round speed limit signs found all across Europe (e.g.).

Second generation systems can also detect overtaking restrictions. It was introduced in 2008 in the Opel Insignia later followed by the Opel Astra and the Saab 9-5. This technology is also available on the 2011 Volkswagen Phaeton and since 2012 in Volvo S80, V70, XC70, XC60, S60, V60 and V40 (model year 2013-), as a technology called Road Sign Information. They are not able to recognize city limit signs, which in most European countries have meaning for speed limits, as they are too equal to direction signs.

**Intelligent Speed Adaptation (ISA)**, also known as Alerting and Intelligent Authority is any system that ensures that vehicle speed does not exceed a safe or legally enforced speed. In case of potential speeding, a human driver can be alerted, or the speed reduced automatically.

Intelligent speed adaptation uses information about the road to determine the required speed. Information can be obtained from knowledge of the vehicle position, taking into account speed limits known for the position, and by interpreting road features such as signs. ISA systems are designed to detect and alert a driver when a vehicle has entered a new speed zone, or when different speed limits are in force according to time of day and conditions. Many ISA systems also provide information about driving hazards (e.g., high pedestrian movement areas, railway crossings, schools, hospitals, etc.) and limits enforced by speed and traffic light cameras. The purpose of ISA is to assist the driver to maintain a safe and lawful speed at all times.

Research has found that, in urban areas, the number of crashes causing casualties is doubled for each 5 km/h over the limit. It is estimated that about 10% of casualties could be prevented if drivers who routinely travel at up to 10 km/h would obey the speed limits. About 20% of casualties could be prevented if all vehicles complied with the speed limits. Fatalities would be reduced even more. Speeding relatively slightly over the limit makes up a large proportion of preventable road trauma. Enforcing speed limits strictly enough to eliminate slight over speed is difficult; ISA helps with this.

**Radio beacons** Roadside radio beacons, or bollards, work by transmitting data to a receiver in the car. The beacons constantly transmit data that the car-mounted receiver picks up as it passes each beacon. This data could include local speed limits, school zones, variable speed limits, or traffic warnings. If sufficient numbers of beacons were used and were placed at regular intervals, they could calculate vehicle speed based on how many beacons the vehicle passed per second. Beacons could be placed in/on speed signs, telegraph poles, other roadside fixtures, or in the road itself. Mobile beacons could be deployed in order to override fixed beacons for use around accident scenes, during poor weather, or during special events. Beacons could be linked to a main computer so that quick changes could be made.

The use of radio beacons is common when ISA systems are used to control vehicle speeds in off road situations, such as factory sites, logistics and storage centres, etc., where

occupational health and safety requirements mean that very low vehicle speeds are required in the vicinity of workers and in situations of limited or obscured visibility.

**Overtaking assistance:** The large number of accidents caused by unsuccessful motor vehicle overtaking manoeuvres on roads is a significant problem with much public debate and concerns. In addition, one accident might cause another follow on accident when there is no signalling/reporting of it occurring, thus causing obstructions and further accidents in the vicinity. This can seriously affect the traffic flows, generating traffic jams and sometimes resulting in more unwarranted accidents, which can be avoided with the application of technology.

The emergence of wireless ad hoc and sensor technologies and their adoption in vehicular networks can provide smart solutions to mitigate the probability of accidents and avoidance of dangerous traffic flow situations. This project assists the driver in overtaking manoeuvres. In order to do that, the application must indicate whether it is safe or not to perform the overtaking manoeuvre. To propose this application, concepts of kinematics techniques are used to model the overtaking manoeuvres and the successful simulation results show this technique to be promising.

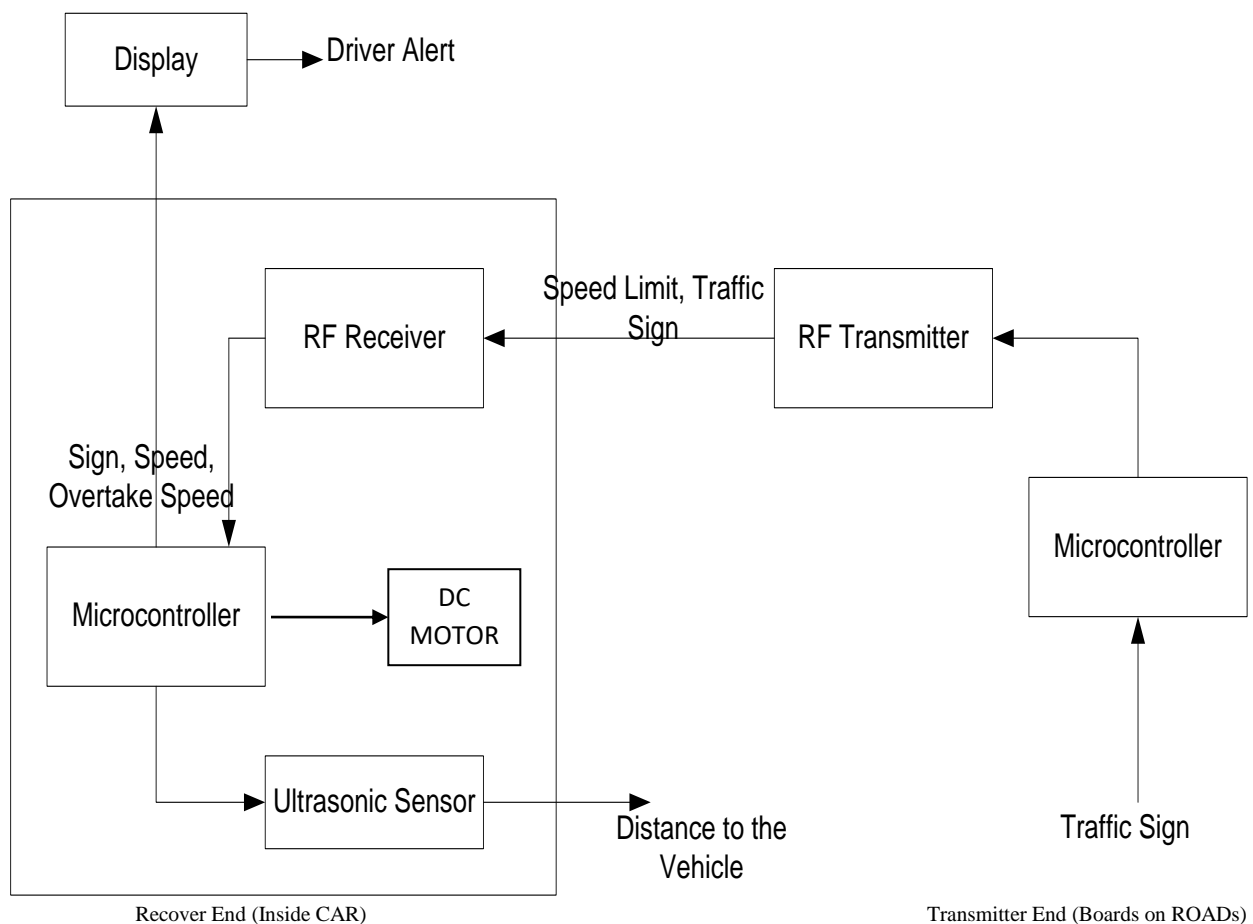
## 1.2 Problem Statement

- 1) Drivers miss out Traffic sign reading due to disturbances in road or due to non-attentiveness while driving due to absence of mind.
- 2) In the monsoon and rainy season the sign might not be seen due to fog.
- 3) Most accidents occurred near school or restricted sites are due to over speeding above the speed limit.
- 4) Present technology limits a maximum speed in every zone but sometimes this might not be useful in various zone, particularly on highways.
- 5) Overtaking without judging about front running vehicles safe distance causes collision accidents.
- 6) The immature drivers don't have the clear idea of bumper clearance, speed and dimensions; and hence would lead to collision.

### 1.3 Existing System

- 1) Image Processing based Vehicle on board Traffic sign alert systems are available, where camera in front of car captures the frames and using image processing analysis to identify traffic signs and provide visual , audio alerts to drivers.
- 2) For speed restriction, Speed Limit systems are used, which limit the speed of vehicle with a threshold speed like 60km/hr. But the speed limit is fixed in this system. We cannot adapt to conditions like 20 Km near to School.
- 3) For overtaking, as of now, it is based only on human judgment.

### 1.4 Proposed System



**Figure 1.1. Proposed System Block Diagram**

- 1) The Traffic signs are sent as Radio Signal and vehicle can receive this radio signal and display visual / audio alerts to Driver.

- 2) The speed limit on that road is sent as radio signal and the vehicle can receive this radio signal and display visual alert.
- 3) And a resultant action is computed based upon the current speed and speed limit. The action is informed to the driver through display and control unit limits the speed to the speed limit value.
- 4) Ultra-sonic sensor fit in front of vehicle and they measure the distance to front running vehicles and based on this distance, the system calculated the desired parameters and provide the driver with the overtaking sight distance, acceleration to apply and the safe speed for overtaking.

## **1.5 Brief outline of the project**

The works carried out at each project phase are outlined below:-

### **A. Learning & Analysis Phase**

This phase includes:

- Gathering knowledge about existing communicating techniques.
- Well understanding of the project design review from the client.
- Learning tools, technologies & programming Language for coding purpose.

### **B. Design & Implementation**

This phase Includes:

- Designing the overall functional view i.e. system architecture of the project.
- Describing the language, platform used in the project implementation.
- Identification and design of the modules for implementing.

### **C. Testing Phase**

This phase includes:

- Writing the test cases for testing the implemented modules.
- Executing the test cases manually, comparing and evaluating the actual result with the expected result.

## Chapter 2

# LITERATURE SURVEY

Literature survey is mainly carried out in order to analyse the background of the current project which helps to find out flaws in the existing system & guides on which unsolved problems we can work out. So, the following topics not only illustrate the background of the project but also uncover the problems and flaws which motivated to propose solutions and work on this project. A variety of research has been done on power aware scheduling. Following section explores different references that discuss about several topics related to power aware scheduling.

## 2.1 Literature Survey

**Y. GU, T. Yendo, M. Tehrani, T. Fujii and M.Tanimoto. 2011. “Traffic sign detection in dual-focal active camera system”, International Conference on Intelligent Vehicle Symposium, IEEE.**

Traffic sign recognition systems can be applied to assist drivers and improve the traffic safety. High resolution image of traffic sign can improve the recognition result, especially for some complex traffic signs. In this paper, a dual-focal active camera system is proposed to obtain a high resolution image of traffic sign. In the system an active telephoto camera is equipped as an assistant of a wide angle camera. To make the proposed system correctly capture a high resolution image of traffic sign, the shape, colour features and the relationship between continuous frames are used together in the traffic sign detection. The experiment results demonstrate that the proposed method makes the dual-focal active camera system effectively work when the system is installed on an automobile and moves on the road.

**Road sign recognition system based on GentleBoost with sharing features,Jin-Yi-Wu; Chien-Chung Tseng; Chun-Hao Chang; Jenn-Jier James Lien; Ju Chin Chen; Ching Ting Tu, Proceedings 2011 International Conference on System Science and Engineering.**

Nowadays, the number of vehicles is growing rapidly, and more and more intelligent transportation systems are developed for assisting drivers. Road sign detection and recognition is extremely important for safe and careful driving, this system can not only inform the driver about the condition of the roadway but also support the driver during the tedious task of remembering the large number of road signs. In this work, we propose a fast road sign detection and recognition system. This system takes advantage of the HSI color space to filter most of the false alarms. Distance to borders (DtBs) and Support Vector Machine (SVM) are then used for the shape detection. Finally, the candidate blobs that pass through the shape detection is recognized by a GentleBoost with sharing features detector and rotation, scale, translation-invariant (RST-invariant) template matching. For recognition step, color information is used for training GentleBoost detector to ensure the accuracy of the system; and the achromatic parts of the candidates are matched to the templates by RST-invariant template matching. The main advantage of this system is that it can detect and recognize road signs efficiently

#### **Robust traffic signs detection by means of vision and V2I communications**

**M. A. García-Garrido; M. Ocaña; D. F. Llorca; M. A. Sotelo; E. Arroyo; A. Llamazares**

**2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC).**

This paper presents a complete traffic sign recognition system, including the steps of detection, recognition and tracking. The Hough transform is used as detection method from the information extracted in contour images, while the proposed recognition system is based on Support Vector Machines (SVM), and is able to recognize up to one hundred of the main road signs. Besides a novel solution to the problem of discarding detected signs that do not pertain to the host road is proposed, for that purpose vehicle-to-infrastructure (V2I) communication and stereo information is used. This paper presents plenty of tests in real driving conditions, both day and night, in which a high success rate and low number of false negatives and true positives were obtained, and an average runtime of 35 ms, allowing real-time performance.

**Design of traffic sign detection, recognition, and transmission systems for smart vehicles, Abdelhamid Mammeri; Azzedine Boukerche; Mohammed Almulla, IEEE Wireless Communications, Year: 2013.**

Traffic sign detection and recognition (TSDR) is an essential component of advanced driver assistance systems (ADAS). It is mainly designed to enhance driver safety through the fast acquisition and interpretation of traffic signs. However, such systems still suffer from the inability to accurately recognize signs. Moreover, the sharing of wirelessly recognized signs among vehicles is not yet supported by current systems. In some safety scenarios, vehicle-to-vehicle communication of traffic sign information is required. In this article, we first address challenges and undesirable factors facing TSDR systems. After that, we show how to design a TSDR system by addressing some useful techniques used in each stage of the system. For each stage, these techniques are regrouped into different categories. Then, for each category, a short description is given followed by some concluding remarks. Finally, the transmission of the recognized signs is briefly investigated.

**Yoshimichi, S., and M. Koji. "Development and evaluation of in-vehicle signing system utilizing RFID tags as digital traffic signs." *International Journal of ITS Research* 2006..**

Traffic signs visually provide drivers with regulatory, warning and guide information. Vehicle drivers are requested to collect dynamic visual information on such matters as other vehicles and traffic signals, and static visual information including traffic signs, and to manoeuvre the vehicle accordingly. Actually, however, traffic signs and other static visual information are more likely to be overlooked than dynamic visual information during manoeuvre. As a solution to the problem, in-vehicle signing systems that are capable of displaying signing on a terminal in the vehicle are expected to provide an effective support. The system has been designated as one of the ITS market packages in the National ITS Architecture of the United States. In Japan, however, the serviceability of the system has yet to win sufficient social recognition although several experiments have been conducted using image processing or DSRC (Dedicated Short Range Communications). Under the circumstances, an attempt is made in this study to apply RFID (Radio Frequency Identification) as digital traffic signing replacing existing signs. Efforts are now being made to seek applications for RFID tags as the next generation of tag systems. Wider use of RFID tags is now being accelerated. As applications of RFID to digital traffic signing, some systems have been developed for pedestrians. For vehicle users, however, no reports have yet been made on specific studies for practical implementation of the tag system although the possibility of using the system has been suggested. In this study, an in-vehicle



signing system is built and assessed that uses general-purpose RFID tags as digital traffic signs and communications between the road surface and vehicle equipment. Then, the serviceability of the system is identified.

**Pérez, Joshué, "An RFID-based intelligent vehicle speed controller using active traffic signals."**

These days, mass-produced vehicles benefit from research on Intelligent Transportation System (ITS). One prime example of ITS is vehicle Cruise Control (CC), which allows it to maintain a pre-defined reference speed, to economize on fuel or energy consumption, to avoid speeding fines, or to focus all of the driver's attention on the steering of the vehicle. However, achieving efficient Cruise Control is not easy in roads or urban streets where sudden changes of the speed limit can happen, due to the presence of unexpected obstacles or maintenance work, causing, in inattentive drivers, traffic accidents. In this communication we present a new Infrastructure to Vehicles (I2V) communication and control system for intelligent speed control, which is based upon Radio Frequency Identification (RFID) technology for identification of traffic signals on the road, and high accuracy vehicle speed measurement with a Hall effect-based sensor. A fuzzy logic controller, based on sensor fusion of the information provided by the I2V infrastructure, allows the efficient adaptation of the speed of the vehicle to the circumstances of the road. The performance of the system is checked empirically, with promising results.

**An Efficient RealTime Traffic Sign Recognition System for Intelligent Vehicles with Smart phones;Ching-Hao Lai; Chia-Chen Yu, 2010 International Conference on Technologies and Applications of Artificial Intelligence, Year: 2010.**

In recent years, intelligent vehicles and smart phones have become more and more popular. The traffic sign recognition system is one kind of driving assistance system (DAS) which is used to automatically inform the driver the traffic sign information by a head up display (HUD), monitor, or speaker device. Hence this system is helpful to reduce driver distraction to increase driver safety. In this paper, an efficient real-time traffic sign recognition scheme is proposed for intelligent vehicles. The proposed scheme can integrate in-vehicle computing devices and smart phones to construe an in-vehicle traffic sign recognition system. This scheme contains four major

stages: video frame capturing and transmitting, image pre-process, traffic sign detection, and character/icon extraction and recognition. The smart phone first captures videos and then extracts video frames in certain frame-rate. These extracted frames can be transmitted to an in-vehicle computing device by a wireless network (Bluetooth, WiMAX, Wi-Fi etc.) The pre-process employs some image processing to improve and transform the video frames to keep a stable quality for following detection and recognition schemes. At following stages, this paper presents some efficient and accurate traffic sign detection and recognition schemes which contain color selection, shape recognition, character/icon extraction and recognition. The experimental results shows that the proposed scheme only averagely spends 0.085 seconds for a video frame and its average accuracy can achieve about 98%. Hence, the proposed scheme can keep lower computing complexity, however it still can obtain a well accuracy.

**Automatic Detection and Classification of Traffic Signs Carlos Filipe Paulo; Paulo Lobato Correia, Image Analysis for Multimedia Interactive Services, 2007. WIAMIS '07. Eighth International Workshop on, Year: 2007.**

This paper proposes algorithms for the automatic detection of traffic signs from photo or video images and their classification to provide a driver alert system. Several examples taken from Portuguese roads are used to demonstrate the effectiveness of the proposed system. Traffic signs are detected by analysing color information, notably red and blue, contained on the images. The detected signs are then classified according to their shape characteristics, as triangular, squared and circular shapes. Combining color and shape information, traffic signs are classified into one of the following classes: danger, information, obligation or prohibition. Both the detection and classification algorithms include innovative components to improve the overall system performance.

## Summary

This chapter mainly discusses about the papers, websites that are referred while making this dissertation report. All these papers and websites provide information related to learning of collective behaviour, their existing solutions, methods used and also their advantages & limitations.

