AUTOMATIC AND MANUAL CONTROL OF SPEED AND DIRECTION OF DC MOTOR WITH RASPBERRY PI3B+

A Project report submitted in partial fulfilment

of the requirements for the degree of B. Tech in Electrical Engineering

By

Sounak Bar (11701616022)
Sayan Mallick (11701616034)
Joyjit Majumder (11701617015)

Under the supervision of :- ***Dr. Debasish Mondal***

(HEAD OF THE DEPARTMENT, DEPARTMENT OF ELECTRICAL ENGINEERING, RCCIIT)

Department of Electrical Engineering
RCC INSTITUTE OF INFORMATION TECHNOLOGY
CANAL SOUTH ROAD,BELIAGHATA,KOLKATA-700015,WEST BENGAL
Maulana Abul Kalam Azad University of Technology(MAKAUT)

CERTIFICATE

To whom it may concern

This is to certify that the project work entitled **AUTOMATIC AND MANUAL CONTROL OF SPEED AND DIRECTION OF DC MOTOR WITH RASPBERRY PI3B+** is the bona fide work carried out by

SOUNAK BAR (11701616022) **SAYAN MALLICK** (11701616034)

JOYJIT MAJUMDER(11701617015)

students of B.Tech in the Dept. of Electrical Engineering, RCC Institute of Information Technology (RCCIIT), Canal South Road, Beliaghata, Kolkata-700015, affiliated to Maulana Abul Kalam Azad University of Technology (MAKAUT), West Bengal, India, during the academic year **2019-20**, in partial fulfillment of the requirements for the degree of Bachelor of Technology in Electrical Engineering and that this project has not submitted previously for the award of any other degree, diploma and fellowship.

_____ _____

Signature of the Guide **Signature of the HOD**

(DR. DEBASISH MONDAL) (DR. DEBASISH MONDAL)

,Electrical Department,RCCIIT HOD,Electrical Department,RCCIIT

_____

Signature of the External Examiner

ACKNOWLEDGEMENT

It is my great fortune that I have got the opportunity to carry out this project work under the supervision of **Prof.(Dr.) Debasish Mondal, HOD,** in the Department of Electrical Engineering, RCC Institute of Information Technology (RCCIIT), Canal South Road, Beliaghata, Kolkata-700015, affiliated to Maulana Abul Kalam Azad University of Technology (MAKAUT), West Bengal, India. I express my sincere thanks and deepest sense of gratitude to my guide for his constant support, unparalleled guidance and limitless encouragement.

I wish to convey my gratitude to **Prof. (Dr.) Debasish Mondal**, HOD, Department of Electrical Engineering, RCCIIT and to the authority of RCCIIT for providing all kinds of infrastructural facility towards the research and project work.

I would also like to convey my gratitude to all the faculty members and staffs of the Department of Electrical Engineering, RCCIIT for their whole hearted cooperation to make this work turn into reality.

Place: KOLKATA ----------

Date: 16/05/2019 **Full Signature of the Student**

CONTENTS

Chapters **Page No**

Acronyms and Abbreviations:

- ***PWM*** : Pulse Width Modulation
- ***GPIO*** : General purpose input/output
- ***RPi***: Raspberry Pi
- ***IP***: Internet Protocol
- ***DHT***: Digital Humidity and Temperature
- ***IR***: Infrared sensor
- ***GUI*** Graphical User Interface
- ***TK*** Tkinter
- ***Wifi*** Wireless Fidelity
- ***SD*** Storage Device
- ***LPDDR*** Low Power Double Data Rate
- ***LAN*** Local Area Network
- ***UI*** User Interface

List of Figures

Abstract:

Raspberry pi3B+ is a versatile and very powerful almost credit card sized computer. In this report we use the raspberry to interface a motor driver with it in order to control the speed and direction of a dc motor through an easy to use straight forward Graphical User Interface. Through this GUI, everyone can easily control and monitor the speed and direction of the DC motor without the need to know technical details about it.. A temperature and humidity sensor is also added to make the operations automatic according to temperature if needed. Furthermore, the inclusion of a tachometer which displays the speed of the motor further helps to monitor and visualize the changes in speed of the motor. An infrared sensor is used to measure the speed of the motor in this project

1.INTRODUCTION:

A DC motor is any of a class of rotary electrical motors that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current in part of the motor.

DC motors were the first form of motor widely used, as they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. Speed of a DC motor can be controlled by raspberry pi 3 b+ using PWM technique.

It is a small board computer, introduced by Raspberry Pi foundation in 14th March 2018 and is the most recent version of the Pi boards.

It is a modified form of its predecessor Raspberry Pi 3 B that was introduced in 2016 and came with CPU, GPU, USP ports and I/O pins. Both versions are almost same in terms of functionality and technical specifications; however, there are some exceptions in the B+ model as it comes with USB boot, network boot, and Power over Ethernet option that are not present in the B model.

Now the PWM technique works by driving the motor with a series of "ON-OFF" pulses and varying the duty cycle, the fraction of time that the output voltage is "ON" compared to when it is "OFF", of the pulses while keeping the frequency constant.

Using the Raspberry Pi, a very user friendly Graphics User Interface can be implemented which is very easy to use and intuitive and in this project we do the same

1.1 Objective:

1. TO FIND THE IP ADDRESS OF RASPBERRY PI USED
2. TO CONECT RASPBERRY PI TO A PC USING ETHERNET AND ACESS THE RASPBERRY PI DEKTOP
3. TO MAKE ALL THE NECESSARY CONNECTION ACCORIDNG TO CIRCUIT DIAGARM
4. TO CREAT A GUI USING TKINTER MODULE OF PYTHON PROGRAMMING.
5. TO USE THE GUI TO CONTROLL THE SPEED AND DIRECTION OF MOTOR.
6. TO INTERFACE DHT11 (TEMPERATURE AND HUMIDITY SENSOR) WITH RASPBERRY PI
7. TO USE THE TEMPERATURE DATA TO CONTROL THE SPEED AUTOMATICALY
8. TO CREATE A RPM MEASURING ARRANGEMENT USING AN IR SENSOR AND DISPLAY IT

2.Literatue Survey:

- ***SPEED CONTROL OF DC MOTOR***

Prabha Malviya(PG Student) * , Menka Dubey(Sr. Asst.prof.) EX department, Malwa institute of Technology, Indore R.G.P.V. ,Bhopal, India

This paper deal with various method of speed control of DC Motor and literature review on speed control of DC motor is presented. DC motors are widely used in industry applications, robotics and domestic appliances because of its low cost and less complex control structure and wide range of speed and torque. So wide range of position control is required. Proportional Integral Derivative (PID) controller is used in industries for wide number of applications .The tuning of PID controller parameters is very important for desired out response there is so many techniques for tuning of PID controller.

- ***R.K Munje,. "Speed control of DC motor using PI and SMC" in proc. Conference on IPEC, Singapore, 27- 29 Oct. 2010.***

In this paper, sliding mode control (SMC) technique is used to control the speed of DC motor. The performance of the SMC is judged via MATLAB simulations using linear model of the DC motor and known disturbance. SMC is then compared with PI controller. The simulation result shows that the sliding mode controller (SMCr) is superior controller than PI for the speed control of DC motor. Since the SMC is robust in presence of disturbances, the desired speed is perfectly tracked. The problem of chattering, resulting from discontinuous controller, is handled by pseudo sliding with smooth control action.

- ***A. Asadi, S. Bagheri, A. Imam, E. Jalayeri, W. Kinsner, and N. Sepehri. Institute of Electrical and Electronics Engineers Inc. (2016)***

The expanding capabilities of today's microcontrollers and other devices lead to an increased utilization of these technologies in diverse fields. The automation and issue of remote control of moving objects belong to these fields. In this project, a microcontroller Raspberry Pi 2B was chosen for controlling DC motors and servo-motors. This paper provides basic insight into issue of controlling DC motors and servo-motors, connection between Raspberry and other components on breadboard and programming syntaxes for controlling motors in Python programming language.

- ***K. Premkumar, K. G. J. Nigel. "Smart Phone Based Robotic Arm Control using Raspberry Pi, Android and Wi-Fi."Institute of Electrical and Electronics Engineers Inc***

Nowadays, there is a high demand for wireless devices controlled by Wi-Fi, GSM or Bluetooth. These technologies offer a simplification of our lives in home automation or entertainment in the form of Radio Controlled (RC) models. At the same time, the possibilities of microcontrollers have risen in many applications. Current control systems are based on special purpose devices. However, little attention is given to universal microcontrollers that are able to perform a wide variety of tasks .The remote control via microcontrollers is more popular among researchers and "RC fans" than among the public . The performance growth of microcontrollers has led to their ability to manage complex applications. At the same time, there is a variety of manufacturers of microcontrollers with many diverse types of processors and performance. Raspberry and Arduino belong to the most widely used microcontrollers . Both of them provide high performance, and they can be used in many challenging automation applications. The second generation of the Raspberry microcontroller provides enough performance to replace a standard PC in some audiovisual and automation applications . With the use of the wireless extension, which can be achieved by a simple antenna, the device becomes in a complex control station.

