A Mini-Project Report

on

**VOICE CONTROLLED ROBOT**

Submitted for partial fulfillment of the requirements for the award of the degree

of

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**BY**

**Ms. Thota Aakanksha (2451-17-733-133)**

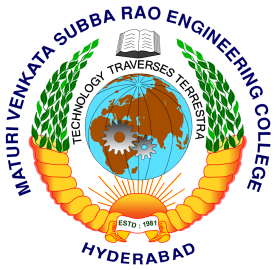
Under the guidance of

**Mr. G.Vijay Kumar**

Associate Professor

Department of CSE

M.V.S.R.E.C., Hyderabad.

****

Department of Computer Science and Engineering

M.V.S.R. ENGINEERING COLLEGE

(Affiliated to Osmania University & Recognized by AICTE)

Nadergul, Saroor Nagar Mandal, Hyderabad – 501 510

2018-19.

**M.V.S.R. ENGINEERING COLLEGE**

(Affiliated to Osmania University, Hyderabad)

Nadergul(P.O.), Hyderabad-501510



**Certificate**

This is to certify that the mini-project work entitled “**Voice Controlled Robot”** is a bonafide work carried out by **Ms. Thota Aakanksha (2451-17-733-133)** in partial fulfillment of the requirements for the award of degree of **BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING** from M.V.S.R. Engineering College, affiliated to OSMANIA UNIVERSITY, Hyderabad, under our guidance and supervision.

The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide                                                                     Faculty Coordinator

**G.Vijay Kumar                                                                                                D. Sirisha**

Associate Professor                                                                                  Assistant Professor

Department of CSE                                                       Department of CSE MVSREC, Hyderabad.                                                                           MVSREC,Hyderabad.

**DECLARATION**

This is to certify that the work reported in the present mini-project entitled “**Voice Controlled Robot”** is a record of bonafide work done by me in the Department of Computer Science and Engineering, M.V.S.R. Engineering College, Osmania University. The reports are based on the mini-project work done entirely by us and not copied from any other source.

The results embodied in this mini-project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of my knowledge and belief.

Thota Aakanksha

2451-17-733-133

**ACKNOWLEDGEMENTS**

I would like to express my sincere gratitude and indebtedness to my mini-project guide **G. Vijay Kumar** for his valuable suggestions and interest throughout the course of this mini-project.

I am also thankful to our principal **Dr.G.Kanaka Durga** and **Dr.Akhil Khare**, Professor and Head, Department of Computer Science and Engineering, MVSR Engineering College, Hyderabad for providing excellent infrastructure for completing this mini-project successfully as a part of our B.E. Degree (CSE). I would like to thank my mini-project coordinator **D.Sirisha** for their constant monitoring, guidance and support.

I convey my heartfelt thanks to the lab staff for allowing me to use the required equipment whenever needed.

Finally, I would like to take this opportunity to thank my family for their support through the work. I sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

Thota Aakanksha (2451-17-733-133)

**ABSTRACT**

A robot is usually an electro mechanical machine that is guided by computer and electronic programming. The purpose of robotics in commercial and residential intentions is becoming essential to make the challenging tasks simpler. Our project is one of the applications on vehicles. We aim to achieve human robotic interaction. It will be useful for applications such as speech controlled vehicle. It can be used for indoor assistance for household works and especially for people with disabilities. This also contributes a major part in industrial applications such as work robots. Security has been a major issue these days. This project can be useful in surveillance. The robot is also useful in places where humans find difficult to reach but human voice can reach e.g. in fire situations. Our proposed model is a prototype of a robotic vehicle which moves as per the voice instructions given by the user. The commands are given through the Google Assistant app. They are then sent to Adafruit.io feed using IFTTT( if +this then +that) platform. The open loop microcontroller ESP8266 reads the commands from Adafruit.io through the code and moves the vehicle accordingly.

**TABLE OF CONTENTS**

                                                                                                                          PAGE NOS.

Certificate ......................................................................................................... i

Declaration ....................................................................................................     ii

Acknowledgements ........................................................................................ iii

Abstract............................................................................................................  iv

Table of contents.............................................................................................. v

List of Figures .................................................................................................    vi

List of Tables .................................................................................................... vii

CONTENTS INSIDE THE DOCUMENT

CHAPTER I

1. INTRODUCTION
   1. PROBLEM STATEMENT
   2. PROPOSED SYSTEM
   3. SCOPE OF THE MINI-PROJECT

CHAPTER II

1. TOOLS AND TECHNOLOGIES
   1. LITERATURE SURVEY
   2. HARDWARE REQUIREMENTS
   3. SOFTWARE REQUIREMENTS

CHAPTER III

1. SYSTEM DESIGN
   1. BLOCK DIAGRAM
   2. SYSTEM ARCHITECTURE

CHAPTER IV

1. SYSTEM IMPLEMENTATION & METHODOLOGIES
   1. ENVIRONMENT SETUP
   2. ALGORITHM
   3. CODE SKELETON
   4. CODE

          CHAPTER V

1. CONCLUSION & FUTURE ENHANCEMENTS

           REFERENCES/ BIBLIOGRAPHY

**LIST OF FIGURES**

|  |
| --- |
| **Figure no NAME OF FIGURE Page no**  Figure 1.2.1 Transmission of voice commands 11  Figure 2.2.1.2 Wemos d1 mini esp8266 microcrontroller 12  Figure 2.2.1.4 Pin diagram of wemos d1 mini 13  Figure 2.2.2 L298N Motor Driver 15  Figure 2.2.2.3 Pin description of L298N Description 16  Figure 2.2.3.1 Robot Chasis 17  Figure 2.2.4.1 12 v rechargeable battery 17  Figure 2.2.4.2 3.7 v rechargeble battery 18  Figure 2.2.5 3S 18650 Battery holder 21  Figure 2.2.6 18650 battery charger 22  Figure 2.2.7 Jumpers 22  Figure 2.2.8 Micro USB Cable 23  Figure 2.3.1 Google Assistant 24  Figure 2.3.2 Adafruit 25  Figure 2.3.3 MQTT Protocol 25  Figure 2.3.2.3 Feeds 27  Figure 2.3.2.5 Creating Feed 28  Figure 2.3.2.6 IFTTT 28  Figure 2.3.3.2 Creating applets 29  Figure 2.3.4 Arduino 31  Figure 3.1.1 Block Diagram 35  Figure 3.1.2 IFTTT Smartphone pic 36  Figure 3.1.3 if + this then that 37  Figure 3.1.4 Google assistant in smartphone 37  Figure 3.1.4 Choose Trigger 38  Figure 3.1.5 Complete trigger fields 39  Figure 3.1.8 Choose action service 41  Figure 3.1.9 Choice action 41  Figure 3.1.10 Complete action field 42  Figure 3.1.11 Review and finish 43  Figure 3.1.12 Adafruit IO 43  Figure 3.1.13 Creating feed in adafruit 44  Figure 3.1.14 Arduino IDE 45  Figure 3.2.1 Schematic Diagram 47 |

**LIST OF TABLES**

Table no.

**Table No Name of Table Page no**

Table 2.2 Table 1 Technical Specification 14

Table 2.2. Table 2 Pin functioning of ESP8266 17

Table 2.2. Table 3 Pin inputs & Motor directions 21

**CHAPTER 1**

**INTRODUCTION**

The purpose of robotics in commercial & residential intention has come to be quite essential for executing challenging work into more conveniently simple way. There are a lot of researches working on to enhance the connection between humans and robot. Our project presents the research of the designing & development of a voice controlled talking robot using mobile phone based on Arduino. The control system of the robot movement will be employed by the voice and the robot will response the commanding persons by generating sounds of human voice with each verbal instruction. The proposed system will be designed based on microcontroller which is connected to smart android phone through Wifi module for receiving voice command. The voice command is converted to text by an app of the android phone and sends necessary data to the microcontroller for controlling robot movement. After receiving the data the robot responses according to the command by performing proper movement to the proper direction according to the voice command. After getting each command the robot will act according to the instruction and will be able to speak different sentences

**1.1 PROBLEM STATEMENT**

**“**Aim to achieve human robotic interaction. A prototype of robotic vehicle which moves as per the voice instructions given by the user.”

**1.2 PROPOSED MODEL**

Voice controlled robot using Google assistance is one of those ideas which could pave path for human machine easy interaction. This model performs according to the instructions given to the robot using Google assistance. The command has a specific user defined format for moving forward, backward and turning in a particular direction. This involves two sections:-

* Transmission section
* Receivers section

**1.2.1 TRANSMISION SECTION**

Initially commands are given through Google aasistant using the shown format of commands:-

* Turn left ‘x’ degrees
* Turn right ‘x’ degrees
* Move forward ‘y’
* Move back ‘y’

Where x is the angle to b specified and y is the distance in centimeters. The obtained commands are send to the adafruit IO feed and then processed further.

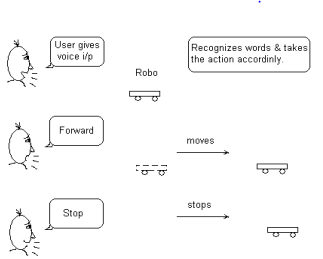


Fig.1.2.1 Transmission of voice commands

**1.2.2 RECIEVER SECTION**

Once the commands stored in adafruit IO feed ,now they must be accessed by the ESP8266 board. This process takes place through a key and password provided by the adafruit. This key place a receivers role in transmitting the data accessed by the feed to the ESP8266 board. Once the command is accessed by the board , it is stored in a string and the code runs as coded and the execution takes place accordingly. Thus the motors receive the signaling inputs from the esp as coded. This makes the robot move in the way specified.