## Chapter 3

### PROBLEM STATEMENT

Road traffic assumes a major importance in modern society organization. To ensure that motorized vehicle circulation flows in a harmonious and safe way, specific rules are established by every government. Some of these rules are displayed to drivers by means of traffic signs that need to be interpreted while driving. This may look as a simple task, but sometimes the driver misses signs, which may be problematic, eventually leading to car accidents, offenses and other incidents. Modern cars already include many safety systems, but even with two cars moving at 40 km/h, their collision consequences can be dramatic. Although some drivers intentionally break the law not respecting traffic signs, an automatic system able to detect these signs can be a useful help to most drivers. One might consider a system taking advantage of the Global Positioning System (GPS). It could be almost flawless if an updated traffic sign location database would be available. Unfortunately, few cars have GPS installed and traffic sign localization databases are not available for download. Installing a low price “traffic sign information” receiver on a car could also be a good idea if traffic signs were able to transmit their information to cars.

Based on the information of the speed limit, the drivers must be informed about the restrictions and actions. A vehicle is considered to be overtaking from the moment it first appears in the side view from the ego-vehicle until more than 10% of the rear of the overtaking vehicle is seen from the ego-vehicle. This completely depends on the experience and judgment of the drivers. In case of any improper judgement or carelessness, then driver gets involved in overtaking manoeuvres. In order to assist the driver, the application must developed to indicate whether it is safe or not to perform the overtaking manoeuvre. To propose this application, concepts of kinematics techniques are used to model the overtaking manoeuvres and the successful simulation results show this technique to be promising.

## Chapter 4

# PROPOSED SYSTEM

### 4.1 Block Diagram

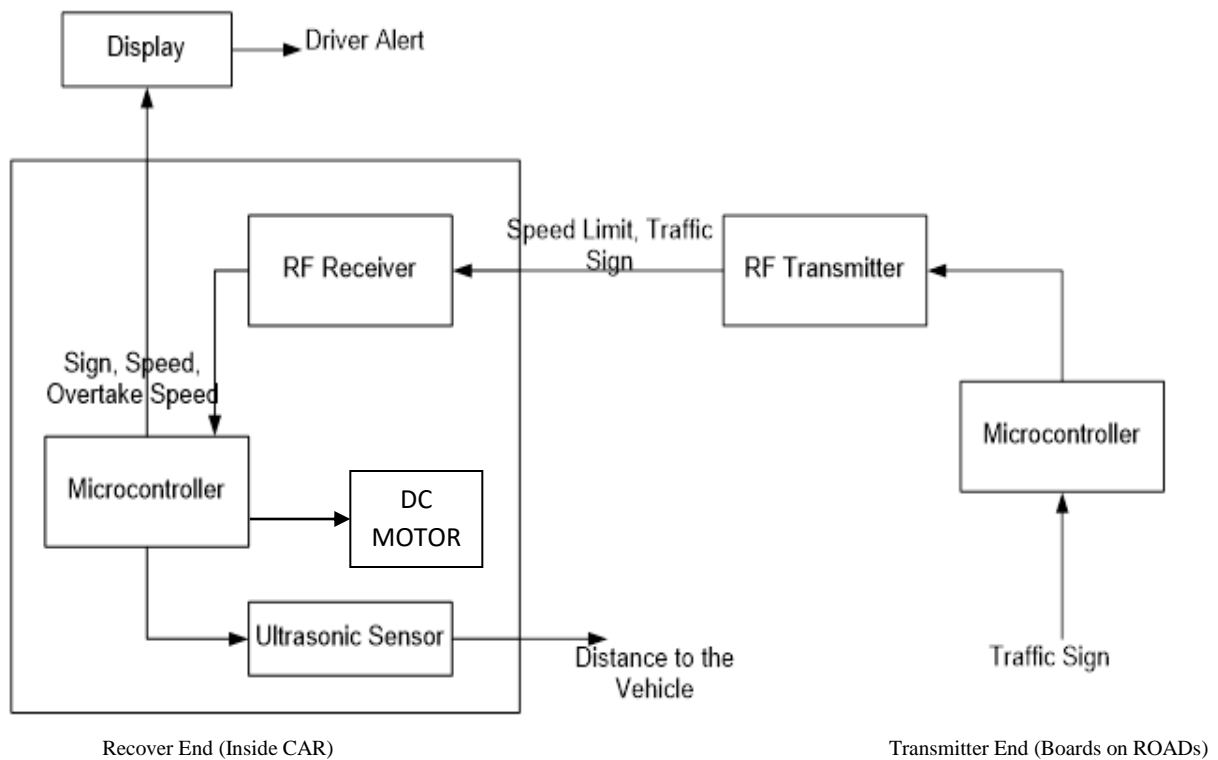


Figure 4.1. Block Diagram

### 4.2 Methodology

The project is developed for helping the driver for safe and efficient driving. In this application based project, we are focusing on solving the problems associated with three scenarios.

Firstly, we would be informing the driver about the traffic sign board ahead. An Arduino along with a RF transmitter is placed at the road (transmitter end) and a receiver block inside the car that will receive and display the signs to the driver along with a visual alert. The RF components are operated at 433MHz band. Arduino on receiving the signals, send the data to MATLAB software which compares them with predefined standard traffic signs in the database to display and inform the driver about the traffic sign.

Secondly, we concentrate to make the car adapt to the predefined and different speed limits. Here, the transmitter block will transmit the speed limit and receiver inside the car will receive; then the present status as well as required action is computed and is implemented on the car speed. The speed limit, present status and required action is alerted to the driver. Here, this scenario is being implemented by simulating a virtual remote car using a DC motor controlled by the Arduino board. The speed to be limited is input from the defined program and speed limit.

Finally, our project focuses on helping the driver during overtaking scenarios. The driver will be informed about the safe distance, safe speed and also inform about the overtaking sight distance. Here an ultrasonic sensor is placed at the front side of the car; it measures the distance between the car and the front vehicle or obstacle using the echo pulses. The ultrasonic sensor is interfaced to an Arduino. The initial data is computed as per algorithm and sent to MATLAB via serial cable. Accordingly, the MATLAB is programmed to compute the speed of the two vehicles and the desired parameters to display and inform the driver through the monitor.

### 4.3 User Case Diagram

A use case diagram is a type of behavioral diagram created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

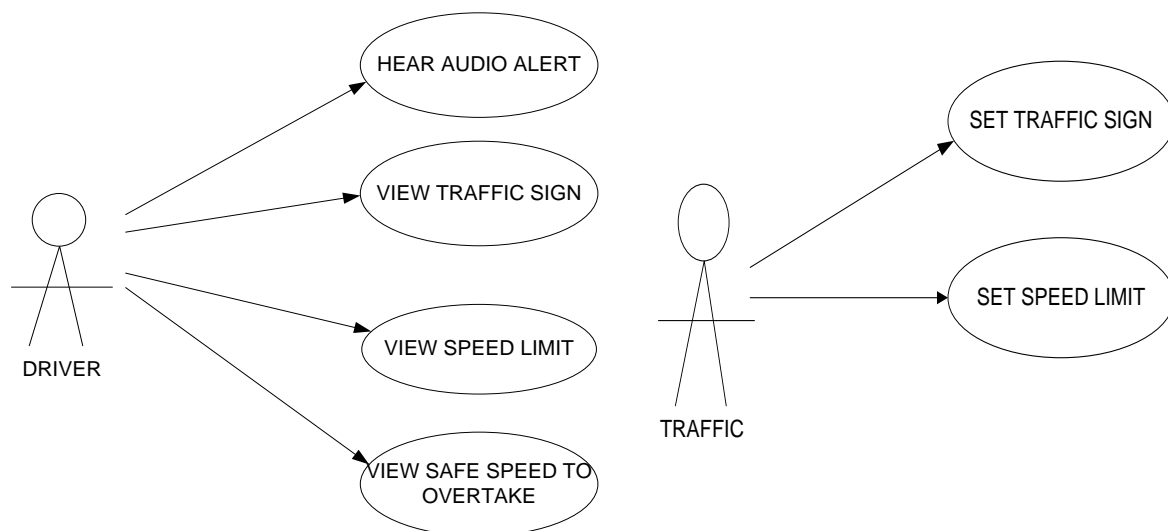
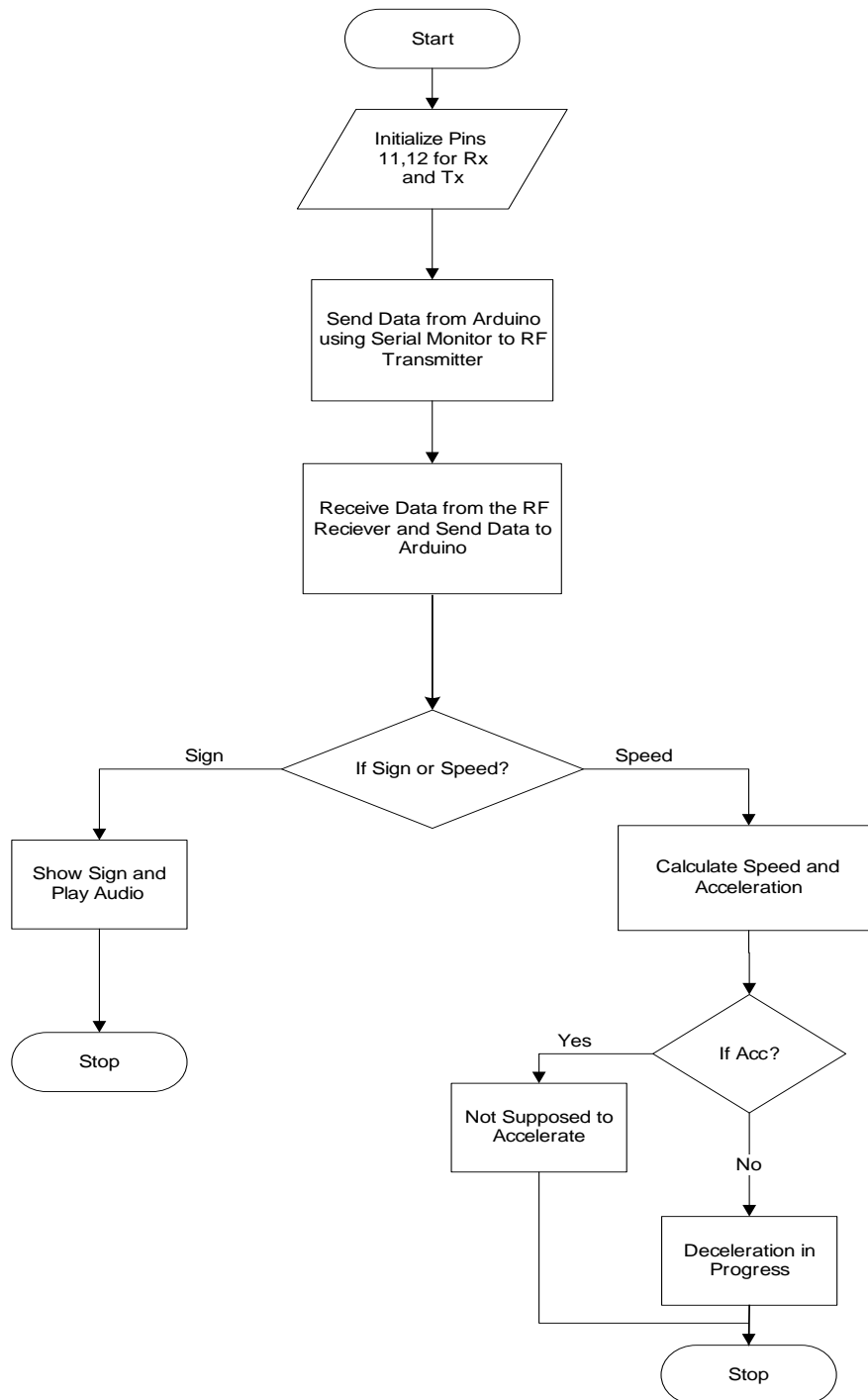


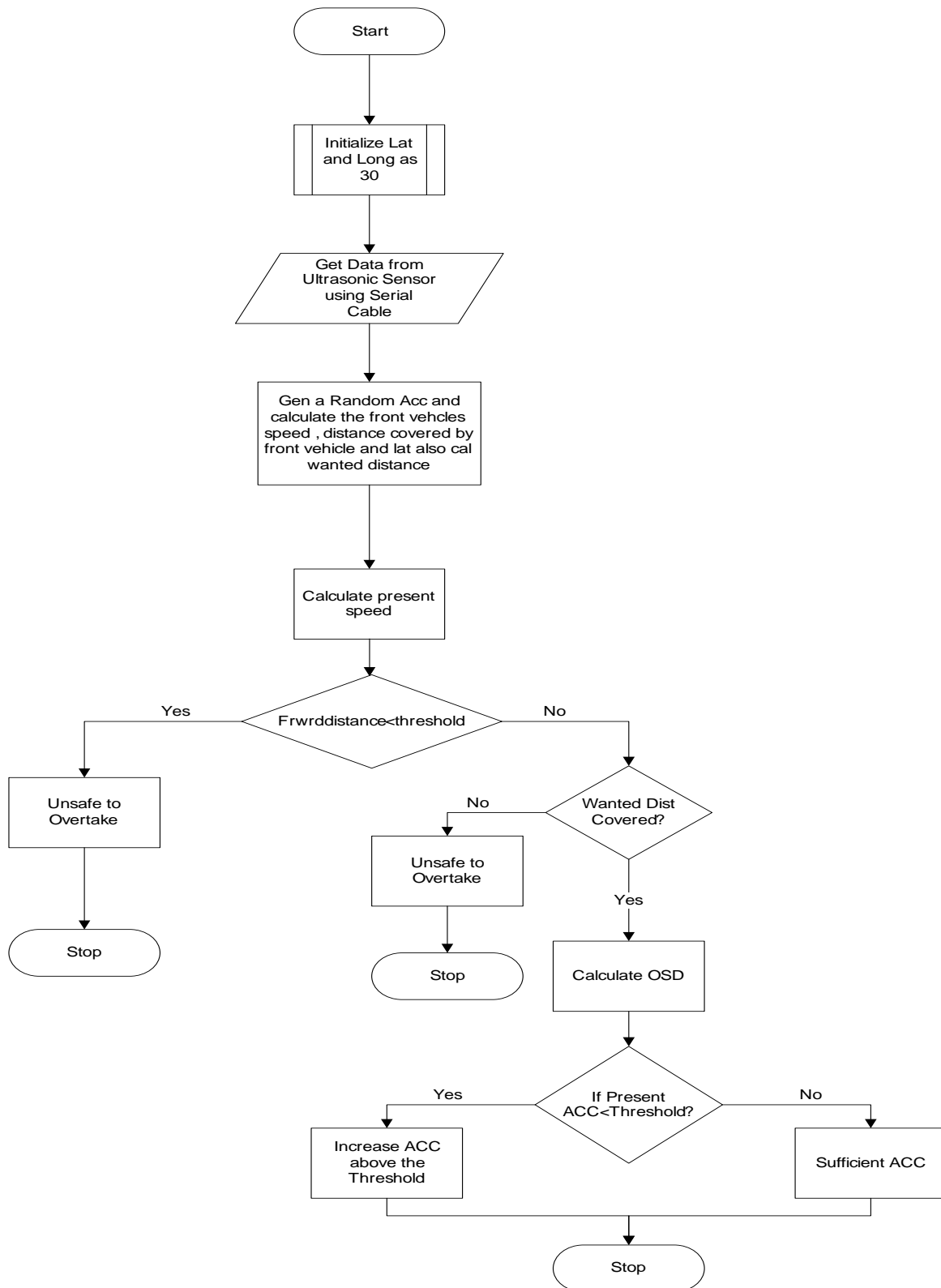
Figure 4.2. User Case Diagram

## 4.4 Flowchart

**CASE 1:** Traffic Sign Recognition and **CASE 2:** Intelligent Speed Adaptation



**Figure 4.3. Case 1 and Case 2 Flowchart**

**CASE 3: Overtaking Assistance****Figure 4.4. Case 3 Flowchart**

## 4.5 Data Flow Diagram of the system

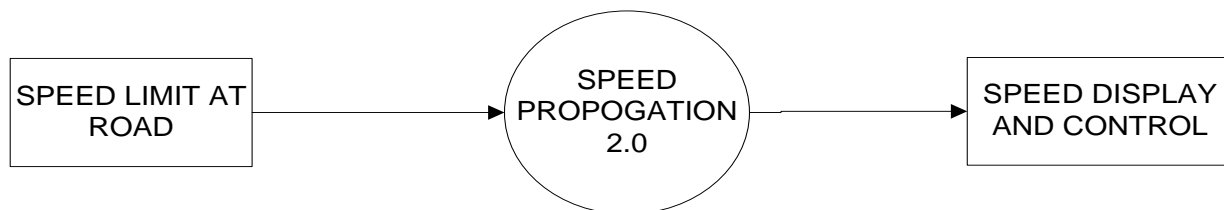
A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

### Data flow diagram

A context-level or level 0 data flow diagram shows the interaction between the system and external agents which act as data sources and data sinks. On the context diagram (also known as the Level 0 DFD) the system's interactions with the outside world are modeled purely in terms of data flows across the system boundary. The context diagram shows the entire system as a single process, and gives no clues as to its internal organization.



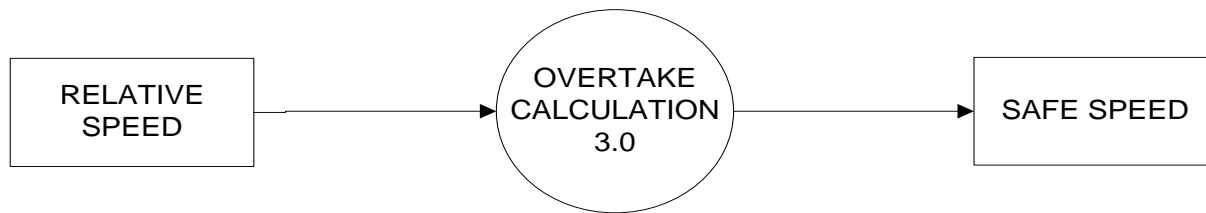
Here in level 0, traffic sign (board) is taken as input and user at receiver end obtains the visual and audio alert of the traffic sign as output. The first main process is transmission, propagation and reception of the traffic sign.



Here in this, speed limit of the road defined by the traffic police department is taken as input and is delivered to target car (receiver ends) which is taken as output. The second main process is speed limit display and control. The speed limit, present status and required action is alerted to the driver. Here, this scenario is being implemented by simulating a virtual car using a



DC motor controlled by the Arduino board. The speed is limited or controlled by defined program and speed limit.



Here in this, vehicle (user) speed, front moving vehicle speed, distance between them is taken as input and based on which relative speed, overtaking sight distance (OSD) and safe speed is calculated. The “SAFE” or “UNSAFE” to overtake message is delivered to target user which is the output. These are depicted by “GREEN” or “RED” signal on the monitor along with other messages. The third main process is safe overtaking speed calculation.