3.OVERVIEW OF THE PROJECT:

3.1 BLOCK DIAGRAM:

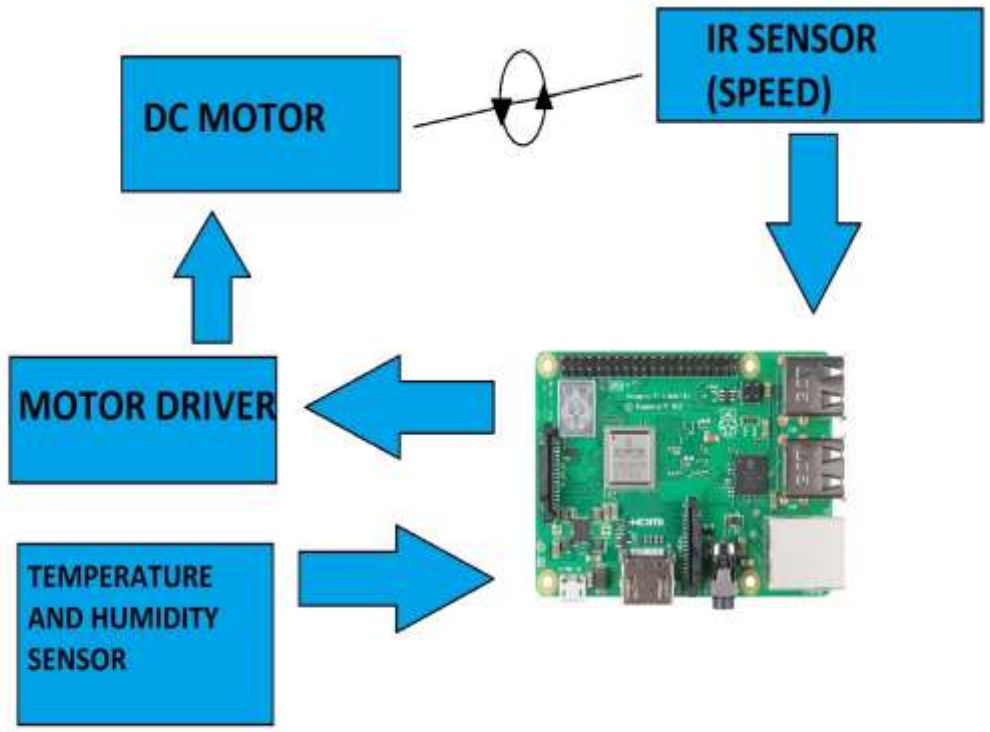


Fig 1: Proposed block diagram of the project work

3.2 LIST OF HARDWARE COMPONENTS

1. RASPBERRY PI 3B+
2. 12V DC MOTOR
3. MOTOR DRIVER L298
4. DHT 11
5. 12V DC POWER SUPPLY
6. 5V DC POWER SUPPY FOR RASPBERRY PI
7. LAN CABLE
8. BREAD BOARD
9. JUMPER WIRE
10. IR SESNOR

3.3 CIRCUIT DIAGRAM:

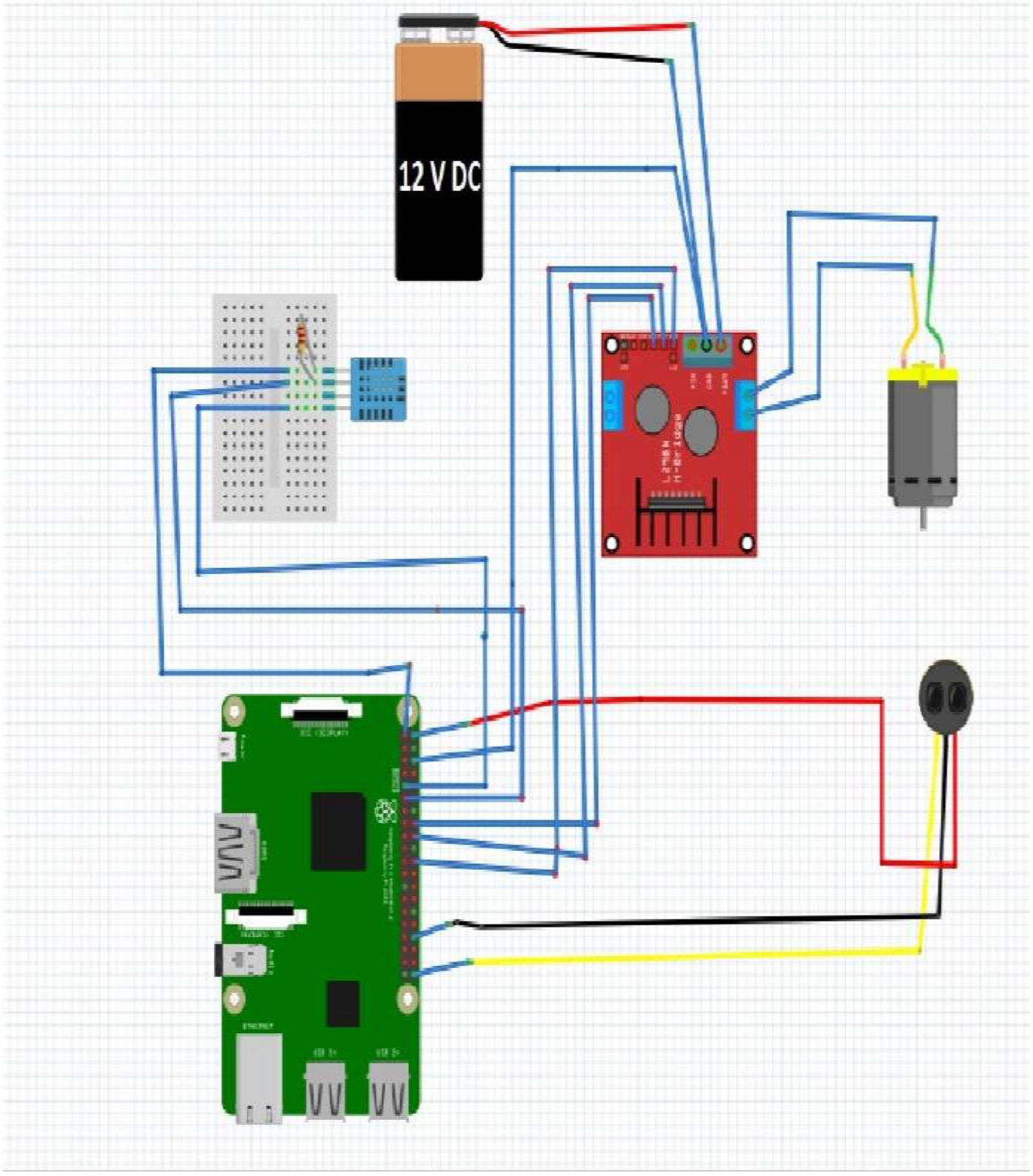


Fig 2:Circuit diagram of the project work

4 COMPONENT DESCRIPTION:

4.1 Raspberry PI3b+ :

Raspberry Pi is a low priced, small sized computer that plugs into a computer monitor or TV, and uses normal peripheral like keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games. The Raspberry Pi was first launched in 2012, and there have been a few changes and variations performed from that point forward. The first Pi had a single core 700MHz CPU and simply 256MB RAM, and the most recent model has a quad-core 1.4GHz CPU with 1GB RAM. All over the world, individuals use Raspberry Pi to get the skills of programming abilities, , do home automation, and even use them in modern applications. The Raspberry Pi works in the open source environment: it runs Linux (diverse distribution), and its principle supported working system, Raspbian, is open source and runs a suite of open source programming. The Raspberry Pi Foundation adds to the Linux part and Raspberry Pi is a low cost credit card size computer that plugs into a computer monitor or TV and uses a standard keyboard and mouse. Most importantly it's open source hardware. Computing Programmable Language like python and scratch under Linux platform different other open source extends just as releasing its very own lot programming as open source. Here Raspberry pi is being used as a main controller to derive other features like face recognition and detection which we are doing in our project.. Raspberry Pi 3 model B has CPU 1400MHZ quad-core ARM cortx-A53 processor. The Ethernet adaptor is connected to an additional USB port. In model A and A+ the USB port is connected directly to the Silicon on Chip (SoC).