**1.3 SCOPE OF THE MINI PROJECT**

* Speech controlled driving

Helps the user to control the car/machine through voice commands. Hence he could drive to any possible place.

* Surveillance application

Monitoring has become a major importance these days.

* Industrial robots

Robots working in industries could reduce life loss and improve efficiency.

* Indoor assistance

Assistance for the one who cannot perform certain tasks (permanently/temporarily) can be looked after by the robot.

* The robot is useful in places where humans find difficult to reach but human voice can reach e.g. in fire situations

**CHAPTER-2**

**TOOLS AND TECHNOLOGIES**

**2.1 LITERATURE SURVEY**

Many research on the design of robotic vehicle using  microcontroller have been reported in the literature.

    Omijeh et al. [1] used an RF transmitter to control a robotic vehicle. Their work used a robotic arm of five degrees of freedom with its base resting directly on top of

the vehicle, a body having four drive wheels coupled to the ends thereof. The wheels were selectively powered to propel the vehicle. The design methodology involves the

hardware and software part. A prototype of the remote-controlled pick and place robotic vehicle was built to validate design specifications. The motion results obtained

were very satisfactory but due to the number of motors clustered on the robotic arm, maximum weight carrying capacity was affected by the weight of the motors. Another

drawback of their proposed system is that RF signals faced disturbance from interference.

    Ritika and Narender [2] used an android based on-screen remote control application to operate a robotic vehicle. Remote buttons in the Android app are the means by

which motion control of the robot is achieved while android communication to the controller is interfaced via Bluetooth using UART protocol. The robot was able to

move forward, backward, left and right side using the android application. The device can be reprogrammed but the absence of an appendage limits its application to motion only.

    Mohammed et al. [3] used a robotic arm vehicle with a soft catching gripper which avoids extra pressure on the object. The robotic vehicle was controlled by an android

application called blue control which had a graphical user interface (GUI). The android application acts as a remote transmitter having the advantage of adequate range, while

the receiver Bluetooth device is connected to the microcontroller to drive D.C. motors via motor driver I.C. for necessary operation. The microcontroller used was Atmega328 and Arduino software used to program it. The pick and place robot had four motors for the operation of the system, two for the operation of moving vehicle having a belt linking the wheels like in the tanks and two for the pick and place operation. The maximum upward and downward motion of the arm was limited by a mechanical push-button type switch. The vehicle was able to move along surfaces being smooth or rough in the direction of forward, backward and left and right while able to carry a maximum weight of 2kg. The device was however on-screen remote controlled rather than voice controlled.

    Bhuvaneshwari et al. [4] constructed a voice-controlled robotic vehicle using 8051 microcontrollers. The two-wheel robot powered by D.C. battery used an android

application called AMR Voice as the transmitter and an HC-05 Bluetooth I.C. as the receiver. The robotic vehicle was able to move forward, backward, left and right using

the voice commands or Bluetooth. The output of the proposed system was not discussed in their write-up neither was the performance of the system evaluated.

Russell and Norvig [5] proposed cooperative Robotics. This has to do with the use of multiple robotics agents assisting each other to perform a task that is either too

difficult or impossible for a single robot to perform alone. The study presents a search-and-rescue algorithm, referred to as SARA-1 that is designed to enable a team of

cooperative autonomous robots to search an area for a stationary target. The robots use wireless communication to build and share collective maps of the environment.

They attempt to spread out their cooperative search and being careful not to explore the same area again. This algorithm is pertinent to both indoor and outdoor

applications. The range of the applications is limited only by the user's imagination and might include such tasks as hazardous waste location and removal, planetary exploration, warehouse organization, and human search-and-rescue.

**2.2 HARDWARE REQUIREMENTS**

**2.2.1 WEMOS D1 MINI ESP826**

**2.2.1.1BASIC INFORMATION**

TECHNICAL SPECS

| **Microcontroller** | **ESP-8266EX** |
| --- | --- |
| Operating Voltage | 3.3V |
| Digital I/O Pins | 11 |
| Analog Input Pins | 1(Max input: 3.2V) |
| Clock Speed | 80MHz/160MHz |
| Flash | 4M bytes |
| Length | 34.2mm |
| Width | 25.6mm |
| Weight | 10g |

**2.2.1.2 DESCRIPTION**

WeMos is a company that develops low cost-effective Internet of Things (IoT) devices for various projects and products. WeMos D1 Mini series are one of the products developed by the company to enable wireless connectivity, simple data traffic and electronic controlling to gramming. electronic projects at the same time

WeMos D1 mini is a mini WiFi board based on ESP8266EX. This board is Arduino IDE compatible, therefore it can be programmed using Arduino or its own Lua compiler. It also supports both serial and OTA programming. This small development board has 4MB Flash memory, based on the specifications of ESP8266EX, has 11 digital IO pins, all IO pins capable of Interrupt / PWM / I2C / 1-wire except D0 pins. It has 1 Analog input (3.3V max) and a Micro USB port to connect with PC for communication and programing.

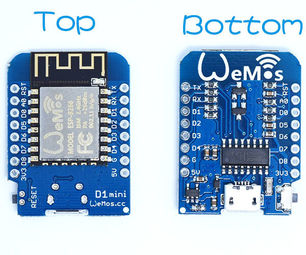


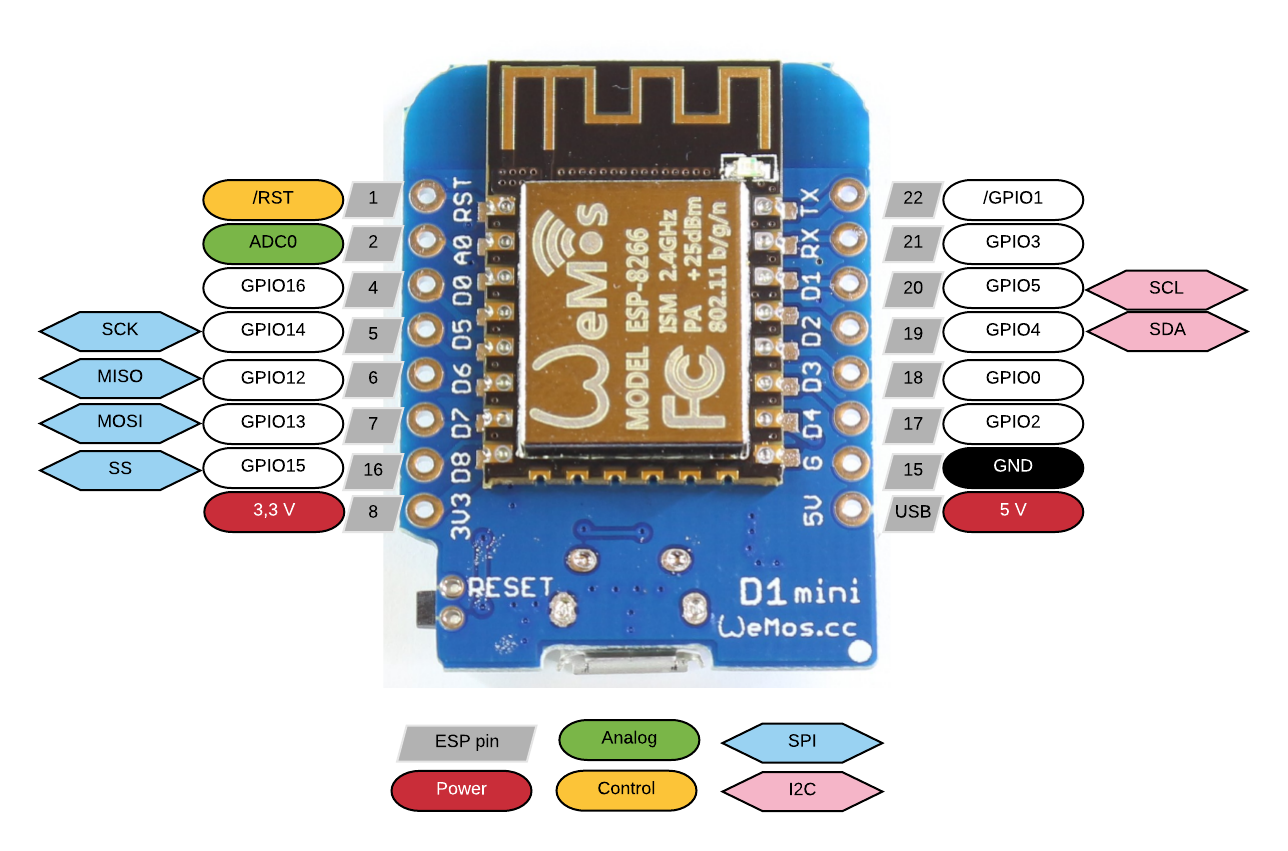
Figure 2.2.1.2 WEMOS D1 MINI ESP8266VMICROCONTROLLER

Wemos D1 mini is like a "little Arduino with wifi" for a great price. It's based around the ESP8266, has one analogue port and 11 digital ports. It's programmed via micro-USB (or remote flash via wifi). You can use it with the Arduino IDE, micropython or NodeMCU..It runs from 5V or 3.3V. Logic levels are 3.3V for all ports. If you use the Arduino IDE, there are a lot of example sketches already provided. This makes life relatively easy to be able to do what you want.I use these to [read voice commands and send the data to ESP8266.](http://raspi.tv/2017/wireless-remote-sensing-with-wemos-d1-mini-arduino-ide-raspberry-pi-and-lighttpd-web-server)