## Summary

This chapter mainly concentrates on system architecture, methodology, use-case diagram, flow chart, data flow diagram etc.

## Chapter 5

# SYSTEM REQUIREMENT SPECIFICATION

System Requirement Specification is a fundamental document, which forms the foundation of the software development process. It not only lists the requirements of a system but also has a description of its major feature. An SRS is basically an organization's understanding (in writing) of a customer or potential client's system requirements and dependencies at a particular point in time (usually) prior to any actual design or development work. It's a two-way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time. The SRS also functions as a blueprint for completing a project with as little cost growth as possible. The SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it. It is important to note that an SRS contains functional and non-functional requirements only; it doesn't offer design suggestions, possible solutions to technology or business issues, or any other information other than what the development team understands the customer's system requirements to be.

## 5.1 Functional Requirement

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. In this system following are the functional requirements:-

1. Traffic sign is encoded and sent via transmitter.
2. Traffic sign are received by the receiver and are decoded.
3. A visual alert of the traffic sign must be displayed.
4. An audio alert of the traffic sign must be provided.

5. The speed limit on the road must be transmitted.
6. Vehicles should receive the speed limit information and display.
7. By comparing the current speed and information received, an action is to be suggested and displayed.
8. Based on which, the speed of the vehicle is restricted to the target speed limit in that zone.
9. Measuring of the distance from the front moving vehicle using ultra sonic sensor, computing the relative speed, overtaking sight distance, safe speed and other parameters as per the derived formulas.
10. Calculate safe speed and overtaking sight distance (OSD) and display the result.

## 5.2 Non-functional Requirement

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Non-functional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as:-

1. Product Requirements; 2.Organizational Requirements; 3.User Requirements;
- 4.Basic Operational Requirements

### 5.2.1 Product Requirements

1. **Portability:** The software and hardware developed in the system can work for any vehicle, and can have minor or no modifications.
2. **Correctness:** The traffic signs must be displayed accurately. It follows a well-defined set of procedures and rules to compute and also rigorous testing is performed to confirm the correctness of the data.

3. **Ease of Use:** The front end is designed in such a way that it provides an interface which allows the user to interact in an easy manner. The system is designed as a plug and play type.
4. **Modularity:** The system must be modular, so that new traffic sign, speed limit zone can be added easily. The complete product is broken up into many modules and well-defined interfaces are developed to explore the benefit of flexibility of the product.
5. **Robustness:** This software is being developed in such a way that the overall performance is optimized and the user can expect the results within a limited time with utmost relevancy and correctness. The system should work and correspond to any traffic sign, any speed and any distance.

Non-functional requirements are also called the qualities of a system. These qualities can be divided into execution quality & evolution quality. Execution qualities are security & usability of the system which are observed during run time, whereas evolution quality involves testability, maintainability, extensibility or scalability.

### 5.2.2 Organizational Requirements

1. **Process Standards:** IEEE standards are used to develop the application which is the standard used by the most of the standard software developers all over the world.
2. **Design Methods:** Design is one of the important stages in the software engineering process. This stage is the first step in moving from problem to the solution domain. In other words, starting with what is needed design takes us to work how to satisfy the needs.

The design of the system is perhaps the most critical factor affecting the quality of the software and has a major impact on the later phases, particularly testing and maintenance. We have to design the product with the standards which has been understood by the developers of the team. We will use Object Oriented decomposition for this work.

### 5.2.3 User Requirements

- The user must be able to visualize Graphical User Interface Window.
- The user must be able to configure all the parameters with neat GUI.
- In case of system malfunction, an error should be displayed.
- In case of system not receiving any data from the transmitter for a long period of time, an error message should be displayed.
- The user must be notified about the parameters and also must be assisted with suitable action.

### 5.2.4 Basic Operational Requirements

The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, will be related to these following points:-

- **Mission profile or scenario:** It describes about the procedures used to accomplish mission objective. It also finds out the effectiveness or efficiency of the system.
- **Performance and related parameters:** It points out the critical system parameters to accomplish the mission. The response time to recognise the traffic sign, speed limit information; calculate safe speed and distance must be in few mini seconds.
- **Utilization environments:** It gives a brief outline of system usage. It finds out appropriate environments for effective system operation. The system should work and operate under 12V DC power supply.
- **Operational life cycle:** It defines the system lifetime. The traffic sign, speed limit information must be sent for infinite period in the range from start to stop of the transmitter.

## Summary

This chapter gives details of the functional requirements, non-functional requirements, resource requirements, hardware requirements, software requirements etc. Again the non-functional requirements in turn contain product requirements, organizational requirements, user requirements, basic operational requirements etc.

## Chapter 6

# HARDWARE REQUIREMENTS

### 6.1 Arduino UNO

Arduino/Genuino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programmed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

The key features are –

- Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
- You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).
- Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.
- Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.

- Finally, Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

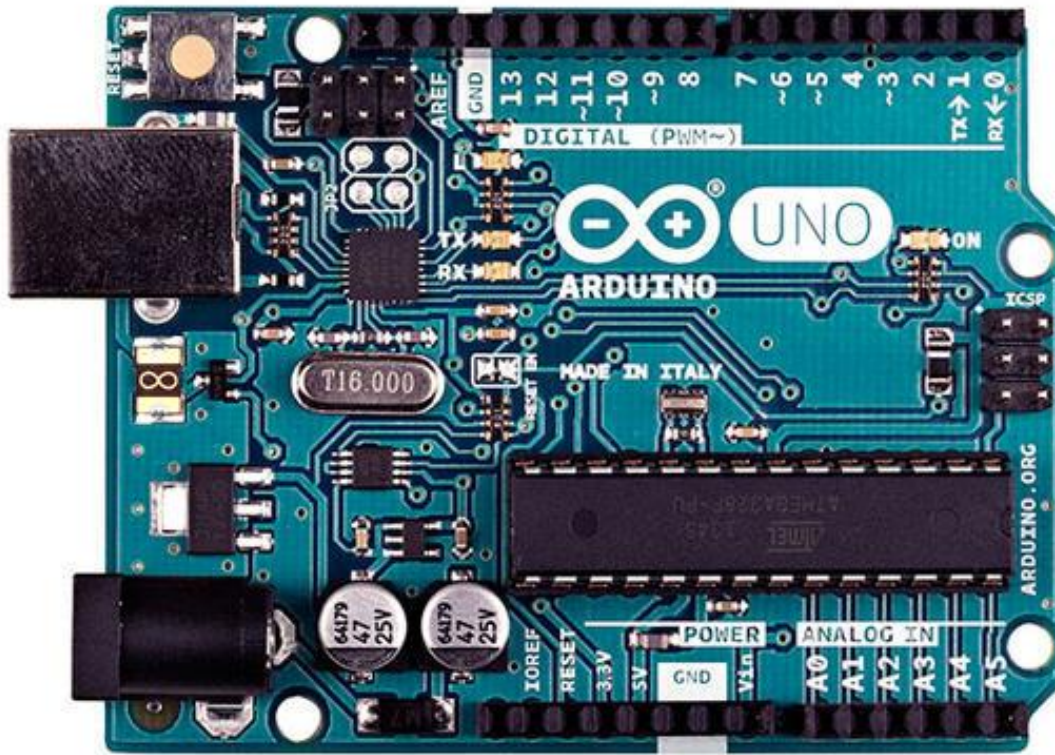


Figure 6.1 Arduino UNO board

Various kinds of Arduino boards are available depending on different microcontrollers used. However, all Arduino boards have one thing in common: they are programmed through the Arduino IDE.

The differences are based on the number of inputs and outputs (the number of sensors, LEDs, and buttons you can use on a single board), speed, operating voltage, form factor etc. Some boards are designed to be embedded and have no programming interface (hardware), which you would need to buy separately. Some can run directly from a 3.7V battery, others need at least 5V.



## 6.1 Board Description

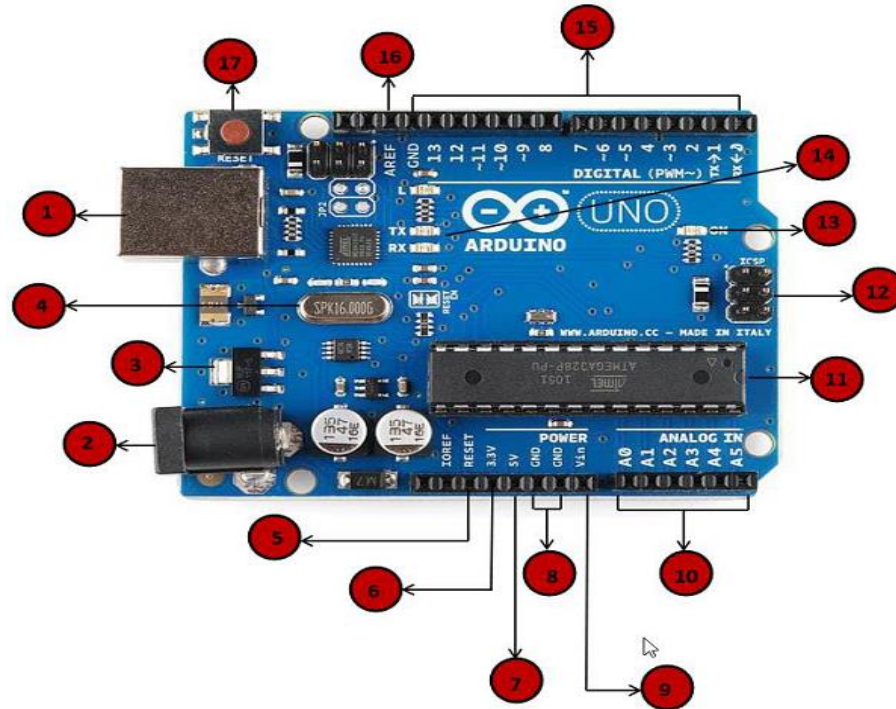







Figure 6.2 Arduino UNO board pins and description

Table 6.1 Arduino UNO Board Description

<p>1</p>	<p><b>Power USB</b></p> <p>Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).</p>
<p>2</p>	<p><b>Power (Barrel Jack)</b></p> <p>Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).</p>
<p>3</p>	<p><b>Voltage Regulator</b></p> <p>The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.</p>

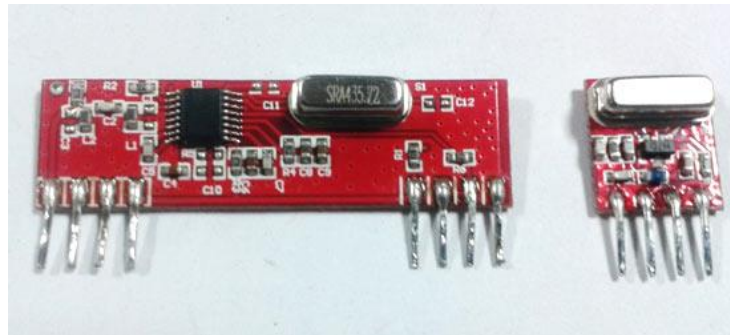


	<p><b>Crystal Oscillator</b></p> <p>The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.</p>
	<p><b>Arduino Reset</b></p> <p>You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).</p>
	<p><b>Pins (3.3, 5, GND, Vin)</b></p> <p>3.3V (6) – Supply 3.3 output volt</p> <p>5V (7) – Supply 5 output volt</p> <p>Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.</p> <p>GND (8) (Ground) – There are several GND pins on the Arduino, any of which can be used to ground your circuit.</p> <p>Vin (9) – This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.</p>
	<p><b>Analog pins</b></p> <p>The Arduino UNO board has five analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.</p>
	<p><b>Main microcontroller</b></p> <p>Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top</p>

	of the IC. For more details about the IC construction and functions, you can refer to the data sheet.
12	<p><b>ICSP pin</b></p> <p>Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.</p>
13	<p><b>Power LED indicator</b></p> <p>This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.</p>
14	<p><b>TX and RX LEDs</b></p> <p>On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.</p>
15	<p><b>Digital I/O</b></p> <p>The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled “~” can be used to generate PWM.</p>
16	<p><b>AREF</b></p> <p>AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.</p>

## 6.2 RF Transmitter and Receiver Module

The RX – ASK is an ASK Hybrid receiver module. It is a effective low cost solution for using 433 MHz. The TX-ASK is an ASK hybrid transmitter module. TX-ASK is designed by the saw resonator, with an effective low cost, small size and simple to use for designing.

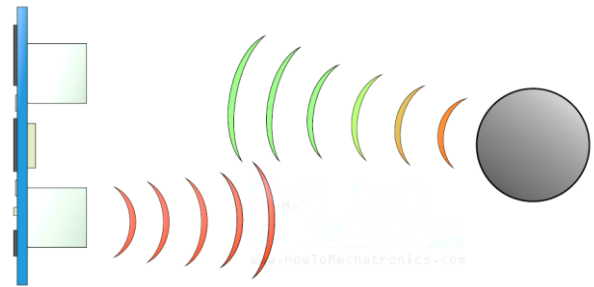


**Figure 6.3. RF Transmitter and Receiver Module**

This radio frequency (RF) transmission system employs Amplitude Shift Keying (ASK) with transmitter/receiver (Tx/Rx) pair operating at 433 MHz. The transmitter module takes serial input and transmits these signals through RF. The transmitted signals are received by the receiver module placed away from the source of transmission. The system allows one way communication between two nodes, namely, transmission and reception.

## 6.3 Ultrasonic Sensor

An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sonar sensor and the object. It emits an ultrasound at 40 000 Hz which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.

**Figure 6.4. Ultrasonic Sensor Module****Figure 6.5. Working Principle of Ultrasonic Sensor**

Since it is known that sound travels through air at about 344 m/s (1129 ft./s), you can take the time for the sound wave to return and multiply it by 344 meters (or 1129 feet) to find the total round-trip distance of the sound wave. Round-trip means that the sound wave traveled 2 times the distance to the object before it was detected by the sensor; it includes the 'trip' from the sonar sensor to the object AND the 'trip' from the object to the Ultrasonic sensor (after the sound wave bounced off the object).

To find the distance to the object, simply divide the round-trip distance in half.

$$\text{distance} = \frac{\text{speed of sound} \times \text{time taken}}{2}$$

**Figure 6.6. Round Trip Distance of Sound Wave**

In order to generate the ultrasound you need to set the Trig on a High State for 10  $\mu$ s. That will send out an 8 cycle sonic burst which will travel at the speed sound and it will be received in the Echo pin. The Echo pin will output the time in microseconds the sound wave traveled.

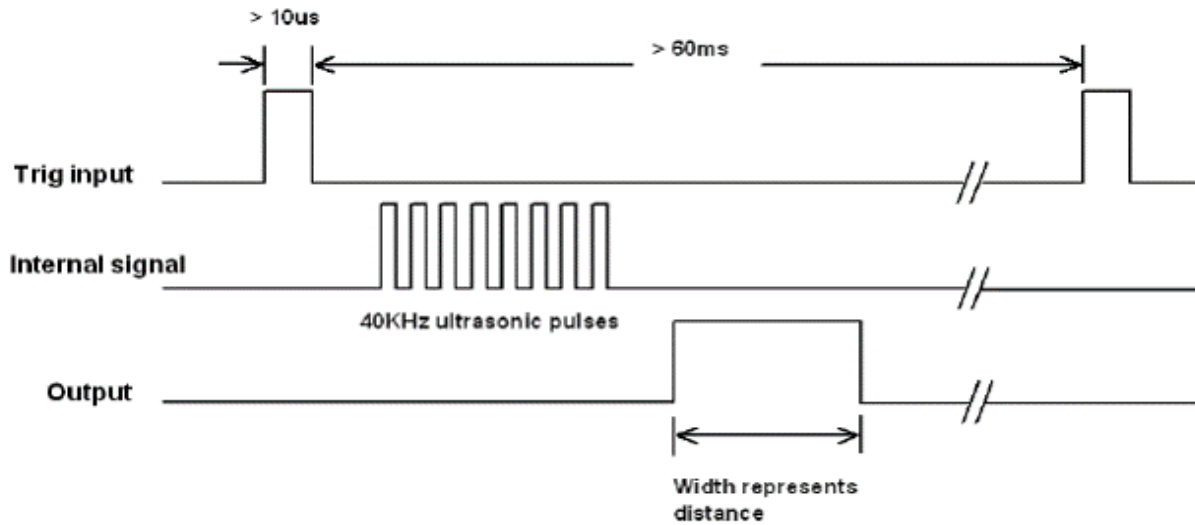


Figure 6.7. HC-SR04 Timing Diagram

**For example**, if the object is 10 cm away from the sensor, and the speed of the sound is 340 m/s or 0.034 cm/µs the sound wave will need to travel about 294 µ seconds. But what you will get from the Echo pin will be double that number because the sound wave needs to travel forward and bounce backward. So in order to get the distance in cm we need to multiply the received travel time value from the echo pin by 0.034 and divide it by 2.

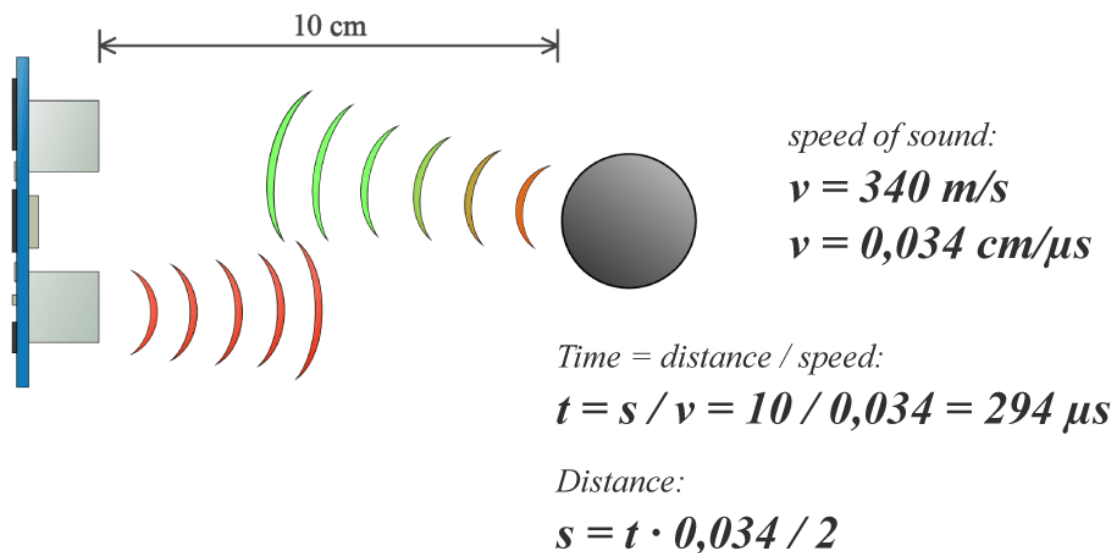


Figure 6.8. Ultrasonic Sensor Working Principle

## 6.4 Buzzer

A buzzer or a beeper is an audio signaling device which may be mechanical, electromechanical or piezoelectric. Typical uses of buzzers and beepers include alarm devices, timers and confirmation of user input such as a mouse click or keystroke.

