AUTOMATED FLOOR DISINFECTING ROBOT USING ARDUINO

A Mini Project Report

submitted in partial fulfilment of the requirements for the

award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICTION ENGINEERING

SUBMITTED BY

A.CHAITANYA	**16H61A0402**
J.LIKHIT SASTRY	**17H61A0417**
ABHIRAM MURARISETTY	**17H61A0433**

Under the guidance of

Dr. E.Srinivas

Associate Professor

Department of ECE

Department of Electronics and Communication Engineering

ANURAG GROUP OF INSTITUTIONS

(Formerly CVSR College of Engineering)

An Autonomous Institution

Approved by AICTE – Accredited by NBA

(Permanently affiliated to Jawaharlal Nehru Technological University, Hyderabad)

Venkatapur(V) ,Ghatkesar(M), Medchal-Malkajgiri Dist-500088

2020-2021

ANURAG GROUP OF INSTITUTIONS

AUTONOMOUS SCHOOL OF ENGINEERING

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad

Venkatapur(V),Ghatkesar(M), Medchal-Malkajgiri Dist-500088

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**AUTOMATED FLOOR DISINFECTING ROBOT USING ARDUINO**" being submitted by

A.CHAITANYA	**16H61A0402**
J.LIKHIT SASTRY	**17H61A0417**
ABHIRAM MURARISETTY	**17H61A0433**

in the partial fulfilment for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering to the Jawaharlal Nehru Technological University, Hyderabad is a record of bonafide work carried out under my guidance and supervision. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Dr. E. SRINIVAS	**Dr. S.SATHEES KUMARAN**
Associate Professor	**Head of the Department**
Department of ECE	**Department of ECE**

External Examiner

ACKNOWLEDGEMENT

This project is an acknowledgement to the inspiration, drive and technical assistance contributed by many individuals. This project would have never seen light of this day without the help and guidance we have received. We would like to express our gratitude to all the people behind the screen who helped us to transform an idea into a real application.

It is our privilege and pleasure to express our profound sense of gratitude to **Dr.E.Srinivas , Associate Professor**, Department of ECE for his guidance throughout this dissertation work.

We express our sincere gratitude to **Dr. S. SATHEESKUMARAN, Head of the Department,** Electronics and Communication Engineering for his precious suggestions for the successful completion of this project. He is also a great source of inspiration to our work.

We would like to express our deep sense of gratitude to **Dr. K.S. RAO**, **Director**, Anurag Group of Institutions for his tremendous support, encouragement and inspiration.

Lastly, we thank almighty, our parents, friends for their constant encouragement without which this assignment would not be possible. We would like to thank all the other staff members, both teaching and non- teaching, which have extended their timely help and eased my work.

A.CHAITANYA	**16H61A0402**
J.LIKHIT SASTRY	**17H61A0417**
ABHIRAM MURARISETTY	**17H61A0433**

DECLARATION

We hereby declare that the result embodied in this project report entitled "**Automated Floor Disinfecting Robot Using Arduino** " is carried out by us during the year 2020-2021 for the partial fulfilment of the award of **Bachelor of Technology** in **Electronics and Communication Engineering,** from **ANURAG GROUP OF INSTITUTION**. We have not submitted this project report to any other Universities / Institute for the award of any degree.

By

A. Chaitanya	**16H61A0402**
J.Likhit Sastry	**17H61A0417**
M.Abhiram	**17H61A0433**

ABSTRACT

Automation has its hands spread out in every field of technology and proven itself efficient than a man. Keeping such a legit point in our mind, our team has decided to take-up a project in the field of automation. Thus, we came up with the idea of making an automated floor disinfecting machine, which will drastically reduce the mundane work of cleaning floors especially in toilets and hospitals.

Our plan is to integrate automated vehicles/cars and efficient sanitizing methods to obtain our end product. We achieved this using Arduino UNO along with Ultrasonic sensor and Motor Shield as our major modules. The sanitizing mechanism would consist of UV-C lights which can kill harmful micro-organisms when exposed for a sufficient time.

TABLE OF CONTENTS

TITLE	PAGE NO.
Abstract	v
List of Figures	viii
List of Tables	ix
1.Introduction	1
1.1 Introduction	1
1.2 Embedded system	1
2.Literature survey	3
3.Methodology	6
3.1 Introduction	6
3.2 Circuit diagram	8
3.3 Flow chart	9
3.4 Algorithm	10
4.Software and hardware requirements	11
4.1Introduction to Arduino IDE	11
4.1.1 Need for Arduino IDE	11
4.1.2 Working with Arduino IDE	12
4.1.3 Creating sketches	12
4.2Introduction to Arduino	15
4.2.1 Arduino features	16
4.2.2 Arduino UNO	19
4.3 L293D motor shield	21
4.4 Bluetooth module	24
4.5 Ultrasonic sensor	25

4.6 Servo motor 26
4.6.1 Working mechanism 26
4.6.2 Working principle 27
4.6.3 Interfacing servo motors with micro controllers 28
4.6.4 Controlling servo motor 28
5.Results and discussion 30
6.Conclusion and future scope 31
7.References 32
8.Appendices 33

.

LIST OF FIGURES

Figure No	Figure Name	Page No.
1.1	General Diagram of Embedded System	2
3.1	Block diagram of Automatic Floor Disinfecting Robot	6
3.2	Arduino Uno and Bluetooth module Connections	7
3.3	Component connections with Motor shield	8
3.4	Flow chart of Automatic Floor Disinfecting Robot	9
4.1	Opening blink program on Arduino IDE	13
4.2	Source code of blink program	14
4.3	Choosing Arduino Uno board from available boards	15
4.4	Arduino Board	16
4.5	Description of Arduino board pin	17
4.6	Arduino UNO (microcontroller board)	20
4.7	IC's present on the L293D motor shield	22
4.8	L293D motor shield power terminals	22
4.9	L293D motor shield output terminals	23
4.10	Bluetooth module pin diagram	24
4.11	Working principle of the Ultrasonic sensor	25
4.12	Servo motor and it's pin diagram	28
4.13	Working mechanism of Servo motor	29

5.1 Mobile application User Interface 30
5.2 Automatic Floor Disinfecting Robot 30

LIST OF TABLES

Table No	Table Name	Page No
4.1	Technical specifications of Arduino Uno	2

CHAPTER 1

INTRODUCTION

1.1 Introduction

Keeping our surroundings clean is one thing that has to be done by everyone and doing it efficiently is also equally important. As humans we might not be able to give our best when it comes to such tiresome and repetitive works.COVID-19 has shown us that humans are vulnerable while doing such activities and this is the main reason that health care workers have been affected the most pandemic. This is where automated machines come into the picture. Automated sanitizing machines are efficient when it comes to cleaning/ sterilizing floors and in the status quo of COVID-19, the necessity of such machines is unavoidable. Our project serves the very same purpose. We have made an automated machine which is efficiently able to sterilize floors via an automated vehicle mounted with a sterilizing mechanism. Thus avoiding human intervention in such activities. Our device is certainly a go-to in such situations. The aim of this project is to implement an efficient automated floor cleaning mechanism. The project is designed for sterilizing floor using efficient UV-C lights. The Corona pandemic has shown us the necessity of using automated robots especially for sanitation purposes. And our project agrees with the previous statement.

1.2. Embedded system

An Embedded System is a combination of computer hardware and software designed to perform a specific function.The various building blocks of the hardware of an embedded system are

- Central Processing Unit (CPU)
- Memory (Read only Memory and Random-Access Memory)
- Input Devices
- Output Devices
- Communication interfaces
- Application-specific circuitry

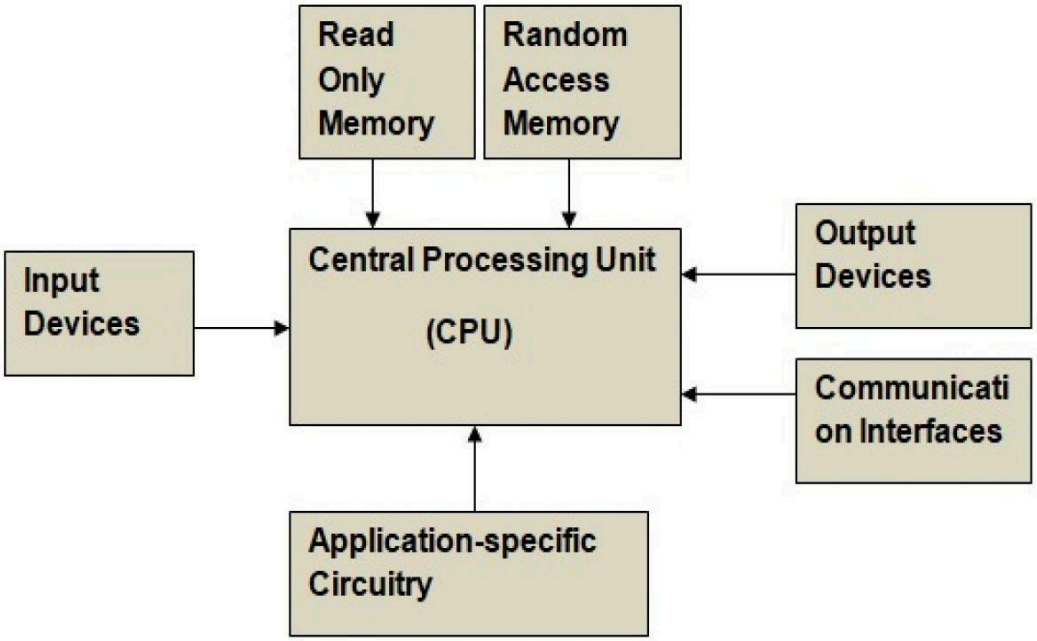

Figure 1.1 General Diagram of Embedded System

Embedded systems are based on microcontrollers (MCUs) and run software with a small memory footprint. Some Linux and Android-based systems can also be described as embedded systems. But usually, these general-purpose operating systems require an application processor, and have additional capabilities such as dynamic application loading.For 8 and 16-bit MCUs, software has often been written using a foreground/background approach (that is, a super-loop). But with 32-bit MCUs dropping in price, a real-time operating system (RTOS) is now the preferred option, allowing for more flexible and extensible software to run on these systems. A complete RTOS – with a kernel, GUI, file system, USB stack, networking, and more – can fit in a memory space of less than 1MB. With an RTOS, the software architecture of an embedded system can be more flexible. Troubleshooting and adding new features becomes dramatically simplified. It is also simpler to perform firmware upgrades. In

summary, it just makes sense to use an RTOS with a 32-bit processor. The project comes under embedded systems and is implemented using Arduino.