 Wemos D1 board is really easy to use and program with Arduino IDE. It has the same footprint of and ordinary Arduino Uno! This way most of Arduino shield will also work with this board. It has built-in Wi-Fi module, so you can use it in a variety of projects.You can also use other ESP8266 based boards

* + - 1. **FEATURES**
* 11 digital input/output pins,all pins have interrupt/pwm/12c/one-wire supported(exceptD0)
* 1 analog input(3.2V max input)
* A micro USB connection
* Compatible with MicroPython,Arduino,nodemcu

**2.2.1.4 PIN DIAGRAM**

 Figure 2.2.1.4 pin diagram of WEMOS D1 MINI

Pin Description

| **Pin** | **Function** | **ESP-8266 Pin** |
| --- | --- | --- |
| TX | TXD | TXD |
| RX | RXD | RXD |
| A0 | Analog input, max 3.3V input | A0 |
| D0 | IO | GPIO16 |
| D1 | IO, SCL | GPIO5 |
| D2 | IO, SDA | GPIO4 |
| D3 | IO, 10k Pull-up | GPIO0 |
| D4 | IO, 10k Pull-up, BUILTIN\_LED | GPIO2 |
| D5 | IO, SCK | GPIO14 |
| D6 | IO, MISO | GPIO12 |
| D7 | IO, MOSI | GPIO13 |
| D8 | IO, 10k Pull-down, SS | GPIO15 |
| G | Ground | GND |
| 5V | 5V | - |
| 3V3 | 3.3V | 3.3V |
| RST | Reset | RST |

**2.2.2 L298N MOTOR DRIVER**

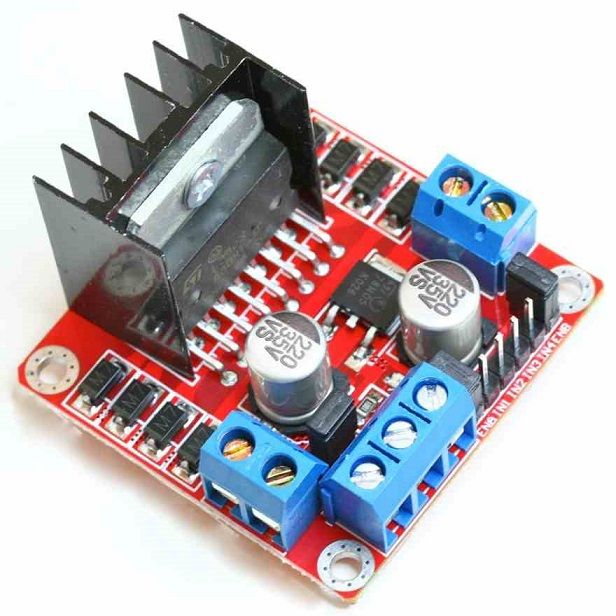


Figure 2.2.2

**2.2.2.1 INTRODUCTION**

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals.The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.Check [**here**](https://wiki.eprolabs.com/images/a/ad/MOT-0008.pdf) for L298 datasheet.

**2.2.2.2 SPECIFICATIONS**

Motor Driver:L298N

* Driver power supply: +5V~+46V
* Driver peak current: 2A
* Logic power output Vss: +5~+7V (internal supply +5V)
* Logic current: 0~36mA
* Controlling level: Low -0.3V~1.5V, high: 2.3V~Vss
* Enable signal level: Low -0.3V~1.5V, high: 2.3V~Vss
* Max drive power: 25W (Temperature 75 )
* Working temperature: -25 ~+130

**2.2.2.3 PIN DESCRIPTION**

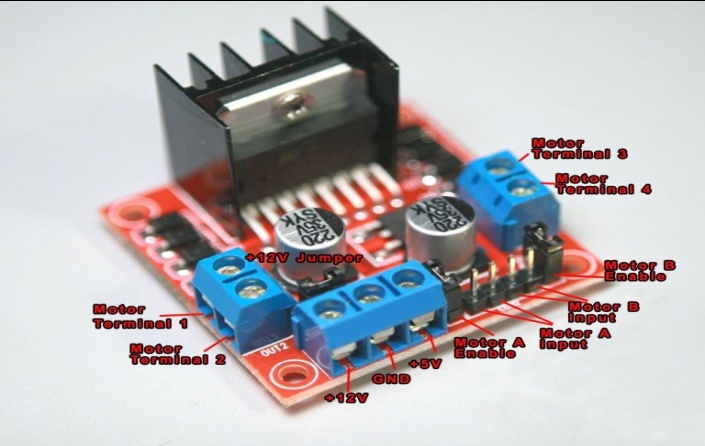


Figure 2.2.2.3

* **Out 1:** Motor A lead out(+ve)
* **Out 2:** Motor A lead out(-ve)
* **Out 3:** Motor B lead out(+ve)
* **Out 4:** Motor B lead out(-ve)
* **12V:**12V input from DC power source(12V jumper – remove this if using a supply voltage greater than 12V DC. This enables power to the on board 5V regulator)
* **GND:** Ground
* **5V:** 5V input (unnecessary if our power source is 7v-35v, if the power source is 7v-35v then it can act as a 5v out)
* **EnA:** Enables PWM signal for Motor A
* **In1:** Enable Motor A
* **In2:** Enable Motor A
* **In3:** Enable Motor B
* **In4:** Enable Motor B
* **EnB:** Enables PWM signal for Motor B

**How To connect motor to L298N H Bridge Motor Driver**

**Hardware and Software Required**

* L298 H Bridge Drive
* Arduino Uno
* Arduino IDE

### Hardware Connections

Here a single DC motor is used and controlled by the H-Bridge driver.It should be connected to Arduino IDE as follows:

**H-Bridge inputs/outputs:**

* ENB pin => Wemos D3 pin
* IN4 pin => Wemos D4 pin
* IN3 pin => Wemos D5 pin
* IN2 pin => Wemos D6 pin
* IN1 pin => Wemos D7 pin
* ENA pin => Wemos D8 pin
* 5V pin => Wemos 5V pin
* Gnd pin => Wemos Gnd pin
* Gnd pin => Battery pack negative wire
* 12V pin => Battery pack positive wire
* OUT1 => Right motor negative wire
* OUT2 => Right motor positive wire
* OUT3 => Left motor positive wire
* OUT4 => Left motor negative wire



**2.2.3 DIY ROBOT CHASSIS TANK**

This awesome kit has with everything you need to built a tank: two DC motors, gears, tracks, bolts, nuts, etc. It already comes with the tools need for assembling the chassis, which is great for beginners!. or else we can buy a car which is already fixed which contains 2 motors ,so that we can save our time



**Fig 2.2.3.1**

**2.2.4 18650 3.7 BATTERIES (X3)**

I used to power the whole circuit. This tank uses 12V motors. I used 12v rechargeable batteries for powering the circuit.Or else we can use three 3.7 v batteries which are rechargeble

Figure 2.2.4.1 3.7v Battery

If we didn’t find 12v rechargeble battery the we can use three 3.7v rechargeble battery.

Figure 2.2.4.2 3.7v rechargeble battery

**2.2.53S 18650 BATTERY HOLDER**

If we use 3.7v rechargeable batteries then we need a battery holder. . It can hold three 18650 batteries in series, and can be easily be attached to the back of the tank.



Figure 2.2.5 3S 18650 Battery holder

**2.2.6 18650 BATTERY CHARGER**

As batteries will eventually run out of power. When that happens, a battery charger will come to rescue.



Figure 2.2.6 18650 battery charger

**2.2.7 JUMPERS**

I used 6 male-female jumpers for signals between the h-bridge an the Wemos, and 2 male-male jumpers for 5V and Gnd. You might need more if you plan to add some sensors.

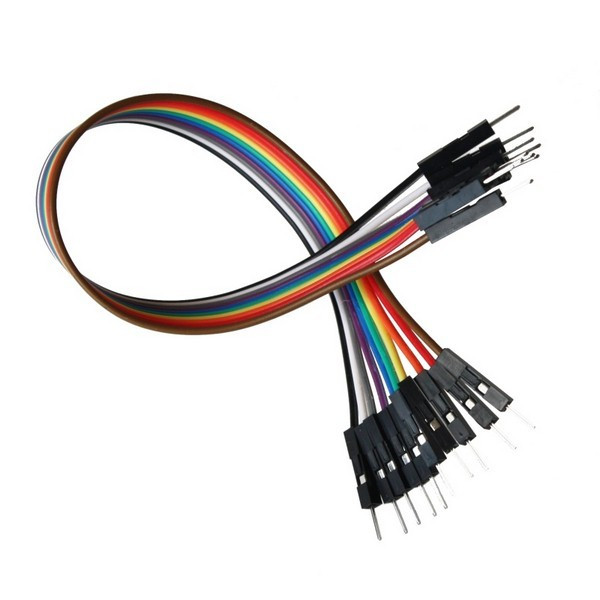


Figure 2.2.7 Jumpers

**2.2.8 MICRO USB CABLE**

You'll need this for uploading our code. Most of the boards already come with its own cable



Figure 2.2.8 Micro USB cable

**2.3 SOFTWARE REQUIREMENTS**

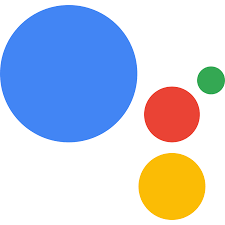
1. GOGGLE ASSISTANT

2. ADAFRUIT.IO

3. IFTTT

4. ARDUINO IDE

**2.3.1 GOOGLE ASSISTANT**

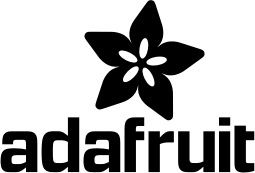
****

**Fig 2.3.1**

Google Assistant is a virtual assistant developed by Google. It's artificial intelligence is able identify voice commands, perform searches, schedule events and alarms, adjust hardware settings on the user's device, and show information from the user's Google account.