A piezoelectric element may be driven by an oscillating electronic circuit or other audio signal source, driven with an amplifier. Sounds commonly used to indicate that a button has been pressed are a click, a ring or a beep.

Two wires are red & black. Black represents ground. Any oscillating voltage applied causes noise. The buzzer case supports the piezo element and has resonant cavity for sound. Buzzer gives the audio indication when Arduino receives data which is transmitted by the transmitter. It indicates the successful reception of data.



### Figure 6.9. Buzzer

## 6.5 LED

A light-emitting diode (LED) is a two-lead semiconductor light source. It is a p-n junction diode that emits light when activated. When a suitable voltage is applied to the leads, electrons are able to recombine with electron holes within the device, releasing energy in the form of photons. This effect is called electroluminescence, and the color of the light (corresponding to the energy of the photon) is determined by the energy band gap of the semiconductor. LEDs are typically small (less than 1 mm<sup>2</sup>) and integrated optical components may be used to shape the radiation pattern.

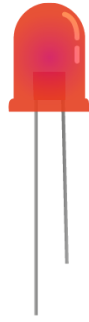


Figure 6.10. LED

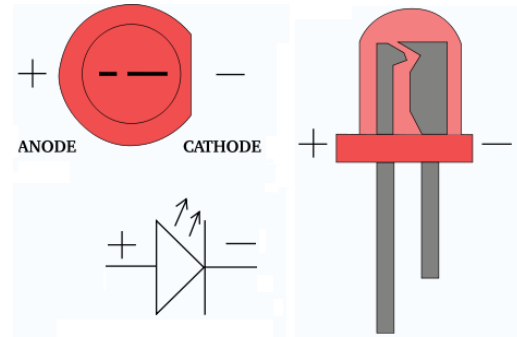


Figure 6.11. LED Internal Structure

## 6.6 DC Motor

A DC Motor is a type of electric motor that converts DC electrical power to mechanical power i.e. a DC supply is converted to rotation or movement. DC motors are one of the commonly used motors in different applications like electronic toys, power tools, portable fans, etc. A DC motor (Direct Current motor) is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.



Figure 6.12. DC Motor Module

## 6.7 H Bridge

An H bridge is an electronic circuit that enables a voltage to be applied across a load in either direction. These circuits are often used in robotics and other applications to allow DC motors to run forwards or backwards. Most DC-to-AC converters (power inverters), most

AC/AC converters, the DC-to-DC push-pull converter, most motor controllers, and many other kinds of power electronics use H bridges. In particular, a bipolar stepper motor is almost invariably driven by a motor controller containing two H Bridges.

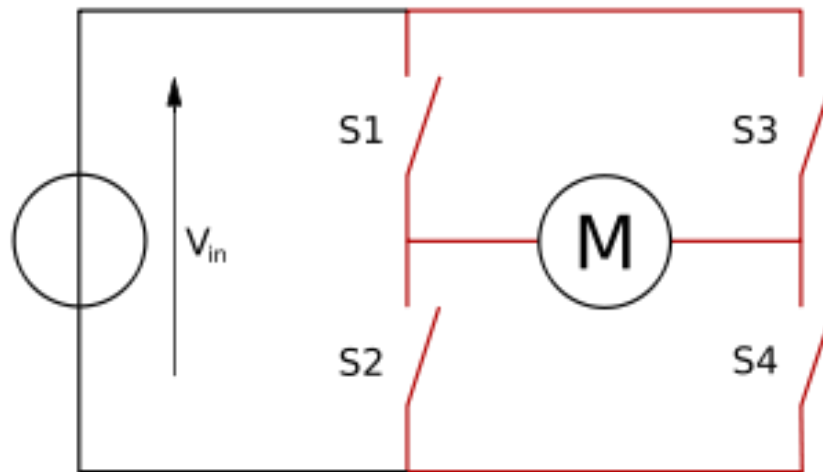


Figure 6.13. H Bridge Circuit Diagram



Figure 6.14. H Bridge Module



## Chapter 7

# SOFTWARE REQUIREMENTS

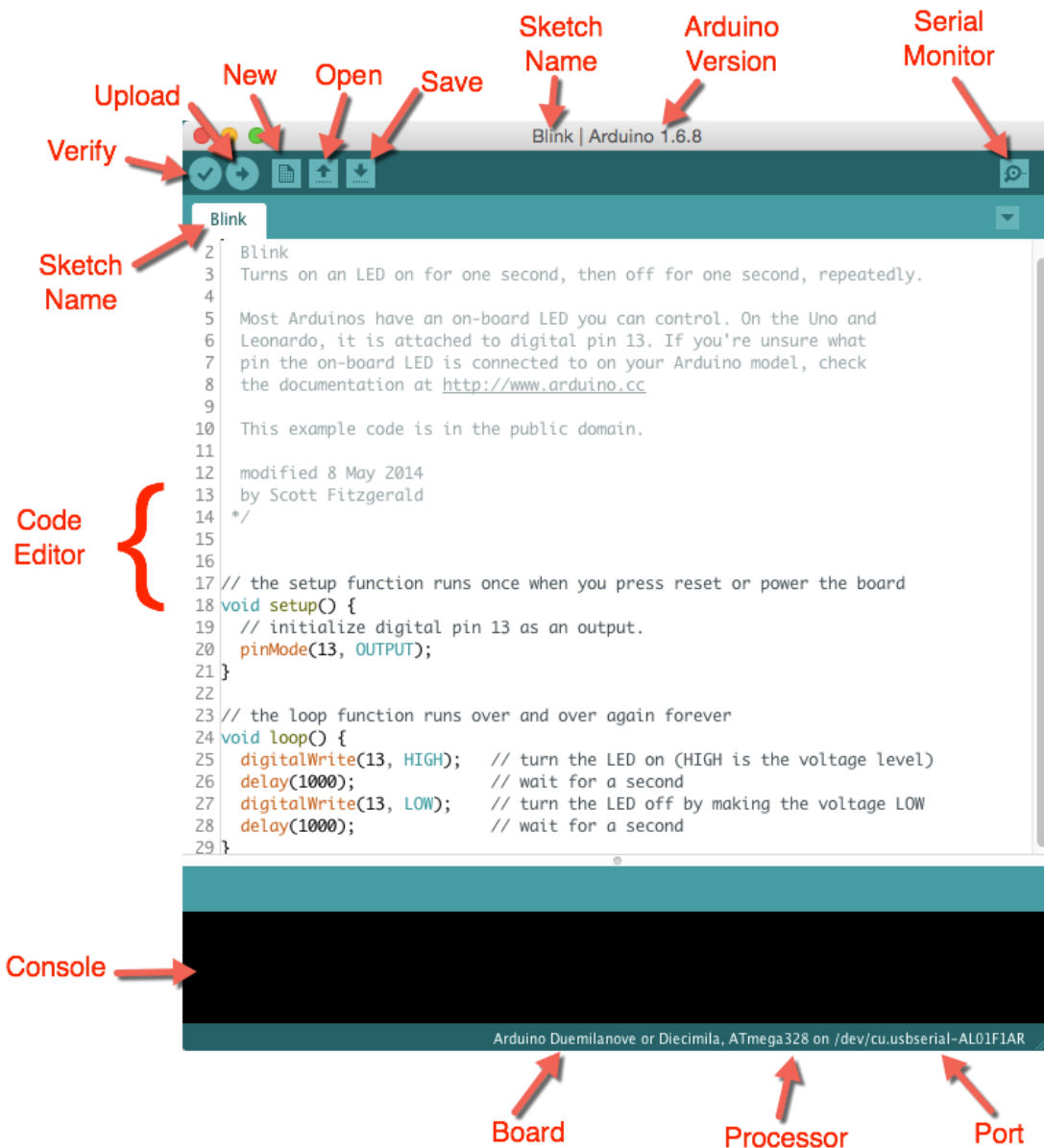
### 7.1 ARDUINO

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.



Figure 7.1. Arduino IDE



**Figure 7.2. Arduino IDE Sketchbook**

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from within the Preferences dialog.

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more `#include` statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its `#include` statements from the top of your code.

Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Displays serial data being sent from the Arduino or Genuino board (USB or serial board). To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to `Serial.begin` in your sketch.

The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

Arduino programs can be divided in three main parts: **Structure**, **Values** (variables and constants), and **Functions**. In this tutorial, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

Let us start with the **Structure**. Software structure consist of two main functions –

- `Setup()` function
- `Loop()` function

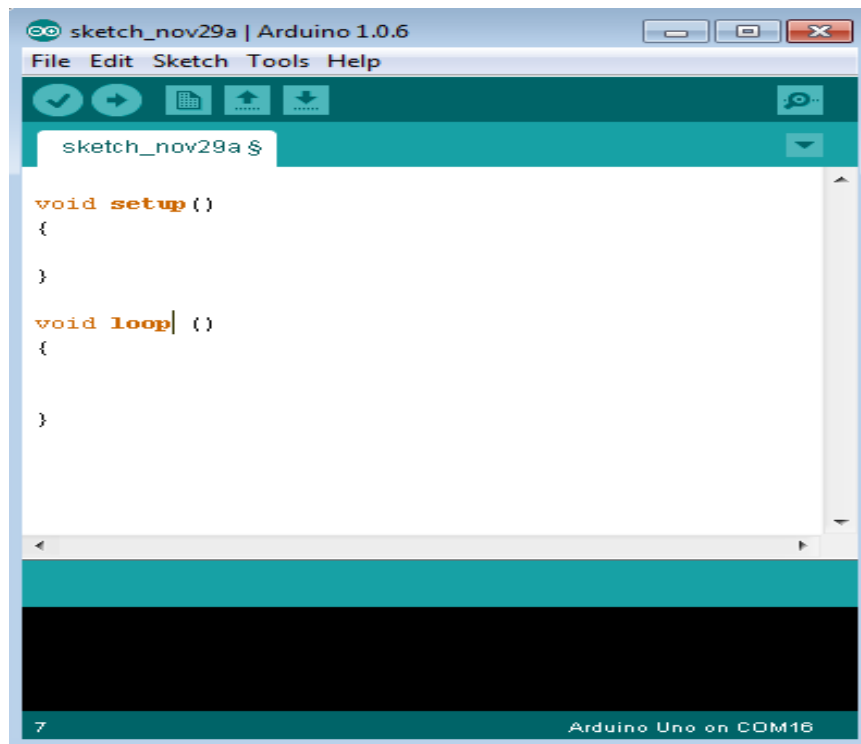


Figure 7.3. Arduino IDE Structure

```
void setup ()
```

```
{  
//  
}
```

**PURPOSE** – The **setup ()** function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

- **INPUT** – -
- **OUTPUT** – -
- **RETURN** – -

```
void loop ()
```

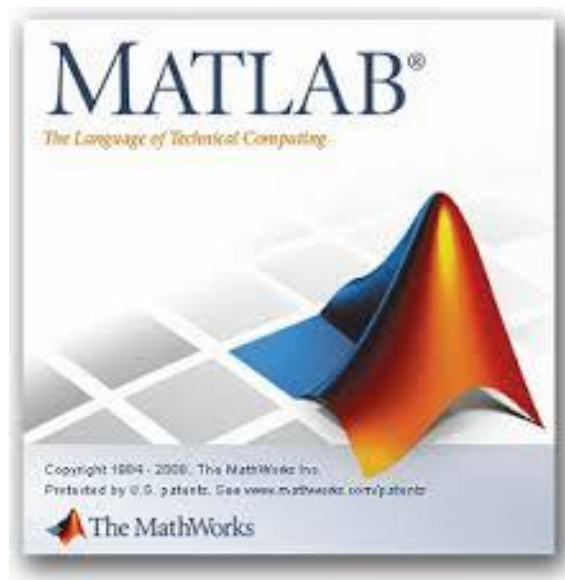
```
{  
// // rest of the code  
}
```

**PURPOSE** – After creating a **setup ()** function, which initializes and sets the initial values, the **loop ()** function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

- **INPUT** – -
- **OUTPUT** – -
- **RETURN** – -

## 7.2 MATLAB

MATLAB is a programming language developed by MathWorks. It started out as a matrix programming language where linear algebra programming was simple. It can be run both under interactive sessions and as a batch job.



**Figure 7.4. MATLAB Software**

MATLAB (MATrix LABoratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietary programming language developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPADsymbolic engine, allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

GUIs (also known as graphical user interfaces or UIs) provide point-and-click control of software applications, eliminating the need to learn a language or type commands in order to run the application.

MATLAB apps are self-contained MATLAB programs with GUI front ends that automate a task or calculation. The GUI typically contains controls such as menus, toolbars, buttons, and sliders. Many MATLAB products, such as Curve Fitting Toolbox, Signal Processing Toolbox, a MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar no interactive language such as C or FORTRAN.

Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

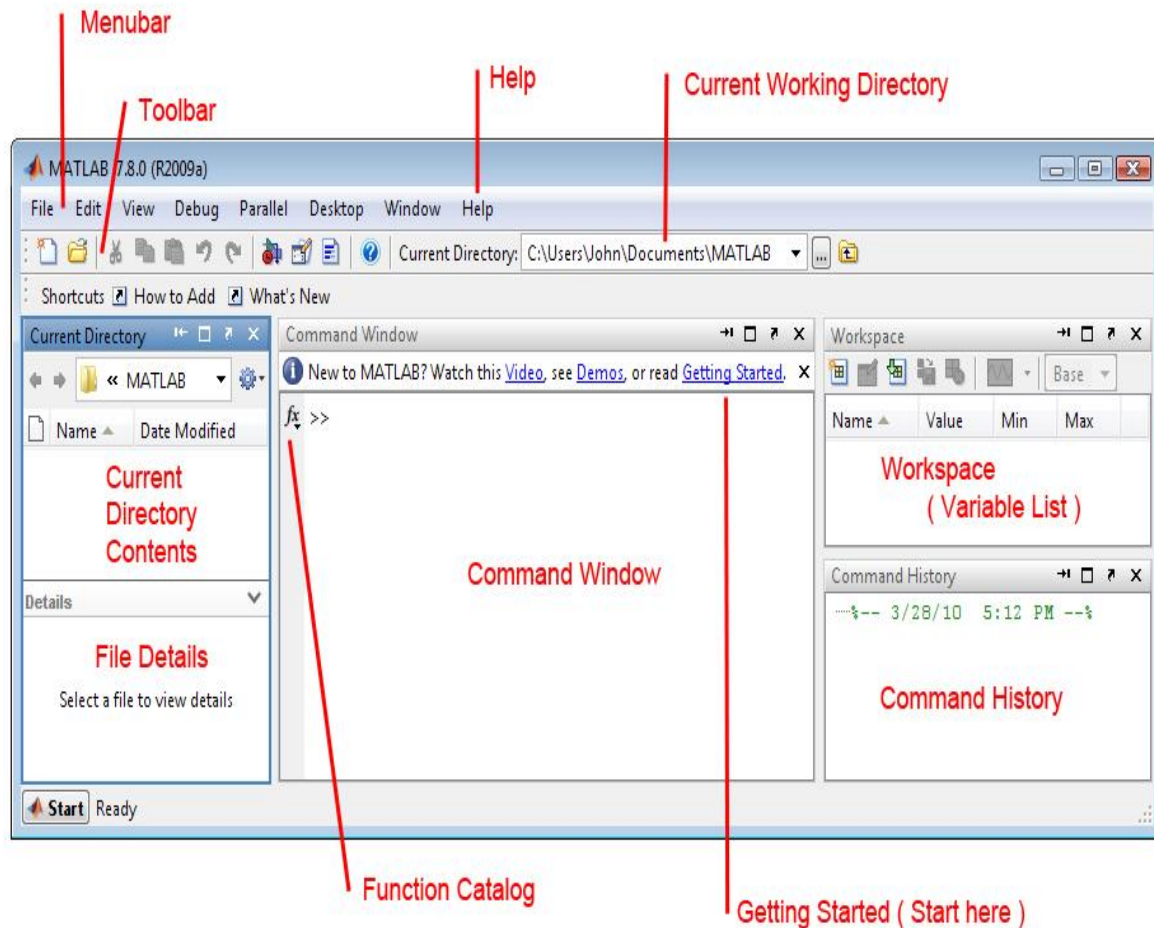


Figure 7.5. MATLAB Software Work Environment

### 7.1.3 Features of MATLAB

Following are the basic features of MATLAB –

- It is a high-level language for numerical computation, visualization and application development.
- It also provides an interactive environment for iterative exploration, design and problem solving.
- It provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.
- It provides built-in graphics for visualizing data and tools for creating custom plots.

- MATLAB's programming interface gives development tools for improving code quality maintainability and maximizing performance.
- It provides tools for building applications with custom graphical interfaces.
- It provides functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET and Microsoft Excel.

### **7.1.1 Uses of MATLAB**

MATLAB is widely used as a computational tool in science and engineering encompassing the fields of physics, chemistry, math and all engineering streams. It is used in a range of applications including –

- Signal Processing and Communications
- Image and Video Processing
- Control Systems
- Test and Measurement
- Computational Finance
- Computational Biology

The MATLAB system consists of five main parts:

1. The MATLAB language.

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

2. The MATLAB working environment.

This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in your workspace and importing and exporting data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB's applications.



### 3. Handle Graphics.

This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete Graphical User Interfaces on your MATLAB applications.

### 4. The MATLAB mathematical function library.

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

### 5. The MATLAB Application Program Interface (API)

This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files. The Control System Toolbox includes apps with custom user interfaces.

## Chapter 8

# HARDWARE IMPLEMENTATION PROCEDURE

## 8.1 Arduino UNO

Arduino UNO is connected to the COMx(x=1,2,3,4,5,1,52 etc.) port of the laptop. This is done in order to keep the board attentive at all the times, to any sort of transmission/reception of data or operations. The data is transmitted from one board is received by the other board which are interfaced with the transmitter and receivers. Upon receiving the data, the second arduino board activates the respective LEDs and buzzer to indicate the reception of particular data. This data is sent to the system serially (i.e. to MatLab); which interprets the data and provide the results on the monitor. The data is fed to the first arduino via Serial Monitor. The boards are powered up using 5V power supply.

