A Major Project Report

on

Movable Road Divider For Organized Vehicular Traffic Control

submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

K. V. Pavan Kumar (16211A05C8)

L. A. Radhakrishna Das (16211A05D3)

M. Nikhil Pavan Sai (16211A05D6)

Under the guidance of

Mr. P. Bhaskar Rao, Assistant Professor,
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B.V.RAJU INSTITUTE OF TECHNOLOGY

(UGC Autonomous, Accredited by NBA & NAAC)

Vishnupur, Narspur, Medak(Dist.), Telangana State, India - 502313

2016 - 2020

B. V. Raju Institute of Technology

(UGC Autonomous, Accredited By NBA & NAAC)

Vishnupur, Narsapur, Medak (Dist.),

Telangana State, India – 502313

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mini Project entitled **"MOVABLE ROAD DIVIDER FOR ORGANIZED VEHICULAR TRAFFIC CONTROL"**, being submitted by

K. V. Pavan Kumar (16211A05C8)

L. A. Radhakrishna Das (16211A05D3)

M. Nikhil Pavan Sai (16211A05D6)

In partial fulfillment of the requirements for the award of degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING to B.V.RAJU INSTITUTE OF TECHNOLOGY is a record of bonafide work carried out during a period from May 2019 to July 2019 by them under the guidance of **Mr. P. Bhaskar Rao** , Assistant Professor, CSE Department.

This is to certify that the above statement made by the students is/are correct to the best of my knowledge.

Mr. P. Bhaskar rao , Assistant Professor

The Project Viva-Voce Examination of this team has been held on

_____.

Mr. Karthik Kovuri	**Dr. Ch. Madhu Babu**	
Project Coordinator	Professor & HoD-CSE	EXTERNAL EXAMINER

B. V. Raju Institute of Technology
(UGC Autonomous, Accredited By NBA & NAAC)
Vishnupur, Narsapur, Medak (Dist.),
Telangana State, India – 502313

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project entitled **"Movable Road Divider For Organized Vehicular Traffic Control"** in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology and submitted in the Department of Computer Science and Engineering , B. V. Raju Institute of Technology, Narsapur is an authentic record of my own work carried out during a period from May 2018 to July 2019 under the guidance of **Mr. P. Bhaskar Rao**, Assistant Professor. The work presented in this project report has not been submitted by us for the award of any other degree of this or any other Institute/University.

K. V. Pavan Kumar (16211A05C8)
L. A. Radhakrishna Das (16211A05D3)
M. Nikhil Pavan Sai (16211A05D6)

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely fortunate to have got this all along the completion. Whatever we have done is due to such guidance and assistance . We would not forget to thank them.

We thank **Mr.P.Bhaskar Rao** for guiding us and providing all the support in completing this project.We are thankful to **Ms. Priyanka** , our section project coordinator for supporting us in doing this project. We are thankful to **Mr. Karthik Kovuri**, project coordinator for helping us in completing the project in time. We thank the person who has our utmost gratitude is **Dr. Ch. Madhu Babu**, Head of CSE Department.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all the staff members of CSE Department.

K. V. Pavan Kumar (16211A05C8)
L. A. Radhakrishna Das (16211A05D3)
M. Nikhil Pavan Sai (16211A05D6)

MOVABLE ROAD DIVIDER FOR ORGANIZED VEHICULAR TRAFFIC CONTROL

ABSTRACT

Road Divider is generically used for dividing the Road for ongoing and incoming traffic. This helps keep the flow of traffic; generally there is an equal number of lanes for both ongoing and incoming traffic. The problem with Static Road Dividers is that the number of lanes on either side of the road is fixed. Since the resources are limited and the population as well as number of cars per family is increasing, there is a significant increase in the number of cars on roads. This calls for better utilization of existing resources like the number of lanes available. Our aim is to formulate a mechanism of automated road divider that can shift lanes, so that we can have a number of lanes in the direction of the rush.With the smarter planet application proposed below, we will also eliminate the dependency on manual intervention and manual traffic coordination so that we can have smarter traffic all over the city. An Automated road divider can provide a solution to the above mentioned problem effectively. Here Low, Medium and High density of traffic value will be posted on IOT server as a graph.

Key Words: Ultrasonic Sensor, Traffic signals, RoadDivider, IoT Server, Automated RoadDivider

TEAM MEMBERS:

1. K. V. Pavan Kumar – 16211A05C8
2. L. A. Radhakrishna Das – 16211A05D3
3. M. Nikhil Pavan Sai – 16211A05D6

GUIDE:

Mr. P. Bhaskar Rao - Signature

CONTENTS

LIST OF FIGURES

Chapter-1

INTRODUCTION

In any city, there is an industrial area or shopping area where the traffic generally flows in one direction in the morning or evening. The other side of Road divider is mostly either empty or very under utilized. This is true for peak morning and evening hours. This results in loss of time for the car owners, traffic jams as well as under utilization of available resources. An Automated road divider can provide a solution to the above mentioned problem effectively. Here Low, Medium and High density of traffic value will be posted on IOT server as a graph. The domain is Embedded Systems. The controller used is ATMEGA-328 and the crystal is 16MHz. We also used DC motors and UltraSonic Sensors in our project.

1.1 Motivation

Road Divider is generically used for dividing the Road for ongoing and incoming traffic. This helps keep the flow of traffic; generally there is an equal number of lanes for both ongoing and incoming traffic. The problem with Static Road Dividers is that the number of lanes on either side of the road is fixed. Since the resources are limited and the population as well as number of cars per family is increasing, there is a significant increase in the number of cars on roads. This calls for better utilization of existing resources like the number of lanes available. Our aim is to formulate a mechanism of automated road divider that can shift lanes, so that we can have a number of lanes in the direction of the rush.With the smarter planet application proposed below, we will also eliminate the dependency on manual intervention and manual traffic coordination so that we can have smarter traffic all over the city. An Automated road divider can provide a solution to the above mentioned problem effectively.

Here Low, Medium and High density of traffic value will be posted on IOT server as a graph.

1.2 Problem Definition

Nearly 99 per cent of the processors manufactured end up in embedded systems. The embedded system market is one of the highest growth areas as these systems are used in every market segment- consumer electronics, office automation, industrial automation, biomedical engineering, wireless communication, data communication, telecommunications, transportation, military and so on.

1.3 Objective of the project

At home we use a number of embedded systems which include digital camera, digital diary, DVD player, electronic toys, microwave oven, remote controls for TV and air-conditioner, VCO player, video game consoles, video recorders etc. Today's high-tech car has about 20 embedded systems for transmission control, engine spark control, air-conditioning, navigation etc. Even wristwatches are now becoming embedded systems. The palmtops are powerful embedded systems using which we can carry out many general-purpose tasks such as playing games and word processing. Office automation: The office automation products using em embedded systems are copying machine, fax machine, key telephone, modem, printer, scanner etc.

1.4 Organization of Documentation:

Every embedded system consists of custom-built hardware built around a Central Processing Unit (CPU). This hardware also contains memory chips onto which the software is loaded. The software residing on the memory chip is

also called the 'firmware'. The embedded system architecture can be represented as a layered architecture as shown in Fig.

The operating system runs above the hardware, and the application software runs above the operating system. The same architecture is applicable to any computer including a desktop computer. However, there are significant differences. It is not compulsory to have an operating system in every embedded system. For small appliances such as remote control units, air conditioners, toys etc., there is no need for an operating system and you can write only the software specific to that application. For applications involving complex processing, it is advisable to have an operating system. In such a case, you need to integrate the application software with the operating system and then transfer the entire software on to the memory chip. Once the software is transferred to the memory chip, the software will continue to run for a long time. You don't need to reload new software.

Now, let us see the details of the various building blocks of the hardware of an embedded system. As shown in Fig. the building blocks are;

· Central Processing Unit (CPU)

· Memory (Read-only Memory and Random Access Memory)

· Input Devices

· Output devices

· Communication interfaces

· Application-specific circuitry

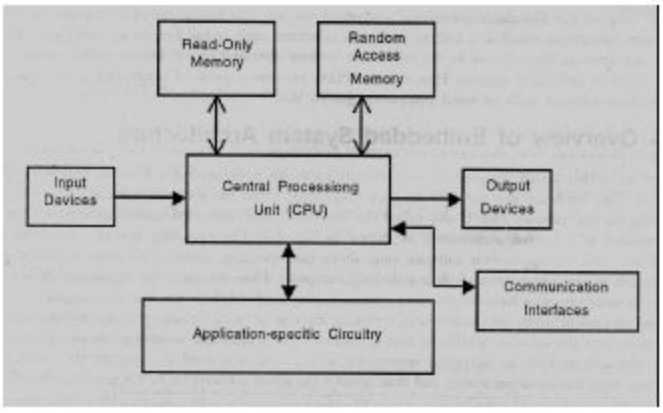


Figure 1.4.1 Image showing the components of the model

Chapter-2

LITERATURE SURVEY

2.1 Introduction

The problem with Static Road Dividers is that the number of lanes on either side of the road is fixed. Since the resources are limited and the population as well as number of cars per family is increasing, there is a significant increase in the number of cars on roads. This calls for better utilization of existing resources like the number of lanes available.This is true for peak morning and evening hours. This results in loss of time for the car owners, traffic jams as well as under utilization of available resources. An Automated road divider can provide a solution to the above mentioned problem effectively. Here Low, Medium and High density of traffic value will be posted on IOT server as a graph.

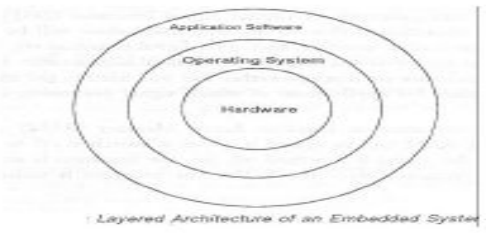
2.2 Existing System

This calls for better utilization of existing resources like the number of lanes available.This is true for peak morning and evening hours. This results in loss of time for the car owners, traffic jams as well as under utilization of available resources. An Automated road divider can provide a solution to the above mentioned problem effectively. Here Low, Medium and High density of traffic value will be posted on IOT server as a graph.

2.2.1 Framework

Every embedded system consists of custom-built hardware built around a Central Processing Unit (CPU). This hardware also contains memory chips onto which the software is loaded. The software residing on the memory chip is also

called the ‘firmware’. The embedded system architecture can be represented as a layered architecture as shown in Fig.



Layered Architecture of an Embedded Syste

The operating system runs above the hardware, and the application software runs above the operating system. The same architecture is applicable to any computer including a desktop computer. However, there are significant differences. It is not compulsory to have an operating system in every embedded system. For small appliances such as remote control units, air conditioners, toys etc., there is no need for an operating system and you can write only the software specific to that application. For applications involving complex processing, it is advisable to have an operating system. In such a case, you need to integrate the application software with the operating system and then transfer the entire software on to the memory chip. Once the software is transferred to the memory chip, the software will continue to run for a long time. You don’t need to reload new software.

2.3 Disadvantages of the Existing system

1. The problem with Static Road Dividers is that the number of lanes on either side of the road is fixed.
2. Since the resources are limited and the population as well as number of cars per family is increasing, there is a significant increase in the number of cars on roads.

3. This calls for better utilization of existing resources like the number of lanes available.

2.4 Proposed System

- Our aim is to formulate a mechanism of automated road dividers that can shift lanes, so that we can have a number of lanes in the direction of the rush.
- The cumulative impact of the time and fuel that can be saved by adding even one extra lane to the direction of the rush will be significant.