Users primarily interact with Google Assistant through natural voice, though keyboard input is also supported. Google Assistant can engage in two-way conversations, using Google's natural language processing algorithm. Google Assistant allows users to activate and modify vocal [shortcut](https://en.wikipedia.org/wiki/Shortcut_(computing)) commands in order to perform actions on their device or configuring it as a hub for [automation](https://en.wikipedia.org/wiki/Home_automation#Applications_and_technologies). It has a great integration with IFTTT service.

**2.3.2 ADAFRUIT.IO**

****

**Fig 2.3.2**

Adafruit.io is a data logging service used for communicating a microcontroller to the web. It is a cloud service. It's meant for storing and then retrieving data from the cloud.

Adafruit.io can do the following tasks:

* Display the data in real-time, online.
* Makes the project internet-connected: Control motors, read sensor data, and more.
* Connects projects to web services like Twitter, RSS feeds, weather services, etc.

Adafruit.io works based on MQTT protocol

**2.3.2.1 MQTT PROTOCOL**

MQTT (Message Queue Telemetry Transport) is a [publish-subscribe](https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern)-based messaging protocol. It works on top of the [TCP/IP protocol](https://en.wikipedia.org/wiki/TCP/IP). The [publish-subscribe messaging pattern](https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern) requires a [message broker](https://en.wikipedia.org/wiki/Message_broker). This system consists of clients communicating with a server, often called a "broker". A client may be either a publisher of information or a subscriber. Each client can connect to the broker.

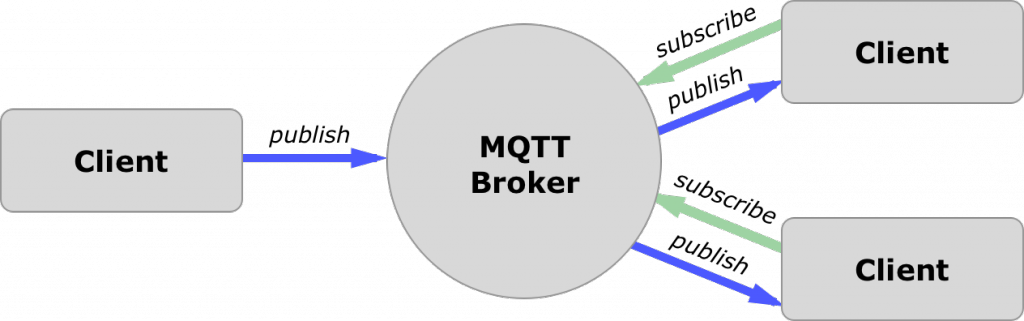


Fig 2.3.3

The Broker or Server plays the prime role in an [**MQTT (Protocol)**](https://www.engineersgarage.com/content/understanding-mqtt-protocol-iot-part-14)based network.  MQTT devices need MQTT broker to communicate with each other. MQTT broker is nothing but a central server connected to the Internet. The broker or server acts as a decision-maker which sits in between the devices. The clients do not know each other, they have to communicate via MQTT broker. The broker is mainly responsible for receiving all messages from publisher clients, filtering them, decide which subscriber is interested in it and then sending the messages to the subscribed clients. One such brokers is adafruit.io

**2.3.2.2 ADAFRUIT.IO AS MQTT BROKER**

It provides highly secure broker. The user has to create an account and get a private key generated for the account to operate this broker. The user has to use that key in the application’s firmware to connect the clients to the broker. In that way, it provides security to the application as well. Adafruit also provides TLS/SSL security.

To create the topics, user has to create the feeds in the account. The user can see the output on the dashboard of his Adafruit broker. It is easy to operate this broker and can be used for the prototype applications.

**2.3.2.3 FEEDS**

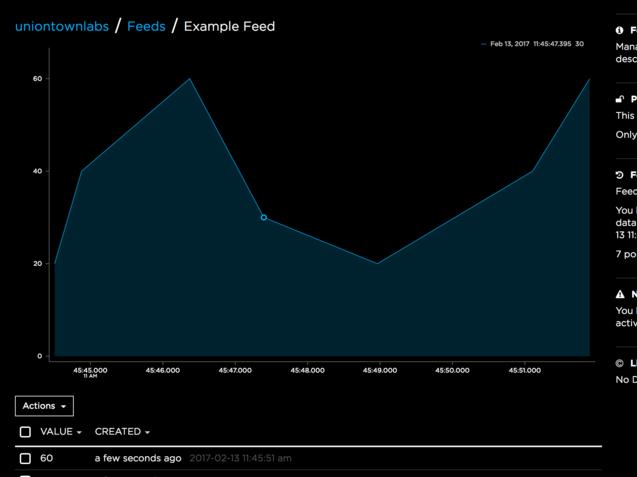


Fig 2.3.4

Adafruit.io can handle and visualize multiple feeds of data. Feeds are the core of Adafruit IO. They hold both the data you uploaded **and** meta-dataabout the data our sensors push to Adafruit IO. This includes settings for whether the data is public or private, what license the stored sensor data falls under, and a general description of the data. The feed also contains the sensor **data**values that get pushed to Adafruit IO from our device. For example, the date and time when it was uploaded or the GPS coordinates where the data came from. We need to create one feed for each unique source of data you send to the system. For example, if you have a project with one temperature sensor and two humidity sensors, you would need to create three feeds. One feed for the temperature sensor, and one feed for each humidity sensor.

**2.3.2.4 CREATING FEEDS**

 Creating a Feed on Adafruit IO is a very simple process.

* login to our [io.adafruit.com](https://io.adafruit.com/) account, you will see a list of our current dashboards
* Click the**Feeds** link on the left hand side of the screen to navigate to the Feeds list.
* Next, click on the **Actions** menu on the left hand side of the screen.
* Click on the **Create a New Feed** menu item.
* A form modal will launch as shown , and you will be presented with two text inputs:
  + **Name** - A short descriptive title of our data. Letters, numbers, and spaces are valid characters, and this field is *required*. The feed name must be *unique* for our account.
  + **Description** - A long form description of our data. This field is not required, but it's useful to provide a detailed description if our feed will be public.

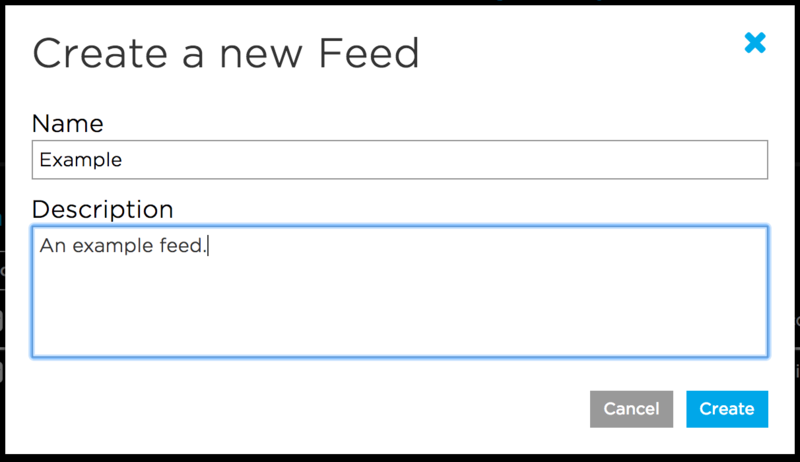
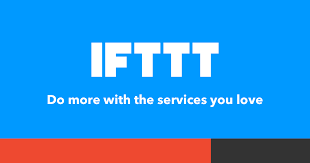


Fig 2.3.5

* Click the Create Feed button once you have finished entering our feed's name and description. You will then be redirected to our new feed.

**2.3.3 IFTTT**

****

**Fig 2.3.6**

**If This Then That**, also known as **IFTTT**  is a free web-based service  to create chains of simple conditional statements, called *applets*. An applet is triggered by changes that occur within other web services. It is a platform that helps we connects apps and devices.

IFTTT uses a "if this then that" logic, where "this" represents a service that will trigger a given action given by "that".

**2.3.3.1 COMPONENTS OF IFTTT**

* **Services** (formerly known as channels) are the basic building blocks of IFTTT. They mainly describe a series of data from a certain web service such as YouTube or [eBay](https://en.wikipedia.org/wiki/EBay). Services can also describe actions controlled with certain [APIs](https://en.wikipedia.org/wiki/API), like SMS. Sometimes, they can represent information in terms of weather or stocks. Each service has a particular set of triggers and actions.
* **Triggers** are the "this" part of an applet. They are the items that trigger the action. For example, from an [RSS feed](https://en.wikipedia.org/wiki/RSS_feed), you can receive a notification based on a keyword or phrase.
* **Actions** are the "that" part of an applet. They are the output that results from the input of the trigger.
* **Applets** (formerly known as recipes) are the [predicates](https://en.wikipedia.org/wiki/First-order_logic) made from Triggers and Actions. For example, if you like a picture on Instagram (trigger), an IFTTT app can send the photo to our [Dropbox](https://en.wikipedia.org/wiki/Dropbox_(service)" \o "Dropbox (service)) account (action).
* **Ingredients** are basic data available from a trigger—from the email trigger, for example; subject, body, attachment, received date, and sender’s address.