CHAPTER 2
LITERATURE SURVEY

As society begins to prepare for the “new normal” following the global impact of COVID-19, there are many changes that will be implemented. From how we travel, to how we interact and socialize with each other, to the increased demand for sanitization and disinfection—the impact will be far-reaching. With an increased demand for regular and consistent cleaning, there is an elevated interest in what technology is available to offset the additional time it will take cleaning teams around the globe to meet this requirement. In the last 3 months across social media, one particular technology that has been garnering attention is UVC light and UVC robots. Through these conversations, the effect of Ultraviolet-C light has been exaggerated beyond any science and practicality based on its wow factor. It went from the ability to kill a virus in 20 minutes in a highly protected environment to be able to kill a virus in seconds, with no harm to humans. Unfortunately, this buzz does not come without a negative impact on our global fight against COVID-19.Ultraviolet germicidal irradiation (UVGI) is a disinfection method that uses short-wavelength ultraviolet **(**ultraviolet C or UVC, commonly referred to wavelengths between 200 – 280 nm) light to kill or inactivate microorganisms by destroying nucleic acids and disrupting their DNA, leaving them unable to perform vital cellular functions .

The history of UVGI air disinfection has been one of promise, disappointment, and rebirth. Investigations of the bactericidal effect of sunlight in the late 19th century planted the seed of air disinfection by UV radiation. First to nurture this seed was William F. Wells, who both discovered the spread of airborne infection by droplet nuclei and demonstrated the ability of UVGI to prevent such spread. Despite early successes in applying UVGI, its use would soon wane due to a variety of reasons that will be discussed in this article. However, with the enduring research of Riley and

others, and an increase in tuberculosis (TB) during the 1980s, interest in UVGI was revitalized. With modern concerns regarding multi- and extensive drug-resistant TB, bioterrorism, influenza pandemics, and severe acute respiratory syndrome, interest in UVGI continues to grow. Research is ongoing, and there is much evidence on the efficacy of UVGI and the proper way to use it, though the technology has yet to fully mature.

In 1877,Downes and Blunt discover the ability of sunlight to prevent microbial growth. It is later shown that the ability of light to inactivate microorganisms is dependent on the dose (intensity x time) and wavelength of radiation and the sensitivity of the specific type of microorganism. In 1930 Gatos publishes the first analytical bactericidal action spectrum with peak effectiveness at 265 nm, very near the 254 nm output of low-pressure Hg germicidal lamps. In 1933 Wells presents the concept of airborne infection via "droplet nuclei" evaporated droplets containing infectious organisms that can remain suspended in the air for extended durations. In 1935 Wells and Fair demonstrate the ability of UVGI to efficiently inactivate airborne microorganisms and prove the concept of Infection via the airborne route. In 1937 Wells et al. use upper-room UVGI to prevent the epidemic spread of measles in suburban Philadelphia day schools where Infection outside the school is unlikely.In 1940s to 1950s Several studies are unable to reproduce Wells et al.'s success in using UVGI to prevent the spread of measles in schoolchildren, contributing to the disillusionment with and abandonment of UVGI for air disinfection. These failures have since been attributed to infections occurring outside the irradiated schools. In 1956-1962 Riley exposes guinea pigs to air originating from an occupied TB ward and proves that TB is spread via the airborne route. A group of guinea pigs receiving infected air via a UVGI irradiated duct were not infected, while a group receiving ar via a non-irradiated duct were infected. [5]

In 1969 to 1972 Riley and colleagues conduct model room studies evaluating the use of upper room UVGI to reduce the concentration of aerosolized test organisms in the lower room. They also show that air mixing between the upper and lower room is imperative for effective disinfection and confirm that UVGI Is less

effective at high humidity. In 1974 to 1975 Riley et al. determine virulent tubercle bacilli and BCG to be equally susceptible to UVG and measure the disappearance rate of aerosolized BCG in a model room with and without upper room UVGI. Upper toom UVGI is shown to be highly effective in disinfecting the lower room quantitatively demonstrating the potential of upper room UVGI to reduce TB infection. In 1985 to 1992 After decades of decline, there is an unexpected rise in TB in the United States, leading to a renewed interest in UVGI for air disinfection. In 1990s to present New in-depth efforts are undertaken, aimed toward quantitatively examining UVGI efficacy and safety and providing guidance for the proper use of UVG.[5]

CHAPTER 3
METHODOLOGY

3.1 Introduction

The working of an automated floor disinfecting robot can be easily explained with the help of the block diagram given below as shown in figure 3.2. As soon as we turn on the robot with adequate power supply, we have to connect our mobile phone with the Bluetooth by entering the correct credentials. Once we are connected to the network, we can set the mode of the robot, which can be either manual or automated. If the mode is automated, the ultrasonic sensor attached on the servo motor of the robot sends out an ultrasonic wave, which reflects in case an obstacle is present in the path in which the wave travels.

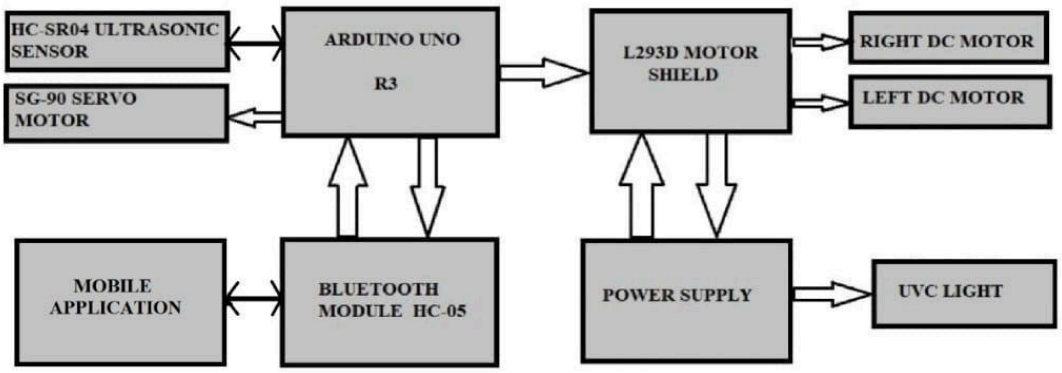

Figure 3.1 Block diagram of Automatic Floor Disinfecting Robot

Using this we are able to calculate the distance between the obstacle and the robot and based on the distance, control signals are sent to the motor shield which in turn controls the speed and direction of rotation of the gear motors. The ultrasonic sensor, which is mounted on the servo motor, turns both left and right when an obstacle is detected in front of it. Comparing the distance calculated in both the directions and following the logic in the code the micro controller sends out appropriate command signals to the motor shield which alters the direction accordingly. If the mode chosen is manual, then the Bluetooth module plays a major role. In this case it continuously searches for any serial data available and if yes then reads the data and send the data to the Arduino present on the vehicle.

3.2 Circuit Diagram

As the project is based on Arduino, the connections are straight forward. The mode of the vehicle is set based on the commands received by the Bluetooth module, which is interfaced with Arduino Uno as shown in figure 3.2. The Arduino is the heart of all the operations. In either mode, the ultrasonic sensor sends out ultrasonic waves, based on which the object is detected and the vehicle moves accordingly. The movement of ultrasonic sensor is possible since it has been attached with the servo motor.The speed of rotation, the delays etc…are defined in the code.