Input devices:

Unlike the desktops, the input devices to an embedded system have very limited capability. There will be no keyboard or a mouse, and hence interacting with the embedded system is no easy task. Many embedded systems will have a small keypad-you press one key to give a specific command. A keypad may be used to input only the digits. Many embedded systems used in process control do not have any input device for user interaction; they take inputs from sensors or transducers 1'fnd produce electrical signals that are in turn fed to other systems.

Output devices:

The output devices of the embedded systems also have very limited capability. Some embedded systems will have a few Light Emitting Diodes (LEDs) to indicate the health status of the system modules, or for visual indication of alarms. A small Liquid Crystal Display (LCD) may also be used to display some important parameters.

Communication interfaces:

The embedded systems may need to interact with other embedded systems as they may have to transmit data to a desktop. To facilitate this, the embedded systems are provided with one or a few communication interfaces such as RS232, RS422, RS485, Universal Serial Bus (USB), IEEE 1394, Ethernet etc.

Application-specific circuitry:

Sensors, transducers, special processing and control circuitry may be required at an embedded system, depending on its application. This circuitry interacts with the processor to carry out the necessary work. The entire hardware has to be given power supply either through the 230 volts main supply or through a battery. The hardware has to design in such a way that the power consumption is minimized.

Chapter-3

ANALYSIS

3.1 Introduction

Analysis is the process of separating or breaking up a whole into its parts in order to gain an understanding of its nature, function and interrelationship.

The above sentence describes the same thing in our present criteria. Here, we are going to perform some set of duties to come on an analysis part. In analysis, mainly how to satisfy the needs of project goals is first and foremost point. Later on, every step of the process, whatever the problem is, could be dealt with as much as possible.

3.2 Requirement Specification

According to the project job needs to be done, here are some of the following factors.

- Hardware Requirements and
- Software Requirements.

3.2.1 Hardware Requirements

The hardware requirements for the project are:

- The user should have basic idea on Arduino microcontrollers.
- The user should have a basic idea on Sonar sensors.
- To run the device there should be enough power supply to the device.

3.2.2 Software requirements

- Design Tool: Arduino IDE. (https://www.arduino.cc/en/main/software)
- Language used: Embedded C
- Technology used: Arduino UNO board, Sonar sensors.

1. **Embedded C:** Embedded C is one of the most popular and most commonly used Programming Languages in the development of Embedded Systems. So, in this article, we will see some of the Basics of Embedded C Program and the Programming Structure of Embedded C. Embedded C is perhaps the most popular languages among Embedded Programmers for programming Embedded Systems. There are many popular programming languages like Assembly, BASIC, C++ etc. that are often used for developing Embedded Systems but Embedded C remains popular due to its efficiency, less development time and portability. Before digging into the basics of Embedded C Program, we will first take a look at what an Embedded System is and the importance of Programming Language in Embedded Systems.
2. **Arduino IDE:** You can tell your Arduino what to do by writing code in the Arduino programming language and using the Arduino development environment good enough to handle the high computational load that happens when running the code.

3.3 Hardware and Technology used

3.3.1 Embedded Systems: An embedded system can be defined as a computing device that does a specific focused job. Appliances such as the air-conditioner, VCD player, DVD player, printer, fax machine, mobile phone

etc. are examples of embedded systems. Each of these appliances will have a processor and special hardware to meet the specific requirement of the application along with the embedded software that is executed by the processor for meeting that specific requirement. The embedded software is also called “firmware”. The desktop/laptop computer is a general purpose computer. You can use it for a variety of applications such as playing games, word processing, accounting, software development and so on. In contrast, the software in the embedded systems is always fixed listed below:

Embedded systems do a very specific task, they cannot be programmed to do different things. . Embedded systems have very limited resources, particularly the memory. Generally, they do not have secondary storage devices such as the CDROM or the floppy disk. Embedded systems have to work against some deadlines. A specific job has to be completed within a specific time. In some embedded systems, called real-time systems, the deadlines are stringent. Missing a deadline may cause a catastrophe-loss of life or damage to property. Embedded systems are constrained for power. As many embedded systems operate through a battery, the power consumption has to be very low. Some embedded systems have to operate in extreme environmental conditions such as very high temperatures and humidity.

3.3.2 Micro-controller:

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, 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 a AC-to-DC adapter or battery to get started.

Features:

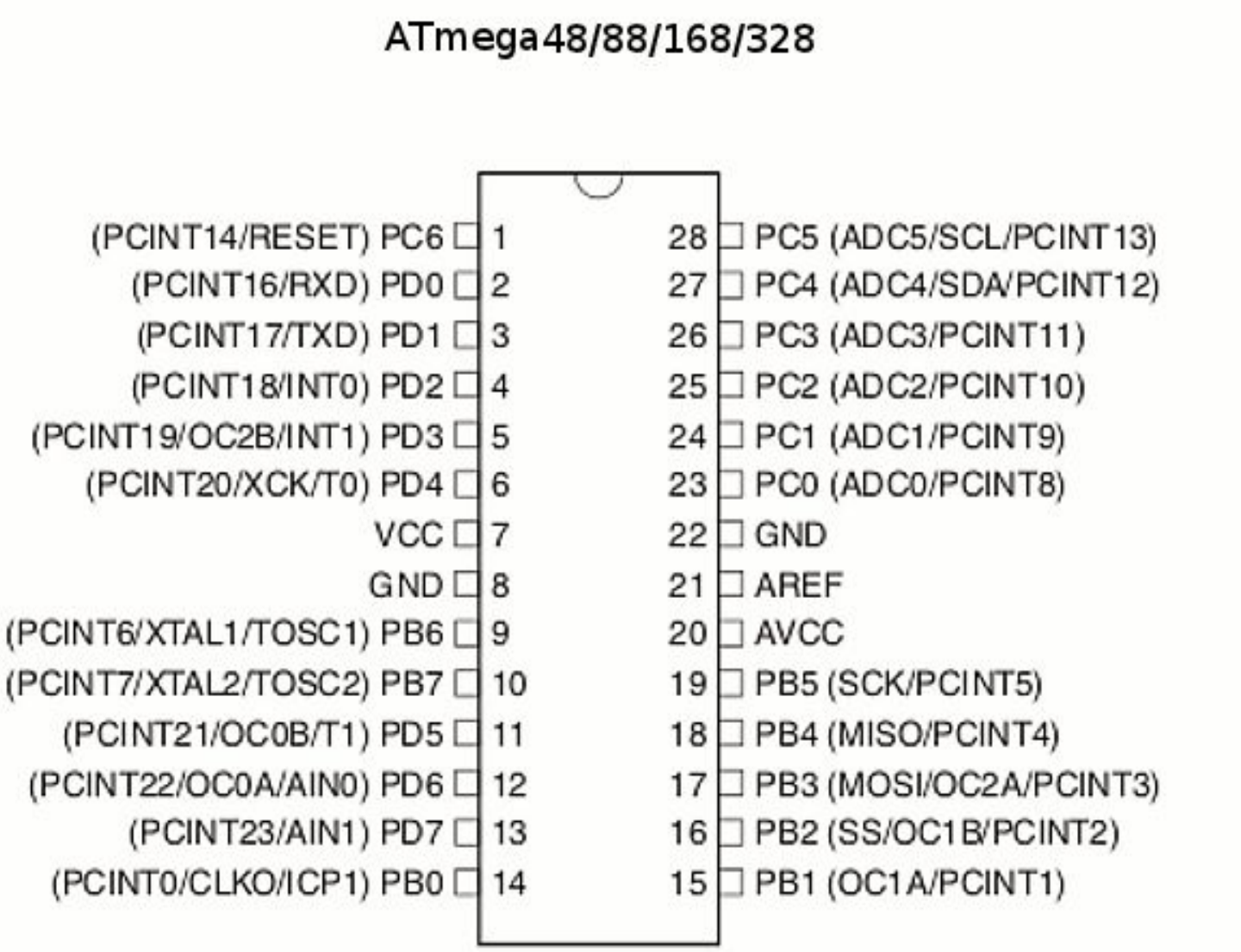
1.0 pin out: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that uses the AVR, which operates with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin that is reserved for future purposes.

"**Uno**" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by boot loader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

3.2.2.1 PIN Diagram:



Pin Descriptions:

VCC: Digital supply voltage

GND: Ground

Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit. Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier. If the Internal Calibrated RC Oscillator is used as chip clock source, PB7.6 is used as TOSC2.1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

Port C (PC5:0)

Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5..0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

PC6/RESET

If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C. If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running.

Port D (PD7:0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

AVCC

AVCC is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. Note that PC6..4 use digital supply voltage, VCC.

AREF

AREF is the analog reference pin for the A/D Converter.

ADC7:6 (TQFP and QFN/MLF Package Only)

In the TQFP and QFN/MLF package, ADC7:6 serves as analog inputs to the A/D converter.

These pins are powered from the analog supply and serve as 10-bit ADC channels.

Chapter-4

DESIGN

4.1 Introduction:

Moving on to the practical side, we want to understand not only how machine learning algorithms operate, but also how the user is situated as an integral part of any machine learning system. Here are two great examples of design approaches for machine learning. One considered the user as an integral part of the system and one focused more on just the algorithm.

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. Design Process is a way to think and design to solve problems. It is a process that can be replicated and used for a variety of different problems for small scale products to complex services. The process has several stages and steps, and each of them can be methodized and be done regularly. In a few words, it is a combination of analytical accuracy, synthesis methods & tools, and is based around the "building up" of ideas.

Typically, we start with something called Design Research. This is to acquire a view of the world and to interact with end-users in order to understand their needs and desires, as well as to understand what opportunities there are for design improvements changes. Design research could potentially lead to the discovery of hidden desires, and also potentially vital design insights. After we have gathered a significant amount of data and insights, we then proceed to `synthesis in order to understand the relationships within data, from which we extract meaningful insights. Synthesis is interpretive, it is to the individual -

which means that each person who synthesizes the data will most likely end up with different results, which is the beauty of synthesis. After the synthesis, we continue with the iterative process of design where we make things, form scenarios, stories in visual forms, and storyboards. We illustrate interaction over time, how people grow with the product, and how they relate to it.

For the last stage, we reach the evaluation, where we test the things we have created to understand if and why people use our products, and market validation to see if people will buy our products in the first place, as well as how people are going to behave once those products have become part of their life.

4.1.1 System Architecture:

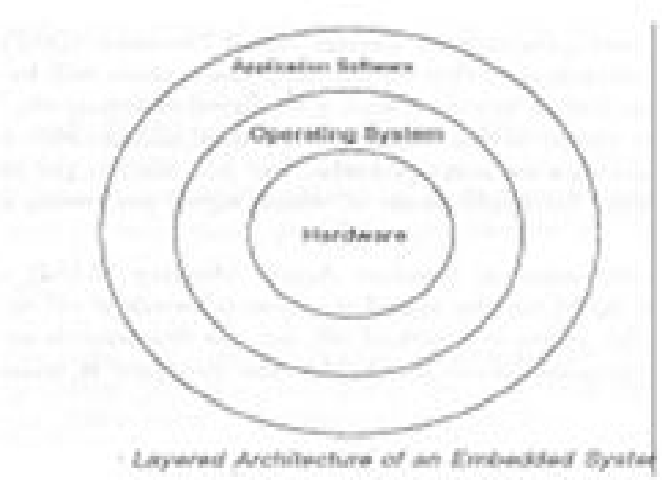


Figure 4.1.1 System Architecture

Initially Every embedded system consists of custom-built hardware built around a Central Processing Unit (CPU). This hardware also contains memory chips onto which the software is loaded. The software residing on the memory

chip is also called the ‘firmware’. The embedded system architecture can be represented as a layered architecture as shown in Fig. 4.1.1.