## 8.2 RF Transmitter and Receiver

433MHz operated RF transmitter and receiver are interfaced to arduino boards; where they are connected with 5V (VCC) and GND (ground) from the board. The data pin of the RF transmitter is connected to IOPIN12 of first arduino board. The data of RF receiver is connected to IOPIN11 of second arduino board. The RF transmitter continuously transmits as available from the Serial Monitor; RF receiver is operated at the same band of frequency checks for data availability. The successful reception is indicated by the LEDs and Buzzer.

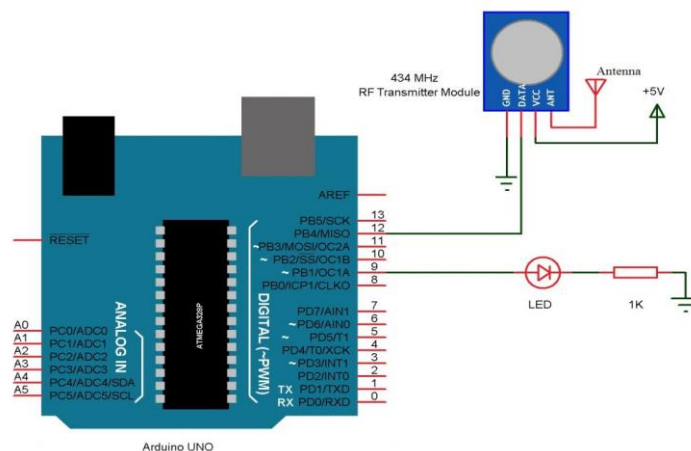


Figure 7.6. Transmitter End Circuit

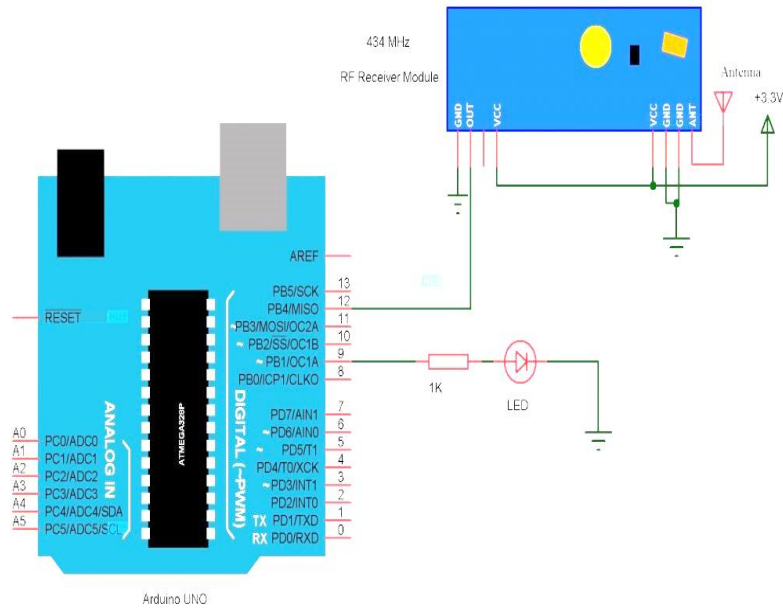


Figure 7.7. Receiver End Circuit

### 8.3 Ultrasonic Sensor

The HC-SR04 Ultrasonic Module has 4 pins, Ground, VCC, Trig and Echo. The ultrasonic sensor is interfaced to the arduino board; The Ground and the VCC pins of the module needs to be connected to the ground and the 5V pins on the arduino board respectively. The TRIG pin of the sensor is connected to IOPIN9 and the ECHO pin is connected to IOPIN10 of the arduino board. On detecting an obstacle or vehicle, it measures the distance between them and outputs to the arduino board. The TRIG pin is the input pin which is used to trigger the ultrasonic sensor at regular intervals to transmit sound pulses. The ECHO pin is the output pin which provides the data based on the detected echoes of the pulses to the Arduino board.

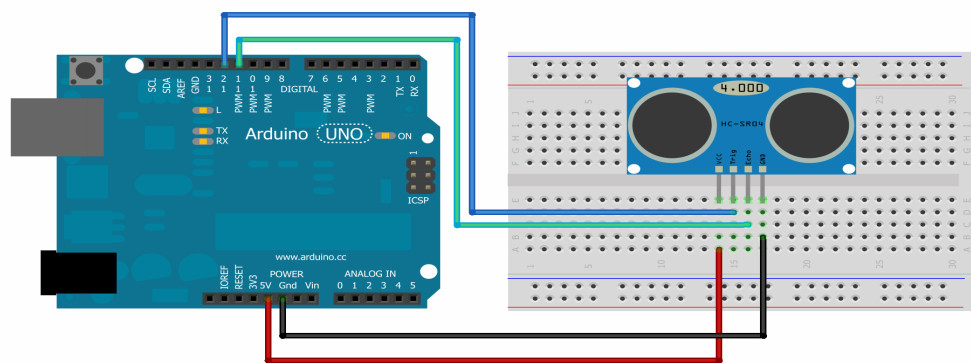


Figure 7.8. Overtaking Assistance Circuit

## Chapter 9

# SOFTWARE IMPLEMENTATION PROCEDURE

**Step 1** – First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.



Figure 8.1. Serial/USB Cable

**Step 2 – Download Arduino IDE Software** - Different versions of Arduino IDE from the Download page on the Arduino Official website. Select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.

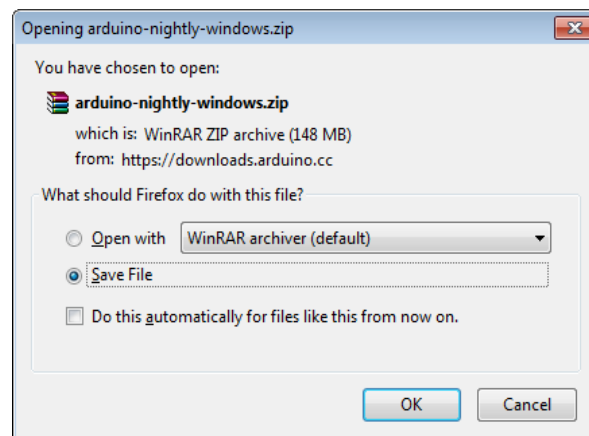


Figure 8.2. Arduino Software (.ZIP) File

**Step 3 – Power up your board.** - The Arduino Uno automatically draw power from either, the USB connection to the computer or an external power supply. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port. Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

**Step 4 – Launch Arduino IDE.** - After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.

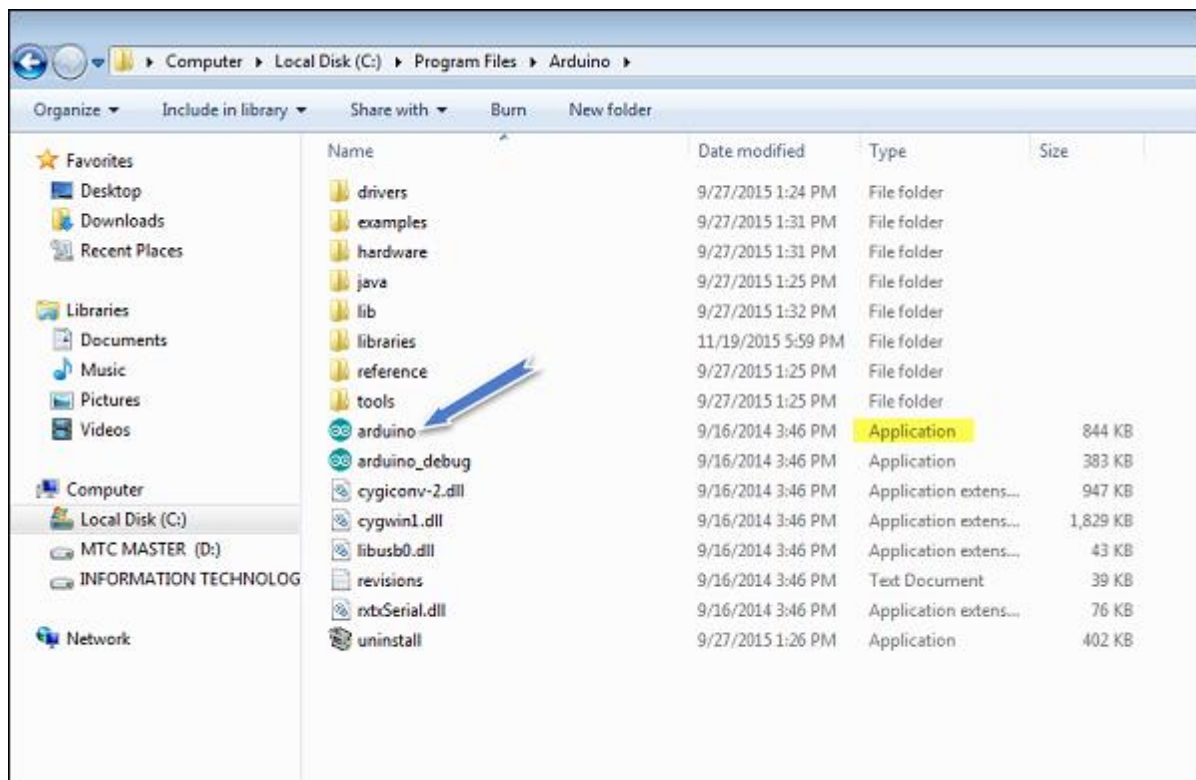


Figure 8.3. Launch Arduino IDE

**Step 5 – Open your first project.** - Once the software starts, you have two options –

- Create a new project.
- Open an existing project example.

To create a new project, select File → New.

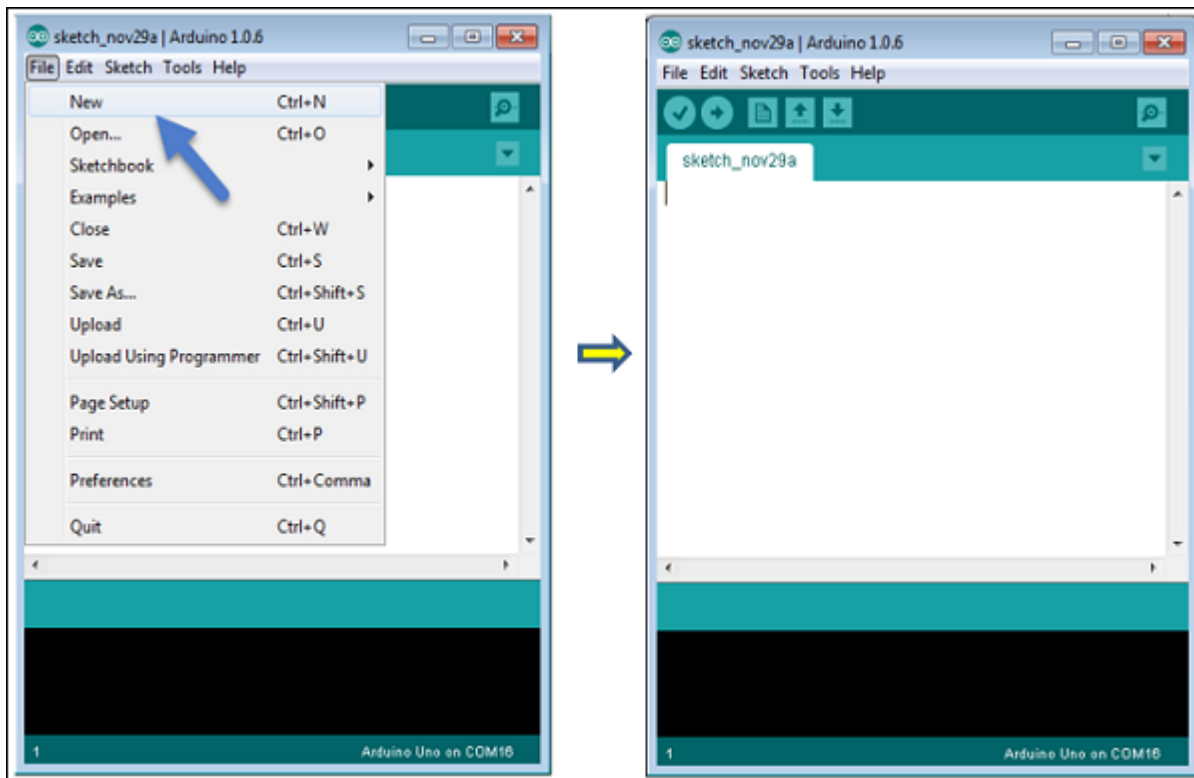


Figure 8.4. Open New Project

To open an existing project example, select File → Example → Basics → Blink. Include the library file I.e. VirtualWire and SoftwareSerial, Sketch → Include Library → Add.ZIP library → library\_name.zip.

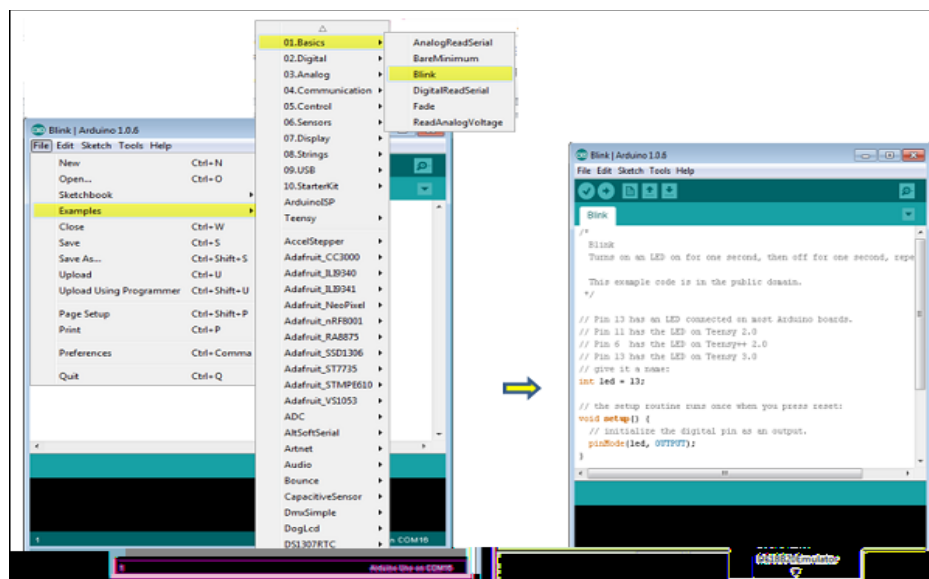
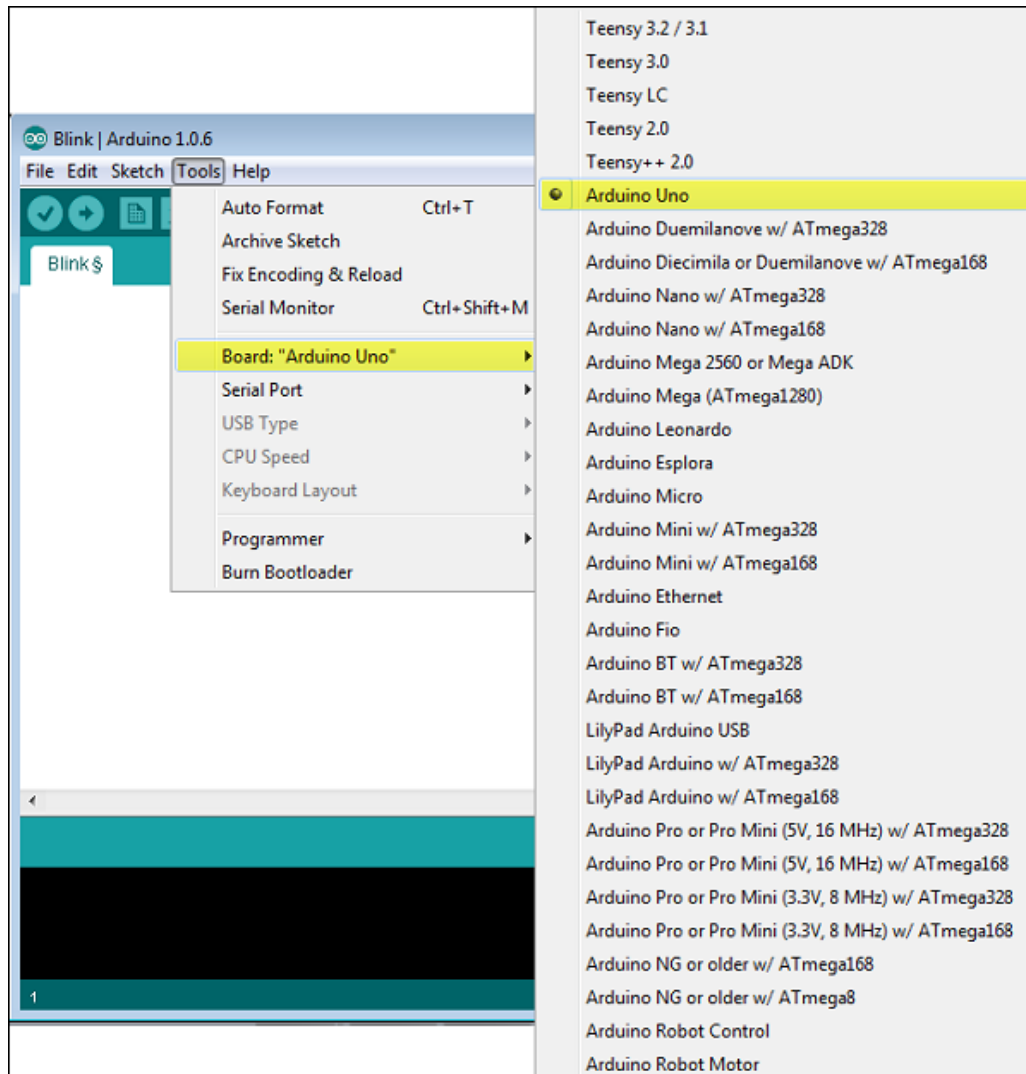


Figure 8.5. Opening an Existing Project

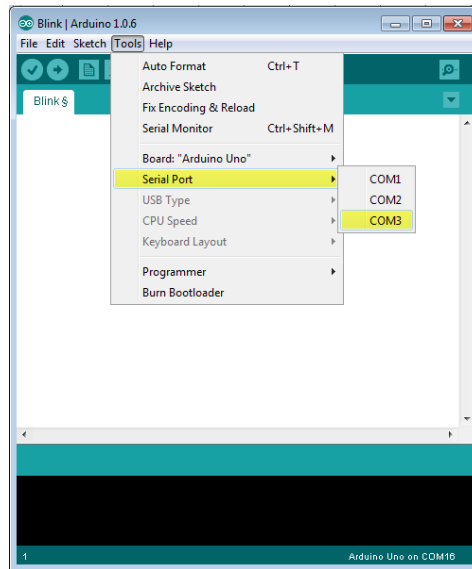
**Step 6 – Select your Arduino board.** - To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

Go to Tools → Board and select your board.