Fig 3: *Visual representation of RPI 3B+*

Pin configuration of Raspberry Pi3B+:

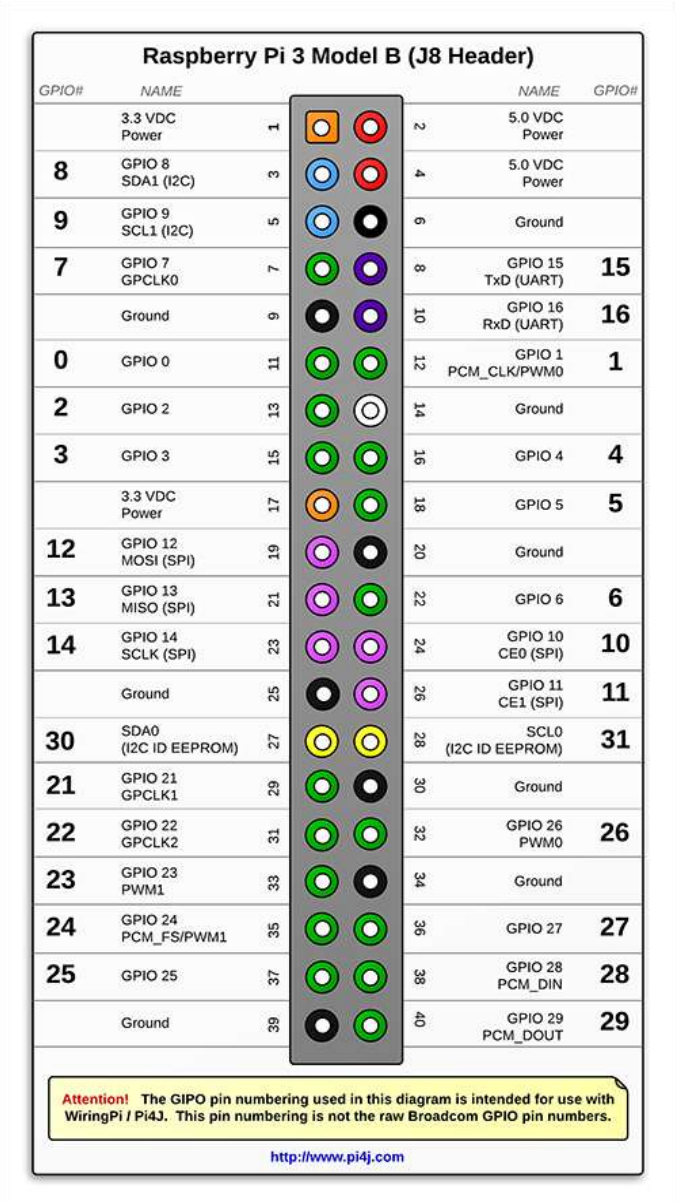


Fig 4:: *GPIO pin configuration of RPI 3B+*

A powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board. A 40-pin GPIO header is found on all current Raspberry Pi boards.

Voltages: Two 5V pins and two 3V3 pins are present on the board, as well as a number of ground pins (0V), which are un-configurable. The remaining pins are all general purpose 3V3 pins, meaning outputs are set to 3V3 and inputs are 3V3-tolerant.

Outputs: A GPIO pin designated as an output pin can be set to high (3V3) or low (0V).

Inputs: A GPIO pin designated as an input pin can be read as high (3V3) or low (0V). This is

made easier with the use of internal pull-up or pull-down resistors. Pins GPIO2 and GPIO3 have fixed pull-up resistors, but for other pins this can be configured in software. The GPIO pins can be used for other functions also. Some are available on all pins and others on some specific pins.

PWM (pulse-width modulation) -Software PWM available on all pins -Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19

SPI

-SPI0: MOSI (GPIO10); MISO (GPIO9); SCLK (GPIO11); CE0 (GPIO8), CE1 (GPIO7) -SPI1: MOSI (GPIO20); MISO (GPIO19); SCLK (GPIO21); CE0 (GPIO18); CE1 (GPIO17);

I2C

-Data: (GPIO2); Clock (GPIO3) -EEPROM Data: (GPIO0); EEPROM Clock (GPIO)

Serial

-TX (GPIO14); RX (GPIO15)

4.2 DC motor:

Fig 5: *Pictorial view of DC motor*

A DC motor is any class of rotary electrical machines that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Almost all types of DC motors have some internal mechanism, either electromechanical or electronic, which periodically changes the direction of current flow in part of the motor. A coil of wire with a current running through it generates an electromagnetic field aligned with the center of the coil. One can change the direction and the magnitude of the magnetic field by changing the direction and magnitude of the current flowing through it.Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic; to

periodically change the direction of current flow in the part of DC motor. DC motors were the first type widely used, since they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances.

4.3 L298N Motor Driver:

L 298 is a dual full bridge driver that has a capability to bear high voltage as well as high current. It receives basic TTL (Transistor Transistor Logic) logic levels and is able to operate the different loads such as DC motors, stepper motors, relays etc. L-298 has two enable input to control any device by enabling or disabling it. L 298 IC is most commonly used to make motor drivers or motor controllers. These motor controllers can be controlled by any micro controller e.g Arduino, PIC, Raspberry Pi etc. hey receives input from micro controllers and operate the load attached to their output terminals correspondingly. L-298 motor driver (H-Bridge) is able to control two different DC motors simultaneously. While it can control a single stepper motor as well. L 298 has two Pulse Width Modulation (PWM) pins. PWM pins are used to control the speed of the motor. By changing the voltage signal's polarity at its input we can rotate the motor in either clockwise or counter clockwise direction. **L298** is a high current and high voltage IC. Its receives TTL logic signals and operates different loads like motors, solenoid, relays etc. It is mostly used in motor driver's designing. It has two specific pins for enabling or disabling the particular device attached at its output. Its features include low saturation voltage, over temperature protection

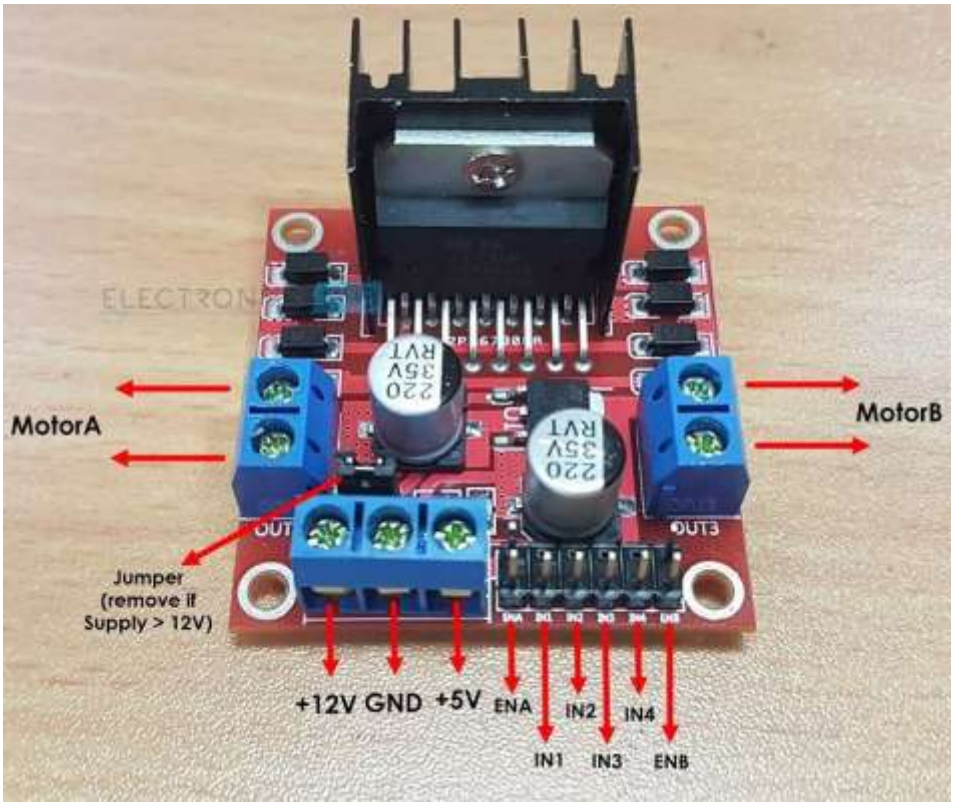


Fig 6: *Schematic view of motor driver L298N*

The L298N works on a H-bridge principle. In the following figure an H bridge is illustrated.