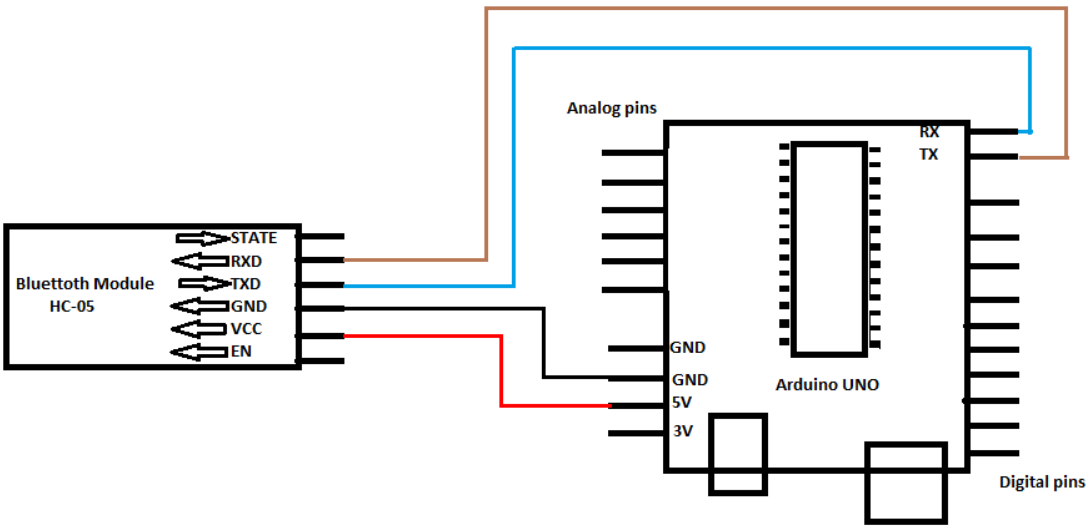

Figure 3.2 Arduino Uno and Bluetooth module Connections

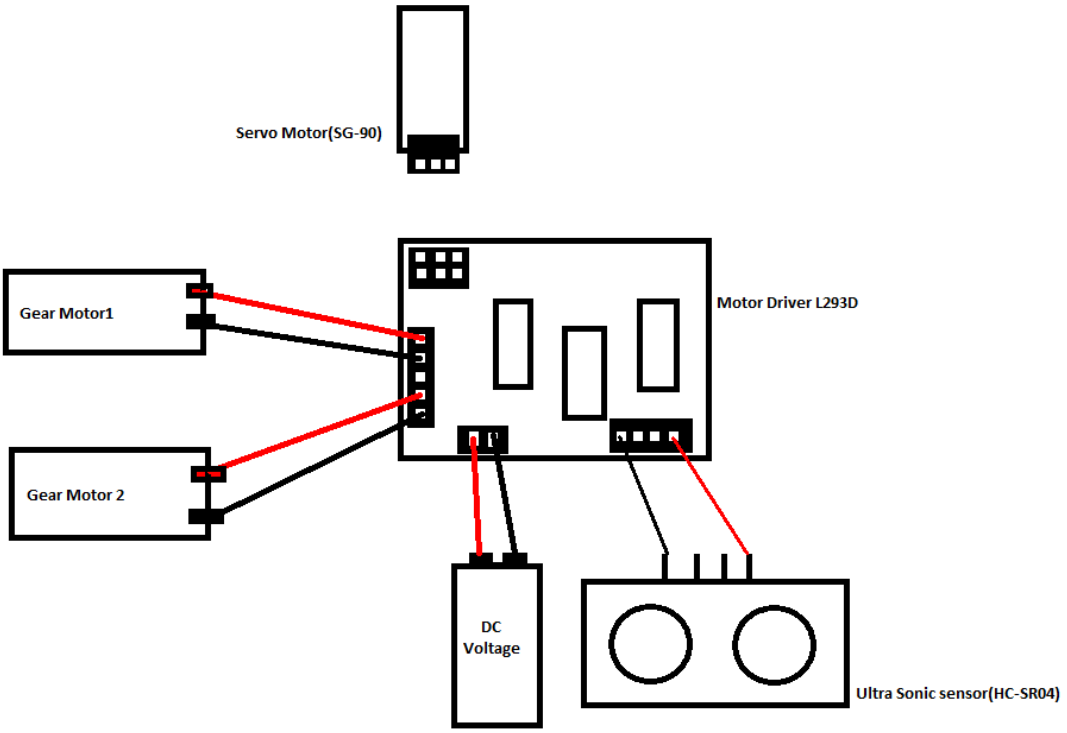

Figure 3.3 Component connections with Motor shield

3.3 Flow Chart

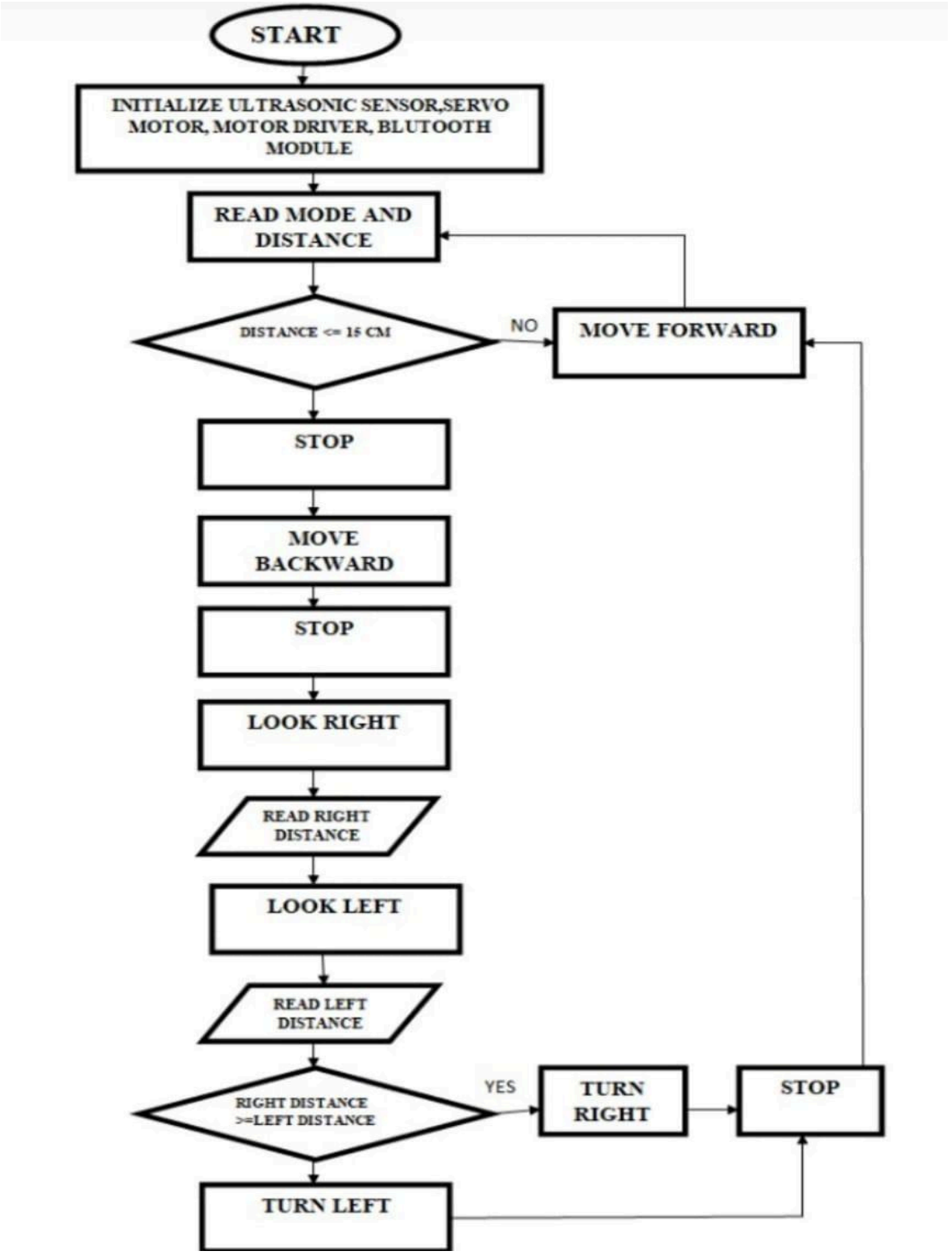

Figure 3.4 Flow chart of Automatic Floor Disinfecting Robot

3.4 Algorithm

Step 1 – Start.

Step 2 – Connect to the blue-tooth network through a Smart phone.

Step 3 – Select the mode of operation.

Step 4 – If the mode is automatic, the vehicle will automatically detect obstacles in its path with the help of ultrasonic sensor and move accordingly.

Step 5 – Simultaneously the UVC light will be disinfecting the surface.

Step 6 – If the mode is manual , the user can control the movement of the robot through the application.

Step 7 – Manual commands are received by the Bluetooth module and given to arduino.

Step 8 – Stop.

CHAPTER 4

SOFTWARE AND HARDWARE REQUIREMENTS

4.1 Introduction to Arduino IDE

Arduino IDE is an open-source electronic platform based on easy-to-use hardware and software. Arduino boards are able to read inputs like light on a sensor, a finger placed on a button, or a Twitter message; and turn it into an output to activating a motor, turn on an LED, publishing something online. One can tell the board what to do by sending a set of commands/instructions to the micro-controller on the board. To do this, the Arduino programming language (based on wiring) is used. Over the years, Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers, students, artists, programmers and professionals have gathered around this open source platform to make their contributions which added up to make an incredible amount of knowledge that is of great help to novices and experts alike. Arduino was born at Ivrea Interaction Design Institution as an easy tool for fast prototyping aimed at students without background in electronics and programming. As soon as it reached a wider community, the Arduino boards started changing to adapt to new needs and challenges; differentiating its offers from simple 8-bit boards to produce for IoT applications, 3D printings and embedded design environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt to their particular needs. The software is also open-source and it is growing by the day through the contributions of users worldwide.

4.1.1 Need for Arduino IDE

Thanks to its simplicity and accessibility nature, Arduino has been used in thousands of different projects and applications. The Arduino software is easy to use for

beginners, yet flexible enough for advance users. It runs on Mac, Windows and Linux based operating systems. Many people use it to build lowcost scientific instruments to prove physics and chemistry principles or to get started with programming, robotics and embedded system designs. Designers and architects build interactive prototypes, musicians and artists use it for installation and to experiment with new musical instruments. Makers on the other hand, use it to build many new things. Anyone can start tinkering just following the step-by-step instructions of a kit, sharing ideas online with other members of the Arduino community. There are many other micro-controllers and micro-controller platforms available for physical computing. Parallax Basic Stamp, Netmedia‟s BX-24, Phidgets, MIT‟s Handy board and many others offer similar functions. All of these tools take messy details of micro-controller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with micro-controllers and offers some advantage for interested users to develop their programming skills.