**Figure 8.6. Selection of Arduino board**

**Step 7 – Select your serial port.** - Select the serial device of the Arduino board. Go to Tools → Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



**Figure 8.7. Selection of Serial Port**

**Step 8 – Upload the program to your board.** - Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



**Figure 8.8. Arduino IDE Toolbar**

**A** – Used to check if there is any compilation error.

**B** – Used to upload a program to the Arduino board.

**C** – Shortcut used to create a new sketch.

**D** – Used to directly open one of the example sketch.

**E** – Used to save your sketch.

**F** – Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.



**Step 9 – Download and Install MATLAB Software** - Different versions of MATLAB are available, same can be checked on Official MATHWORKS website. Select your software, i.e. R2015a or whichever is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file, install and activate the software with license. After your MATALB software is downloaded, you need to open the folder. Inside the folder, you can find the application icon with an infinity label (matlab.exe). Double-click the icon to start the MATLAB.

Once the software starts, you have two options –

- Create a new script.
- Open an existing script.

Here, Program is coded in MATLAB; saved in .m format along with necessary databases. The program is saved and executed by clicking “RUN” icon. It will monitor for real time data from the serial port. Note: The COM port to which arduino is connected; is correctly specified in the command - serial ('COMx'). Upon receiving the data, it detects, analyses, computes and displays the output messages (along visual alerts) is simulated to the driver through the MATLAB GUI (Graphical User Interface).

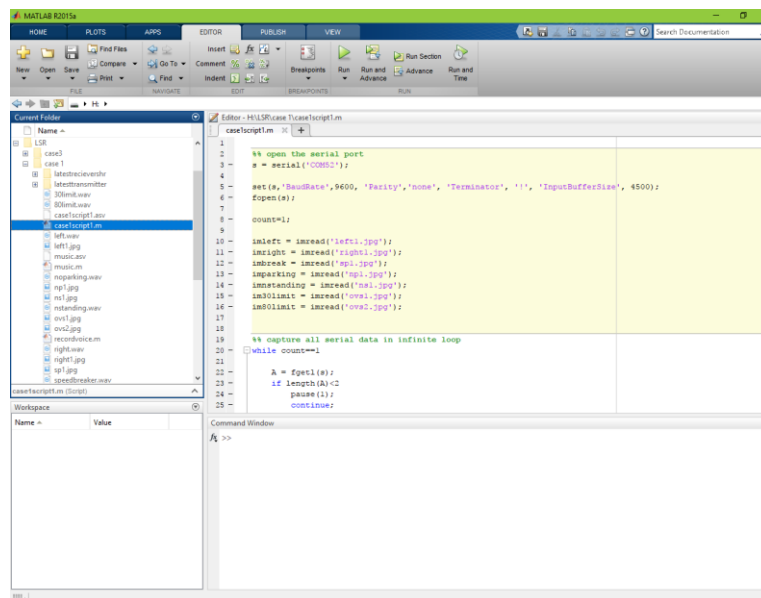


Figure 8.9. MATLAB R2015a GUI

## Chapter 10

# SYSTEM ANALYSIS

Analysis is the process of finding the best solution to the problem. System analysis is the process by which we learn about the existing problems, define objects and requirements and evaluates the solutions. It is the way of thinking about the organization and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis which gives the target for design and development.

### 10.1 Feasibility Study

Depending on the results of the initial investigation the survey is now expanded to a more detailed feasibility study. “FEASIBILITY STUDY” is a test of system proposal according to its workability, impact of the organization, ability to meet needs and effective use of the resources.

Eight steps involved in the feasibility analysis are:

- ✓Form a project team and appoint a project leader.
- ✓Enumerate potential proposed system.
- ✓Define and identify characteristics of proposed system.
- ✓Determine and evaluate performance and cost effective of each proposed system.
- ✓Weight system performance and cost data.
- ✓Select the best proposed system.
- ✓Prepare and report final project directive to management.

Three key considerations involved in the feasibility analysis are

- ✓ECONOMICAL FEASIBILITY
- ✓TECHNICAL FEASIBILITY

## ✓SOCIAL FEASIBILITY

**10.1.1 Economic Feasibility**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Component	Quantity(Nos.)	Cost (₹.)
Arduino Microcontroller Board	2	800
RF Transmitter	1	150
RF Receiver	1	200
Ultrasonic Sensors	1	200
DC power supply	1	150
DC Motor	1	200
H Bridge	1	230
Wiring	-	50
Buzzer	1	05
LEDs and Resistors	5 each	15
MISC	-	500
	Total	₹ 2500/-

**Table 8.1. Expenditure****10.1.2 Technical Feasibility**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This

will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system. The system uses platforms – Arduino and MATLAB which are open source platforms and also available at no cost to the users.

### **10.1.3 Social Feasibility**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system. The system uses a frequency band of 433MHz. As the RF transmission and reception is a continuously process, the user and environment are exposed to this band. The system is accepted as there is no threat to the users and the environment as they spectrum is safe and causes no harm to them.

## **Summary**

The main aim of this chapter is to find out whether the system is feasible enough or not. For these reasons different kinds of analysis, such as performance analysis, technical analysis, economic analysis etc. is performed.

## Chapter 11

# SYSTEM DESIGN

Design is a creative process; a good design is the key to effective system. The system “Design” is defined as “The process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. Various design features are followed to develop the system. The design specification describes the features of the system, the components or elements of the system and their appearance to end-users.

### 11.1 Fundamental Design Concepts

A set of fundamental design concepts has evolved over the past three decades. Although the degree of interest in each concept has varied over the years, each has stood the test of time. Each provides the software designer with a foundation from which more sophisticated design methods can be applied. The fundamental design concepts provide the necessary framework for “getting it right”. The fundamental design concepts such as abstraction, refinement, modularity, software architecture, control hierarchy, structural partitioning, data structure, software procedure and information hiding are applied in this project to getting it right as per the specification.

#### 11.1.1 Input Design

The input Design is the process of converting the user-oriented inputs in to the computer-based format. The goal of designing input data is to make the automation as easy and free from errors as possible. Providing a good input design for the application easy data input and selection features are adopted. The input design requirements such as user friendliness, consistent format and interactive dialogue for giving the right message and help for the user at right time are also considered for the development of the project. Input design is a part of overall system design which requires very careful attention. Often the collection of input data is the most expensive part of the system, which needs to be route through number of modules .It is the

point where the user ready to send the data to the destination machine along with known IP address; if the IP address is unknown then it may prone to error.

### 11.1.2 Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other systems through outputs. It is most important and direct source information to the user. Efficient and intelligent output improves the systems relationship with source and destination machine. Outputs from computers are required primarily to get same packet that the user has send instead of corrupted packet and spoofed packets. They are also used to provide to permanent copy of these results for later consultation.

### 11.1.3 The MVC Design Method

Swing actually makes use of a simplified variant of the MVC design called the model-delegate. This design combines the view and the controller object into a single element that draws the component to the screen and handles GUI events known as the UI delegate. Communication between the model and the UI delegate becomes a two-way street. Each Swing component contains a model and a UI delegate. The model is responsible for maintaining information about the component's state. The UI delegate is responsible for maintaining information about how to draw the component on the screen. The UI delegate (in conjunction with AWT) reacts to various events that propagate through the component.

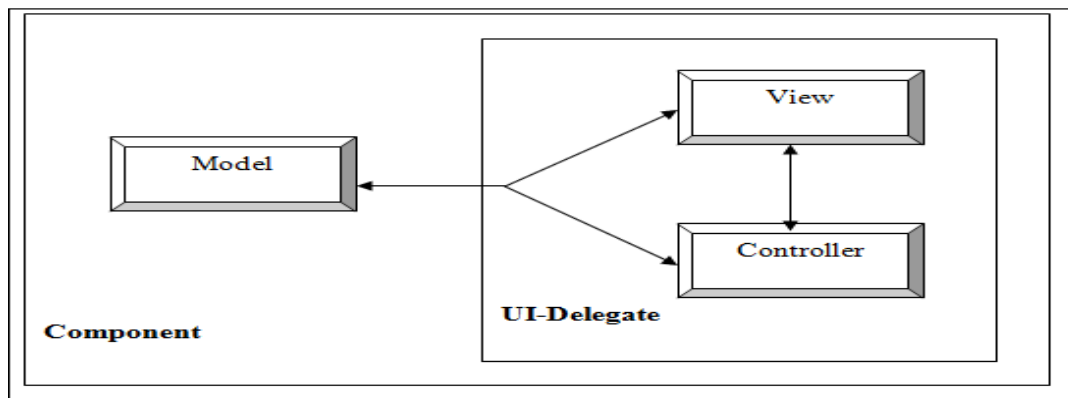


Figure 11.1. Combination of View & Controller into a UI delegate object

The design method that has been followed to design the architecture of the system is MVC design pattern. Swing uses the model-view-controller (MVC) architecture as the fundamental design behind each of its components. Essentially, MVC breaks GUI component into three elements. Each of these elements plays a crucial role in how the component behaves. The MVC design pattern separates a software component into three distinct pieces: a model, a view, and a controller.

## Model

The *model* is the piece that represents the state and low-level behavior of the component. It manages the state and conducts all transformations on that state. The model has no specific knowledge of either its controllers or its views. It encompasses the state data for each component. There are different models for different types of components. For example, the model of a scrollbar component might contain information about its current position of its adjustable “thumb”, its minimum and maximum values, and the thumb’s width. A menu on the other hand, may simply contain a list of the menu items the user can select from. The system itself maintains links between model and views and notifies the views when the model changes state.

## View

The view refers to how you see the component in the screen. It is the piece that manages the visual display of the state represented by the model. Almost all window frames will have a title bar spanning the top of the window. However the title bar may have a close box on the left side or on the right side. These are the examples of different types of views for the same window object. A model can have more than one view, but that is typically not the case in this.

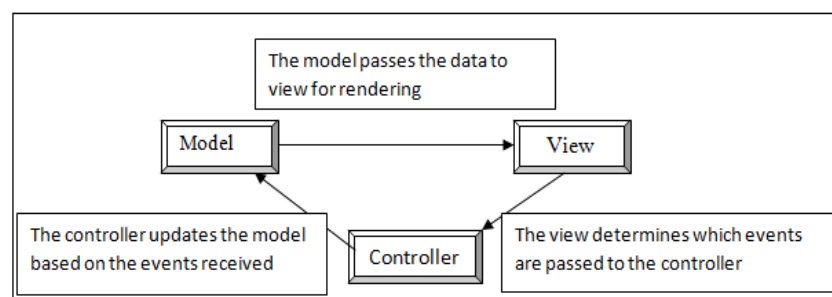


Figure 11.2. Communication through the MVC Architecture

## Controller

The *controller* is the piece that manages user interaction with the model. It provides the mechanism by which changes are made to the state of the model. It is the portion of the user interface that dictates how the component interacts with events.

The view cannot render the scrollbar correctly without obtaining information from the model first. In this case the scrollbar will not know where to draw its “thumb” unless it can obtain its current position and width relative to the minimum and maximum. Likewise the view determines if the component is the recipient of user events, such as mouse clicks. The view passes these events on to the controller, which decides how to handle them best. Based on the controller’s decision the values in the model may need to be altered. If the user drags the scrollbar thumb, the controller will react by incrementing the thumb’s position in the model. At that point the whole cycle can repeat.

The JFC user interface component can be broken down into a model, view, and controller. The view and controller are combined into one piece, a common adaptation of the basic MVC pattern. They form the user interface for the component.

## 11.2 System Development Methodology

System development method is a process through which a product will get completed or a product gets rid from any problem. Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps which is used for product progress. The development method followed in this project is waterfall model.

### 11.2.1 Model phases

` The waterfall model is a sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Requirement initiation, Analysis, Design, Implementation, Testing and maintenance.



**Requirement Analysis:** This phase is concerned about collection of requirement of the system. This process involves generating document and requirement review.

**System Design:** Keeping the requirements in mind the system specifications are translated in to a software representation. In this phase the designer emphasizes on:-algorithm, data structure, software architecture etc.

**Coding:** In this phase programmer starts his coding in order to give a full sketch of product. In other words system specifications are only converted in to machine readable compute code.

**Implementation:** The implementation phase involves the actual coding or programming of the software. The output of this phase is typically the library, executables, user manuals and additional software documentation

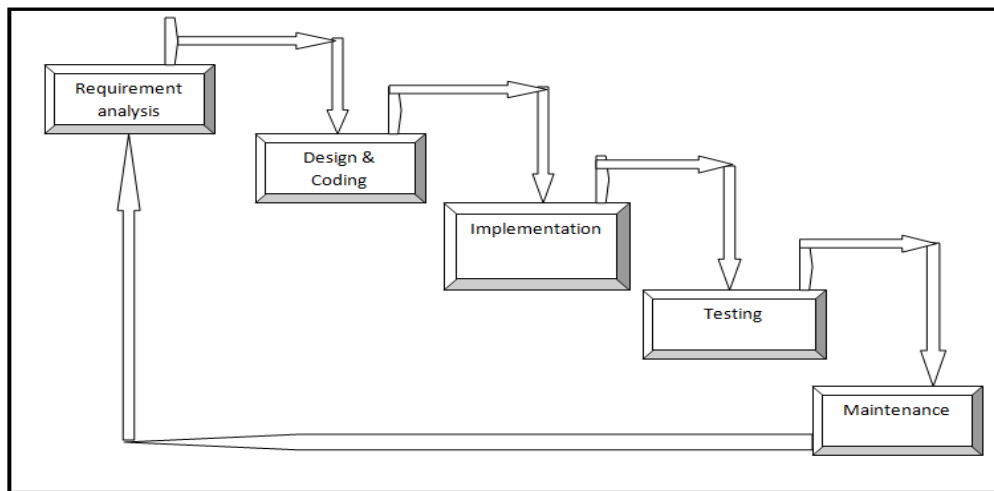
**Testing:** In this phase all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation.

**Maintenance:** The maintenance phase is the longest phase in which the software is updated to fulfill the changing customer need, adapt to accommodate change in the external environment, correct errors and oversights previously undetected in the testing phase, enhance the efficiency of the software.

### 11.3 Reason for choosing Waterfall Model as Development Method

- Clear project objectives.
- Stable project requirements.
- Progress of system is measurable.
- Strict sign-off requirements.
- Helps you to be perfect.
- Logic of software development is clearly understood.

- Production of a formal specification
- Better resource allocation.
- Improves quality. The emphasis on requirements and design before writing a single line of code ensures minimal wastage of time and effort and reduces the risk of schedule slippage.
- Less human resources required as once one phase is finished those people can start working on to the next phase.

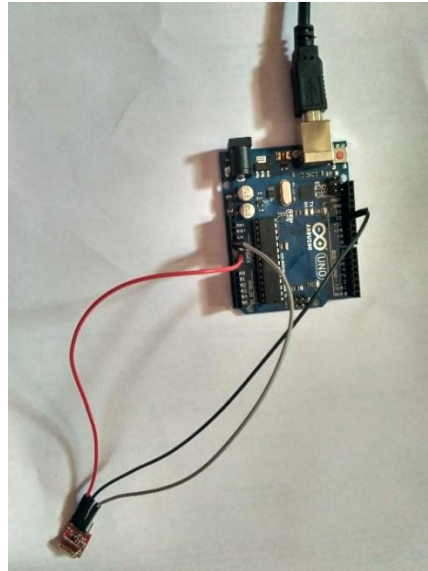


**Figure 11.3. Waterfall model**

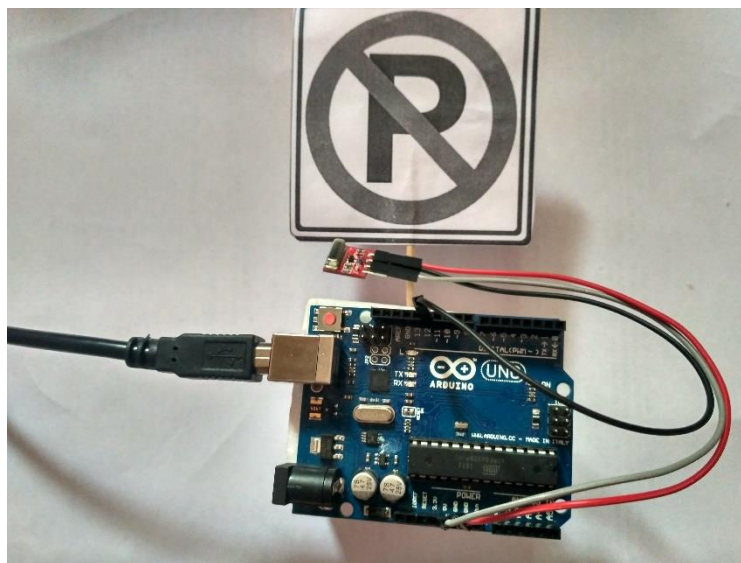
## Chapter 12

# RESULTS

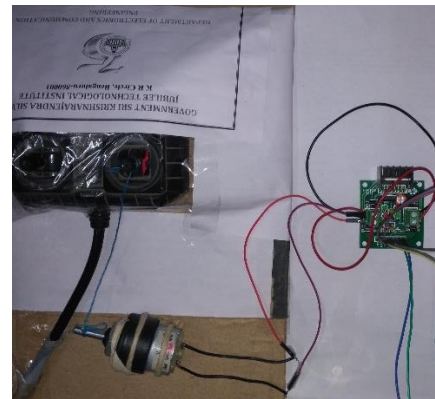
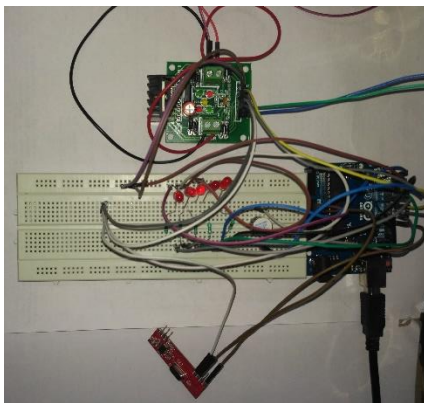
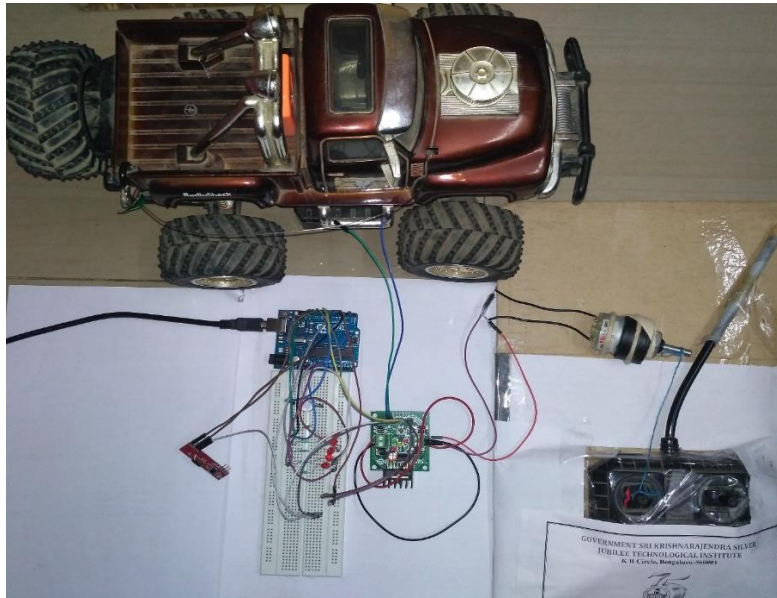
The hardware was built and testing of each and every component was done, for its correct functionality. All the hardware components are interfaced as per design and integrated with software to serve the purpose.