**2.3.3.2 CREATING APPLETS**

**Fig 2.3.7**

* Navigate to our username in the upper right corner
* Select **New Applet** from the dropdown menu
* Click the blue 'this' text
* Choose a service
* Choose a **Trigger**
* Complete Trigger fields
* Click the blue 'that' text
* Choose an **Action** service
* Choose an **Action**
* Complete Action fields
* Click **Finish**

This way we can create small applets connecting webservices and devices.

**2.3.3.3 APPLET FEATURES**

* Applets can run when you tell them to, with a tap [via widgets](https://ifttt.com/blog/2017/01/control-your-world-with-widgets?utm_medium=Help&utm_source=Basics&utm_campaign=Using_IFTTT) (That!).
* Applets can have multiple actions (e.g. if this, then that...and that).
* And some Applets even have filters that mean they only run when a specific condition is met (e.g. if this *and* that, then that).

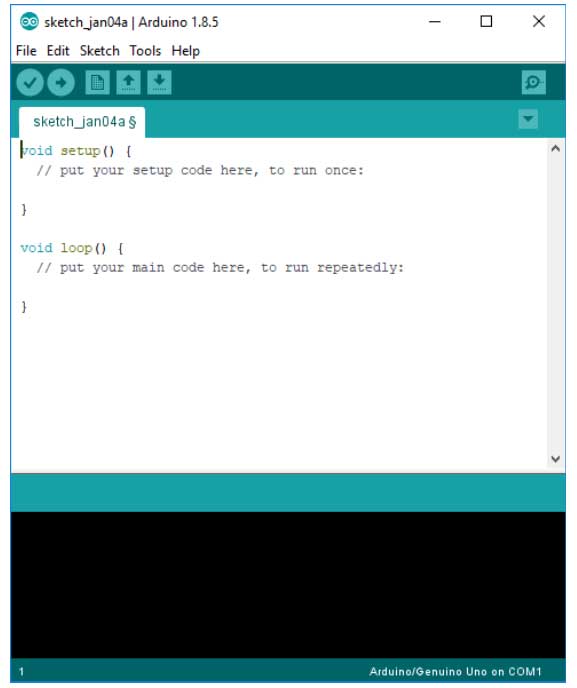
**2.3.4 ARDUINO IDE**

****

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. It is used to write and upload programs to Arduino compatible boards.

The Arduino IDE supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B) using special rules of code structuring. The Arduino IDE supplies a [software library](https://en.wikipedia.org/wiki/Software_library) from the [Wiring](https://en.wikipedia.org/wiki/Wiring_(development_platform)" \o "Wiring (development platform))project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable [cyclic executive](https://en.wikipedia.org/wiki/Cyclic_executive) program.The Arduino IDE converts the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

The top menu bar has the standard options, including “File” (new, load save, etc.), “Edit” (font, copy, paste, etc.), “Sketch” (for compiling and programming), “Tools” (useful options for testing projects), and “Help”. The middle section of the IDE is a simple text editor that where you can enter the program code. The bottom section of the IDE is dedicated to an output window that is used to see the status of the compilation, how much memory has been used, any errors that were found in the program, and various other useful messages.



Software structure of a program in arduino ide called as “sketch” consist of two main functions −

* Setup( ) function
* Loop( ) function

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

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

 Most projects will rely on the six buttons found below the menu bar.

Image of The button bar

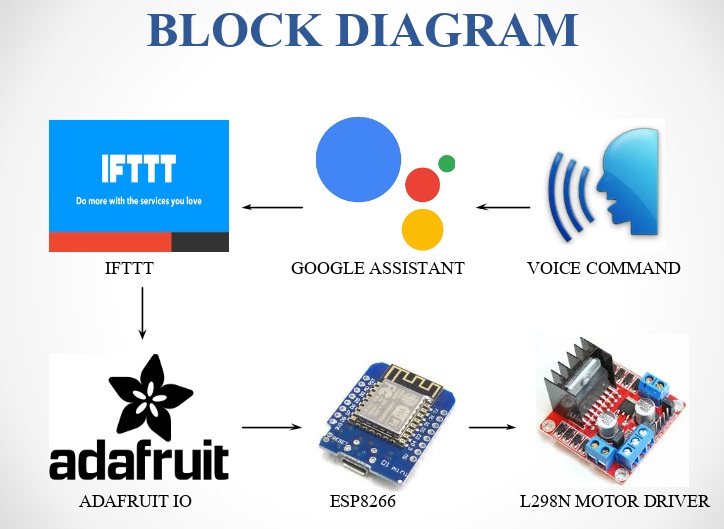
*The button bar*

1. The **check mark** is used to verify our code. Click this once you have written our code. It highlights all the errors that you have in the sketch.
2. The **arrow** uploads our code to the Arduino to run. When you use this button you will normally see two LED’s light up on our board, the TX and RX. These LED’s light up when there is information being passed between the board and the IDE.
3. The **dotted paper** will create a new file.
4. The **upward arrow** is used to open an existing Arduino project.
5. The **downward arrow** is used to save the current file.
6. The far right button is a **serial monitor**, which is useful for sending data from the Arduino to the PC for debugging purposes.

**CHAPTER 3**

**SYSTEM DESIGN**

**3.1 BLOCK DIAGRAM**



**Fig3.1.1**

**3.1.1 EXPLANATION**

* **Voice command:**

Any voice command given by us is directly accessed by Google Assistant.

* **Google Assistant:**

An application which provides an access to the given voice command.

* **IF This Then That(IFTTT):**

IFTTT is a free platform that helps you connects apps and devices. You can use it to connect our smartphone with other gadgets, or to share data between our favourite webservices (like Google, Facebook, Twitter, Instragram, etc.) and other physical devices, for instance. And the best part is that it’s really easy to use!

IFTTT uses a “if this then that” logic, where “this” represents a service that will trigger a given action given by “that”. This way you create small applets connecting webservices and devices. For the project described , you can use Google Assistant to send voice commands from a smartphone to Adafruit.io, which can then be received by the robot.

First you’ll have to sign in at:

<https://ifttt.com/>

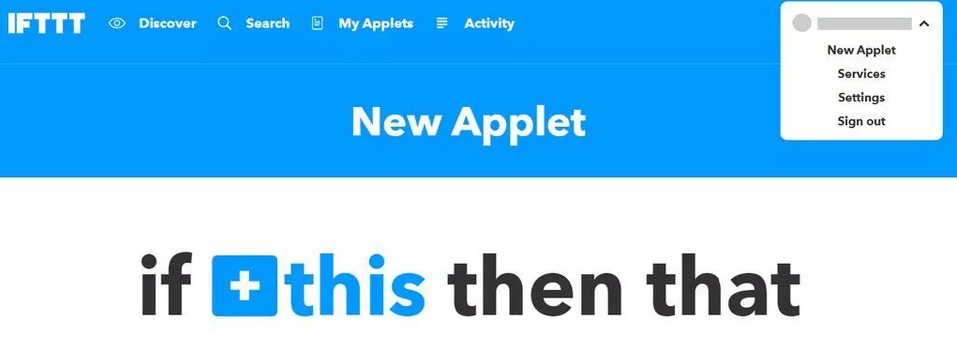
Then install IFTTT app on our smartphone.

****

**Fig.3.1.2**

On the website, navigate to New Applet (click the arrow button next to our login to access the menu).

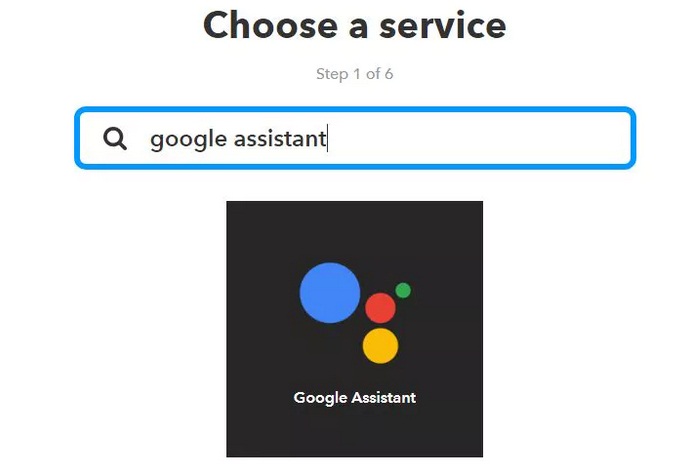
For this applet we will create a voice command that will make the robot move in a given direction or turn left/right.