4.1.2 Working with Arduino IDE

The Arduino Integrated Development (IDE) contains a text editor for writing codes, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It can be interfaced with hardware to upload programs.

4.1.3 Creating Sketches

Programs written using Arduino software 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 as well shows the errors if any. The console displays text output by the Arduino software including complete error messages and other information or suggestions on how to correct the errors. The bottom righthand corner of the window displays the configured boards and serial port.

The toolbar buttons allow you to verify and upload programs, create, open and save sketches. It also allows you to open the serial monitor or serial plotter for value and graphical displays respectively.

Various options available on the Arduino IDE are as follows:

- Open: Presents a menu of all the sketches in your sketchbook. Clicking open will open it within the current window overwriting its contents.
- New: Creates a new sketch file.
- Save: Saves your current sketch file.
- Verify: Checks for errors in the sketch while compiling.
- Upload: Compiles the present code and uploads it to the configured board.
- Serial Monitor and Plotter: Opens the serial monitor or plotter depending which is clicked for analyzing of data.
- Additional commands are found within the five menus: File, Edit, Sketch and Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out will be available.

Steps for creating a sketch

Step 1: Open the Arduino software and click the File, Examples, Basics, then click Blink as shown in figure 4.1.

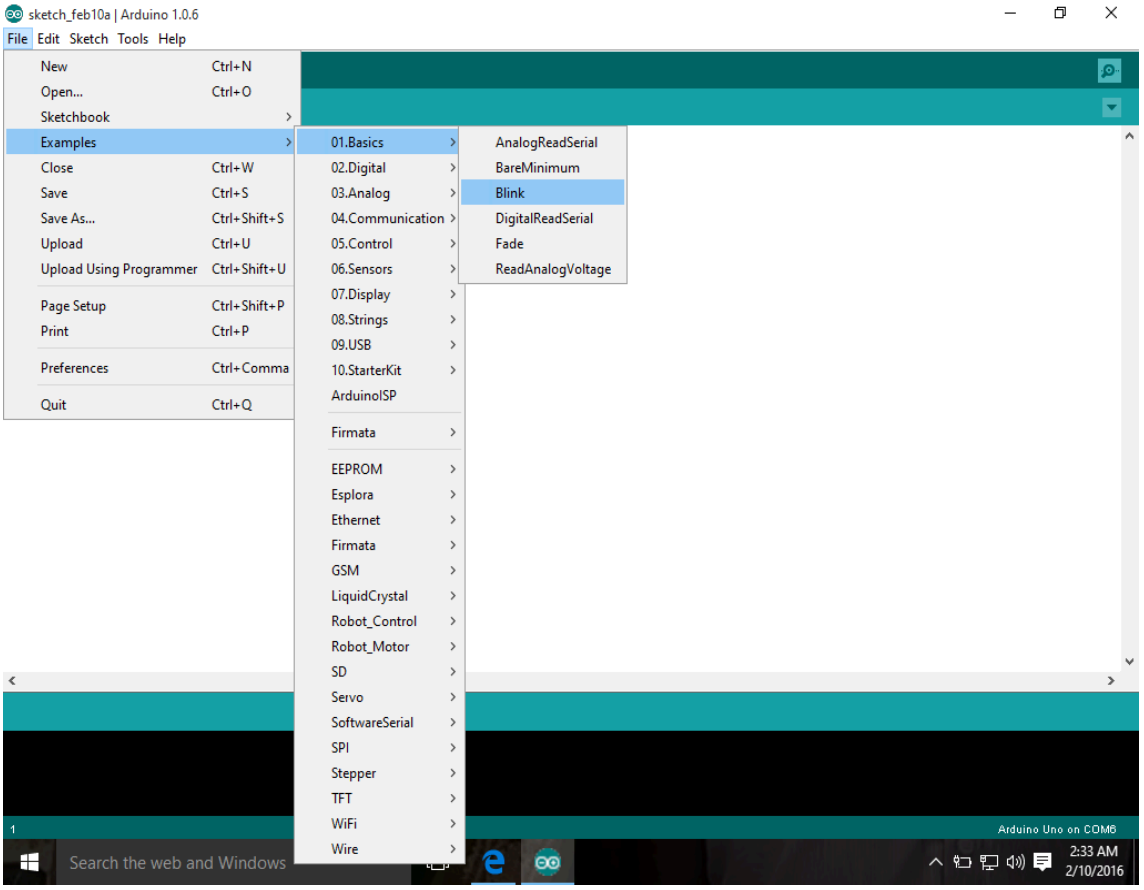

Figure 4.1 Opening Blink Program on Arduino IDE

Step 2**:** Sketch the program in the tab as shown in figure 4.2.

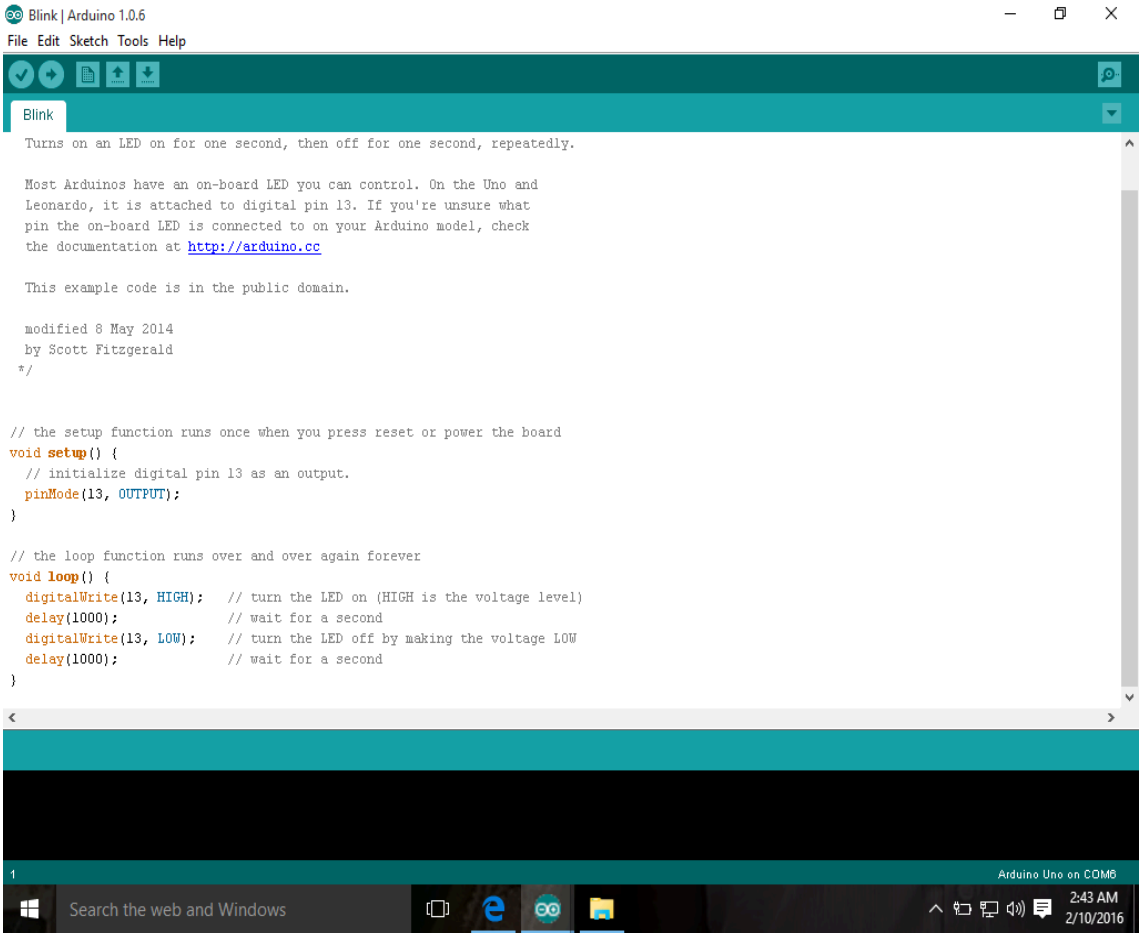

Figure 4.2 Source code of Blink program

Step 3**:** Before verifying the program change the board in the toolbar through Tools**,** Board**,**then Arduino Mega UNO or ATmega 328. Now we can verify the program and then upload the program to the board as shown in figure 4.3.