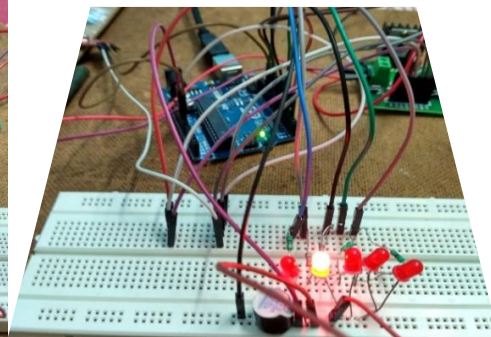
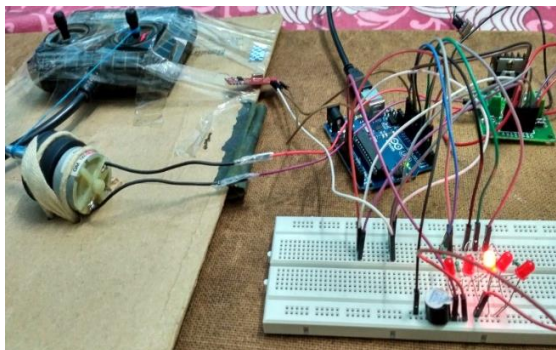
**Figure 12.1. Transmitter Circuit of CASE 1 and CASE 2**



**Figure 12.2. Transmitter sends the data in form radio signal of the associated traffic sign and speed limit warnings**



**Figure 12.3. Receiver End Circuit and Control Unit of CASE 1 & CASE 2**



**Figure 12.4. Output on LEDs when data is successfully received by the receiver along with a buzzer alert and is sent to alert the driver by display message and audio alert**



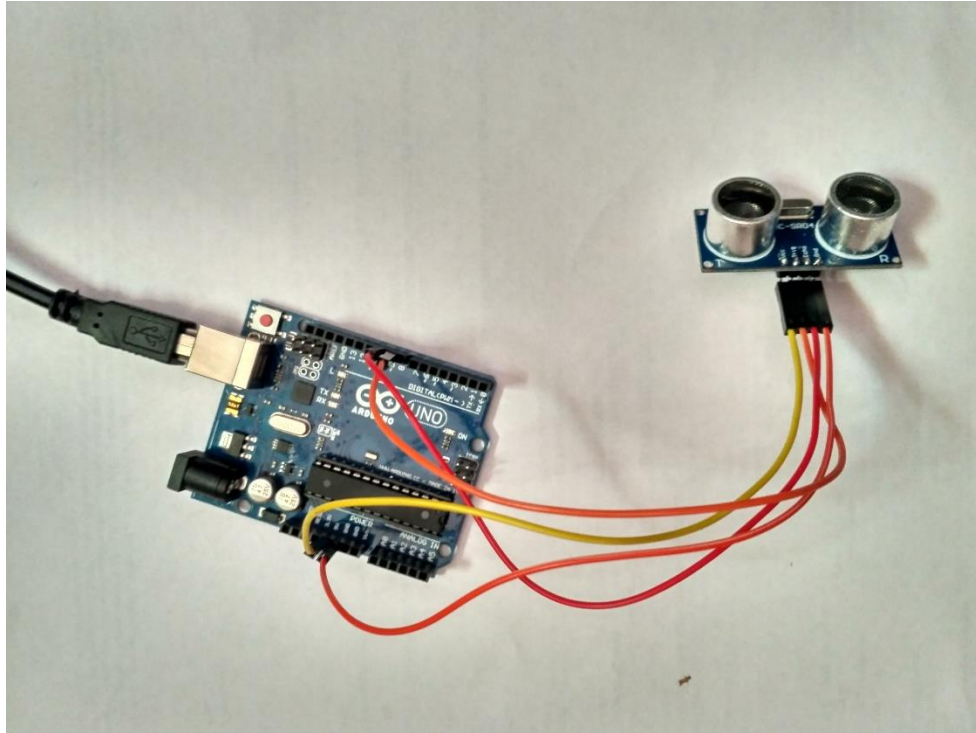


Figure 12.5. CASE 3 Overtaking Assistance Circuit – HC-SR04 is interfaced to the arduino board



Figure 12.6. Ultrasonic sensor is placed at the front of vehicle to obtain the distance from the front running vehicle or obstacle present and sends the data to arduino

## CASE 1: TRAFFIC SIGN RECOGNITION

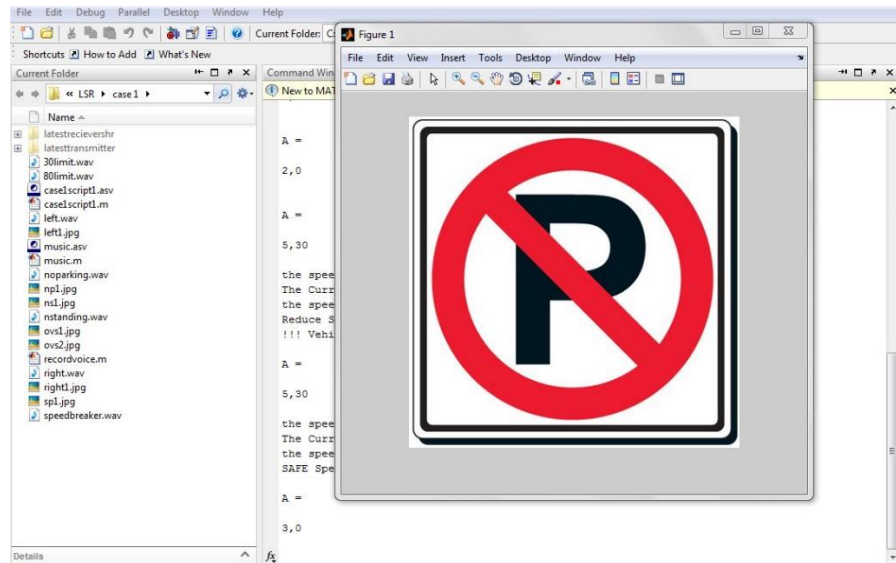


Figure 12.7. Traffic Sign Sample 1: “NO Parking” received and displayed to driver

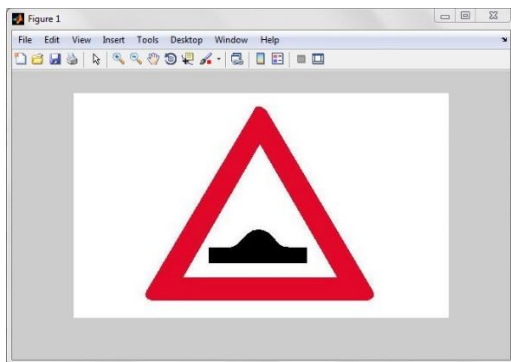


Figure 12.8.a. Sample 2: “SPEED Breaker”

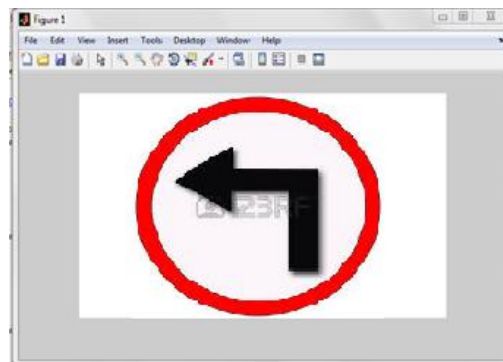


Figure 12.8.b. Sample 3: “LEFT Turn Ahead”

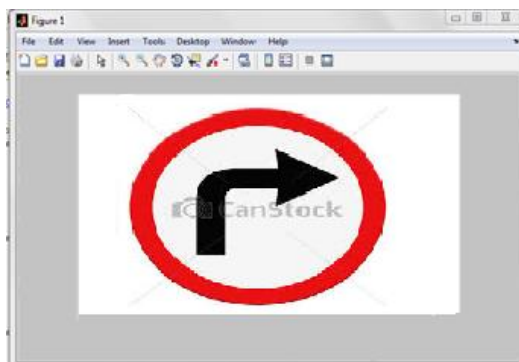


Figure 12.8.c. Sample 4: “RIGHT Turn Ahead”

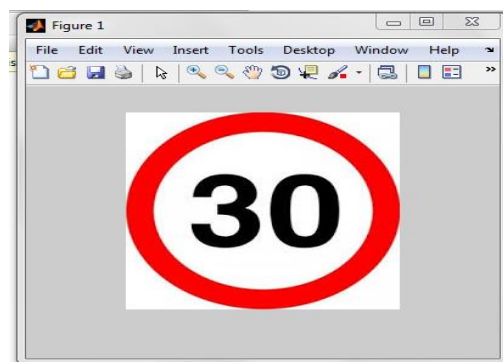


Figure 12.8.d. Sample 5: “SPEED LIMIT  
30km/hr.”

Figure 12.8. Other Traffic Signs received and displayed with visual alert to driver

## CASE 2: INTELLIGENT SPEED ADAPTATION

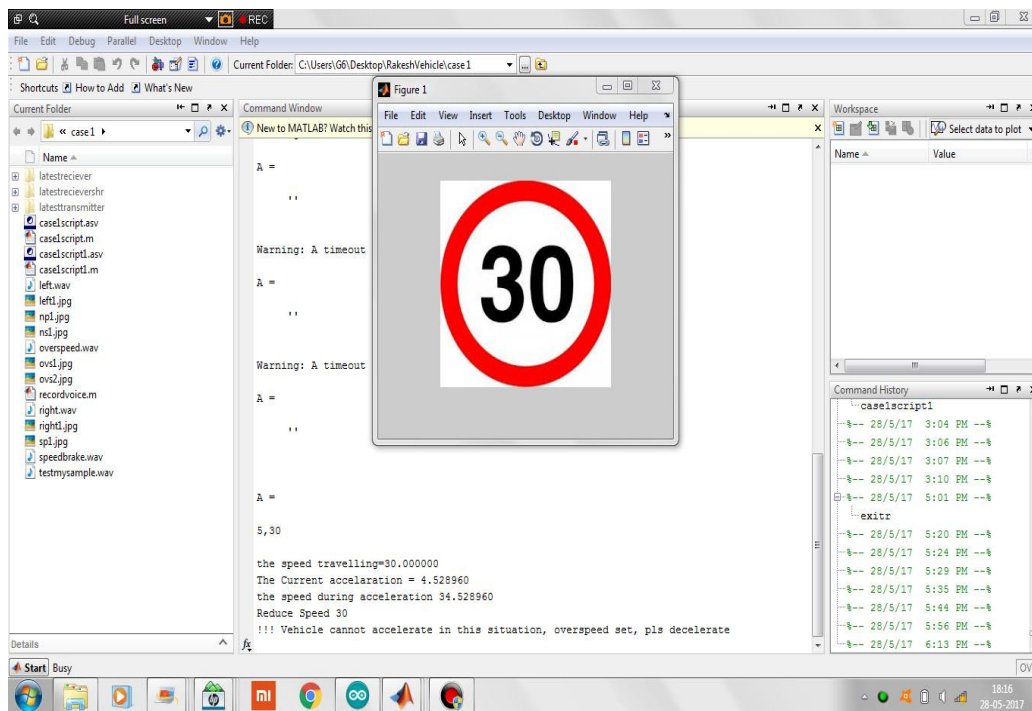


Figure 12.9. Speed Limit and Action message displaying to driver

The picture shows the case 2 output; the speed limit is displayed with visual alert. The driver is informed about speed of the vehicle, current acceleration and action to be taken. After a certain time lag, control unit is enabled to limit the speed to the certain speed.

### TEST CASE 1: DRIVING WITHIN LIMIT

```
the speed travelling=25.000000
The Current acceleration = -1.661150
the speed during acceleration 23.338850
SAFE Speed 30
```

### TEST CASE 2: DRIVING OVER THE LIMIT

```
the speed travelling=30.000000
The Current acceleration = 4.528960
the speed during acceleration 34.528960
Reduce Speed 30
!!! Vehicle cannot accelerate in this situation, overspeed set, pls decelerate
```



Figure 12.10. Case 2: Speed limit and Control message

## CASE 3: OVERTAKING ASSISTANCE

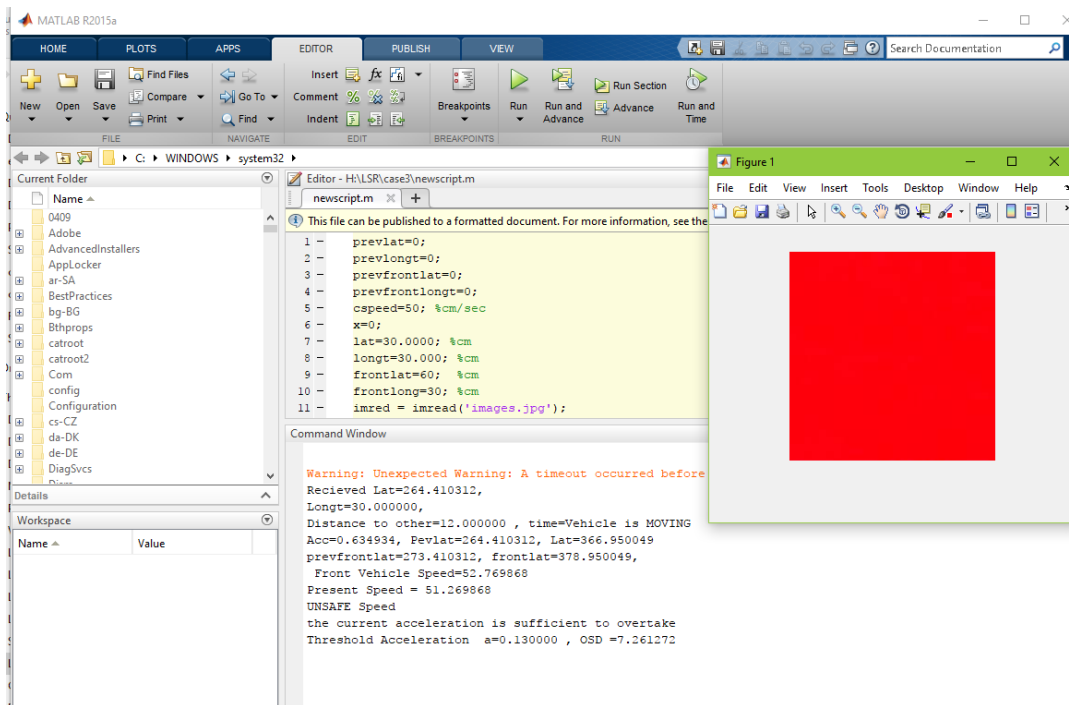


Figure 12.11. Output on monitor displaying that it is “NOT SAFE” to overtake

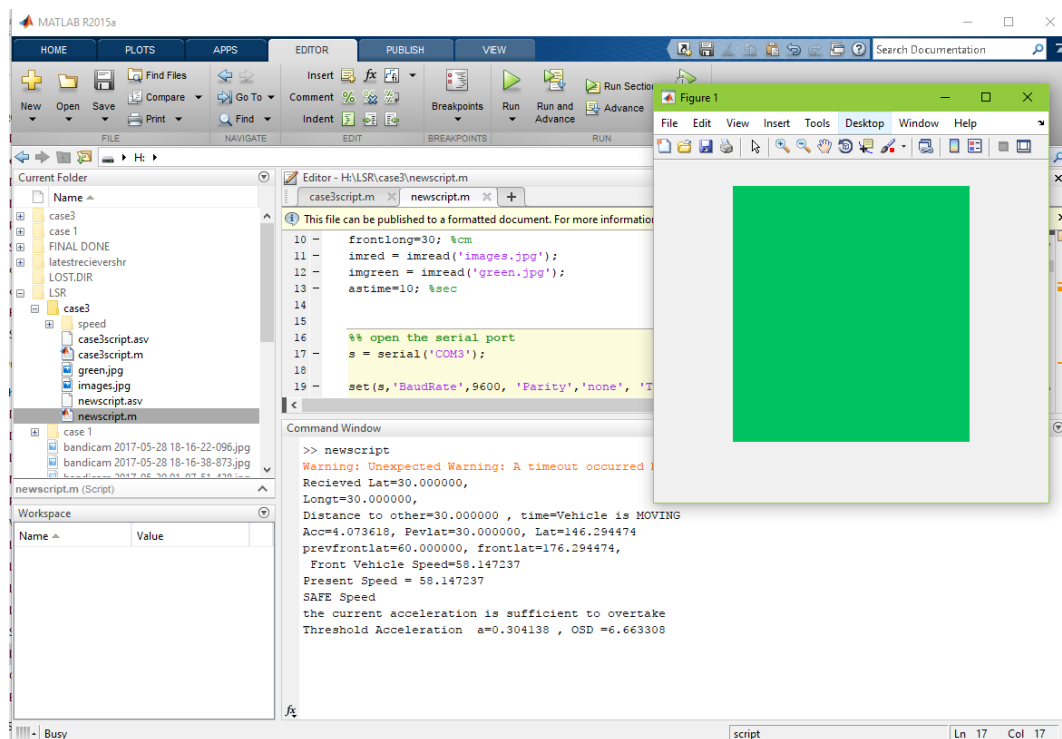


Figure 12.12. Output on monitor displaying that it is “SAFE” to overtake



## Chapter 13

# CONCLUSION

**Traffic Sign** give information about the road conditions ahead, provide instructions to be followed at the major crossroads or junctions; warn or guide drivers, and ensure proper functioning of road traffic. **Speed Limits** are used in most countries to inform the maximum (or minimum in some cases) speed at which road vehicles may legally travel on particular stretches of road. Speed limits may be variable. Speed limits are normally indicated on a traffic sign. These are commonly set by the legislative bodies of nations or provincial governments and enforced by national or regional police and/or judicial bodies. **Overtaking** is the act of one vehicle going past another slower moving vehicle, travelling in the same direction, on a road. The lane used for overtaking another vehicle is almost always a passing lane further from the road shoulder which is to the left in places that drive on the right and to the right in places that drive on the left

- We have proposed this system to provide the driver assistance with the traffic sign, speed limit, speed, distance and every relative movement of the vehicle.
- The System is implemented to help while driving the vehicle on the road so as to avoid the damage of the vehicle; avoid over speeding at various zones; wrong parking or prevent violations; and to prevent premature driving while overtaking.