The operating system runs above the hardware, and the application software runs above the operating system. The same architecture is applicable to any computer including a desktop computer. However, there are significant differences. It is not compulsory to have an operating system in every embedded system. For small appliances such as remote control units, air conditioners, toys etc., there is no need *for* an operating system and you can write only the software specific to that application. For applications involving complex processing, it is advisable to have an operating system. In such a case, you need to integrate the application software with the operating system and then transfer the entire software on to the memory chip. Once the software is transferred to the memory chip, the software will continue to run *for* a long time. You don’t need to reload new software.

Now, let us see the details of the various building blocks of the hardware of an embedded system. As shown in Fig. the building blocks are;

· Central Processing Unit (CPU)

· Memory (Read-only Memory and Random Access Memory)

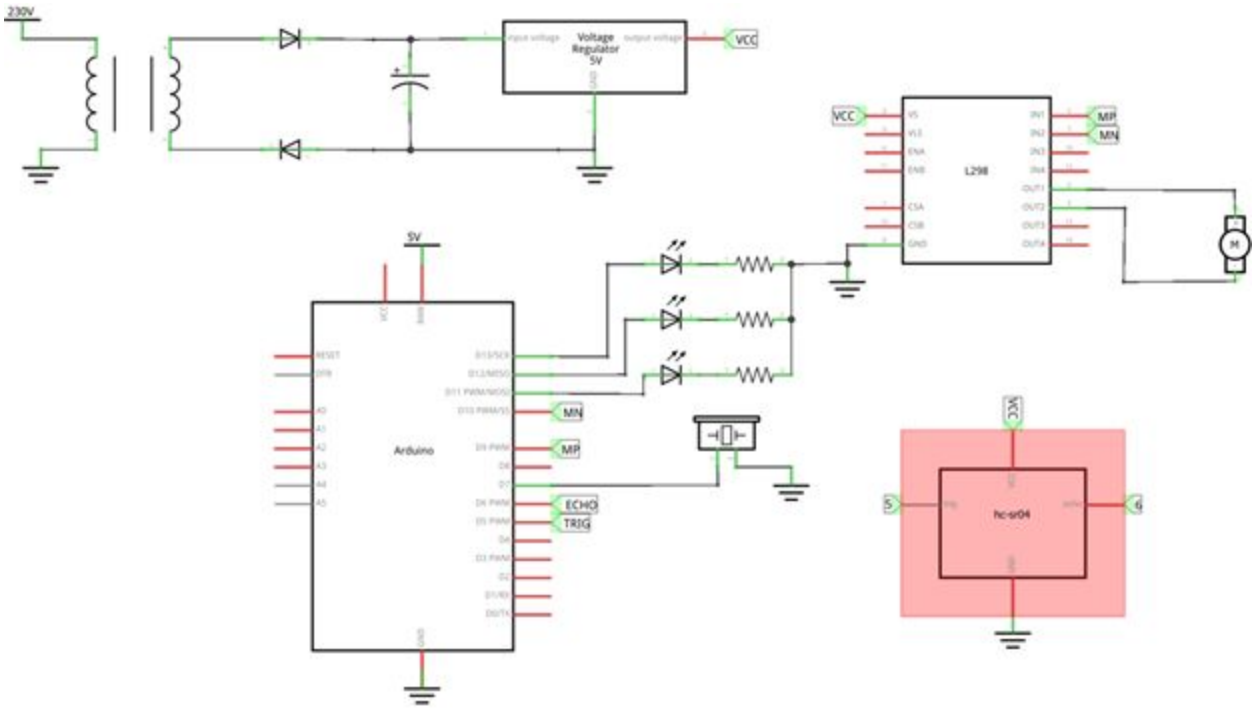
· Input Devices

· Output devices

· Communication interfaces

· Application-specific circuitry

4.1.2 Schematic Structure:



The schematic structure is defined as the reference for the flow of the model. In the figure above we can observe the circuit diagram which will show all the components involved in our model as a paperwork for our model to be built.

4.1.3 Central Processing Unit (CPU):

The Central Processing Unit (processor, in short) can be any of the following: microcontroller, microprocessor or Digital Signal Processor (DSP). A micro-controller is a low-cost processor. Its main attraction is that on the chip itself, there will be many other components such as memory, serial communication interface, analog-to digital converter etc. So, for small applications, a microcontroller is the best choice as the number of external components required will be very less. On the other hand, microprocessors are

more powerful, but you need to use many external components with them. D5P is used mainly for applications in which signal processing is involved such as audio and video processing.

4.1.4 Memory:

The memory is categorized as Random Access 11emory (RAM) and Read Only Memory (ROM). The contents of the RAM will be erased if power is switched off to the chip, whereas ROM retains the contents even if the power is switched off. So, the firmware is stored in the ROM. When power is switched on, the processor reads the ROM; the program is executed.

4.1.5 Input devices:

Unlike the desktops, the input devices to an embedded system have very limited capability. There will be no keyboard or a mouse, and hence interacting with the embedded system is no easy task. Many embedded systems will have a small keypad-you press one key to give a specific command. A keypad may be used to input only the digits. Many embedded systems used in process control do not have any input device *for* user interaction; they take inputs *from* sensors or transducers 1'fnd produce electrical signals that are in turn fed to other systems.

4.1.6 Output devices:

The output devices of the embedded systems also have very limited capability. Some embedded systems will have a *few* Light Emitting Diodes (LEDs) *to* indicate the health status of the system modules, or *for* visual indication of alarms. A small Liquid Crystal Display (LCD) may also be used to display *some* important parameters.

4.1.7 Communication interfaces:

The embedded systems may need to interact with other embedded systems at they may have to transmit data to a desktop. To facilitate this, the embedded systems are provided with one or a *few* communication interfaces such as RS232, RS422, RS485, Universal Serial Bus (USB), IEEE 1394, Ethernet etc.

4.1.8 Application-specific circuitry:

Sensors, transducers, special processing and control circuitry may be required for an embedded system, depending on its application. This circuitry interacts with the processor to carry out the necessary work. The entire hardware has to be given power supply either through the 230 volts main supply or through a battery. The hardware has to design in such a way that the power consumption is minimized.

4.2 DFD / ER / UML diagram:

UML Diagrams UML stands for unified modeling language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed and was created by the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. UML consists of two major components: A Meta-model and a notation. In the future, some form of method or process may also be added to; or association with UML. It is important to distinguish between the UML model and the set of diagrams of a system.

A diagram is a partial graphic representation of a system's model. The set of diagrams need not completely cover the model and deleting a diagram does not change the model. The model may also contain documentation that drives the

model elements and diagrams. In 1997 UML was adopted as a standard by the Object Management Group (OMG), and has been managed by this organization ever since. In 2005 UML was also published by the International Organization for Standardization (ISO) as an approved ISO standard. Since then it has been periodically revised to cover the latest revision of UML. 17 UML is not a development method by itself; however, it was designed to be compatible with the leading object-oriented software development methods of its time, for example OMT, Booch method, Objector and especially RUP that it was originally intended to be used with when work began at Rational Software. UML (Unified Modelling Language) is a standard notation for the modelling of real-world objects as a first step in developing an object-oriented design methodology. The major perspectives of a UML are Design, Implementation, Process and Deployment. The center is the Use Case view which connects all these four. A Use case represents the functionality of the system.

The following are the UML diagrams designed for the proposed project-

- Use Case diagram.
- Collaboration diagram.
- Sequence diagram.
- Activity diagram.
- Statechart diagram.
- Component diagram.
- Deployment diagram.

4.2.1 Use Case diagram:

• Use Case diagrams in the UML is a type of behaviour diagrams defined by and created from a Use-case analysis.

• It is used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors).

• Use case diagrams are drawn to capture the functional requirements of a system.

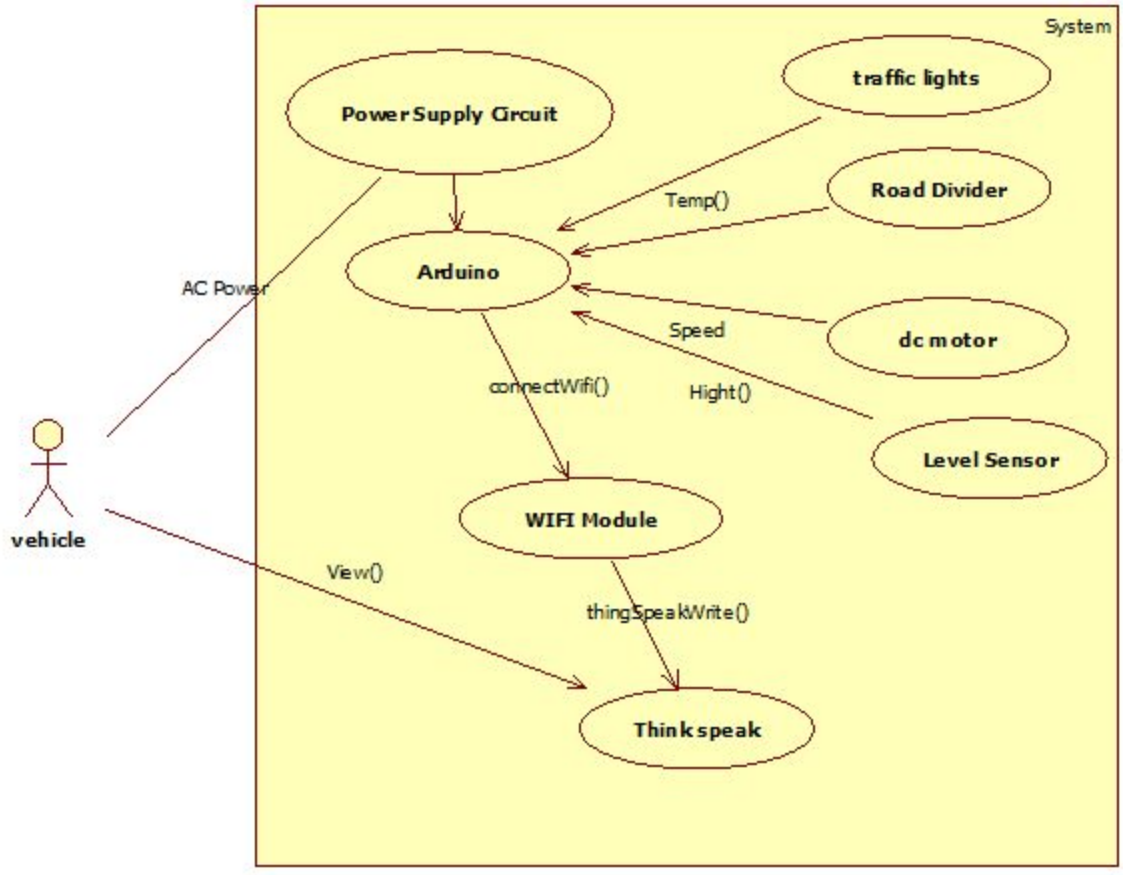


Figure 4.2.1 Use Case diagram of Movable Road Divider For Organized Vehicular Traffic Control

4.2.2 Collaboration diagram:

• A Collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML).