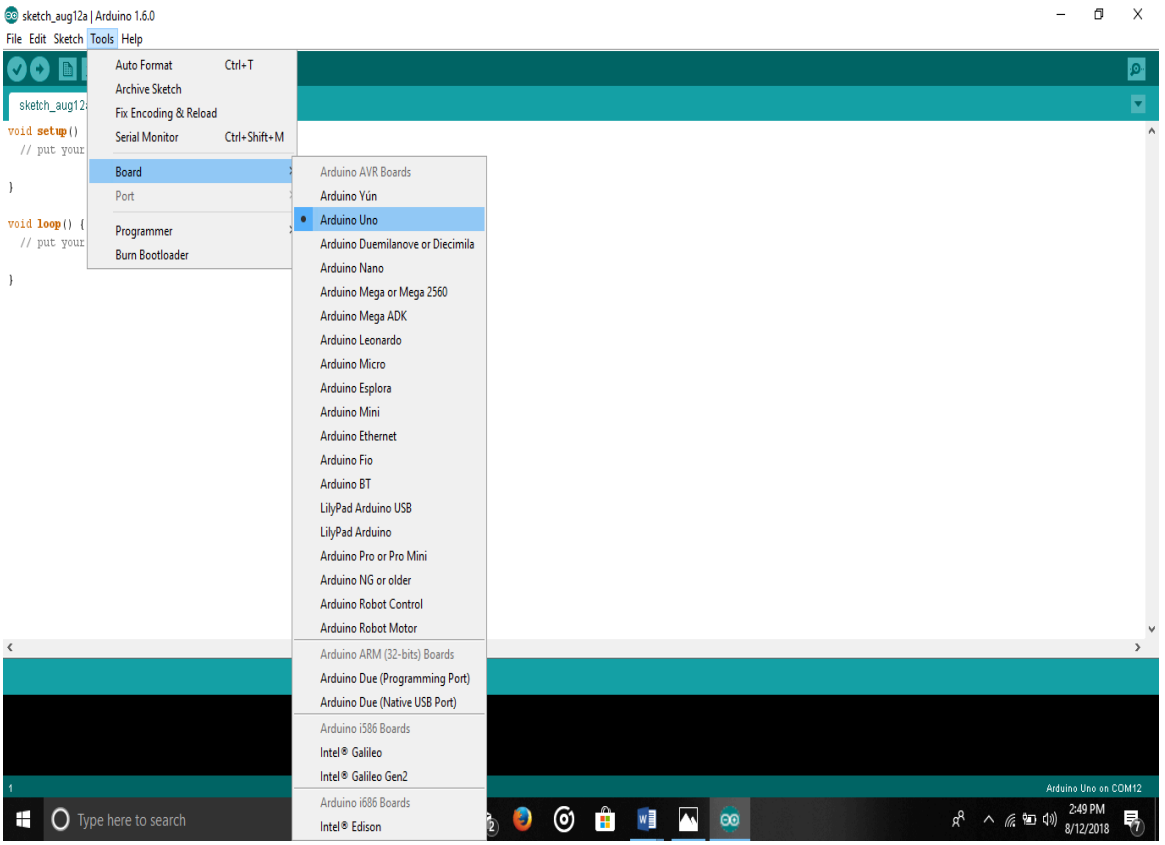

Figure 4.3 Choosing Arduino Uno board from available boards

4.2 Introduction to Arduino

The Arduino is a family of microcontroller boards to simplify electronic design, prototyping and experimenting for artists, hackers, hobbyists, but also many professionals. People use it as brains for their robots, to build new digital music instruments, or to build a system that lets your house plants tweet you when they're dry. Arduino (we use the standard Arduino Uno) are built around an AT mega microcontroller — essentially a complete computer with CPU, RAM, Flash memory, and input/output pins, all on a single chip. Unlike, say, a Raspberry Pi, it's designed to attach all kinds of sensors, LEDs, small motors and speakers, servos, etc as shown in

figure 4.4. directly to these pins, which can read in or output digital or analog voltages between 0 and 5 volts. The Arduino connects to your computer via USB, where you program it in a simple language (C/C++, similar to Java) from inside the free Arduino IDE by uploading your compiled code to the board. Once programmed, the Arduino can run with the USB link back to your computer, or stand-alone without it — no keyboard or screen needed, just power.

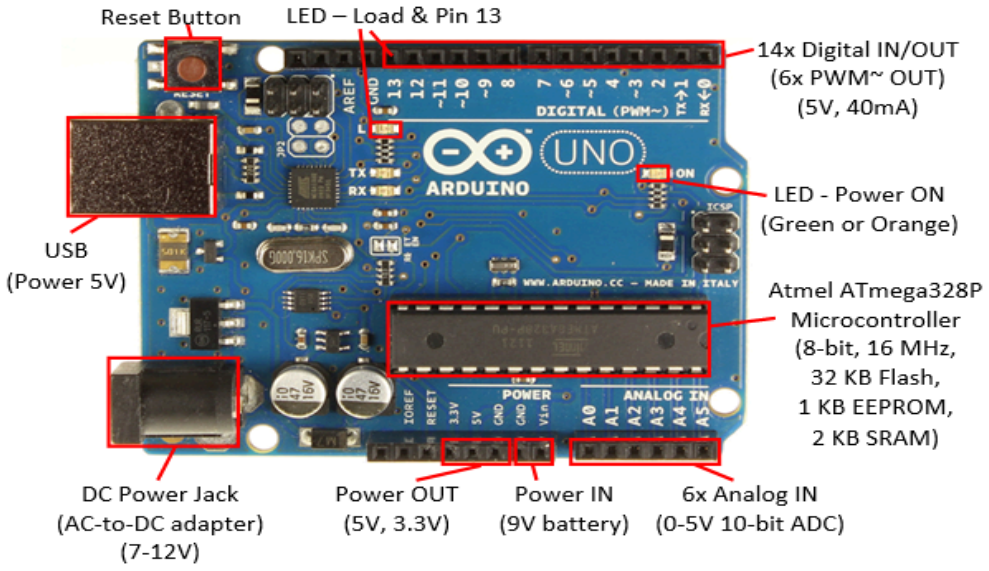

Figure 4.4 Arduino Board

4.2.1 Arduino Features

Starting clockwise from the top centre in the figure 4.5:

- Analog Reference pin (orange)
- Digital Ground (light green)
- Digital Pins 2-13 (green)
- Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - These pins cannot be used for digital i/o (Digital Read and Digital Write) if you are also using serial communication (e.gSerial.begin).
- Reset Button - S1 (dark blue)
- In-circuit Serial Programmer (blue-green)

- Analog In Pins 0-5 (light blue)
- Power and Ground Pins (power: orange, grounds: light orange)
- External Power Supply In (9-12VDC) - X1 (pink)
- Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
- USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow)

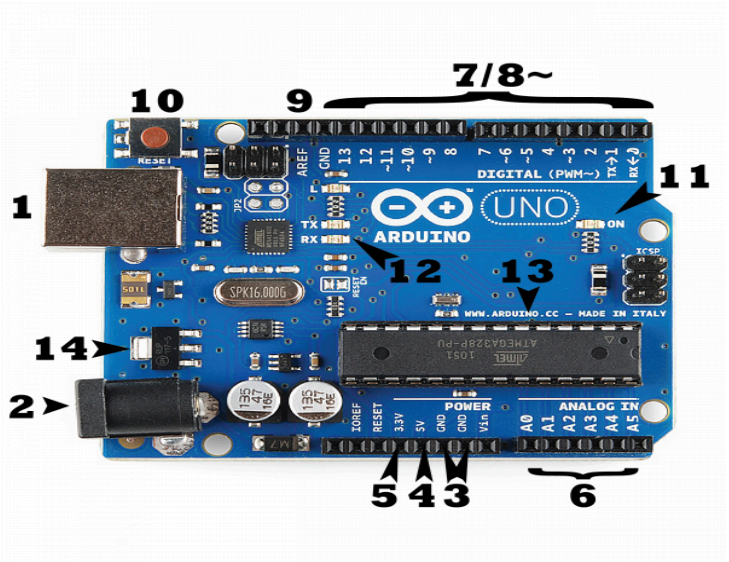

Figure 4.5 Description of Arduino board pins

Power (USB / Barrel Jack):

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply that is terminated in a barrel jack. In the picture above the USB connection is labelled **(1)** and the barrel jack is labelled **(2)**. The USB connection is also how you will load code onto your Arduino board.

Do NOT use a power supply greater than 20 Volts as you will overpower (and thereby destroy) your Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts.

Pins:(5V, 3.3V, GND, Analog, Digital, PWM, AREF)

- The pins on Arduino are the places where wires are connected to construct a circuit. They usually have black plastic ‘headers’ that allow you to just plug a wire right into the board. The Arduino has various kinds of pins, each of which is labeled on the board and used for distinct functions.
- GND (3)**:** Short for ‘Ground’. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- 5V (4) & 3.3V (5)**:** The 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run off of 5 or 3.3 volts.
- Analog (6)**:** The area of pins under the ‘Analog in’ label (A0 through A5 on the UNO) is Analog in pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
- Digital (7)**:** Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
- PWM (8): You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins but can also be used for Pulse-Width Modulation (PWM).
- AREF (9):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.

Reset Button:

The Arduino has a reset button **(10)**. Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn’t repeat, but you want to test it multiple times.

Power LED Indicator:

Just beneath and to the right of the word “UNO” on your circuit board, there’s a tiny LED next to the word ‘ON’ **(11)**. This LED should light up whenever you plug your Arduino into a power source. If this light doesn’t turn on, there’s a good chance something is wrong.

TX RX LEDs:

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs **(12)**. These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we’re loading a new program onto the board).

Main IC:

The black thing with all the metal legs is an IC, or Integrated Circuit **(13)**. It is the brain of our Arduino. The main IC on the Arduino is slightly different from board type to board type but is usually from the ATmega line of IC’s from the ATMEL Company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC.

Voltage Regulator:

The voltage regulator **(14)** is not actually something you can interact with on the Arduino. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. It is as a kind of gatekeeper; which will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don’t hook up your Arduino to anything greater than 20 volts.