We have presented traffic sign detection methods which are often used as building blocks of complex detection systems. We think that the complexity of traffic sign detection systems will diminish in the future, as technology advances. With the advancement of technology, high quality transmitter and sensors will become cheaper and more available in mass production. If in the future every car is equipped with intelligent system as proposed by our project, a GPS receiver and an odometer, ultrasonic sensors and other sensors, the problem of traffic sign detection will be infinitely more simple than it is now. However, the advancement will probably proceed slowly, because of the persistent need to minimize production costs.

The system has an RF transmitter and receiver interfaced with arduino which indicates the vehicle when it passes or nears a traffic sign/warning or enters speed limit zone. Hence by using the accelerometer to monitor the speed of the vehicle, and accelerator unit to control the speed; the speed of the vehicle can be maintained in the limited speed without the intervention of the driver. This is also known as electromechanical system. If this can be implemented effectively, rash driving and over speeding in the speed limit zones can be reduced to a large extent, thus decreasing the total number of road accidents in our country.

The system is capable to diminish the risk of overtaking maneuvers in single lanes, which are particularly dangerous according to the literature. The system under consideration claims to have certain originality at several fronts:

- The sensing onboard unit consists of only vehicular navigation sensors, such as a GNSS receiver, an accelerometer and one gyro.
- Outputs of the sensors are fused (not only matched) with a digital map.
- Cellular networks are exploited as a means to share data of interest for avoiding an inter-vehicular collision.

### **13.1 Advantages**

- ✓ Low cost and one time investment.
- ✓ Less complexity and provides wide range applicability.
- ✓ Circuit is flexible and can be implemented easily.
- ✓ By the using of this system we can control the speed very easily.
- ✓ Easy traffic regulation in busy cities such as Bengaluru, Mumbai, Delhi etc.
- ✓ Power requirements are minimum 5V and max 12V; hence can run on batteries/power banks; thus limiting the reliance on electrical sources for operation.

### **13.2 Disadvantages**

- ✗ This system can operate certain distance only.
- ✗ Sensors and RF components failure can cause problems.
- ✗ Battery/power supply need to be checked periodically.

### **13.3 Applications**

- This system can be used to avoid accidents.
- Transportation applications.

## REFERENCES

- [1] Road sign recognition system based on GentleBoost with sharing features, Jin-Yi-Wu; Chien-Chung Tseng; Chun-Hao Chang; Jenn-Jier James Lien; Ju Chin Chen; Ching Ting Tu, Proceedings 2011 International Conference on System Science and Engineering.
- [2] Y. GU, T. Yendo, M. Tehrani, T. Fujii and M.Tanimoto. 2011. "Traffic sign detection in dual-focal active camera system", International Conference on Intelligent Vehicle Symposium, IEEE.
- [3] Automatic Detection and Classification of Traffic Signs Carlos Filipe Paulo; Paulo Lobato Correia, Image Analysis for Multimedia Interactive Services, 2007. WIAMIS '07. Eighth International Workshop on, Year: 2007.
- [4] Robust traffic signs detection by means of vision and V2I communications, M. A. García-Garrido; M. Ocaña; D. F. Llorca; M. A. Sotelo; E. Arroyo; A. Llamazares 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), Year: 2011,
- [5] "Design of traffic sign detection, recognition, and transmission systems for smart vehicles",- Abdelhamid Mammeri; Azzedine Boukerche; Mohammed Almulla, IEEE Wireless Communications, Year: 2013, Volume: 20.
- [6] Yoshimichi, S., and M. Koji. "Development and evaluation of in-vehicle signing system utilizing RFID tags as digital traffic signs." International Journal of ITS Research 4.1 (2006).
- [7] An Efficient RealTime Traffic Sign Recognition System for Intelligent Vehicle With Smart Phones, Ching-Hao Lai; Chia-Chen Yu, 2010 International Conference on Technologies and Applications of Artificial Intelligence, Year: 2010.
- [8] Pérez, Joshué, et al. "An RFID-based intelligent vehicle speed controller using active traffic signals." Year (2010).
- [9] <http://www.tutorialspoint.com/arduino/>
- [10] <http://www.mapsofindia.com/my-india/government/traffic-signs-and-road-safety>
- [11] National Crime Records Bureau, Ministry of Road Transport & Highway, Law commission of India, Global status report on road safety 2016
- [12] <https://www.create.arduino.cc/projecthub/>

## APPENDIX A

### A.1 ARDUINO CODE FOR TRANSMITTER END – CASE 1& 2

```
#include <VirtualWire.h>

const int led_pin = 11;
const int transmit_pin = 12;
const int receive_pin = 2;
const int transmit_en_pin = 3;

void setup()
{
    Serial.begin(9600);
    // Initialise the IO and ISR
    vw_set_tx_pin(transmit_pin);
    vw_set_rx_pin(receive_pin);
    vw_set_ptt_pin(transmit_en_pin);
    vw_set_ptt_inverted(true); // Required for DR3100
    vw_setup(2000); // Bits per sec
}

byte count = 1;

void loop()
{
    char msg[7] = {60};

    Serial.print(msg);
    Serial.print("\n");

    if (Serial.available() > 0)
        switch(Serial.read())
        {
            case 'L':
                msg[0] = 65;
                Serial.print(msg[0]);
            break;
        }
    }
}
```

```
        Serial.print("\n");  
    break;  
    case 'R':  
        msg[0]=66;  
        Serial.print(msg[0]);  
        Serial.print("\n");  
    break;  
    case 'B':  
        msg[0]=67;  
        Serial.print(msg[0]);  
        Serial.print("\n");  
    break;  
    case 'P':  
        msg[0]=68;  
        Serial.print(msg[0]);  
        Serial.print("\n");  
    break;  
    case 'S':  
        msg[0]=69;  
        Serial.print(msg[0]);  
        Serial.print("\n");  
    break;  
    case 'O':  
        msg[0]=30;  
        Serial.print(msg[0]);  
        Serial.print("\n");  
    break;  
    case 'N':  
        msg[0]=80;  
        Serial.print(msg[0]);
```

```
        Serial.print("\n");

        break;

    }

    msg[6] = count;

    digitalWrite(led_pin, HIGH); // Flash a light to show transmitting

    vw_send((uint8_t *)msg, 7);

    vw_wait_tx(); // Wait until the whole message is gone

    digitalWrite(led_pin, LOW);

    delay(5000);

    count = count + 1;

}
```

## **A.2 AURDINO CODE FOR RECIEVER END – CASE 1& 2**

```
#include <VirtualWire.h>

int buzz_pin = 5;

const int break_pin = 7;

const int lturn_pin = 8;

const int rreak_pin = 9;

const int parking_pin = 6;

const int nstanding_pin = 13;

const int overalertpin = 10;


const int transmit_pin = 12;

const int receive_pin = 11;

const int transmit_en_pin = 3;


int speedv=40;

int times=0;


void setup()

{
```

```
    delay(1000);

    Serial.begin(9600); // Debugging only
    //Serial.println("setup");

    pinMode(break_pin, OUTPUT);
    pinMode(lturn_pin, OUTPUT);
    pinMode(rreak_pin, OUTPUT);
    pinMode(parking_pin, OUTPUT);
    pinMode(nstanding_pin, OUTPUT);
    pinMode(overalertpin, OUTPUT);
    pinMode(buzz_pin, OUTPUT);

    // Initialise the IO and ISR
    vw_set_tx_pin(transmit_pin);
    vw_set_rx_pin(receive_pin);
    vw_set_ptt_pin(transmit_en_pin);
    vw_set_ptt_inverted(true); // Required for DR3100
    vw_setup(2000); // Bits per sec

    vw_rx_start(); // Start the receiver PLL running
}

void loop()
{
    uint8_t buf[VW_MAX_MESSAGE_LEN];
    uint8_t buflen = VW_MAX_MESSAGE_LEN;

    if (vw_get_message(buf, &buflen)) // Non-blocking
    {
        int i;

        // Message with a good checksum received, print it.
        //Serial.print("Got: ");

        /*
        for (i = 0; i < buflen; i++)
```

```
{  
    Serial.print(buf[i]);  
    Serial.print(' ');  
}  
  
Serial.println();  
*/  
//Serial.println(buf[0]);  
if (buf[0]==60)  
{  
  
}  
else if (buf[0]==65)  
{  
    Serial.print(0);  
    Serial.print(',');  
    Serial.print(0);  
    Serial.print('!');  
    //Left turn  
    digitalWrite(lturn_pin, HIGH);  
    tone(buzz_pin,1000,500);  
    times=times+1;  
}  
else if (buf[0]==66)  
{  
    // right pin  
    Serial.print(1);  
    Serial.print(',');  
    Serial.print(0);  
    Serial.print('!');
```



```
        //Right turn
        digitalWrite(rreak_pin, HIGH);
        tone(buzz_pin,1000,500);
        times=times+1;
    }
    else if (buf[0]==67)
    {
        // breaking
        Serial.print(2);
        Serial.print(',');
        Serial.print(0);
        Serial.print('!');
        //Speed Breaker
        digitalWrite(break_pin, HIGH);
        tone(buzz_pin,1000,500);
        times=times+1;
    }
    else if (buf[0]==68)
    {
        Serial.print(3);
        Serial.print(',');
        Serial.print(0);
        Serial.print('!');
        //No Parking
        digitalWrite(parking_pin, HIGH);
        delay(1000);
        digitalWrite(parking_pin, LOW);
        tone(buzz_pin,1000,500);
        times=times+1;
    }
}
```

```
else if (buf[0]==69)
{
    Serial.print(4);
    Serial.print(',');
    Serial.print(0);
    Serial.print('!');
    //No Parking No Standing
    digitalWrite(nstanding_pin, HIGH);
    delay(1000);
    digitalWrite(nstanding_pin, LOW);
    tone(buzz_pin,1000,500);
    times=times+1;
}
else if (buf[0]==30)
{
    digitalWrite(overalertpin, HIGH);
    // over speed case
    Serial.print(5);
    Serial.print(',');
    Serial.print(30);
    Serial.print('!');
    tone(buzz_pin,1000,500);
}
else if (buf[0]==80)
{
    digitalWrite(overalertpin, LOW);
    // not over speeding
    Serial.print(6);
    Serial.print(',');
    Serial.print(80);
```

```
        Serial.print('!');

        tone(buzz_pin,1000,500);

    }

}

if (times>=15)

{

    Serial.print(3);

    Serial.print(',');

    Serial.print(0);

    Serial.print('!');

}

delay(3000);

digitalWrite(break_pin, LOW);

digitalWrite(lturn_pin, LOW);

digitalWrite(rreak_pin, LOW);

digitalWrite(overalertpin, LOW);

}
```

### A.3 MATLAB CODE FOR DISPLAYING RESULTS AT RECIEVER END

```
%% open the serial port

s = serial('COM51');

set(s,'BaudRate',9600, 'Parity','none', 'Terminator', '!', 'InputBufferSize',
4500);
fopen(s);
count=1;

imleft = imread('left1.jpg');
imright = imread('right1.jpg');
imbreak = imread('sp1.jpg');
imparking = imread('np1.jpg');
imnstanding = imread('ns1.jpg');
im30limit = imread('ovs1.jpg');
im80limit = imread('ovs2.jpg');

%% capture all serial data in infinite loop
while count==1
```

```
A = fgetl(s);
if length(A)<2
    pause(1);
    continue;
end
A=strrep(A, '!', '');
res = regexp(A, ',', 'split');
tsign=str2num(char(res(1)));
tspeed=str2num(char(res(2)));

if tsign==0
    [x,Fs,nbits]= wavread('left.wav');
    wavplay(x,Fs);
    figure
    imshow(imleft);

end

if tsign==1
    [x,Fs,nbits]= wavread('right.wav');
    wavplay(x,Fs);
    imshow(imright);
end

if tsign==2

    [x,Fs,nbits]= wavread('speedbreaker.wav');
    wavplay(x,Fs);
    imshow(imbreak);
end

if tsign==3

    [x,Fs,nbits]= wavread('noparking.wav');
    wavplay(x,Fs);
    imshow(imparking);
end

if tsign==4

    [x,Fs,nbits]= wavread('nstanding.wav');
    wavplay(x,Fs);
    imshow(imnstanding);
end

if tsign==5

    [x,Fs,nbits]= wavread('30limit.wav');
    wavplay(x,Fs);
    imshow(im30limit);
    speed=25;
    fprintf('the speed travelling=%f \n', speed);
    accr=randn*10;
```

```
if(accr>10)
    accr=accr-20;
    fprintf(1,'The Current accelaration = %f \n', accr);
else
    fprintf(1,'The Current accelaration = %f \n', accr);
end
sspeed=speed+accr;
fprintf(1,'the speed during acceleration %f \n', sspeed);
if (tspeed>sspeed)
    fprintf(1,'SAFE Speed %d \n', tspeed);
else
    fprintf(1,'Reduce Speed %d \n', tspeed);

    if(accr>0)
        fprintf(1,'!!! Vehicle cannot accelerate in this situation,
overspeed set, pls decelerate \n');
    else
        fprintf(1,'!!! Vehicle is decelerating \n');
    end
    %[x, Fs, nbits]= wavread('overspeed.wav');
end

end

if tsign==6

    [x,Fs,nbits]= wavread('80limit.wav');
    wavplay(x,Fs);
    imshow(im80limit);
    speed=60;
    fprintf('the speed travelling=%f \n',speed);
    accr=randn*20;
    if(accr>10)
        accr=accr-20;
        fprintf(1,'The Current accelaration = %f \n', accr);
    else
        fprintf(1,'The Current accelaration = %f \n', accr);
    end
    sspeed=speed+accr;
    fprintf(1,'the speed during acceleration %f \n', sspeed);

    if (tspeed>sspeed)
        fprintf(1,'SAFE Speed %d \n', tspeed);
    else
        fprintf(1,'Reduce Speed %d \n', tspeed);

        if(accr>0)
            fprintf(1,'!!! Vehicle cannot accelerate in this situation,
overspeed set, pls decelerate \n');
        else
            fprintf(1,'!!! Vehicle is decelerating \n');
        end
        %[x, Fs, nbits]= wavread('overspeed.wav');
    end
end
```

```
    if tsign==7
        fprintf(1, 'Vehicle Stopped\n');
        break;

    end

    pause(10);
end
fclose(s);
```

## A.4 ARDUINO CODE FOR OVERTAKING ASSISTANCE- CASE 3

```
#include <SoftwareSerial.h>

float bxpos=30;
float bypos=30;
int cspeed=50;
int ultrasonicdis=0;
int maxthres=35;
SoftwareSerial mySerial(7, 8);

// defines pins numbers
int trigPin = 9;
int echoPin = 10;
// defines variables
long duration;
float distance;
void setup() {
    // put your setup code here, to run once:
    mySerial.begin(9600);
    Serial.begin(9600);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT); // Sets the echoPin as an Input
    delay(100);
```

```
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
  
    //float disttravel=cspeed*0.277778*5;  
  
    //bxpos=bxpos+disttravel;  
  
    digitalWrite(trigPin, LOW); // setting trigpin to low and wait for  
    2microsec of delay  
  
    delayMicroseconds(2); // Sets the trigPin on HIGH state for 10 micro seconds  
  
    digitalWrite(trigPin, HIGH); //again setting pin to high for trigpin and  
    adding delay of 10microsec  
  
    delayMicroseconds(10);  
  
    digitalWrite(trigPin, LOW); //setting trigpin to low  
  
    // Reads the echoPin, returns the sound wave travel time in microseconds  
    duration = pulseIn(echoPin, HIGH);  
  
    // Calculating the distance  
    distance= duration*0.034/2;  
    ultrasonicdis=distance;  
  
    if ((ultrasonicdis<=0) || (ultrasonicdis>=maxthres))  
    {  
        // no need to send value  
    }  
  
    else  
    {  
        Serial.print(bxpos);  
        Serial.print(",");  
        Serial.print(bypos);  
        Serial.print(",");  
        Serial.print(ultrasonicdis);  
        Serial.print(",");  
    }  
  
    delay(2000);  
}
```

## A.5 MATLAB CODE FOR OVERTAKING ASSISTANCE- CASE 3

```
prevlat=0;
prevlongt=0;
prevfrontlat=0;
prevfrontlongt=0;

%% open the serial port
s = serial('COM4');

set(s, 'BaudRate', 9600, 'Parity', 'none', 'Terminator', '!', 'InputBufferSize',
4500);
fopen(s);

count=1;
collectioninterval=2;
a=0;
%% infinite loop
while count==1

    A = fgetl(s);
    if length(A)<2
        continue;
        pause(1);
    end
    A=strrep(A, '!', '');
    res = regexp(A, ',', 'split');
    lat=str2num(char(res(1)));
    longt=str2num(char(res(2)));
    distofront=str2num(char(res(3)));

    if (lat==0) && (longt==0)
        fprintf(1, 'Vehicle Stopped \n');
        break;
    end

    frontlat=lat+distofront;
    frontlong=longt;

    fprintf('recieved Lat=%f , Longt=%f , Distance to other=%f
\n', lat, longt, distofront);

    if ((prevlat==0) && (prevlongt==0))
        prevlat=lat;
        prevlongt=longt;

        prevfrontlat=frontlat;
        prevfrontlongt=frontlong;

    else
```



```
distance = sqrt((lat - prevlat)^2 + (longt - prevlongt)^2);
fdistance = sqrt((frontlat - prevfrontlat)^2 + (frontlong -
    prevfrontlongt)^2);
cspeed= distance*1.0/collectioninterval;
%fspeed=fdistance*1.0/collectioninterval;
fspeed=30;
fprintf('Speed of current vehicle %f , Speed of forward vehicle=%f
\n', cspeed, fspeed);
cspeed=cspeed+a/collectioninterval;
if (cspeed>100)
    cspeed=100;
end
% vb is the slowest vehicle
vb=cspeed;
if vb<fspeed
    vb=fspeed;
end
% s is distance between vehicle
sid=distofront;
% reaction time is 5 sec for the driver
t=5;
% acceleration calculation
a = (cspeed-fspeed)/collectioninterval;
osd=vb*t + 2*sid + vb* sqrt(4*sid/a);
fprintf(1, 'Acceleartion to apply a=%d , OSD =%f \n', a, osd);

end

pause(collectioninterval);
end
```