• These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object

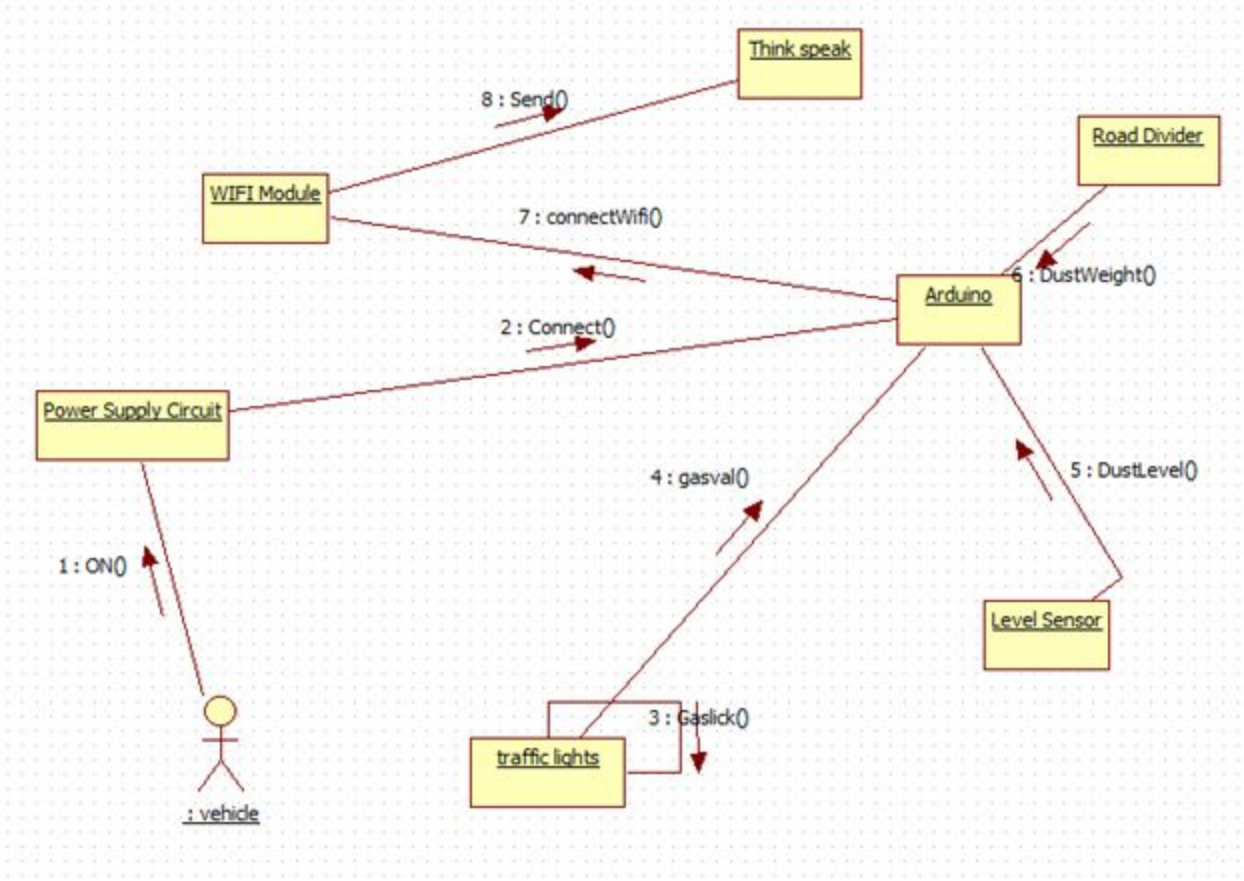


Figure 4.2.2 Collaboration diagram of Movable Road Divider For Organized Vehicular Traffic Control

4.2.3 Sequence Diagram:

• A sequence diagram shows object interactions arranged in time sequence.

• It depicts objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

• A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order.

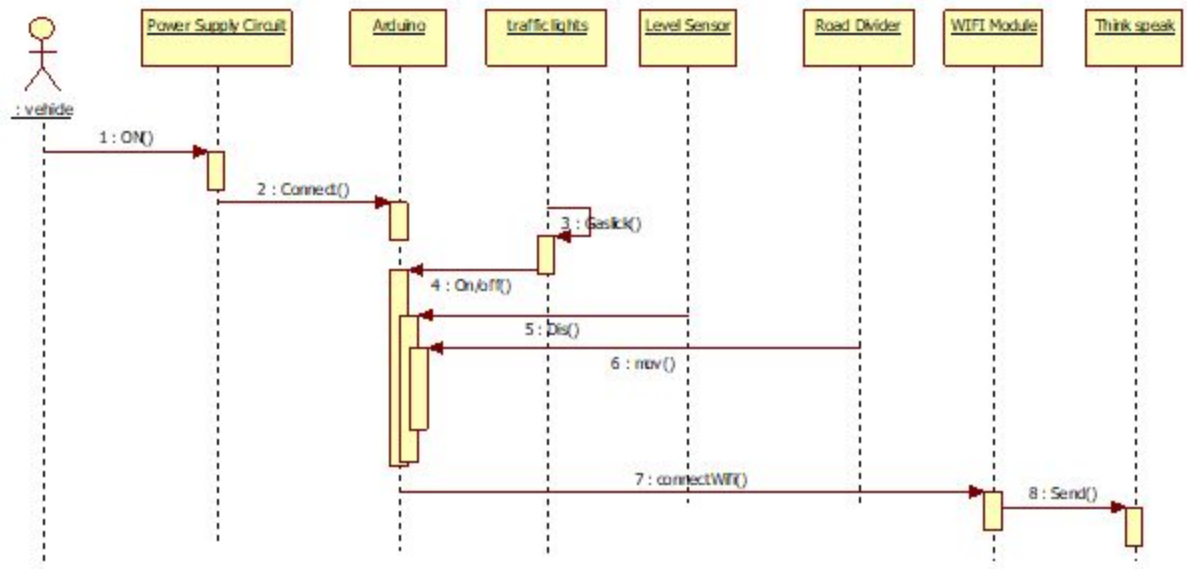


Figure 4.2.3 Sequence diagram of Movable Road Divider For Organized Vehicular Traffic Control

4.2.4 Activity Diagram:

Activity diagram is defined as a UML diagram that focuses on the execution and flow of the behavior of a system instead of implementation. It is also called object-oriented flowchart. Activity diagrams consist of activities that are made up of actions which apply to behavioral modeling technology.

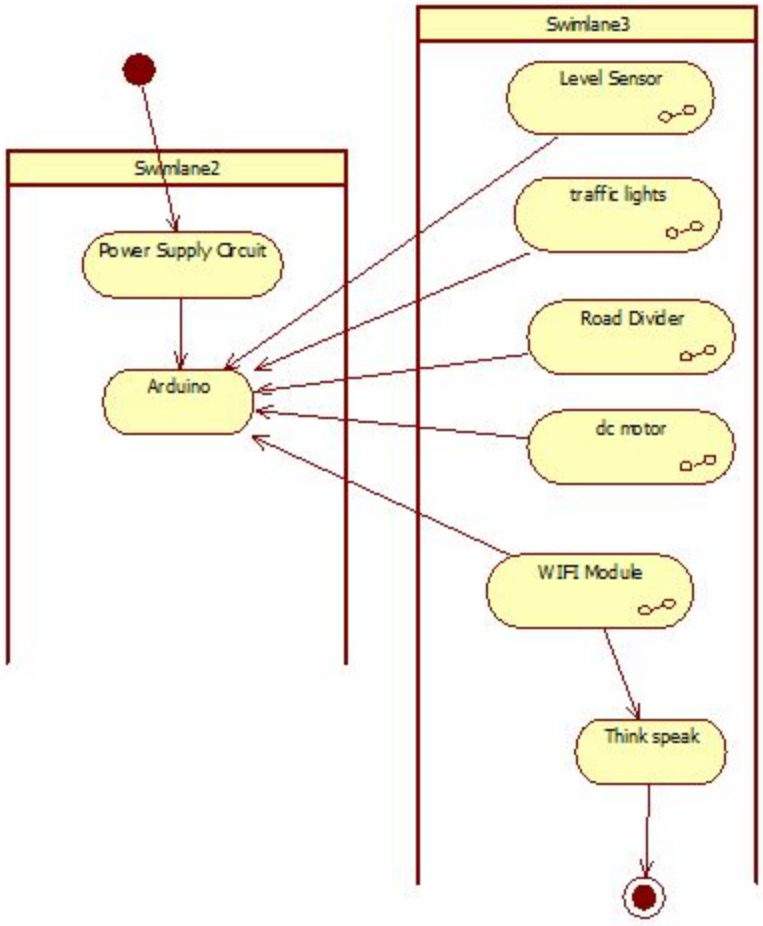


Figure 4.2.4 Activity diagram of Movable Road Divider For Organized Vehicular Traffic Control

4.2.5 Statechart Diagram:

Statechart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. Statechart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model the lifetime of an object from creation to termination.

Statechart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using Statechart diagrams −

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its lifetime.
- Define a state machine to model the states of an object.

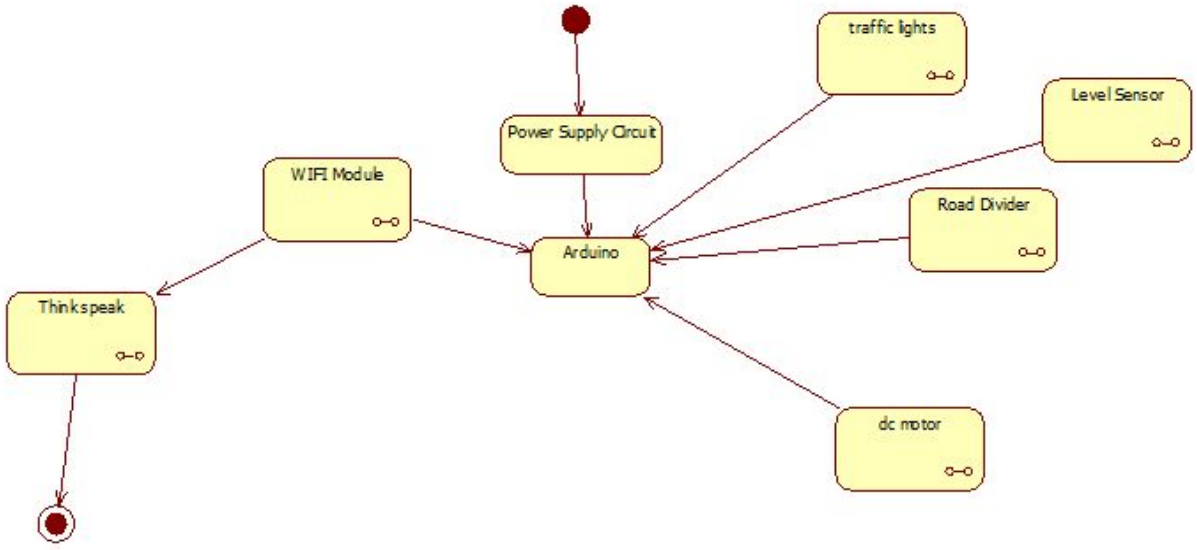


Figure 4.2.5 Statechart diagram of Movable Road Divider For Organized Vehicular Traffic Control

4.2.6 Component diagram:

• Component diagrams are different in terms of nature and behaviour. Component diagrams are used to model the physical aspects of a system. Physical aspects are elements such as executables, libraries, files, documents, etc. which reside in a node.

• Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

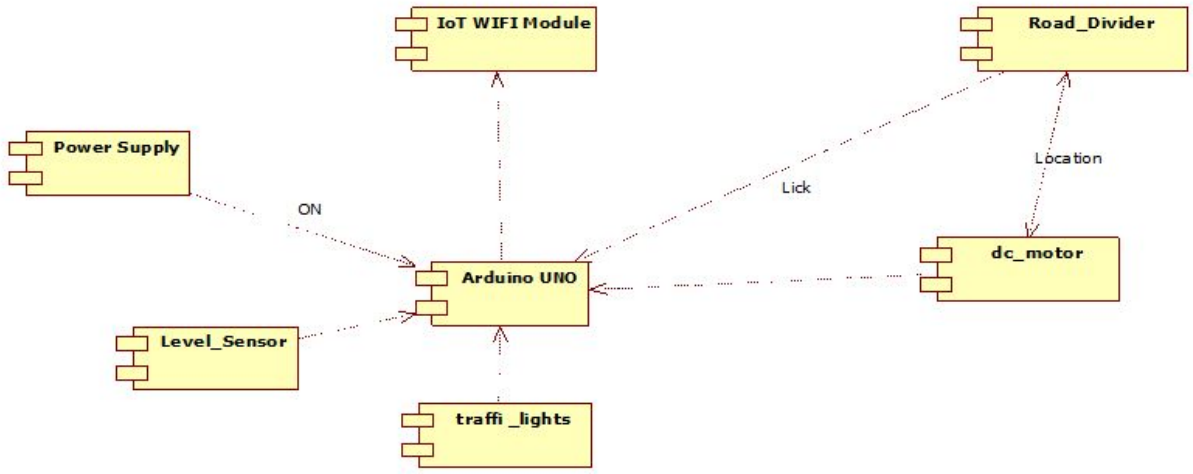


Figure 4.2.6 Component diagram of Movable Road Divider ForOrganized Vehicular Traffic Control

4.2.7 Deployment diagram:

• The deployment diagram is a structure diagram that shows the architecture of the system as a deployment (distribution) of software artifacts to deployment targets.

• Artifacts represent concrete elements in the physical world that are the result of a development process. Examples of artifacts are executable files, libraries, archives, database schemas, configuration files, etc.

• Deployment target is usually represented by a node which is either a hardware device or some software execution environment. Nodes could be connected through communication paths to create networked systems of arbitrary complexity.

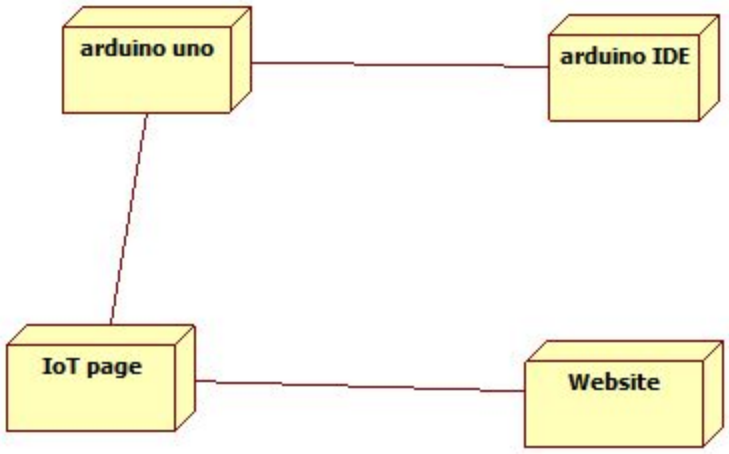


Figure 4.2.7 Deployment diagram of Movable Road Divider For Organized Vehicular Traffic Control

Chapter– 5

IMPLEMENTATION AND RESULTS

5.1 Introduction

5.1.1 Arduino Microcontroller

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, 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 a AC-to-DC adapter or battery to get started.

5.1.2 Features:

1.0 pin out: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that uses the AVR, which operates with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin that is reserved for future purposes.

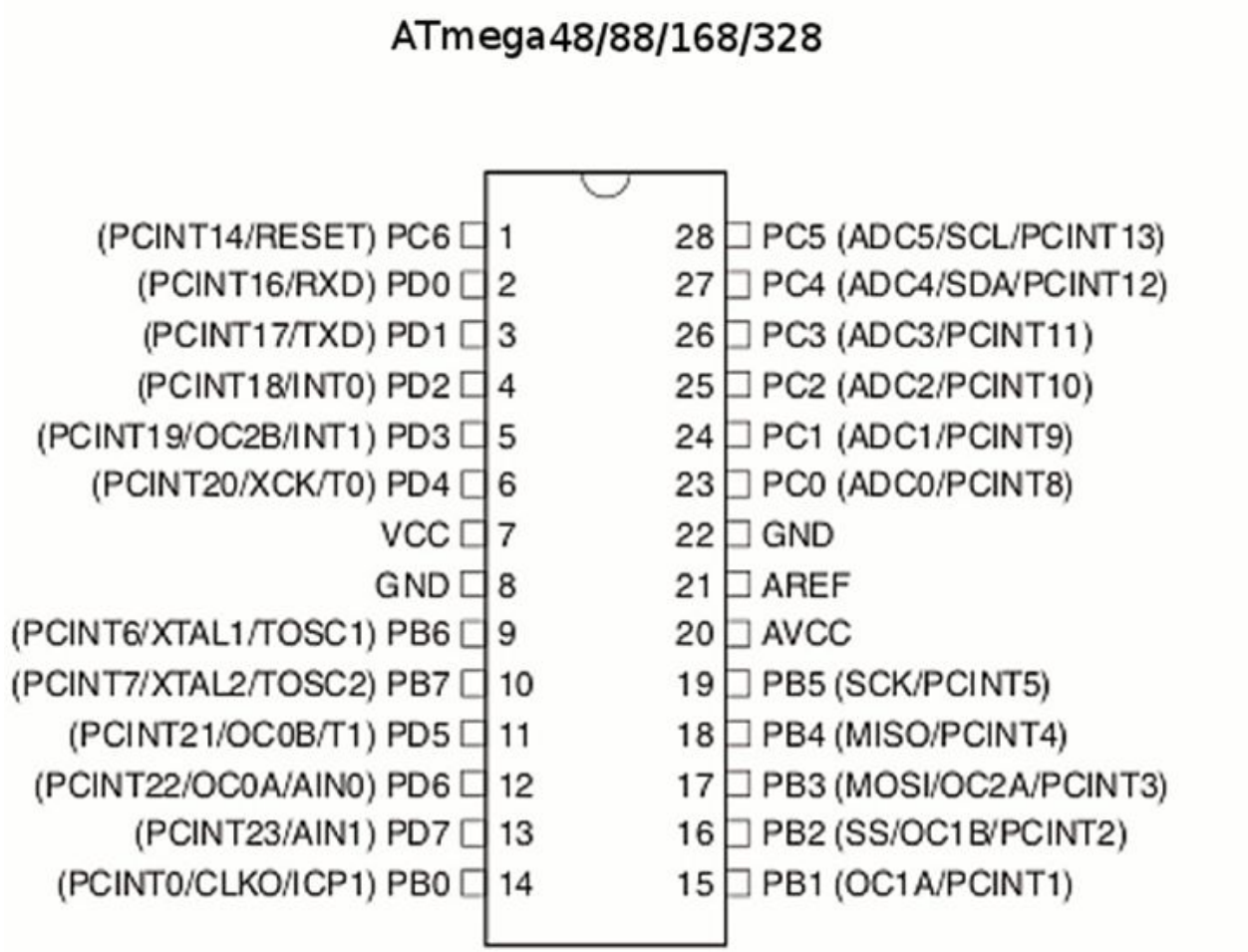
"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

5.1.3 Summary:

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory by boot loader	32 KB (ATmega328) of which 0.5 KB used
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

5.2 Explanation

5.2.1. Pin diagram



5.2.2 Power:

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the

board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

5.2.2.1 VIN:

The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5.2.2.2 5V:

This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

5.2.2.3 3V3:

A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

5.2.2.4 GND: Ground pins.

5.2.3 Memory:

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

5.2.4 Input and Output:

5.2.4.1 Pin Descriptions:

VCC: Digital supply voltage

GND: Ground

5.2.4.2 Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2:

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.

If the Internal Calibrated RC Oscillator is used as chip clock source, PB7.6 is used as TOSC2.1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

5.2.4.3 Port C (PC5:0):

Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5..0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated.

The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

5.2.4.4 PC6/RESET:

If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C. If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running.

5.2.4.5 Port D (PD7:0):

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

5.2.4.6 AVCC:

AVCC is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. Note that PC6..4 use digital supply voltage, VCC.

5.2.4.7 AREF:

AREF is the analog reference pin for the A/D Converter.

5.2.4.8 ADC7:6 (TQFP and QFN/MLF Package Only)

In the TQFP and QFN/MLF package, ADC7:6 serves as analog inputs to the A/D converter.

These pins are powered from the analog supply and serve as 10-bit ADC channels.

5.2.5 Communication:

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

5.2.6 Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the

ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the boot loader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the boot loader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Over current Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and over current. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

5.3 POWER SUPPLY

5.3.1 Introduction

The input to the circuit is applied from the regulated power supply. The a.c. input i.e., 230V from the mains supply is step down by the transformer to 12V and is fed to a rectifier. The output obtained from the rectifier is a pulsating d.c voltage. So in order to get a pure d.c voltage, the output voltage from the rectifier is fed to a filter to remove any a.c components present even after rectification. Now, this voltage is given to a voltage regulator to obtain a pure constant dc voltage.

5.3.2 Transformer

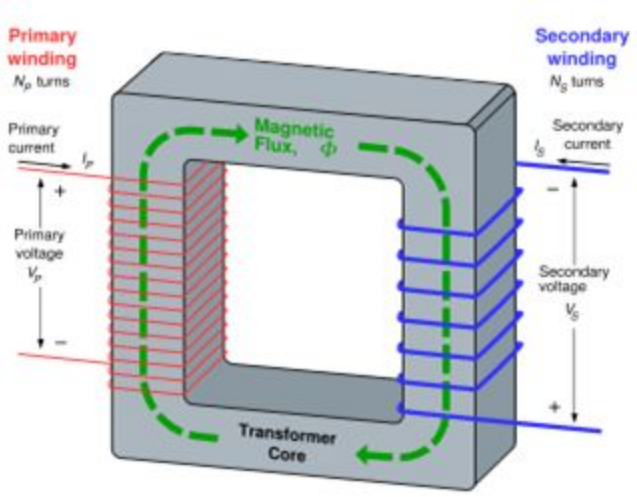
Usually, DC voltages are required to operate various electronic equipment and these voltages are 5V, 9V or 12V. But these voltages cannot be obtained directly. Thus the a.c input available at the mains supply i.e., 230V is to be brought down to the required voltage level. This is done by a transformer. Thus, a step down transformer is employed to decrease the voltage to a required level.

A transformer is an electrical device that transfers energy from one circuit to another by magnetic coupling with no moving parts. A transformer comprises two or more coupled windings, or a single tapped winding and, in most cases, a magnetic core to concentrate magnetic flux. A changing current in one winding creates a time-varying magnetic flux in the core, which induces a voltage in the other windings.

5.3.2.1 Basic principles:

A simple transformer consists of two electrical conductors called the primary winding and the secondary winding. Energy is coupled between the windings by the time-varying magnetic flux that passes through (links) both primary and secondary windings. Whenever the amount of current in a coil changes

(including when the current is switched on or off), a voltage is induced in the neighboring coil. The effect, called mutual inductance, is an example of electromagnetic induction.



If a time-varying voltage is applied to the primary winding of turns, a current will flow in it producing a magneto motive force (MMF). Just as an electromotive force (EMF) drives current around an electric circuit, so MMF tries to drive magnetic flux through a magnetic circuit. The primary MMF produces a varying magnetic flux in the core, and, with an open circuit secondary winding, induces a back electromotive force (EMF) in opposition to . In accordance with Faraday's law of induction, the voltage induced across the primary winding is proportional to the rate of change of flux:

$$v_P = N_P \frac{d\Phi_P}{dt} \qquad \text{and} \qquad v_S = N_S \frac{d\Phi_S}{dt}$$

where

vP and vS are the voltages across the primary winding and secondary winding,

NP and NS are the numbers of turns in the primary winding and secondary winding,

dΦP / dt and dΦS / dt are the derivatives of the flux with respect to time of the primary and secondary windings.

Saying that the primary and secondary windings are perfectly coupled is equivalent to saying that . Substituting and solving for the voltages shows that:

where

vp and vs are voltages across primary and secondary,

Np and Ns are the numbers of turns in the primary and secondary, respectively.