**Fig.3.1.3**

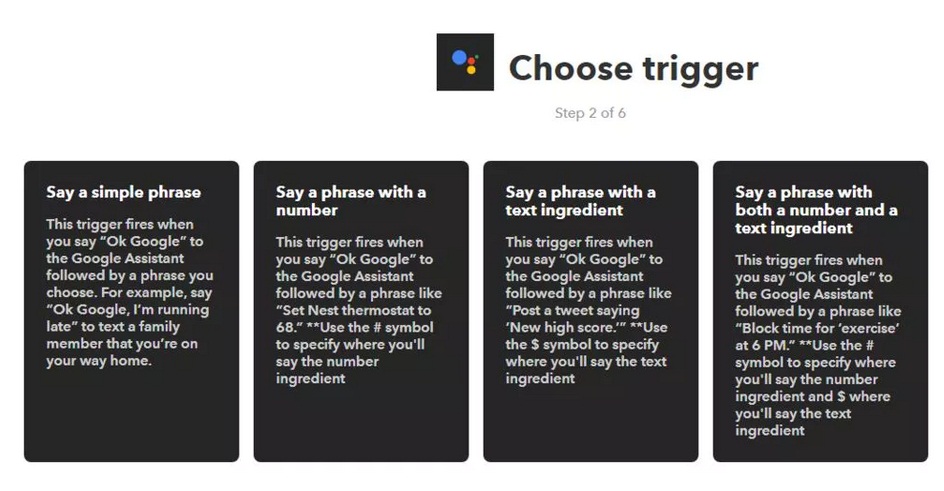
To creat an applet

First click on +this and type "Google Assistant" on Seach service text box and select Google Assistant widget as shown in fig.



**Fig.3.1.4**

Choose "Say a phrase with both a number and a text ingredient" trigger. As it's described on IFTTT website, it will create a trigger that fires every time you say "Ok Google" followed by a phrase with a text ingredient and a number ingredient as shown in fig.

****

**Fig.3.1.4**

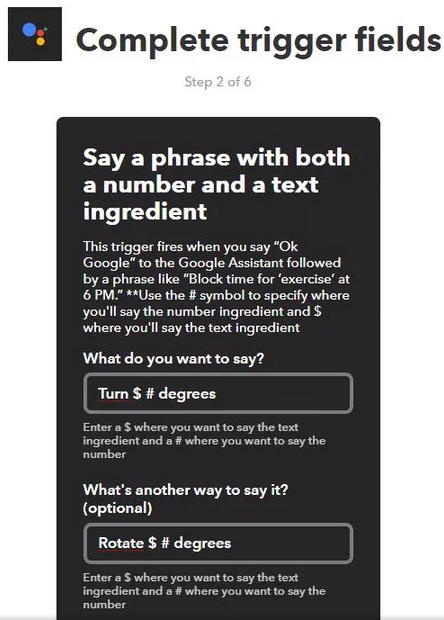
This platform provides different types of triggers for various applications . thus triggers are of 4 types .

A simple phrase

A phrase with a number

A phrase with text ingredient

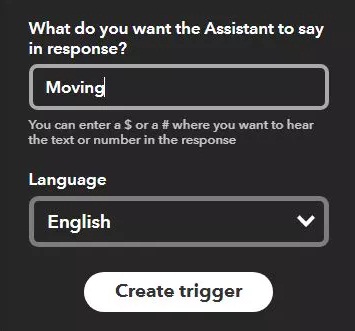
A phrase with both number and text ingredient



**Fig.3.1.5**

To complete the trigger, you'll have to define the phrase you want to say. IFTTT allows you to configure more than one phrase as a trigger. $ represent the text ingredient and # the number ingredient. In my case, I define the following commands:

* Turn $ # degrees
* Rotate $ # degrees
* Move $ #



**Fig.3.1.6**

You'll also have to define the message it will say in response if it understands our command. In this case it is configured to Say "Turning $". Now click on “Create trigger” button to create customised trigger as shown in the fig.



**Fig.3.1.7**

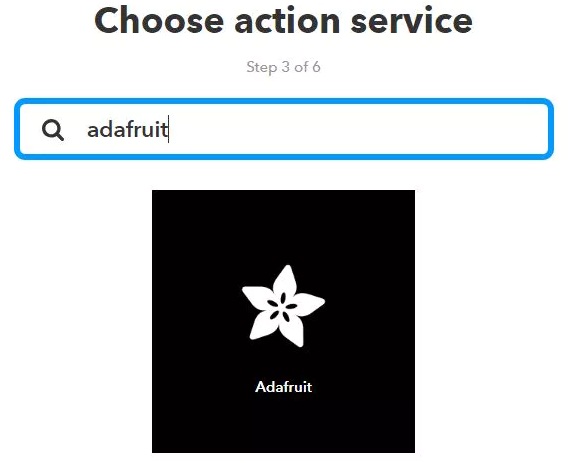
Now choose +That

This will help us choose the responding platform to which the message accepted by Google assistant is sent.

+that can hold any application or app which can serve the user best.

Here we need a platform which can hold the data store it as a feed.

One such application which holds data as feed is ADAFRUIT.



**Fig.3.1.8**

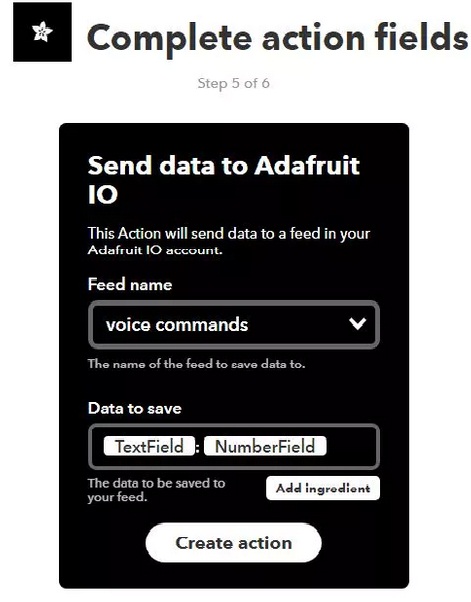
Type "Adafruit" and select Adafruit > Send data to Adafruit IO. This will send data to a feed in our Adafruit IO account whenever the trigger you configured previously (+This) is activated as shown in fig.



**Fig.3.1.9**

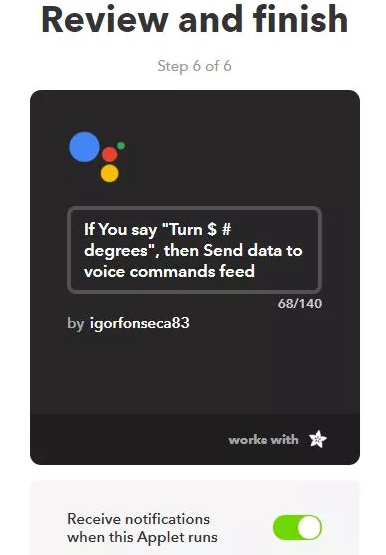
Choose an action from the options. ‘send data to AdafruitIO’ is the presently required action.

This action will send data to feed in our Adafruit IO account as shown in fig.



**Fig.3.1.10**

Configure Feed name as "voice command". In "Data to save" choose "Add ingredient" and add the TextField and NumberField. Add ":" between then. This will work as a delimiter later on Arduino code.

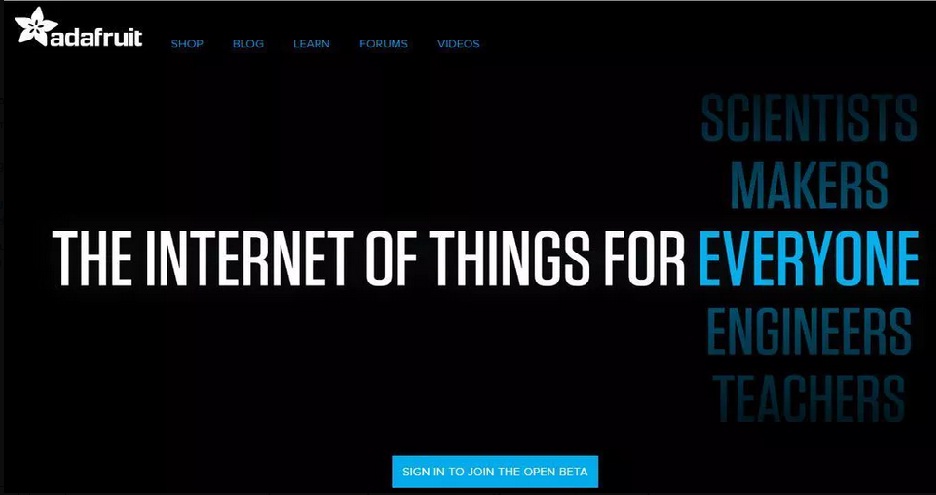


**Fig.3.1.11**

Finish our applet and turn it on.

* **Adafruit IO:**

There are a lot of datalogging services available for communicating a microcontroller to the web. With those services you can upload/download data to/from the cloud, and do a lot of cool stuff.



**Fig.3.1.12**

Adafruit.IO is one of those free services. It's really easy to use and promises to bring internet of things to everyone

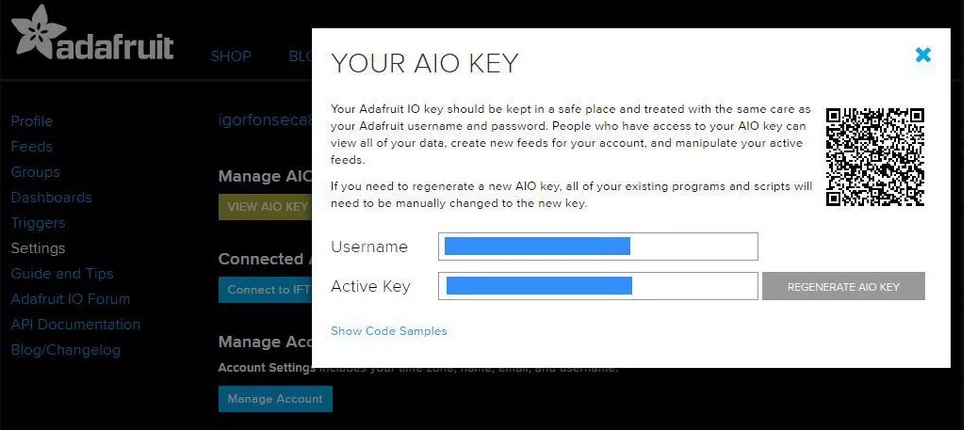
Create Adafruit IO Web Feed

Under *Feeds >Create a new Feed* add a new feed named "voice commands". It will create a database, and we will use it store the commands received by the gadget.