Memory:

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

4.2.2 Arduino Uno

The Arduino Uno board is a microcontroller based on the ATmega328. It has 14 digital input/output pins in which 6 can be used as PWM outputs, a 16 MHz ceramic resonator, an ICSP header, a USB connection, 6 analog inputs, a power jack and a reset button. This contains all the required support needed for microcontroller.

Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board as shown in figure 4.6.

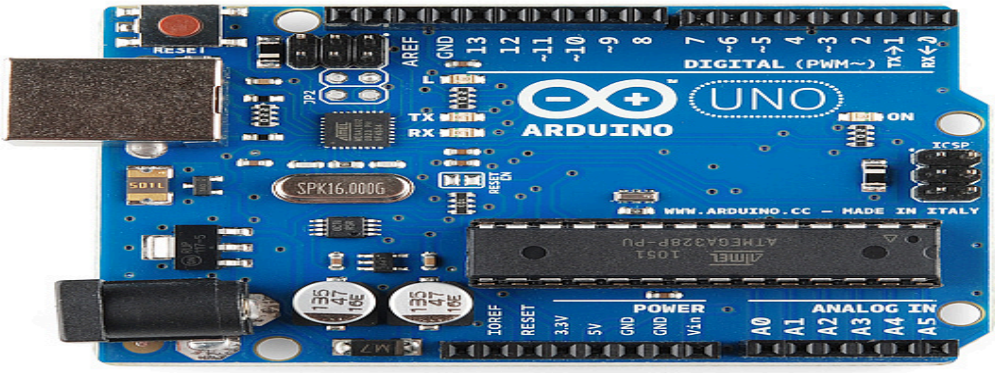

Figure 4.6 Arduino UNO (microcontroller board)

Serial 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. For SPI communication, use the SPI library.

Table 4.1 Technical Specifications of Arduino Uno

Micro controller	ATmega 328
Operating Voltage	5 V
Input Voltage(recommended)	7-12 V
Input Voltage(limits)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog pins	6

DC current per I/O Pin	40 mA
DC current for 3.3 V pin	50 Ma
Flash memory	32 KB out of which 0.5 KB used by boot loader
SRAM	2 KB
EEPROM	1 KB
Clock speed	16 MHz

4.3 L293D Motor shield

If you are planning on assembling your new robot, you will eventually want to control variety of motors like DC motors, Stepper motors & servos. One of the easiest and inexpensive way to do that is to interface L293D Motor Driver Shield with Arduino. It's a full-featured motor shield – perfect for many robot projects.

It can drive :

- 4 bi-directional DC motors with 8-bit speed selection(0-255)
- stepper motors (unipolar or bipolar) with single coil, double coil, interleaved or micro-stepping.
- servo motors

Figure 4.7 IC's present on the L293D motor shield

The L293D is a dual-channel H-Bridge motor driver capable of driving a pair of DC motors or single stepper motor.As the shield comes with two L293D motor driver chipsets, that means it can individually drive up to four DC motors making it ideal for building four-wheel robot platforms. The shield offers total 4 H-Bridges and each H-bridge can deliver up to 0.6A to the motor.The shield also comes with a 74HC595 shift register that extends 4 digital pins of the Arduino to the 8 direction control pins of two L293D chips as shown in figure 4.7.

Power Supply:

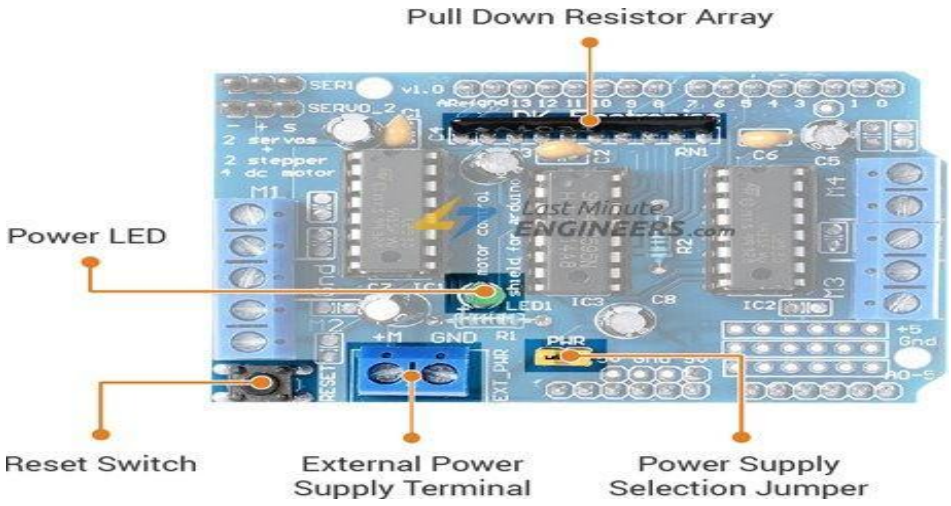

Figure 4.8 L293D motor shield power terminals

There exists three scenarios when it comes to supplying power for the motors through shield.

- Single DC power supply for both Arduino and motors. If you would like to have a single DC power supply for both Arduino and motors, simply plug it into the DC jack on the Arduino or the 2-pin EXT_PWR block on the shield. Place the power jumper on the motor shield. You can employ this method only when motor supply voltage is less than 12V.
- (Recommended) Arduino powered through USB and motors through a DC power supply. If you would like to have the Arduino powered off of USB and the motors powered off of a DC power supply, plug in the USB cable. Then connect the motor supply to the EXT_PWR block on the shield. Do not place the jumper on the shield.
- Two separate DC power supplies for the Arduino and motors. If you would like to have 2 separate DC power supplies for the Arduino and motors. Plug in the supply for the Arduino into the DC jack, and connect the motor supply to the EXT_PWR block. Make sure the jumper is removed from the motor shield.

As a bonus, the shield offers below features:

- The shield comes with a pull down resistor array to keep motors switched off during power-up.
- The on-board LED indicates the motor power supply is Okay. If it is not lit, the motors will not run.
- The RESET is nothing but Arduino's reset button. It just brought up top for convenience.

Output Terminals:

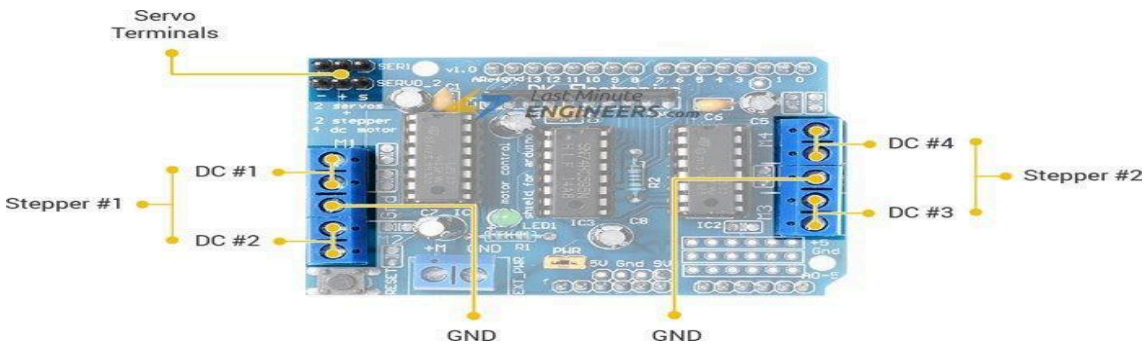

Figure 4.9 L293D motor shield output terminals

The output channels of both the L293D chips are broken out to the edge of the shield with two 5-pin screw terminals viz. M1, M2, M3 & M4. You can connect four DC motors having voltages between 4.5 to 25V to these terminals. Each channel on the module can deliver up to 600mA to the DC motor. However, the amount of current supplied to the motor depends on system's power supply. The GND terminal is also provided if you happen to have a unipolar stepper motor. You can connect the center taps of both stepper motors to this terminal.

4.4 Bluetooth module

The HC-05 has two operating modes, one is the Data mode in which it can send and receive data from other Bluetooth devices and the other is the command mode where the default device settings can be changed. We can operate the device in either of these two modes by using the key pin as shown in the figure 4.10.

It is very easy to pair the HC-05 module with microcontrollers because it operates using the Serial Port Protocol (SPP). Simply power the module with +5V and connect the Rx pin of the module to the Tx of MCU and Tx pin of module to Rx of MCU as shown in the figure below

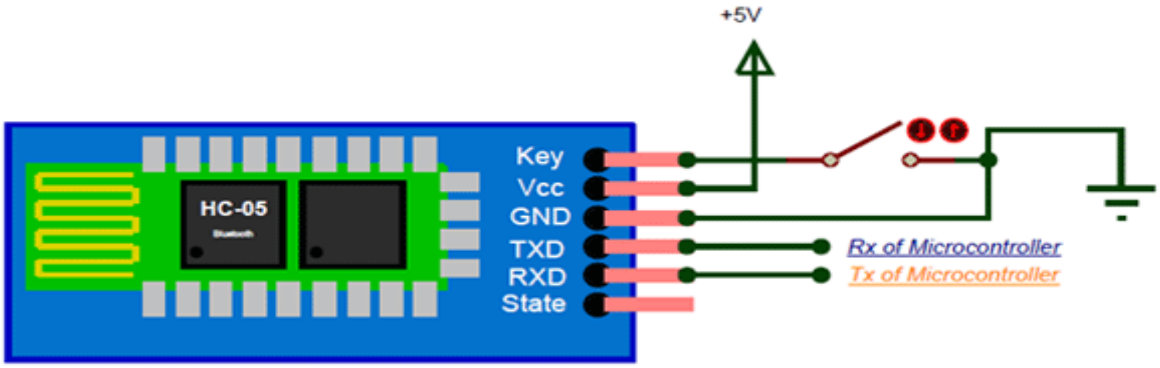

Figure 4.10 Bluetooth module pin diagram

During power up the key pin can be grounded to enter into command mode, if left free it will by default enter into the data mode. As soon as the module is powered you should be able to discover the Bluetooth device as "HC-05" then connect with it using

the default password 1234 and start communicating with it. The name password and other default parameters can be changed by entering into the command mode.