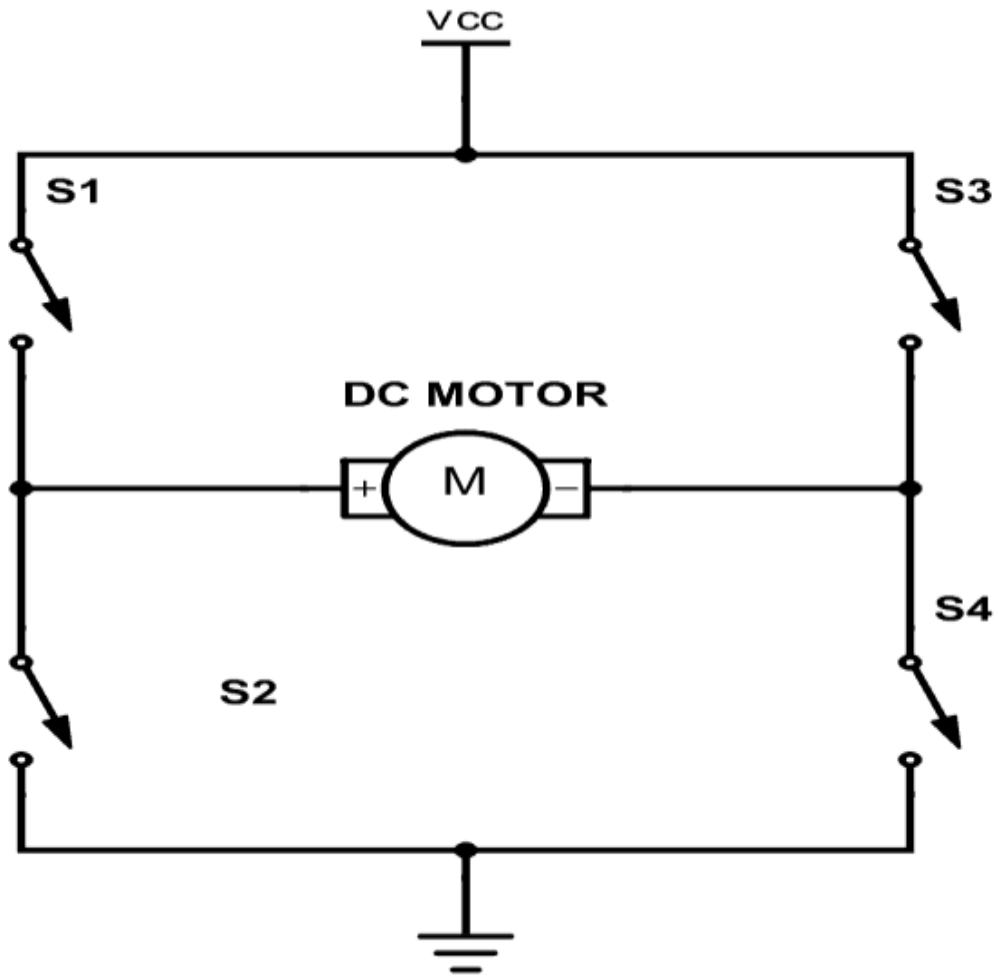


Fig 7: *Schematic view of H-Bridge*

When switch S1 and S4 are ON, motor spins in one direction. Now when switch s3 and S2 are On motor spins in opposite direction. By changing which switches are turned On we can change the direction of motor. The speed of the motor is controlled by turning the switches on and off continuously as explained in PWM .

- L 298 motor controller's each pin has different functions.
- The function associated with each of the pin are given in the table shown below.

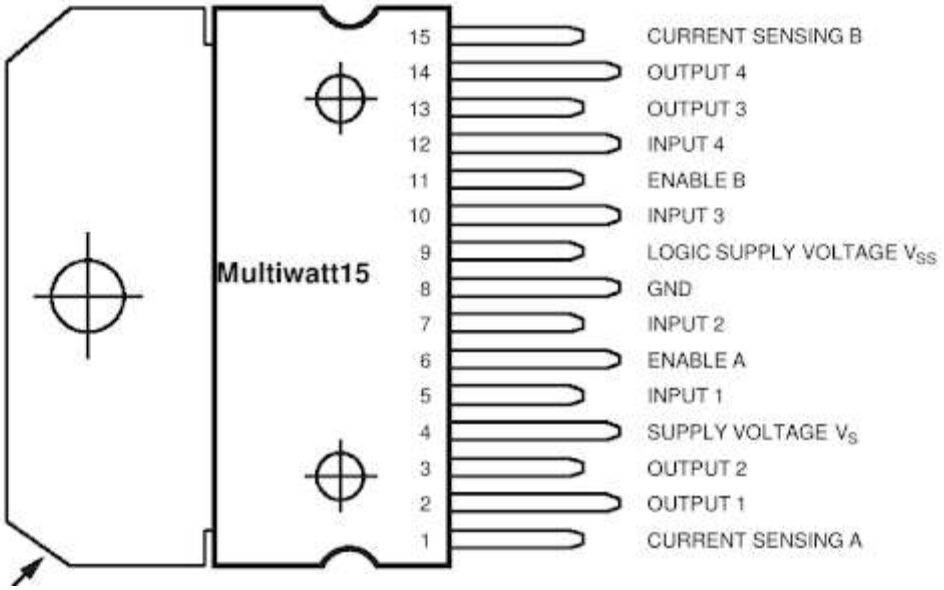


Fig 8 Pin out of Motor Driver L298N

4.4 IR Sensor:

An infrared sensor is an electronic device, that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion.These types of sensors measures only infrared radiation, rather than emitting it that is called as a passive IR sensor. Usually in the infrared spectrum, all the objects radiate some form of thermal radiations. These types of radiations are invisible to our eyes, that can be detected by an infrared sensor.The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR photodiode which is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photodiode, The resistances and these output voltages, change in proportion to the magnitude of the IR light received.

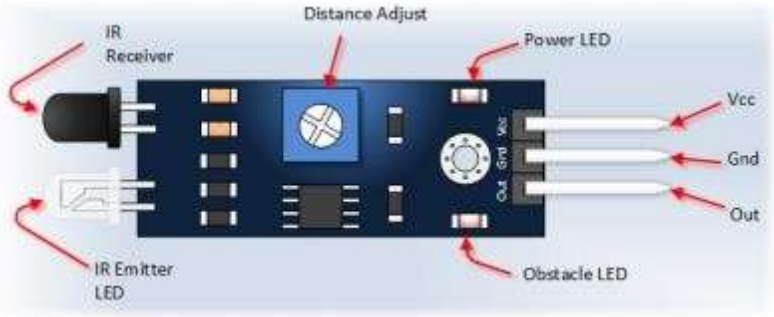


Fig 9: *schematic representation of IR sensor*

4.5 DHT 11:

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and gives out a digital signal on the data pin (no analog input pins needed). Simple to use, but requires careful timing to grab data. We can only get new data from it once every 2 seconds, so when using our library, sensor readings can be up to 2 seconds old..

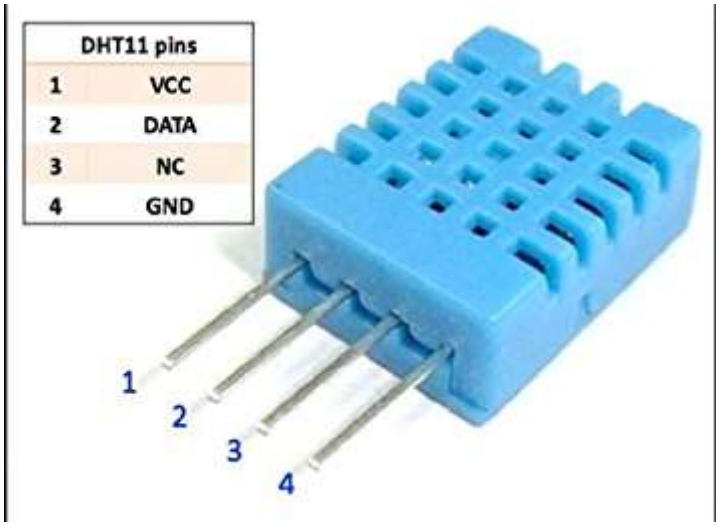


Fig 10: *schematic representation of DHT 11 sensor*

4.6 12V DC SUPPLY FOR MOTOR DRIVER

12V power supplies (or 12VDC power supplies) are one of the most common power supplies in use today. In general, a 12VDC output is obtained from a 120VAC or 240VAC input using a combination of transformers, diodes and transistors.We use a 1.5A 12V DC adapter in the project.