Sign in at <https://io.adafruit.com/>

On the next step I will show we how to configure IFTTT, another platform I used in this project. The idea here is simple: IFTTT will have some triggers configured and send some data to Adafruit.IO platform when a given logic is true. The gadget will be able to read the data stored in a given feed on Adafruit.IO, execute some logic andperform some actions**.**

Copy our Adafruit.IO key, which will be later used for allowing our device accessing the database. Navigate for Settings > View AIO key and copy the active key code. You'll need it for our Arduino (ESP8266) code on next steps.



**Fig.3.1.13**

* **Wemos D1 mini(ESP8266) using Arduino IDE:**

It is really easy to use and program with Arduino IDE. It has the same footprint of and ordinary Arduino Uno! This way most of Arduino shield will also work with this board. It has built-in Wi-Fi module, so we can use it in a variety of projects. We can also us other ESP8266 based boards.



**Fig.3.1.14**

The code from Arduino IDE can be put into ESP8266 using a usb port.

* **Motor driver:**

Signal inputs are given by ESP8266 to the motor driver. Thus motor driver controls the speed,direction,timing of the motors used for the robot to move.

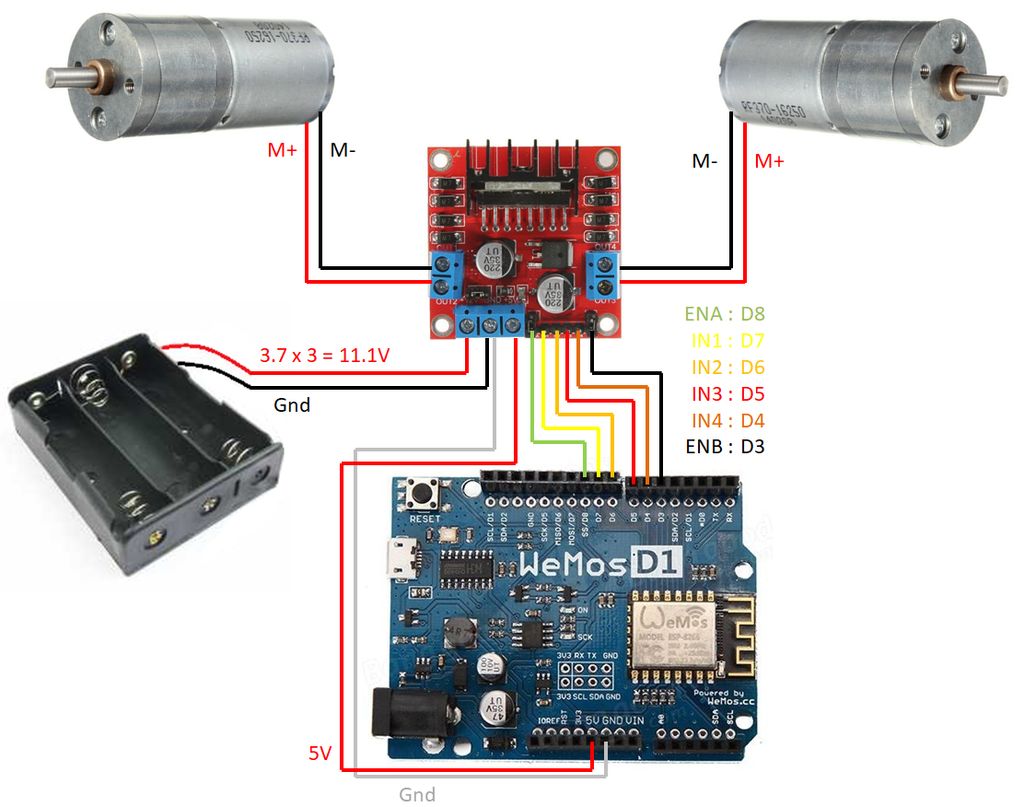
The entire control depends on the code put in the wemos d1 mini board.

**3.1.2 WORKING**

* Instructionsfrom the user are taken by Google assistant application.
* These instructions are sent ti IFTTT platform and a response of “moving” is sent back to Google assistant application.
* This instruction is now sent to the Adafruit IO feed “voice commands”.
* Since Wemos D1 mini(ESP8266) has an access to this adafruit IO account, this command is now read by the microcontroller. This is possible because we provided it with access to internet.
* Now according to the instruction read , signals are sent to the motor driver.
* This motor driver thus controls the motors accordingly.
* Hence the robot moves as per the given instruction.

**3.2 SYSTEM ARCHITECHTURE**

**3.2.1 SCHEMATIC DIAGRAM**

****

**Fig.3.2.1**

* + 1. **DESCRIPTION**

Here we have used 2 motors to control two wheels for navigation of the robot.

These two motors are connected to the two input ports of the motor driver which have an access to the signals given by esp8266.

A L298N dual channel H-bridge was used for the control of the motors. It receives some 3.3V signals of the Wemos board, and provide higher voltages for the motors. It also allow the motors to run in both directions, depenting on the combination of those input signals.

Each device was connected according to the schematics.

Follows a list of the pins you'll need to connect:

**Wemos D1 inputs/outputs:**

* Digital pin D3 => H-Bridge ENB pin
* Digital pin D4 => H-Bridge IN4 pin
* Digital pin D5 => H-Bridge IN3 pin
* Digital pin D6 => H-Bridge IN2 pin
* Digital pin D7 => H-Bridge IN1 pin
* Digital pin D8 => H-Bridge ENA pin
* 5V pin => H-Bridge 5V pin
* Gnd pin => H-bridge Gnd pin

**H-Bridge inputs/outputs:**

* ENB pin => Wemos D3 pin
* IN4 pin => Wemos D4 pin
* IN3 pin => Wemos D5 pin
* IN2 pin => Wemos D6 pin
* IN1 pin => Wemos D7 pin
* ENA pin => Wemos D8 pin
* 5V pin => Wemos 5V pin
* Gnd pin => Wemos Gnd pin
* Gnd pin => Battery pack negative wire
* 12V pin => Battery pack positive wire
* OUT1 => Right motor negative wire
* OUT2 => Right motor positive wire
* OUT3 => Left motor positive wire
* OUT4 => Left motor negative wire

All the pins when properly pinned results out in accurate work.

**CHAPTER 4**

**SYSTEM IMPLEMENTATION AND METHODOLOGIES**

Once all the connections have been made according to the schematic diagram and the configurations of IFTTT and Adafruit.io have been made, the final task is to implement the code.

Our Wemos D1 ESP8266 have been coded by using Arduino IDE. Arduino IDE already comes with support to a lot of different boards: Arduino Nano, Mine, Uno, Mega, Yún, etc. Unfortunately ESP8266 isn't by default among those supported development boards. So in order to upload our codes to a ESP8266 based board, we'll have to add its properties to Arduino's software first.

**4.1 ENVIRONMENT SETUP**

**4.1.1 ADDING ESP8266 BOARD TO ARDUINO IDE**

Navigate to File > Preferences (Ctrl +, on Windows OS);

* Add the following URL to Additional Boards Manager textbox (the one on the bottom of the Preferences window):

<http://arduino.esp8266.com/stable/package_esp8266com_index.json>

Navigate for Tools > Board > Boards Manager for adding our ESP8266 board.

* If the text box wasn't blank, it means had already add other boards before on Arduino IDE before. Add a comma at the end of the previous URL and the one above.
* Hit "Ok" button and close the Preferences Window.
* Type "ESP8266" on the search text box, select "esp8266 by ESP8266 Community" and install it.

Now our Arduino IDE will be ready to work with a lot of ESP8266 based development boards, like the generic ESP8266, NodeMcu, Adafruit Huzzah, Sparkfun Thing and the WeMos D1 mini (which is our board), etc.

**4.1.2ADDING LIBRARIES**

The following libraries will be used for our Arduino code:

**ESP8266WiFi LIBRARY**

ESP8266 is all about Wi-Fi. To connect your new ESP8266 module to Wi-Fi network to start sending and receiving data this library is used. The scope of functionality offered by [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) library is quite extensive

**ARDUINOHTTPCLIENT LIBRARY**

 Easily interact with web servers from Arduino, using HTTP and WebSocket's. This library can be used for HTTP (get, post, put, delete) requests to a web server. It also supports exchanging messages with WebSocket servers.

**ARDUINOMQTTCLIENT LIBRARY**

Allows us to send and receive MQTT messages using Arduino. This library provides a client for doing simple publish/subscribe messaging with a server that supports MQTT

**ADAFRUIT MQTT LIBRARY**

It is a MQTT library that supports the ESP8266 and generic Arduino Client hardware. It supports the bare minimum to publish and subscribe to topics.

Navigate to Sketch-> Include Library -> Manage Libraries... on Arduino IDE and add the libraries above.