Hence in an ideal transformer, the ratio of the primary and secondary voltages is equal to the ratio of the number of turns in their windings, or alternatively, the voltage per turn is the same for both windings.

5.3.2.2 Transformer losses arise from:

5.3.2.2.1 Winding resistance

Current flowing through the windings causes resistive heating of the conductors (I2 R loss). At higher frequencies, skin effect and proximity effect create additional winding resistance and losses.

5.3.2.2.2 Eddy currents

Induced eddy currents circulate within the core, causing resistive heating. Silicon is added to the steel to help in controlling eddy currents. Adding silicon also has the advantage of stopping aging of the electrical steel that was a problem years ago.

5.3.2.2.3 Hysteresis losses

Each time the magnetic field is reversed, a small amount of energy is lost to hysteresis within the magnetic core. The amount of hysteresis is a function of the particular core material.

5.3.2.2.4 Magnetostriction

Magnetic flux in the core causes it to physically expand and contract slightly with the alternating magnetic field, an effect known as magnetostriction. This in turn causes losses due to frictional heating in susceptible ferromagnetic cores.

5.3.2.2.5 Mechanical losses

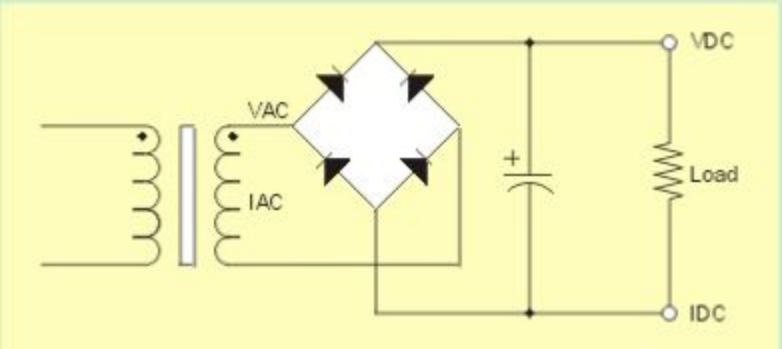
In addition to magnetostriction, the alternating magnetic field causes fluctuating electromagnetic forces between the primary and secondary windings. These incite vibrations within nearby metalwork, creating a familiar humming or buzzing noise, and consuming a small amount of power.

5.3.2.2.6 Stray losses

Not all the magnetic field produced by the primary is intercepted by the secondary. A portion of the leakage flux may induce eddy currents within nearby conductive objects, such as the transformer's support structure, and be converted to heat.

5.3.3 Rectifier:

The output from the transformer is fed to the rectifier. It converts A.C. into pulsating D.C. The rectifier may be a half wave or a full wave rectifier. In this project, a bridge rectifier is used because of its merits like good stability and full wave rectification.

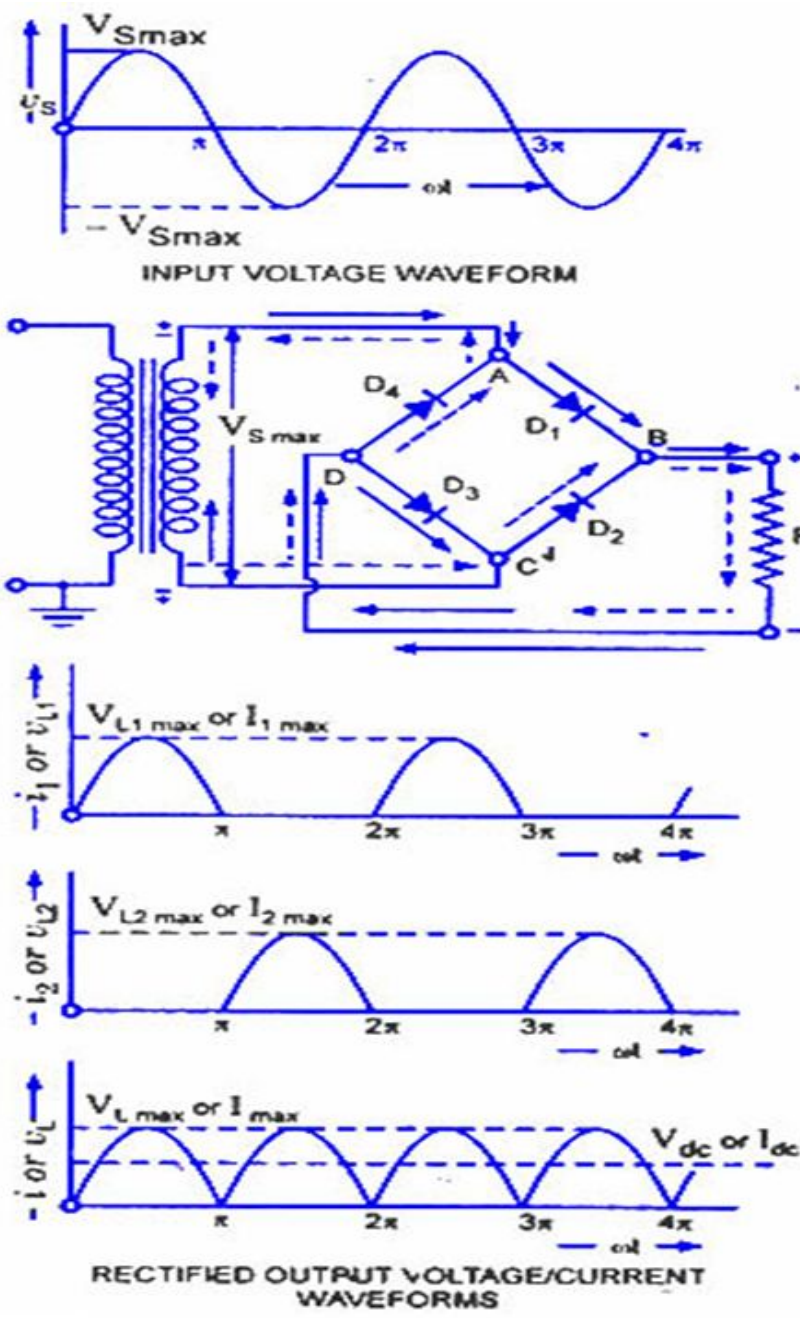


5.3.3.1 Bridge Rectifier

The Bridge rectifier is a circuit, which converts an ac voltage to dc voltage using both half cycles of the input ac voltage. The Bridge rectifier circuit is shown in the figure. The circuit has four diodes connected to form a bridge. The ac input voltage is applied to the diagonally opposite ends of the bridge. The load resistance is connected between the other two ends of the bridge.

For the positive half cycle of the input ac voltage, diodes D1 and D3 conduct, whereas diodes D2 and D4 remain in the OFF state. The conducting diodes will be in series with the load resistance R_L and hence the load current flows through R_L.

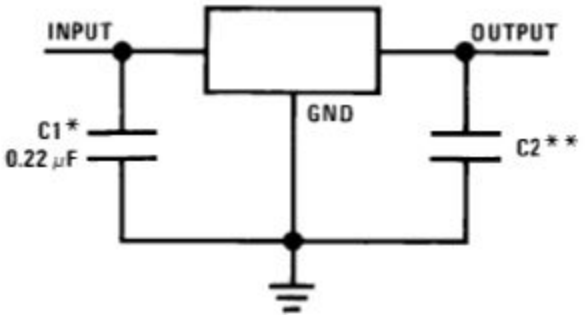
For the negative half cycle of the input ac voltage, diodes D2 and D4 conduct whereas, D1 and D3 remain OFF. The conducting diodes D2 and D4 will be in series with the load resistance R_L and hence the current flows through R_L in the same direction as in the previous half cycle. Thus a bi-directional wave is converted into a unidirectional wave.



Bridge Rectifier

5.3.4 Filter:

Capacitive filter is used in this project. It removes the ripples from the output of rectifier and smoothens the D.C. Output received from this filter is constant until the mains voltage and load is maintained constant. However, if either of the two is varied, D.C. voltage received at this point changes. Therefore a regulator is applied at the output stage.



5.3.5 Voltage regulator:

As the name itself implies, it regulates the input applied to it. A voltage regulator is an electrical regulator designed to automatically maintain a constant voltage level. In this project, power supply of 5V and 12V are required. In order to obtain these voltage levels, 7805 and 7812 voltage regulators are to be used. The first number 78 represents positive supply and the numbers 05, 12 represent the required output voltage levels. The L78xx series of three-terminal positive regulators is available in TO-220, TO-220FP, TO-3, D2PAK and DPAK packages and several fixed output voltages, making it useful in a wide range of applications. These regulators can provide local on-card regulation, eliminating the distribution problems associated with single point regulation. Each type employs internal current limiting, thermal shut-down and safe area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1 A output current. Although

designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltage and currents.

5.4 SONAR

5.4.1 Ultrasonic Waves

Humans can normally hear sound frequencies between 20 and 20,000 Hz (20 kHz). When a sound wave's frequency lies above 20 kHz, it is called an ultrasonic wave. While we cannot hear ultrasonic waves, we apply them in various technologies such as sonar systems, sonograms, surgical tools, and cleaning systems. Some animals also use ultrasonic waves in a specialized technique called echolocation that allows them to pinpoint objects and other animals, even in the dark.

The project uses 5 standard transistors to receive and transmit the ultrasound and a comparator to set the threshold echo detection level - so there are no special components other than the microcontroller. The ultrasonic transducers are standard 40 kHz types. Note that the internal oscillator of the MC micro is used and this saves two pins - that can be used for normal I/O,

Ultrasonic sensors (also known as transceivers when they both send and receive) work on a principle similar to radar or sonar which evaluate attributes of a target by interpreting the echoes from radio or sound waves respectively. Ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor. Sensors calculate the time interval between sending the signal and receiving the echo to determine the distance to an object.

This technology can be used for measuring: wind speed and direction (anemometer), fullness of a tank and speed through air or water. For measuring speed or direction a device uses multiple detectors and calculates the speed from the relative distances to particulates in the air or water. To measure the amount of liquid in a tank, the sensor measures the distance to the surface of the fluid. Further applications include: humidifiers, sonar, medical ultrasonography, burglar alarms and non-destructive testing.

Systems typically use a transducer which generates sound waves in the ultrasonic range, above 18,000 hertz, by turning electrical energy into sound, then upon receiving the echo turn the sound waves into electrical energy which can be measured and displayed.

The technology is limited by the shapes of surfaces and the density or consistency of the material. For example foam on the surface of a fluid in a tank could distort a reading.

5.4.2 Ultrasonic Range Finding:

A common use of ultrasound is in range finding; this use is also called SONAR, (sound navigation and ranging). This works similarly to RADAR (radio detection and ranging): An ultrasonic pulse is generated in a particular direction. If there is an object in the path of this pulse, part or all of the pulse will be reflected back to the transmitter as an echo and can be detected

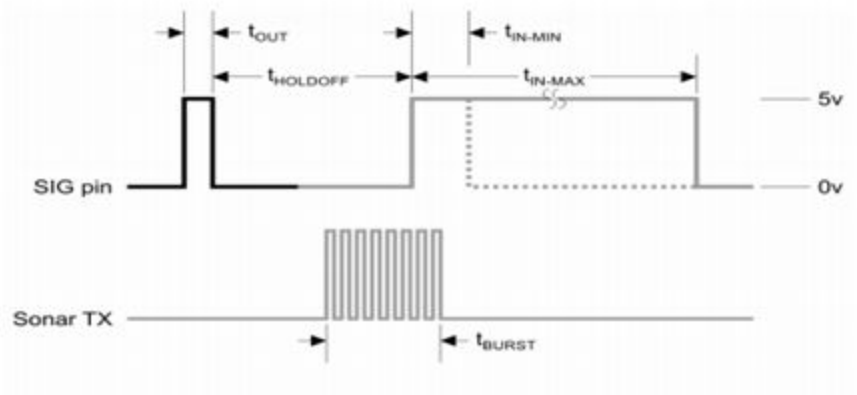
through the receiver path. By measuring the difference in time between the pulse being transmitted and the echo being received, it is possible to determine how far away the object is.