Fig 11: *Pictorial view of 12V DC supply*

4.7 5V DC SUPPLY FOR RASPBERRY PI3B+

Recommended and easiest way to power the Raspberry Pi is via the Micro USB port on the side of the unit. The recommended input voltage is 5V, and the recommended input current is 2A. A 5V 2.4A DC power supply is used in the project.

Fig 12: *Pictorial view of 5V DC supply*

5 Software Requirements:

A. **Raspbian**: Raspbian is the recommended operating system for normal use on a Raspberry Pi.

B. Python:

Python is a widely used high-level programming language for general-purpose programming first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly braces or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.

C. Tera Term :

Used to connect devices over IP address

D. Advance IP scanner:

Used to scan the IP address of the devices.

E. Vnc viewer:

Used to remotely control the pi over a GUI.

6 THEORY:

The speed of a permanent magnet DC motor can be controlled by changing the voltage applied to the armature.

The main principle in controlling a DC Motor with Raspberry Pi lies with the Motor Driver. A Motor Driver is a special circuit or IC that provides the necessary power (or rather the current) to the motor for smooth and safe operation.

Even a small 5V DC Motor draws a high initial current of around 300 – 400 mA. This current will then fall down 150 – 200 mA as the motor gains speed.

This is a huge current for devices like Microcontrollers, Arduino, Raspberry Pi etc. Hence, we should never connect a motor directly to Raspberry Pi

Here, a motor driver (L298N) will be used for varying speed of the motor based on PWM technique and the motor driver is controlled via Raspberry Pi 3B+.

The motor driver (L298N) is given with two control signals from Raspberry Pi through GPIO Pins. As per the Python Program, the motor will rotate in either forward or reverse direction.

Raspberry Pi 3B+ is an ARM architecture processor based board. There are 40 GPIO pins on the board for connecting other modules and sensors. The Raspberry PI 3B+ will be given standard power supply for its operation (5V) and internal use. To drive the DC motor one external 12 V DC power supply to be connected in the circuit. It has to be noted that it is not recommended to draw more than 50mA current from the Raspberry PI3B+ GPIO pins so an external dc power supply is used.

In order to make a GUI, the Tkinter Python module is used. It is an inbuilt Python library used to make GUI. The button feature in Tkinter is used to make buttons in GUI. These buttons are then associated with their corresponding function like start, stop, increase speed, decrease speed, stop ,exit.

To change the speed of the motor, the “ChangeDutyCycle()” function is used. The function takes in a value from 0-100 which translates from slowest speed to max speed.

For the automatic control of speed, a program is written that reads the temperature and humidity reading from the DHT11 sensor and displays it as a label in Tkinter.

The temperature and humidity sensor DHT 11 is used to control the speed of the motor according to the Ambient temperature.

To visualize the changes in speed of the motor a tachometer is needed. An IR sensor is used to implement a non-contact type tachometer to measure the RPM of the DC motor.

7 Methodology:

7.1 Configuring the Raspberry pi

In order to use the Raspberry Pi first an operating system is needed .

A SD card is used to install the raspberry pi. The Raspbian operating system is downloaded on it and the sd card is inserted in the raspberry pi. It is then connected to a monitor using a HDMI cable and powered by a 5V 2.4A micro USB power supply. The Raspberry Pi boots for the first time. The terminal window is opened on the raspberry pi desktop and "ifconfig" command is run which shows the IP address of the Raspberry Pi. This IP address is used to connect it to a laptop in future and no monitor is needed henceforth.

7.2 Connecting the Raspberry Pi to PC

When the Raspberry Pi IP address is found, now to connect to a pc we need a software called Tera Term.

Using this software, we can connect to the Raspberry Pi and access the shell or non-graphical UI of the raspberry.

A Rj45 or LAN cable is used to connect Raspberry Pi to the PC. Then Tera Term is opened and the IP address of the raspberry Pi is give in. The SSH connection is selected since the connection mode is SSH.

7.2.1 SSH

The SSH protocol uses encryption to secure the connection between a client and a server. All user authentication, commands, output, and file transfers are encrypted to protect against attacks in the network We can access the command line of a Raspberry Pi remotely from another computer or device on the same network using SSH. The Raspberry Pi will act as a remote device: we can connect to it using a client on another machine.

The IP address of our particular Raspberry Pi is put in and connection is made along with the mentioned settings.

The default username and password is pi and raspberry respectively

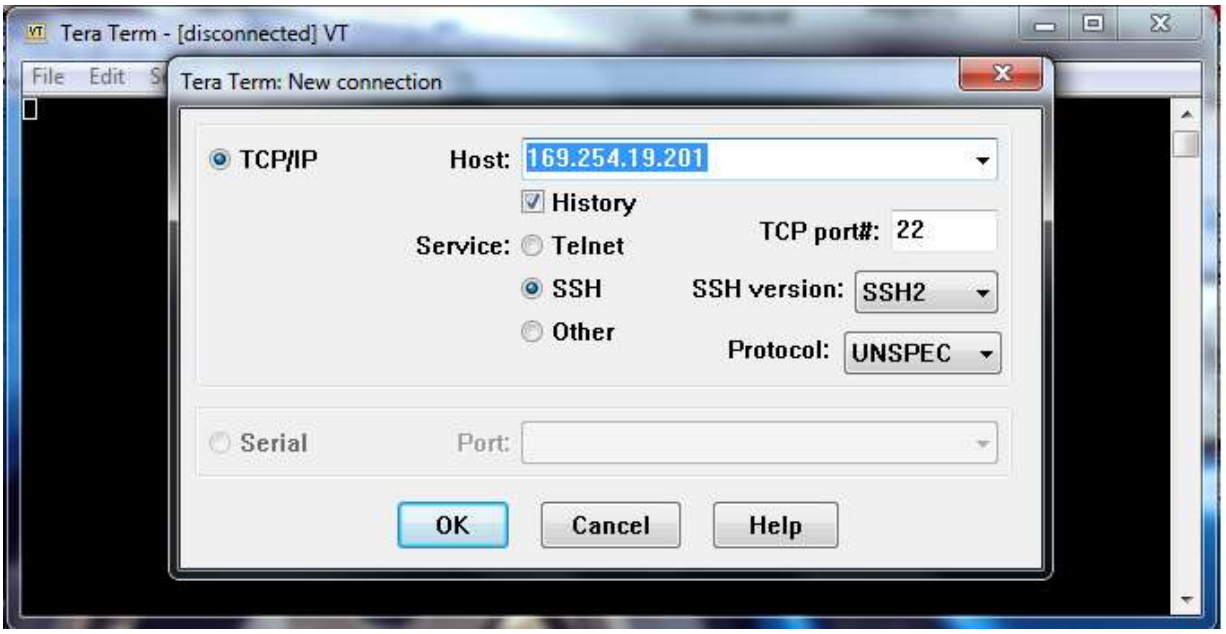


Fig 13: *Tera Term connection screen*

After successful connection, in order to access the Raspberry Pi desktop we need another software Tight VNC Viewer.
First we need to install tight VNC server. We do that by typing following command in shell UI

```
sudo apt-get update
sudo apt-get install realvnc-vnc-server
```

Then we open Tight VNC viewer and again give ip , username and password same as before.