**4.2 ALGORITHM**

* Connect ESP8266 to the Wifi
* Connect it with Adafruit.io’s voice commands feed where the commands are stored
* Read the command
* Handle the command i.e give inputs to the motor driver as per the command

**4.3 CODE SKELETON**

//Include libraries

//Configure wifi

//Configure adafruit.io

//Define pins

void setup()

{

// Configure pins

// Disable both the motors

// Connect ESP8266 to Wifi

// Subscribe to voice commands feed

}

void loop()

{

//read the commands from the feed and call handle message function

}

void handlemessage(char\* data)

{

// separate text and number ingredients

// perform movements

}

**4.4 CODE**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* WIFIConfiguration\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define WIFI\_SSID "Asus"

#define WIFI\_PASS "08n11a25s13a"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Main Program Starts Here \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <ESP8266WiFi.h>

#include <Adafruit\_MQTT.h>

#include <ArduinoHttpClient.h>

//Adafruit io configuration

#include "Adafruit\_MQTT\_Client.h"

#define AIO\_SERVER "io.adafruit.com"

#define AIO\_SERVERPORT 1883

#define AIO\_USERNAME "AakankshaThota"

#define AIO\_KEY "b594fa274ffc4ed78ae669314a2166ae"

WiFiClient client;

Adafruit\_MQTT\_Client mqtt(&client, AIO\_SERVER, AIO\_SERVERPORT, AIO\_USERNAME, AIO\_KEY);

Adafruit\_MQTT\_Subscribe voicecommands = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/voice-commands");

boolean MQTT\_connect()

{

int8\_t ret;

if (mqtt.connected()) {return true; }

uint8\_t retries = 3;

while ((ret = mqtt.connect()) != 0)

{

mqtt.disconnect(); delay(2000); retries--;

if (retries == 0) { return false; }

}

return true;

}

void handlemessage(char\* data);

void frontside(int);

void backside(int);

void leftside(int);

void rightside(int);

// Defining pins

#define ENB D3 //ENB

#define MOTORB\_1 D4 //IN3

#define MOTORB\_2 D5 //IN4

#define MOTORA\_1 D7 //IN1

#define MOTORA\_2 D6 //IN2

#define ENA D8 //ENA

// Setup

void setup()

{

// Configure pins

pinMode(ENA, OUTPUT);

pinMode(MOTORA\_1, OUTPUT);

pinMode(MOTORA\_2, OUTPUT);

pinMode(ENB, OUTPUT);

pinMode(MOTORB\_1, OUTPUT);

pinMode(MOTORB\_2, OUTPUT);

// Start serial communication

Serial.begin(115200);

// disable both motors

digitalWrite(ENA,LOW);

digitalWrite(ENB,LOW);

//Connect to wifi

WiFi.mode(WIFI\_STA);

WiFi.begin(WIFI\_SSID,WIFI\_PASS );

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

//Subscribe to voice commands feed

mqtt.subscribe(&voicecommands);

}

void loop()

{ // read the command

if (MQTT\_connect()) {

Adafruit\_MQTT\_Subscribe \*subscription\_name;

while ((subscription\_name = mqtt.readSubscription(5000))) {

if (subscription\_name == &voicecommands){

char\* command= (char \*)voicecommands.lastread;

handleMessage(command);

}}}

}

// this part of the code runs whenever there's a new message on Adafruit.io feed

void handleMessage(char \*data) {

String commandStr = String(data); // store the incoming commands in a string

// received message

Serial.print("received <- ");

Serial.println(commandStr);

String cmd;

int angle;

// separate text and number ingredients

for (int i = 0; i < commandStr.length(); i++) {

if (commandStr.substring(i, i+1) == ":") {

cmd = commandStr.substring(0, i);

angle = commandStr.substring(i+1).toInt();

break;

}

}

// print command

Serial.println(cmd);

Serial.println(angle);

// perform movements

// LEFT

if (cmd.equalsIgnoreCase("left")){

leftside(angle);

}

// RIGHT

if (cmd.equalsIgnoreCase("right")){

rightside(angle);

}

// FORWARD

if (cmd.equalsIgnoreCase("forward")){

frontside(angle);

}

// BACK

if (cmd.equalsIgnoreCase("back")){

backside(angle);

}

}

void leftside(int angle)

{

Serial.println("Turning left");

digitalWrite(ENA,HIGH);

digitalWrite(ENB,HIGH);

digitalWrite(MOTORA\_1,HIGH);

digitalWrite(MOTORA\_2,LOW);

digitalWrite(MOTORB\_1,LOW);

digitalWrite(MOTORB\_2,HIGH);

delay(angle\*12);

Serial.println("Stop");

digitalWrite(ENA,LOW);

digitalWrite(ENB,LOW);

digitalWrite(MOTORA\_1,LOW);

digitalWrite(MOTORA\_2,LOW);

digitalWrite(MOTORB\_1,LOW);

digitalWrite(MOTORB\_2,LOW);

}

void rightside(int angle)

{Serial.println("Turning right");

digitalWrite(ENA,HIGH);

digitalWrite(ENB,HIGH);

digitalWrite(MOTORA\_1,LOW);

digitalWrite(MOTORA\_2,HIGH);

digitalWrite(MOTORB\_1,HIGH);

digitalWrite(MOTORB\_2,LOW);

delay(angle\*12);

Serial.println("Stop");

digitalWrite(ENA,LOW);

digitalWrite(ENB,LOW);

digitalWrite(MOTORA\_1,LOW);

digitalWrite(MOTORA\_2,LOW);

digitalWrite(MOTORB\_1,LOW);

digitalWrite(MOTORB\_2,LOW);

}

void frontside(int angle)

{Serial.println("Moving forward");

digitalWrite(ENA,HIGH);

digitalWrite(ENB,HIGH);

digitalWrite(MOTORA\_1,LOW);

digitalWrite(MOTORA\_2,HIGH);

digitalWrite(MOTORB\_1,LOW);

digitalWrite(MOTORB\_2,HIGH);

delay(angle\*71.4283);

Serial.println("Stop");

digitalWrite(ENA,LOW);

digitalWrite(ENB,LOW);

digitalWrite(MOTORA\_1,LOW);

digitalWrite(MOTORA\_2,LOW);

digitalWrite(MOTORB\_1,LOW);

digitalWrite(MOTORB\_2,LOW);

}

void backside(int angle)

{

Serial.println("Moving back");

digitalWrite(ENA,HIGH);

digitalWrite(ENB,HIGH);

digitalWrite(MOTORA\_1,HIGH);

digitalWrite(MOTORA\_2,LOW);

digitalWrite(MOTORB\_1,HIGH);

digitalWrite(MOTORB\_2,LOW);

delay(angle\*71.4283);

Serial.println("Stop");

digitalWrite(ENA,LOW);

digitalWrite(ENB,LOW);

digitalWrite(MOTORA\_1,LOW);

digitalWrite(MOTORA\_2,LOW);

digitalWrite(MOTORB\_1,LOW);

digitalWrite(MOTORB\_2,LOW);

}

**CHAPTER 5**

**CONCLUSIONS AND FUTURE ENHANCEMENTS**

**5.1 CONCLUSION**

In this module, we have discussed about voice controlled robot which can serve us in many stages of life. We have also learned how to make use of applications like IFTTT and Google assistance to work out with the social problems. Our day to day life unfortunately encounters many problems which we, humans cannot solve but something like a robotic assistance could be a good choice to overcome that situation.

People assume that the rise of robots could be destructive. But it has to be noticed that development in the field of robotics is very much essential and should always be controlled by humans. This prescribed model of robotic assistance can serve is in many beneficial ways.

**5.2 FUTURE ENHANCEMENTS**

* For more accurate working, servo motors can be deployed.
* Automatic Targeting System can be implemented in the robot for tracking the target.
* We can also proceed to construct a full fledged robot to carry out more complex tasks.
* We can extend to advanced applications by using GPS.
* Image processing can be implemented in the robot to detect the colour and the objects.
* A camera can be attached when used for remote or high risk operation in order to monitor its surrounding on screen.
* A solar panel can be attached to the device in order to recharge the D.C. battery thereby reducing energy cost.
* Increase the weight it can carry for it to carry heavyobjects.

**5.3 RESULT**

This project when enhanced could result in more sophisticated living. This could be enhanced with an addition of a camera which could move to places where humans can’t reach and record the then instances. This could also help the handicapped to move according to the voice command. This could also be added with movable hands which could pick things and displace them accordingly through voice commands. Hence the implementation of this model could pave a path for better future.

**REFERENCES**

**1.B.O. Omijeh, R. Uhunmwangho and M. Ehikhamenle.**

“Design Analysis of a Remote Controlled Pick and Place Robotic Vehicle”, International Journal of Engineering Research and Development, volume 10, 2014, pp. 57-68.

**2.J.N. Mohammed, J. Neetha, F. Muhammed, M. Midhun, S. Mithun and C,N. Safwan.**

“Wireless Control of Pick and Place Robotic Arm Using an Android Application”,

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Volume 4, Issue 4, 2015, pp. 2410-2416.

**3.P. Ritika and K. Narender. “Android Mobile Phone**

Controlled Bluetooth Robot Using Microcontroller”, International Journal of Scientific Engineering and Research (IJSER), Volume 2, Issue 7, 2014, pp. 14-17.

**4.J. Bhuvaneshwari, A. Mohnish,, G. Rohan, B. Chetan.**

“Voice Controlled Robotic Vehicle”, International Research Journal of Engineering and Technology (IRJET), Volume 04, Issue 06, 2017, pp. 191-193.

**5.S. J. Russell and P. Norvig.**

“Artiﬁcial Intelligence: A Modern Approach” Third Edition, Prentice Hall Series,

Pearson Education, Inc., Upper Saddle River, New Jersey, 2003.