4.5 Ultrasonic Sensor

The HC-SR04 Ultrasonic (US) sensor is a 4 pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver as shown in figure 4.11. The sensor works with the simple high school formula that

Distance = Speed × Time

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module as shown in the picture below. Now, to calculate the distance using the above formulae, we should know the Speed and time. Since we are using the Ultrasonic wave we know the universal speed of US wave at room conditions which is 330m/s. The circuitry inbuilt on the module will calculate the time taken for the US wave to come back and turns on the echo pin high for that same particular amount of time, this way we can also know the time taken. Now simply calculate the distance using a microcontroller or microprocessor.

Figure 4.11 Working principle of the Ultrasonic sensor

Power the Sensor using a regulated +5V through the Vcc ad Ground pins of the sensor. The current consumed by the sensor is less than 15mA and hence can be directly powered by the on board 5V pins (If available). The Trigger and the Echo pins are both I/O pins and hence they can be connected to I/O pins of the microcontroller. To start the measurement, the trigger pin has to be made high for 10uS and then turned off. This action will trigger an ultrasonic wave at frequency of 40Hz from the transmitter and the receiver will wait for the wave to return. Once the wave is returned after it getting reflected by any object the Echo pin goes high for a particular amount of time which will be equal to the time taken for the wave to return back to the sensor.The amount of time during which the Echo pin stays high is measured by the MCU/MPU as it gives the information about the time taken for the wave to return back to the Sensor. Using this information, the distance is measured.

4.6 Servo motor

A servo motor is a type of motor that can rotate with great precision. Normally this type of motor consists of a control circuit that provides feedback on the current position of the motor shaft, this feedback allows the servo motors to rotate with great precision. If you want to rotate an object at some specific angles or distance, then you use a servo motor. It is just made up of a simple motor which runs through a servo mechanism. If motor is powered by a DC power supply then it is called DC servo motor, and if it is AC-powered motor then it is called AC servo motor. For this tutorial, we will be discussing only about the DC servo motor working. Apart from these major classifications, there are many other types of servo motors based on the type of gear arrangement and operating characteristics. A servo motor usually comes with a gear arrangement that allows us to get a very high torque servo motor in small and lightweight packages. Due to these features, they are being used in many applications like toy car, RC helicopters and planes, Robotics, etc. Servo motors are rated in kg/cm (kilogram per centimetre) most hobby servo motors are rated at 3kg/cm or 6kg/cm or 12kg/cm. This kg/cm tells you how much weight your servo motor can lift at a particular distance. For example: A 6kg/cm Servo motor should be able to lift 6kg if the load is suspended 1cm away from the motors shaft, the greater the distance the lesser

the weight carrying capacity. The position of a servo motor is decided by electrical pulse and its circuitry is placed beside the motor.

4.6.1 Working Mechanism

It consists of three parts

- Controlled device
- Output sensor
- Feedback system

It is a closed-loop system where it uses a positive feedback system to control motion and the final position of the shaft. Here the device is controlled by a feedback signal generated by comparing output signal and reference input signal.Here reference input signal is compared to the reference output signal and the third signal is produced by the feedback system. And this third signal acts as an input signal to the control the device. This signal is present as long as the feedback signal is generated or there is a difference between the reference input signal and reference output signal. So the main task of servomechanism is to maintain the output of a system at the desired value at presence of noises.

4.6.2 Working Principle

A servo consists of a Motor (DC or AC), a potentiometer, gear assembly, and a controlling circuit. First of all, we use gear assembly to reduce RPM and to increase torque of the motor. Say at initial position of servo motor shaft, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer. Now an electrical signal is given to another input terminal of the error detector amplifier. Now the difference between these two signals, one comes from the potentiometer and another comes from other sources, will be processed in a feedback mechanism and output will be provided in terms of error signal. This error signal acts as the input for motor and motor starts rotating. Now motor shaft is connected with the potentiometer and as the motor rotates so the potentiometer and it will generate a signal. So as the potentiometer's angular position changes, its output

feedback signal changes. After sometime the position of potentiometer reaches at a position that the output of potentiometer is same as external signal provided. At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer, and in this situation motor stops rotating.

4.6.3 Interfacing Servo Motors with Microcontrollers

Interfacing hobby Servo motors like s90 servo motor with MCU is very easy. Servos have three wires coming out of them. Out of which two will be used for Supply (positive and negative) and one will be used for the signal that is to be sent from the MCU. An MG995 Metal Gear Servo Motor which is most commonly used for RC cars humanoid bots etc. The picture of MG995 is shown below:

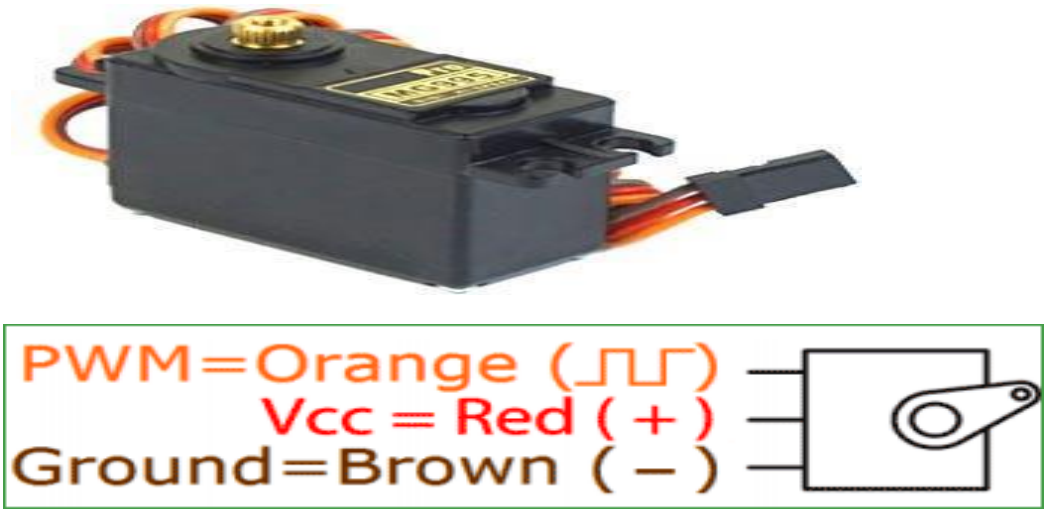

Figure 4.12 Servo motor and it's pin diagram

The colour coding of your servo motor might differ hence check for your respective datasheet as shown in figure 4.11.All servo motors work directly with your +5V supply rails but we have to be careful on the amount of current the motor would consume if you are planning to use more than two servo motors a proper servo shield should be designed.

4.6.4 Controlling Servo Motor:

All motors have three wires coming out of them. Out of which two will be used for Supply (positive and negative) and one will be used for the signal that is to be sent from the MCU. Servo motor is controlled by PWM (Pulse with Modulation) which is provided by the control wires. There is a minimum pulse, a maximum pulse and a repetition rate. Servo motor can turn 90 degree from either direction form its neutral position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90° position, such as if pulse is shorter than 1.5ms shaft moves to 0° and if it is longer than 1.5ms than it will turn the servo to 180° as shown in figure 4.13.

Servo motor works on PWM (Pulse width modulation) principle, means its angle of rotation is controlled by the duration of applied pulse to its Control PIN. Basically servo motor is made up of DC motor which is controlled by a variable resistor (potentiometer) and some gears. High speed force of DC motor is converted into torque by Gears. We know that WORK= FORCE X DISTANCE, in DC motor Force is less and distance (speed) is high and in Servo, force is High and distance is less. The potentiometer is connected to the output shaft of the Servo, to calculate the angle and stop the DC motor on the required angle.Servo motor can be rotated from 0 to 180 degrees, but it can go up to 210 degrees, depending on the manufacturing. This degree of rotation can be controlled by applying the Electrical Pulse of proper width, to its Control pin. Servo checks the pulse in every 20 milliseconds. The pulse of 1 ms (1 millisecond) width can rotate the servo to 0 degrees, 1.5ms can rotate to 90 degrees (neutral position) and 2 ms pulse can rotate it to 180 degree.