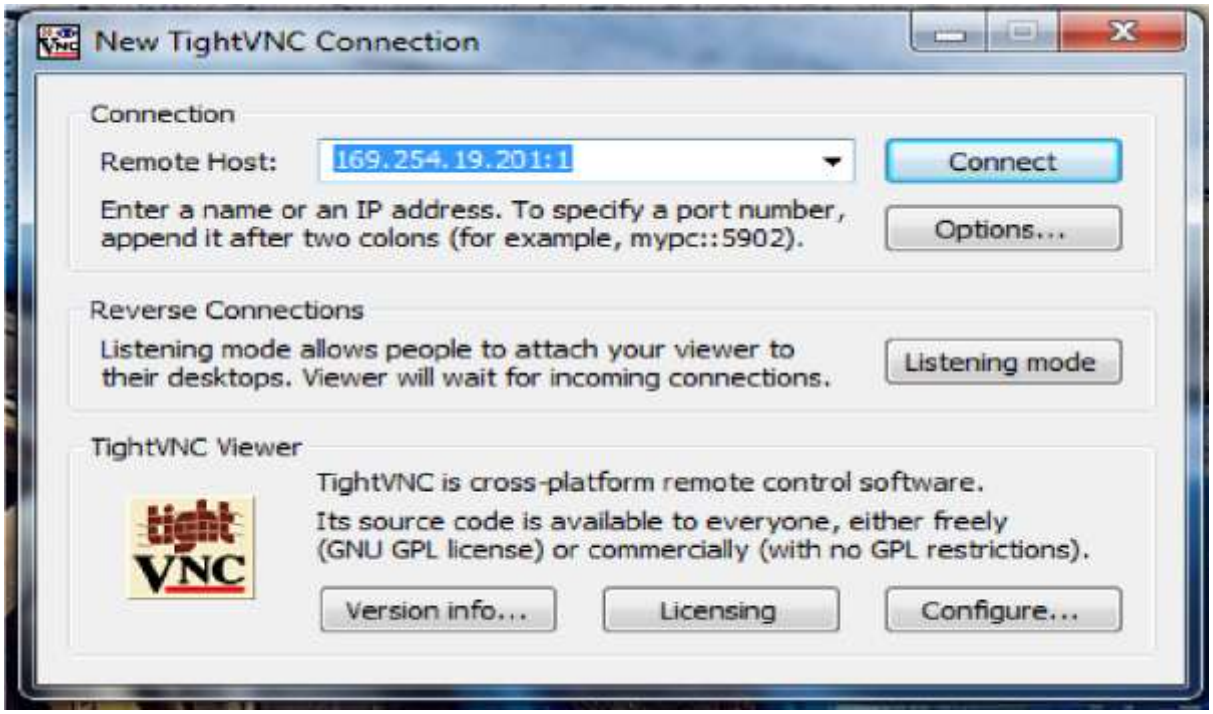


Fig 14: Tight VNC *connection screen*

On successful connection it will look like this

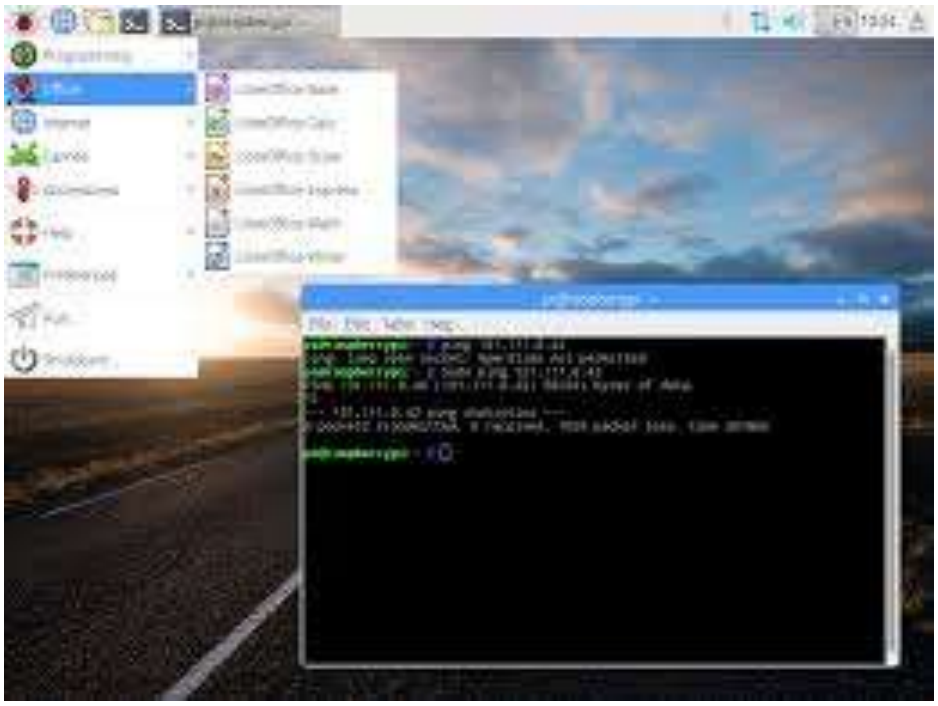


Fig 15: *Raspberry Pi desktop*

7.3 Required programs are written in Python environment based on following :

7.3.1 Speed and direction control of a DC motor

The speed of a permanent magnet dc motor can be controlled by changing the voltage applied to the armature. If the polarity is reversed on the armature ,the motor would spin in the reverse direction. Using this principle , the motor's speed and direction is controlled. However, the raspberry pi cannot directly control the speed and direction of the motor as it does not provide the necessary current and voltage for driving the motor. Moreover, if more current is drawn from the pins of the raspberry pi ,it may get damaged.

Hence, a motor driver is needed that interfaces between the raspberry pi and the motor. The motor driver can handle high voltage and current. In our project we use a l 298 motor driver. It is dual bridge motor driver which means it can run two motors from one motor driver.

In order to control the speed of the motor, pulse width modulation or pwm technique is used..

7.3.2 PWM (Pulse width modulation)

Pulse-width modulation (**PWM**), or **pulse-duration modulation** (**PDM**), is a method of reducing the average power delivered by an electrical signal, by effectively chopping it up into discrete parts. The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast rate. The longer the switch is on compared to the off periods, the higher the total power supplied to the load PWM is particularly suited for running inertial loads such as motors, which are not as easily affected by this discrete switching, because they have inertia to react slow. The PWM switching frequency has to be high enough not to affect the load, which is to say that the resultant waveform perceived by the load must be as smooth as possible.

The main advantage of PWM is that power loss in the switching devices is very low. When a switch is off there is practically no current, and when it is on and power is being transferred to the load, there is almost no voltage drop across the switch. Power loss, being the product of voltage and current, is thus in both cases close to zero. PWM also works well with digital controls, which, because of their on/off nature, can easily set the needed duty cycle..

The term duty cycle describes the proportion of 'on' time to the regular interval or 'period' of time; a low duty cycle corresponds to low power, because the power is off for most of the time. Duty cycle is expressed in percent, 100% being fully on. When a digital signal is on half of the time and off the other half of the time, the digital signal has a duty cycle of 50% and resembles a "square" wave. When a digital signal spends more time in the on state than the off state, it has a duty cycle of >50%. When a digital signal spends more time in the off state than the on state, it has a duty cycle of <50%. Here is a pictorial that illustrates these three scenarios:

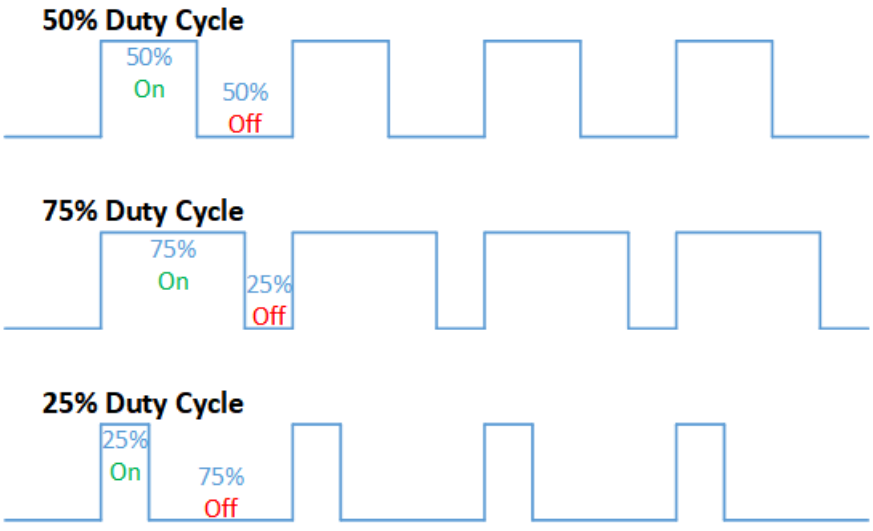


Fig 16: *Duty Cycle illustration*

7.4 Tkinter module of Python

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

A program is written to incorporate a GUI to control the speed and direction of the motor.

The button and label functions are used here in the project to make the GUI. To make all the buttons look like as needed, an image is displayed on the buttons which when pressed gives the desired result.

Here is the “button” function of Tkinter which on pressing executes the associated function.

Another program is written to automatically control the speed and direction of the motor depending on the temperature measured by the temperature and humidity sensor DHT 11.