The measured travel time of SONAR pulses in water is strongly dependent on the temperature and the salinity of the water. Ultrasonic ranging is also applied for measurement in air and for short distances. Such a method is capable for easily and rapidly measuring the layout of rooms.

Although range finding underwater is performed at both sub-audible and audible frequencies for great distances (1 to several kilometers), ultrasonic range finding is used when distances are shorter and the accuracy of the distance measurement is desired to be finer. Ultrasonic measurements may be limited through barrier layers with large salinity, temperature or vortex differentials. Ranging in water varies from about hundreds to thousands of meters, but can be performed with centimeters to meters accuracy.

5.4.3 Theory:

The Ping sensor detects objects by emitting a short ultrasonic burst and then "listening" for the echo. Under control of a host microcontroller (trigger pulse), the sensor emits a short 40 kHz (ultrasonic) burst. This burst travels through the air at about 1130 feet per second, hits an object and then bounces back to the sensor. The PING sensor provides an output pulse to the host that will terminate when the echo is detected; hence the width of this pulse corresponds to the distance to the target.

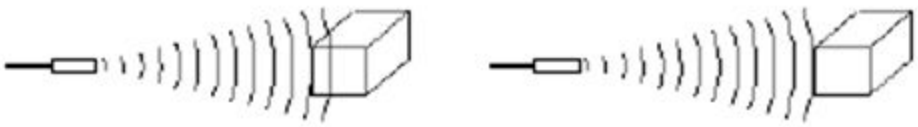


5.4.4 Two Ultrasonic Sensor Types

The following diagrams summarize the distinctions between proximity and ranging ultrasonic sensors:

5.4.4.1 Proximity Detection

An object passing anywhere within the preset range will be detected and generate an output signal. The detect point is independent of target size, material, or degree of reflectivity.

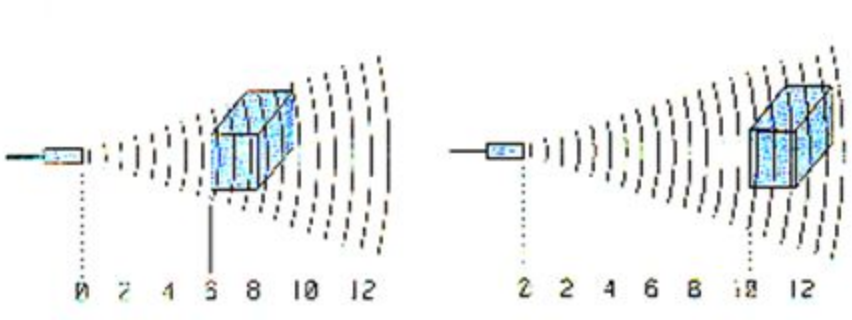


Objected detected - YES Objected detected - NO

5.4.4.2 Ranging Measurement:

Precise distance(s) of an object moving to and from the sensor are measured via time intervals between transmitted and reflected bursts of ultrasonic sound. The example shows a target detected at six inches from the

sensor and moving to 10 inches. The distance change is continuously calculated and outputted.



5.4.4.3 Calculation for target finding:

The time from transmission of the pulse to reception of the echo is the time taken for the signal energy to travel through the air to the object and back again. Since the speed of signal is constant through air measuring the echo reflection time lets you calculate the distance to the object using the DST equation:

Distance = (s * t)/2 (in meters)

You need to divide by 2 as the distance is the round trip distance i.e. from transmitter to object and back again.

Where:

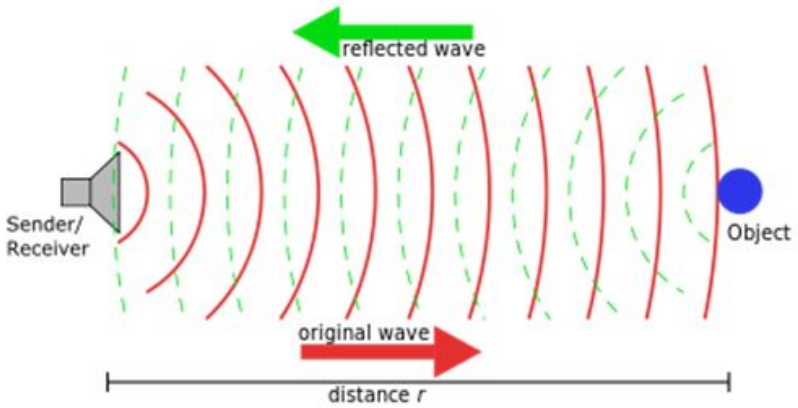
s [m/s]	the speed of sound in air
t [s]	The round trip echo time.

Some delay times:

Round trip echo time	Distance
t = 588us	10cm
t = 5.8ms	1m

You can get ultrasonic transducers optimized for 25 kHz, 32 kHz, 40 kHz or wide bandwidth transducers. This project uses a 40 kHz transducer but it will still work with the others if you make simple changes to the software (where it generates the 40kz signal). The receiver and generator circuits will work as they are. If you use a different transducer you must change the software to generate the correct frequency for the transducer as they only work at their specific operating frequency. The 40kz signal is easily generated by the microcontroller but detection requires a sensitive amplifier. I have used a three transistor amplifier for the receiver.

This is followed by a peak detector and comparator which set the sensitivity threshold so that false reflections (weaker signals) are ignored.



5.4.4.4 Time mode

If you store the value of timer 1 and then the counter value starts counting up to the Rx echo completion or as soon as an ultrasonic echo is received. Counter value gives the time delay in machine cycles. Since the project uses a 4MHz main clock then the time delay will be measured in microseconds.

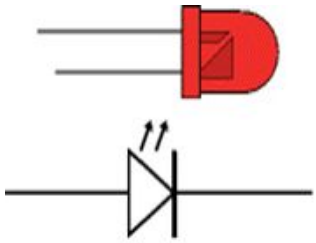
The minimum distance of this scheme is about 5cm. looking at the output of the first receiver amplifier shows that it should be more accurate at lower distances - it is inaccurate by about 2cm which is still quite good. Probably the addition of amplifiers for the longer range stops accurate short range operation. The maximum distance is limited by the sensitivity, gain and noise performance of the receive amplifier and also the transmit power and duration of transmission. For this circuit the maximum distance is about 3m.

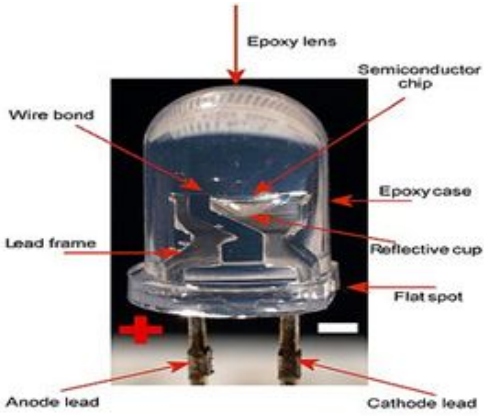
MC Sonar Specification

Range	~0cm - 4m
Accuracy	100 %
Transducer frequency	40kHz
internal oscillator frequency	4MHz

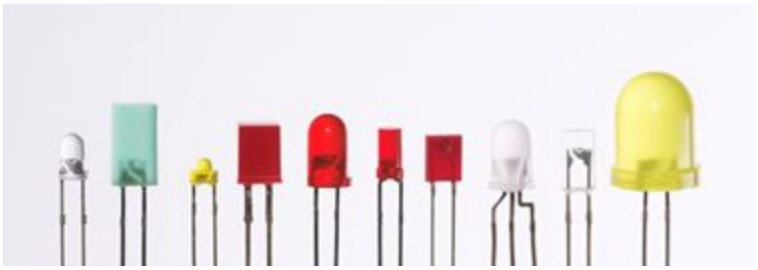
5.5 Light Emitting Diode (LED):

The longer lead is the anode (+) and the shorter lead is the cathode (&minus). In the schematic symbol for an LED (bottom), the anode is on the left and the cathode is on the right. Light Emitting diodes are elements for light signalization in electronics.





They are manufactured in different shapes, colors and sizes. For their low price, low consumption and simple use, they have almost completely pushed aside other light sources- bulbs at first place.

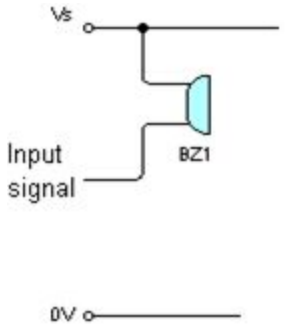


It is important to know that each diode will be immediately destroyed unless its current is limited. This means that a conductor must be connected in parallel to a diode. In order to correctly determine the value of this conductor, it is necessary to know the diode's voltage drop in forward direction, which depends on what material a diode is made of and what colors it is. Values typical for the most frequently used diodes are shown in the table below: As seen, there are three main types of LEDs. Standard ones get full brightness at 20mA. Low Current diodes get full brightness at ten time’s lower current while Super Bright diodes produce more intensive light than Standard ones.

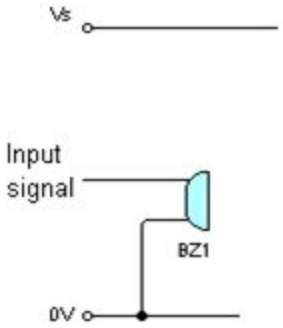
5.6 Buzzer

5.6.1 Introduction

The buzzer subsystem produces an audible tone when powered



Buzzer circuit



Buzzer circuit for use with higher current process units

5.6.2 Method Of Operation

5.6.2.1 Buzzer circuit

Buzzers come in a variety of voltages and currents. The power supply for the buzzer (which can be separate from the supply for the rest of the electronics) must provide the voltage needed by the buzzer.

Piezo sounders are a type of buzzer. They should not be confused with Piezo transducers – which require an a.c. input voltage to drive them.

Some process units provide enough current to drive buzzers. Typical buzzers require currents in the range 10 – 35mA.

If CMOS ICs or a higher current buzzer are used then a driver (transistor, Darlington or MOFET) is needed to boost the current. The circuit on the left shows the circuit needed with a driver.

5.6.2.2 Buzzer circuit for use with higher current process units

PICs, 555 Timer ICs and the LM324 op-amp can provide higher currents and can drive some buzzers directly.

Check the data for the buzzer and the process unit to make sure that the process unit can provide more current than is needed by the buzzer.

If this is possible, the buzzer is connected to the 0V rail (as on the left) rather than to +Vs.

Buzzers can either be PCB-mounted or connected to the circuit with flying leads. Usually it is neater to mount them on the PCB.

5.6.3 Application

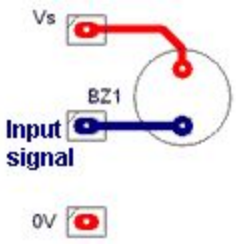
5.6.3.1 Making

Buzzers have a positive and a negative terminal, marked on their case. The positive terminal should be connected to the positive voltage supply. The negative terminal should be connected to the signal from the driver. The graphic on the left shows how part of the PCB might look for a PCB-mounted buzzer connected to a driver.

How part of the PCB might look

If a buzzer with flying leads is used then a terminal block is mounted on the PCB and wires from this are connected to the buzzer.

Build and test the unit that will provide the driving input signal before adding the buzzer.