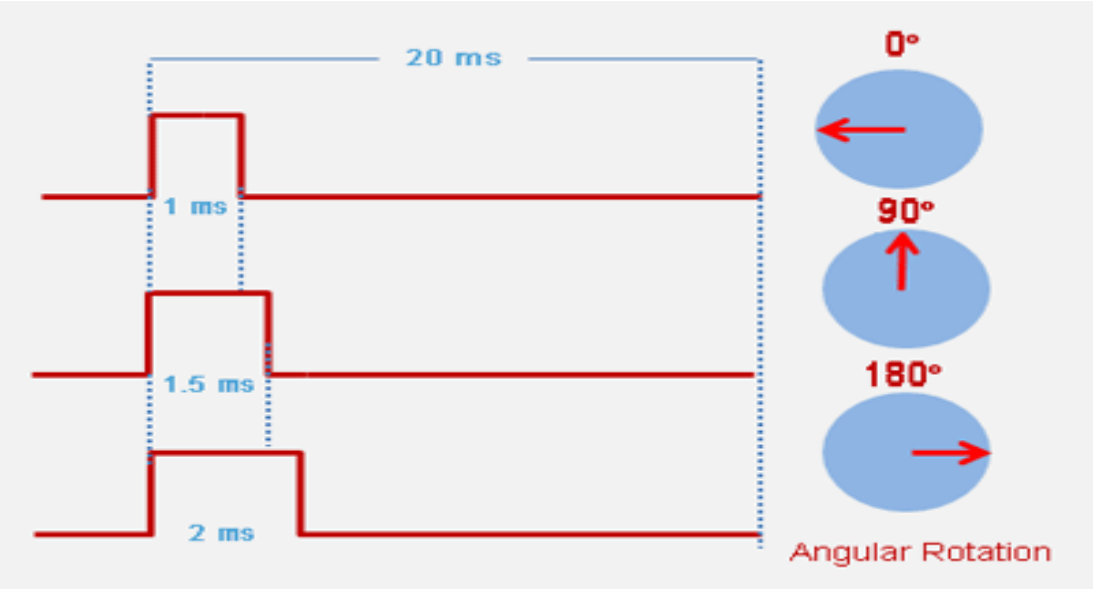

Figure 4.13 Working mechanism of Servo motor

CHAPTER 5

RESULTS AND DISCUSSION

We turn on the robot and connect it to the smart phone with the help of blue-tooth module. The mobile application is shown in figure 5.1 through which we select mode of operation and the robot acts accordingly avoiding obstacles and simultaneously disinfecting the surface using UVC lights. The Automatic Floor Disifecting robot is as shown in figure 5.2.

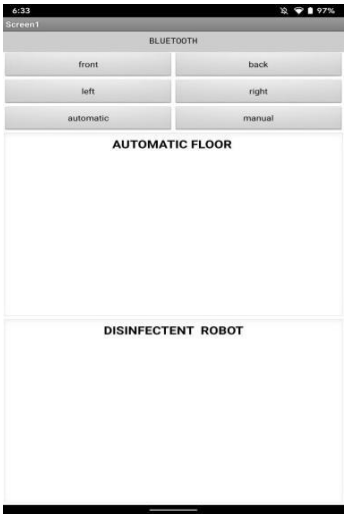

Figure 5.1 Mobile Application User Interface

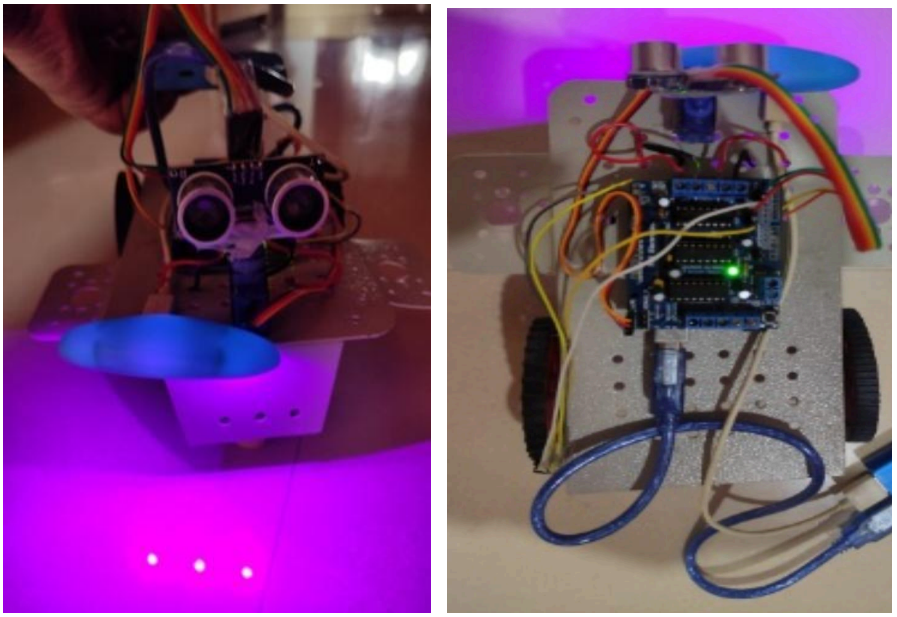

Figure 5.2 Automatic floor disinfecting robot

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

The Arduino based floor automated machine has been designed and tested in a real time situation. The user can decide the mode of operation of the device(automated/ manual) making it suitable for different scenarios. The bluetooth module used in the project gives us the flexibility to switch modes. The sterilization of surfaces using

UV-C light is a proven phenomena and works effectively to eliminate contamination from the floors .The vehicle is backed-up by a proper power source, increasing it's active time and thus providing a longer service.

Future Scope

Similar variants of such automated vehicles can be developed by making small changes like installing security cameras, electrostatic sprayers. These can be used instead of UVC lights which makes the mechanism more effective. These robots are not only useful in hospitals but also at home, other work areas etc.

REFERENCES

[1] https://learn.adafruit.com/afmotor-library-reference

[2] https://bitbucket.org/teckel12/arduino-new-ping/wiki/Home

[3] https://www.arduino.cc/reference/en/language/functions/communication/serial/

[4] https://abionline.com/is-uv-sterilization-effective-for-viruses-and-bacteria/

[5] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2789813/

APPENDICES

Source Code

```
#include <AFMotor.h>
#include <NewPing.h>
#include <Servo.h>
#define TRIG_PIN A0
#define ECHO_PIN A1
#define MAX_DISTANCE 200
#define MAX_SPEED 250
#define MAX_SPEED_OFFSET 20
NewPing sonar(TRIG_PIN, ECHO_PIN, MAX_DISTANCE);
AF_DCMotor motor3(3, MOTOR34_1KHZ);
AF_DCMotor motor4(4, MOTOR34_1KHZ);
Servo myservo;


int high=0;
boolean goesForward=false;
int distance = 100;
int speedSet = 0;
String readString;
int cm;
int readPing() {
 delay(70);
 int cm = sonar.ping_cm();
 if(cm==0)
 {
  cm = 250;
 }
 return cm;
}
int lookRight()
```

```
{
   myservo.write(30);
   delay(500);
   int distance = readPing();
   delay(100);
   myservo.write(90);
   return distance;
}
int lookLeft()
{
   myservo.write(150);
   delay(500);
   int distance = readPing();
   delay(100);
   myservo.write(90);
   return distance;
   delay(100);
}
void moveStop()
{
  motor3.run(RELEASE);
  motor4.run(RELEASE);
}
void moveForward() {

 if(!goesForward)
  {
   goesForward=true;
   motor3.run(FORWARD);
   motor4.run(FORWARD);
```

```
   for (speedSet = 0; speedSet < MAX_SPEED; speedSet +=2)
   {
    motor3.setSpeed(speedSet);
    motor4.setSpeed(speedSet);
    delay(5);
   }
  }
}
void moveBackward()
{
    goesForward=false;
    motor3.run(BACKWARD);
    motor4.run(BACKWARD);
  for (speedSet = 0; speedSet < MAX_SPEED; speedSet +=2)
  {
    motor3.setSpeed(speedSet);
    motor4.setSpeed(speedSet);
    delay(5);
  }
}

void turnRight()
{
  motor3.run(FORWARD);
  motor4.run(BACKWARD);
  delay(450);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}
```

```
void turnLeft()
{
  motor3.run(BACKWARD);
  motor4.run(FORWARD);
  delay(450);
  motor3.run(FORWARD);
  motor4.run(FORWARD);

}
void setup() {
  int readPing();
  Serial.begin(9600);
  myservo.attach(10);
  myservo.write(90);
  delay(2000);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  motor3.setSpeed(250);
  motor4.setSpeed(250);
}
void loop()
{
  while(Serial.available())
   {
```

```
  delay(50);
  char c=Serial.read();
  readString=c;
  Serial.println(readString);
}
if(readString.length()>0)
{
  Serial.println(readString);
  if(readString=="6")
    {
     Serial.println("m");
     high=1;
     readString=" ";
    }
   if(readString=="5")
   {
   Serial.println("a");
   high=2;
   readString=" ";
   }
}
if(high==2)
{
  int distanceR = 0;
  int distanceL =  0;
  delay(40);
  if(distance<=15)
  {
   moveStop();
   delay(100);
```

```
      moveBackward();
     delay(300);
      moveStop();
      delay(200);
      distanceR = lookRight();
      delay(200);
      distanceL = lookLeft();
      delay(200);
      if(distanceR>=distanceL)
      {
        turnRight();
        moveStop();
      }
      else
      {
        turnLeft();
        moveStop();
      }
      }
    else
 {
  moveForward();
 }
 distance = readPing();
  }
if(high==1)
   {
    delay(100);
    Serial.println(readString);
    if (readString =="1"){
```

```
  Serial.println("f");
  motor3.run (FORWARD);
  motor4.run (FORWARD);
  delay(500);
}
if (readString =="2"){
  Serial.println("b");

  motor3.run (BACKWARD);
  motor4.run (BACKWARD);
  delay(500);
}
if (readString =="4"){
  Serial.println("r");
  motor3.run (BACKWARD);
  motor4.run (FORWARD);
  delay(500);
}
if (readString =="3"){
  Serial.println("l");
  motor3.run (FORWARD);
  motor4.run (BACKWARD);
  delay(500);
}
if (readString =="7"){
  Serial.println("s");
  motor3.run (RELEASE);
  motor4.run (RELEASE);
  delay(500);
}
```

```
  //readString=" ";
}
```