Fig 17: *Tkinter Button Illustration*

To format the default buttons, we display relevant images on the buttons itself. These are the images incorporated in the buttons

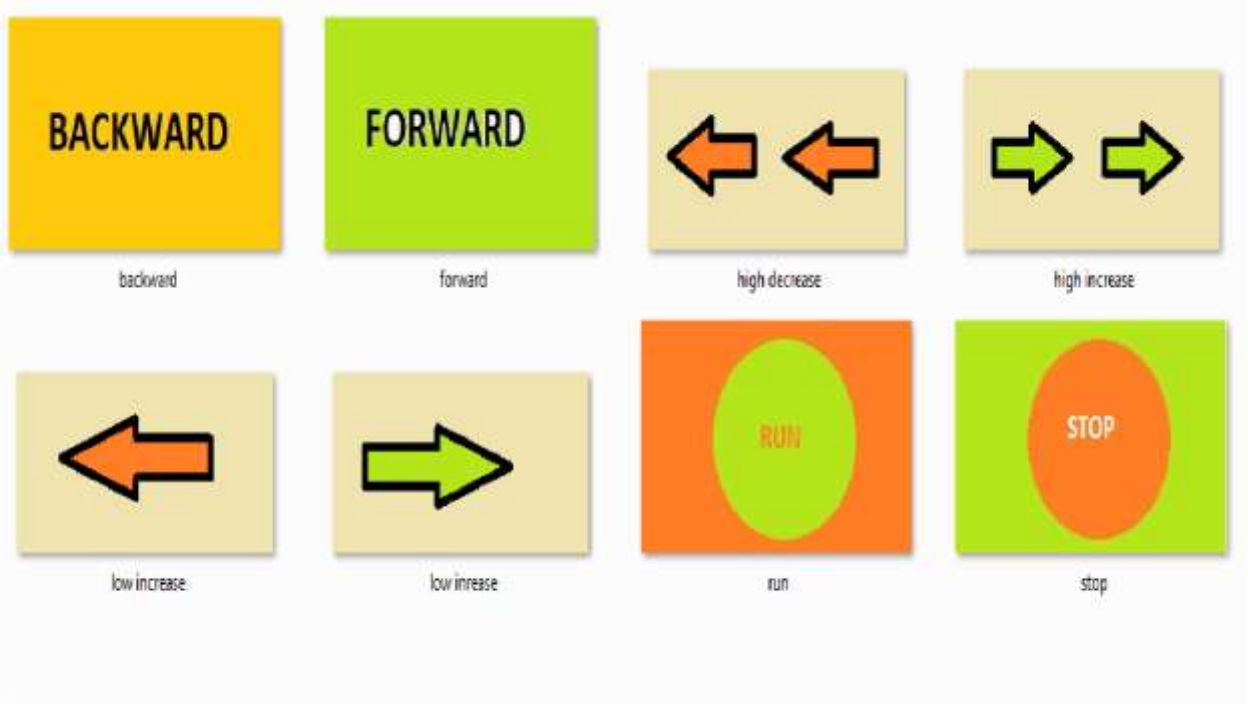


Fig 18: *Tkinter Button Formatting Images*

7.5 Making the Tachometer

A tachometer (revolution-counter, tacho, rev-counter, RPM gauge) is an instrument measuring the rotation speed of a shaft or disk, as in a motor or other machine

A disc is attached to the shaft of the motor. The disc is painted solid black with one white strip of paper pasted to it.

An IR sensor is so placed that is detects the changed of the disc color from white to black when the disc rotates.

On each occasion of the white strip, the program is configured to read one rpm.

The program measures how long it takes for 30 rotations to complete and then it converts the value to rotations per minute.

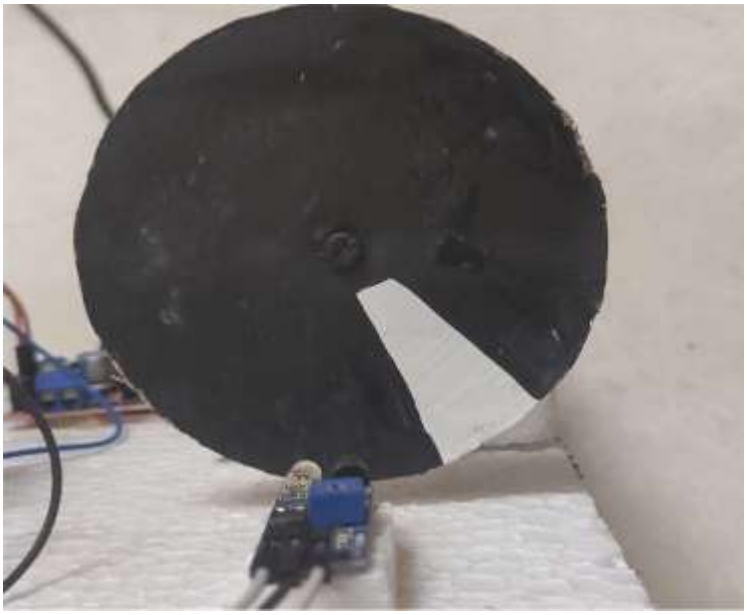


Fig 19: *RPM measuring disc arrangement*

8 Physical connection layout of project:

The connection is made as per circuit diagram. Motor is connected to motor driver. The motor driver is connected to Raspberry at GPIO pins 23,24 and 25.

The IR sensor is connected to pin 21 and a 3.3V pin out from raspberry is used to power the IR sensor.

The DHT11 data pin is connected to pin 17 in raspberry PI and a 5V supply is taken from the Raspberry pins to power it.

The raspberry is powered by the 5V supply and motor driver is powered by the 12V DC supply.

The final connection is shown in the figure.