5.6.3.2 Testing: Make sure that the buzzer switches on and off as power is applied from the driver unit.

5.7 Method of Implementation

```
#define MP 9
#define MN 10
#define buz 13
#define red 5
#define yel 4
#define grn 3
#define trigPin 12
#define echoPin 11

int duration, distance;

void setup()
{
  Serial.begin(9600);
  pinMode (trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(MP, OUTPUT);
  pinMode(MN, OUTPUT);
  pinMode(red,OUTPUT);
  pinMode(yel,OUTPUT);
  pinMode(grn,OUTPUT);
  pinMode(buz,OUTPUT);
  Serial.println("Setup Completed");
}

void loop()
{
   int l;
   digitalWrite(red,1);
   delay(1000);
   digitalWrite(red,0);
   digitalWrite(yel,1);
   delay(500);
   digitalWrite(yel,0);
   digitalWrite(grn,1);
   l = density();
```

```
   Serial.println("Sensor Value = " + String(l));
   if(l<10)
   {
      Serial.println("Traffic is High");
      digitalWrite(buz,1);
      delay(3000);
      digitalWrite(buz,0);
      divider_front();
      delay(8000);
      divider_back();
      delay(3000);
      digitalWrite(grn,0);
   }
   else
   {
    Serial.println("Traffic is Normal");
    delay(2000);
    digitalWrite(grn,0);
   }

  Serial.println();
  delay(500);
}

void divider_front()
{
  Serial.println("MOVING DIVIDER");
  analogWrite(MP,80);
  //digitalWrite(MP,1);
  analogWrite(MN,0);
  delay(800);
  analogWrite(MP,0);
  analogWrite(MN,0);
}

void divider_back()
{
  Serial.println("Moving Divider Back");
```

```
  analogWrite(MP,0);
  analogWrite(MN,80);
  //digitalWrite(MN,1);
  delay(800);
  digitalWrite(MP,0);
  digitalWrite(MN,0);
}

int density()
{
  digitalWrite (trigPin, HIGH);
  delay(50);
  digitalWrite (trigPin, LOW);
  duration=pulseIn(echoPin,HIGH);
  distance=(duration*0.034)/2;
  return distance;
}
```

Chapter 6

TESTING AND VALIDATION

6.1 Introduction

The process of executing a system with the intent of finding an error is called testing. Testing is defined as the process in which defects are identified, isolated or subjected for rectification and ensured that the product is defect free in order to produce the quality product and hence customer satisfaction. Quality is defined as justification of the requirements. Defect is nothing but deviation from the requirements. It is nothing but a bug.

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects). Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test: meets the requirements that guided its design and development, responds correctly to all kinds of inputs, performs its functions within an acceptable time, is sufficiently usable, can be installed and run in its intended environments, and Achieves the general result its stakeholder‟s desire. As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources.

As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects). The job of testing is an iterative process as when one bug is fixed; it can illuminate other, deeper bugs, or can even create new ones. Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors. Software testing can be conducted as soon as executable software (even if partially complete) exists.

The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system Identifying the amount of fertilization required for Rice Crops using Image Processing system to increase the yield requirements have been defined and then implemented in testable programs. In contrast, under an Agile approach, requirements, programming, and testing are often done concurrently.

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

TESTING METHODOLOGY

EMBEDDED TESTING is checking the functional and non-functional attributes of both software and hardware in an embedded system. The purpose of Embedded test is to verify and validate the Embedded software as well as hardware against client requirement. Embedded Software testing checks and

ensures the concerned software is of good quality and complies with all the requirements it should meet. Embedded software testing is an excellent approach to guarantee security in critical applications like medical equipment, railways, aviation, vehicle industry, etc. Strict and careful testing is crucial to grant software certification.

HOW TO PERFORM EMBEDDED SOFTWARE TESTING

In general, you test for four reasons:

- To find bugs in software
- Helps to reduce risk to both users and the company
- Cut down development and maintenance costs
- To improve performance

In Embedded Testing, the following activities are performed:

1. The software is provided with some inputs.

2. A Piece of the software is executed.

3. The software state is observed, and the outputs are checked for expected properties like whether the output matches the expected outcome, conformance to the requirements and absence of system crashes.

EMBEDDED SOFTWARE TESTING TYPES

Fundamentally, there are five levels of testing that can be applied to embedded software

Software Unit Testing

The unit module is either a function or class. Unit Testing is performed by the development team, primarily the developer and is usually carried out in a

peer-review model. Based on the specification of the module test cases are developed.

Integration Testing

Integration testing can be classified into two segments:

1. Software integration testing
2. Software/hardware integration testing.

In the end, the interaction of the hardware domain and software components are tested. This can incorporate examining the interaction between built-in peripheral devices and software. Embedded software development has a unique characteristic which focuses on the actual environment, in which the software is run, is generally created in parallel with the software. This causes inconvenience for testing since comprehensive testing cannot be performed in a simulated condition.

System Unit Testing

Now the module to be tested is a full framework that consists of complete software code additionally all real-time operating system (RTOS) and platform-related pieces such as interrupts, tasking mechanisms, communications and so on. The Point of Control protocol is not anymore a call to a function or a method invocation, but rather a message sent/got utilizing the RTOS message queues. System resources are observed to evaluate the system's ability to support embedded system execution. For this aspect, gray-box testing is the favored testing method. Depending on the organization, system unit testing is either the duty of the developer or a dedicated system integration team.

System Integration Testing

The module to be tested begins from a set of components within a single node. The Points of Control and Observations (PCOs) are a mix of network related communication protocols and RTOS, such as network messages and RTOS events. Additionally to a component, a Virtual Tester can likewise play the role of a node.

System Validation Testing

The module to be tested is a subsystem with a complete implementation or the complete embedded system. The objective of this final test is to meet external entity functional requirements. Note that an external entity either be a person, or a device in a telecom network, or both.

6.2 Validation

Validation is a process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements. Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfils its intended use when deployed in an appropriate environment. Here we validate the software and hardware units as a system and a V model of verification and validation technique is applied to test the embedded system as a whole.

Validation means observing the behaviour of the system. The verification and validation means that will ensure that the output of a phase is consistent with its input and that the output of the phase is consistent with the overall requirements of the system. This is done to ensure that it is consistent with the required output. If not, apply certain mechanisms for repairing and thereby achieving the requirement.

Here we validate the concept of operations with requirements and detailed design with last stage of operational maintenance is carried on.

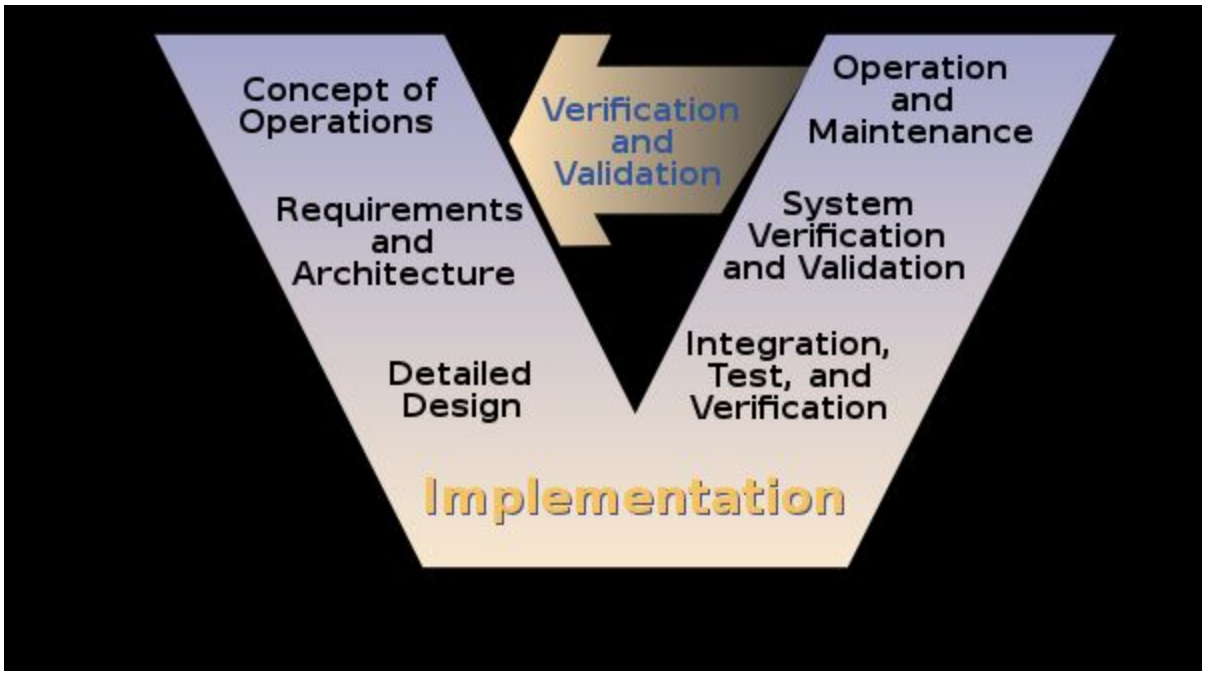


Figure 6.2.1 Validation

Chapter -7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

Our system can be used in multiple scenarios where the traffic on roads creates a lot of problems and public safety issues. For example, governments can use it to make traffic in an organized manner and thus reduce a lot of pollution and public safety measures can be achieved. It can be used in areas where ongoing and incoming traffic and the road can be divided accordingly by it to make effective use of roads and the traffic system.With growing population and the vehicles are increasing exponentially,there is limitation of resources and leads to many number of vehicles causing vast congestion of vehicles at places where there is more traffic.But due to this system,effective usage of width of roads happen and traffic can be controlled in highly populated areas.

7.2 Future Work

The present system effectively utilizes the road making changes in the widths of road and thus organizing traffic in an ordered and optimized manner. The system also fails in some cases subject to higher volumes of traffic. This work is a true reflection of what embedded systems can do in the field of road transportation and regulating traffic congestion problems. After successful implementation of this work, in future due to the technological advances and increase in population,this system may fail .So for this an algorithm can be developed in viewing and analyzing the traffic from a distant place so that the traffic volume can be analyzed at a nearer place next to the analyzed place so that the effectiveness of system increases.

REFERENCES

1. K.Vidhya, A.Bazila Banu, Density Based Traffic Signal System", Volume 3, Special Issue 3, March 2014.
2. Priyanka Khanke, Prof. P. S. Kulkarni , "A Technique on Road Trance Analysis using Image Processing", Vol. 3 Issue 2, February 2014.
3. Rajeshwari Sundar, Santhoshs Hebbar, and Varaprasad Golla, Implementing intelligent Traffic Control System for Congestion Control, Ambulance Clearance, and Stolen Vehicle Detection" IEEE Sensors Journal, Vol. 15, No. 2, February 2015.
4. Ms.Pallavi Choudekar, Ms.Sayanti Banerjee , Prof.M.K.Muju, Real Time Traffic Light Control Using Image Processing" Vol. 2, No. March.
5. S.Lokesh, “An Adaptive Traffic Control System Using Raspberry PI”, International journal of engineering sciences & research Technology, IEEE conference June 2014, pp 831-83.
6. Shabbir Bhusari, “Traffic control system using Raspberry-pi”, Global Journal of Advanced Engineering Technologies ISSN (Online), Volume 4, Issue 4- 2015, pp 413-415.MARCH 2015.
7. Soufiene Djahel, “Reducing Emergency Services Response Time in Smart Cities: An Advanced Adaptive and Fuzzy Approach”, IEEE 2015, pp 978-986.
8. George Kiokes, “Development of an Integrated Wireless Communication System for Connecting Electric Vehicles to the Power Grid”, IEEE conf. 2015, pp 296-301.