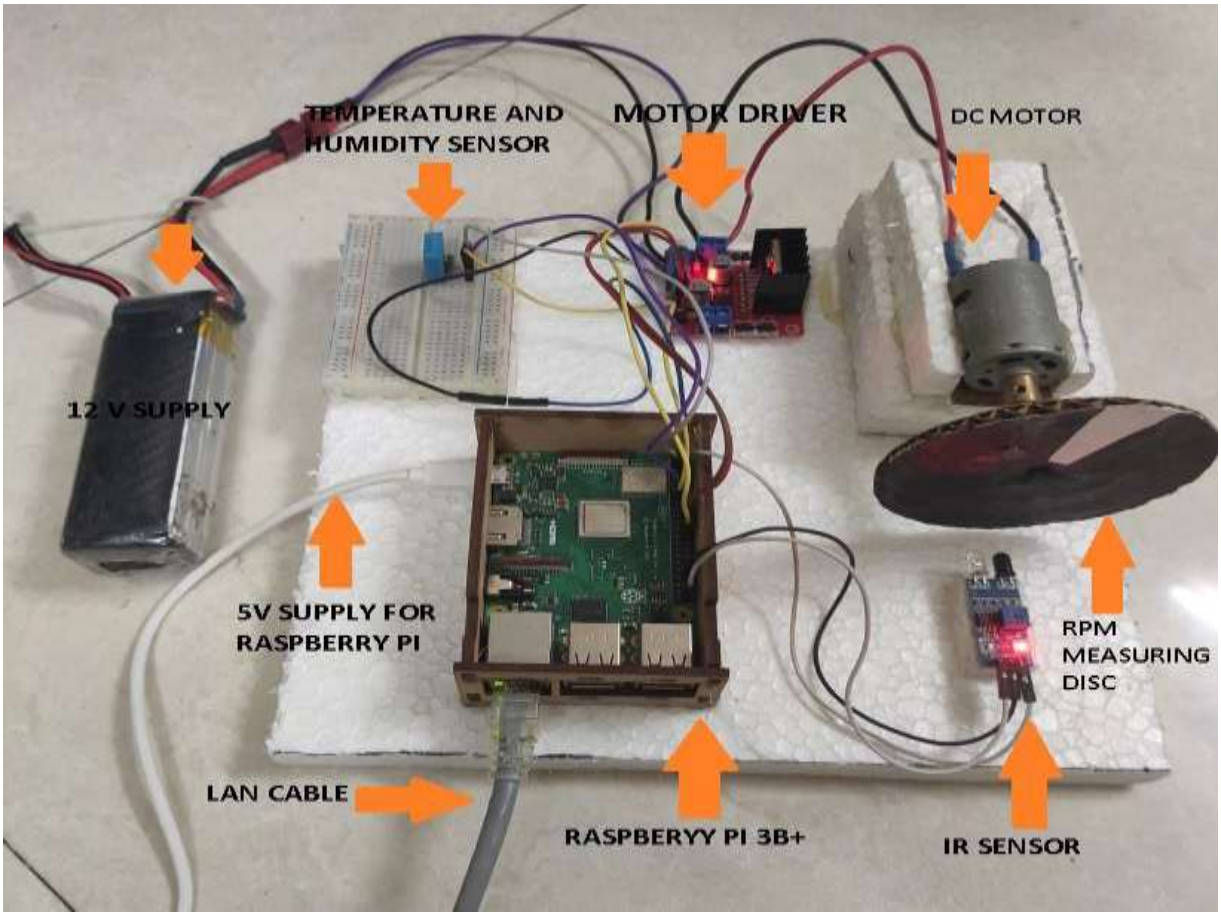


Fig 20: physical connection layout of Raspberry Pi with breadboard

9 Software program developed for the proposed project:

9.1 Manual speed and Direction control using GUI code

```
from tkinter import*
import tkinter.font
import RPi.GPIO
import RPi.GPIO as GPIO
import time
from time import sleep
RPi.GPIO.setmode(RPi.GPIO.BCM)

in1 = 24
in2 = 23
en = 25
sensor = 21 # define the GPIO pin our sensor is attached to

GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor,GPIO.IN) # set our sensor pin to an input
GPIO.setup(in1,GPIO.OUT)
GPIO.setup(in2,GPIO.OUT)
GPIO.setup(en,GPIO.OUT)
GPIO.output(in1,GPIO.LOW)
GPIO.output(in2,GPIO.LOW)
p=GPIO.PWM(en,1000)
limit=20
p.start(limit)
sample = 15# how many half revolutions to time
count = 0

start = 0
end = 0



## GUI def
win=Tk()
win.title("Speed and Direction Control")
myFont=tkinter.font.Font(family="Helvetica",size= 12 ,weight ="bold")

ph=PhotoImage(file="//home/pi/Desktop/run.png")
ph1=PhotoImage(file="//home/pi/Desktop/stop.png")
ph2=PhotoImage(file="//home/pi/Desktop/low increase.png")
ph3=PhotoImage(file="//home/pi/Desktop/high increase.png")
ph4=PhotoImage(file="//home/pi/Desktop/low decrease.png")
ph5=PhotoImage(file="//home/pi/Desktop/high decrease.png")
```

```
ph6=PhotoImage(file="//home/pi/Desktop/forward.png")
ph7=PhotoImage(file="//home/pi/Desktop/backward.png")
count =0
def rpm():
 def set_start():
  global start
 start = time.time()

 def set_end():
  global end
 end = time.time()

 def get_rpm(c):
  global count
# delcear the count variable global so we can edit it

 if not count:
set_start() # create start time
count = count + 1 # increase counter by 1
 else:
count = count + 1

 if count==sample:
set_end() # create end time
delta = end - start # time taken to do a half rotation in seconds
delta = delta / 60 # converted to minutes
rpm = (sample / delta) # converted to time for a full single rotation
 print ("RPM   = ")
  print (rpm)
count = 0 # reset the count to 0

GPIO.add_event_detect(sensor, GPIO.RISING, callback=get_rpm) # execute the get_rpm function
when a HIGH signal is detected




##define
def run():
  temp1=1
 print("run")
startButton["text"]="Motor is on"
if(temp1==1):
GPIO.output(in1,GPIO.HIGH)
GPIO.output(in2,GPIO.LOW)
print("forward")
```

```
x='z'
else:
GPIO.output(in1,GPIO.LOW)
GPIO.output(in2,GPIO.HIGH)
print("backward")
x='z'
rpm()
def stop():
print("stop")
stopButton["text"]="Stop a"
GPIO.output(in1,GPIO.LOW)
GPIO.output(in2,GPIO.LOW)
x='z'

def forward():
print("forward")
GPIO.output(in1,GPIO.HIGH)
GPIO.output(in2,GPIO.LOW)
temp1=1
x='z'
rpm()

def backward():
print("backward")
GPIO.output(in1,GPIO.LOW)
GPIO.output(in2,GPIO.HIGH)
temp1=0
x='z'

def lowd():
global limit
print("low decrease")
limit=limit-10
if limit<20:
limit=25
p.ChangeDutyCycle(limit)
x='z'

def mediumd():
global limit
print("hi decrease")
limit=limit-30
if limit<20:
limit=20
p.ChangeDutyCycle(limit)
x='z'
```

```
def lowinc():
    global limit
print("low increase")
limit=limit+10
 if limit>100:
    limit=100
p.ChangeDutyCycle(limit)
x='z'

def mediuminc():
    global limit
print("high increase")
limit=limit+30
 if limit>100:
    limit=100
p.ChangeDutyCycle(limit)
x='z'


def exi():
  GPIO.cleanup()
  win.destroy()


###widgets
startButton=Button(win, text =" Start Motor ", font=myFont ,command = run)
startButton.config(image=ph,compound=BOTTOM)
startButton.grid(row=0,column=0)

stopButton=Button(win, text =" Stop Motor ",font = myFont , command = stop)
stopButton.config(image=ph1,compound=BOTTOM)
stopButton.grid(row=0,column=1)

fwdButton=Button(win, text =" Forward Rotation ",font = myFont , command =forward)
fwdButton.config(image=ph6,compound=BOTTOM)
fwdButton.grid(row=2,column=0)

bwdButton=Button(win, text =" Backward Rotation ",font = myFont , command =backward)
bwdButton.config(image=ph7,compound=BOTTOM)
bwdButton.grid(row=2,column=1)

ldButton=Button(win, text ="Low Decrease",font = myFont , command =lowd)
ldButton.config(image=ph4,compound=BOTTOM)
ldButton.grid(row=4,column=0)
```

```
hdButton=Button(win, text =" High Decrease ",font = myFont , command =mediumd)
hdButton.config(image=ph5,compound=BOTTOM)
hdButton.grid(row=4,column=1)

liButton=Button(win, text ="Low Increase",font = myFont , command =lowinc)
liButton.config(image=ph2,compound=BOTTOM)
liButton.grid(row=3,column=0)

hiButton=Button(win, text =" High Increase ",font = myFont , command =mediuminc)
hiButton.config(image=ph3,compound=BOTTOM)
hiButton.grid(row=3,column=1)
```

9.2 Automatic speed control based on temperature in GUI code

```
import sys
import Adafruit_DHT
from tkinter import*
import tkinter.font
import RPi.GPIO
import RPi.GPIO as GPIO
import time
from time import sleep
RPi.GPIO.setmode(RPi.GPIO.BCM)
sensor1 = 21
in1 = 24
in2 = 23
en = 25
sensor=Adafruit_DHT.DHT11
gpio=17

GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor1,GPIO.IN)
GPIO.setup(in1,GPIO.OUT)
GPIO.setup(in2,GPIO.OUT)
GPIO.setup(en,GPIO.OUT)
GPIO.output(in1,GPIO.LOW)
GPIO.output(in2,GPIO.LOW)
```

```
p=GPIO.PWM(en,1000)
limit=15
p.start(limit)

sample = 20# how many half revolutions to time
count = 0

start = 0
end = 0


win=Tk()
var1=StringVar()
var2=StringVar()
myFont=tkinter.font.Font(family="Helvetica",size= 30 ,weight ="bold")
win.title("Automatic Control")
Label1=Label(win,textvariable=var1,width=30,height=5).pack()

Label2=Label(win,textvariable=var2,width=30,height=5).pack()
rpm1=0
def rpm():

def set_start():
    global start
start = time.time()

def set_end():
    global end
end = time.time()

def get_rpm(c):
    global count
# delcear the count variable global so we can edit it

if not count:
set_start() # create start time
count = count + 1 # increase counter by 1
else:
count = count + 1

if count==sample:
set_end() # create end time
delta = end - start # time taken to do a half rotation in seconds
delta = delta / 60 # converted to minutes
rpm1 = (sample / delta) # converted to time for a full single rotation
```

```
print ("RPM   = ")
print (rpm1)
time.sleep(4)
count = 0 # reset the count to 0

GPIO.add_event_detect(sensor1, GPIO.RISING, callback=get_rpm) # execute the get_rpm function
when a HIGH signal is detected
rpm()
while(1):
humidity, temperature = Adafruit_DHT.read_retry(sensor,gpio)
print('Temp={0:0.1f}*C Humidity={1:0.1f}%'.format(temperature,humidity))
GPIO.output(in1,GPIO.HIGH)
GPIO.output(in2,GPIO.LOW)

var1.set("Centigrade ^C ---> "+str(temperature))
var2.set("Humidity % ------> "+str(humidity))
limit=25+((temperature-25)*2.5)
print(limit)
if limit>100:
limit=100
p.ChangeDutyCycle(limit)
win.update()

continue




exiButton=Button(win, text =" EXIT ",font = myFont , command =exi)
exiButton.grid(row=5,column=1)
```

10 Observation and Experimental Results :

10.1 Manual Speed and Direction control using GUI Observation and Results :

In the manual control mode, the program when executed opens a GUI which lets the user control the motor. The Start motor starts the motor and stop motor stops it. The forward and backward rotation buttons spin the motor in clockwise and anticlockwise direction when pressed respectively. The low increase and low decrease buttons increase and decrease the motor speed slowly.

While the high increase and high decrease buttons increase and decrease the motor speed rapidly.

The exit button closes the windows. Since the RPM function updates the rpm very rapidly, it is displayed in the shell of the Python UI when run. Below is the pictorial view of the manual GUI

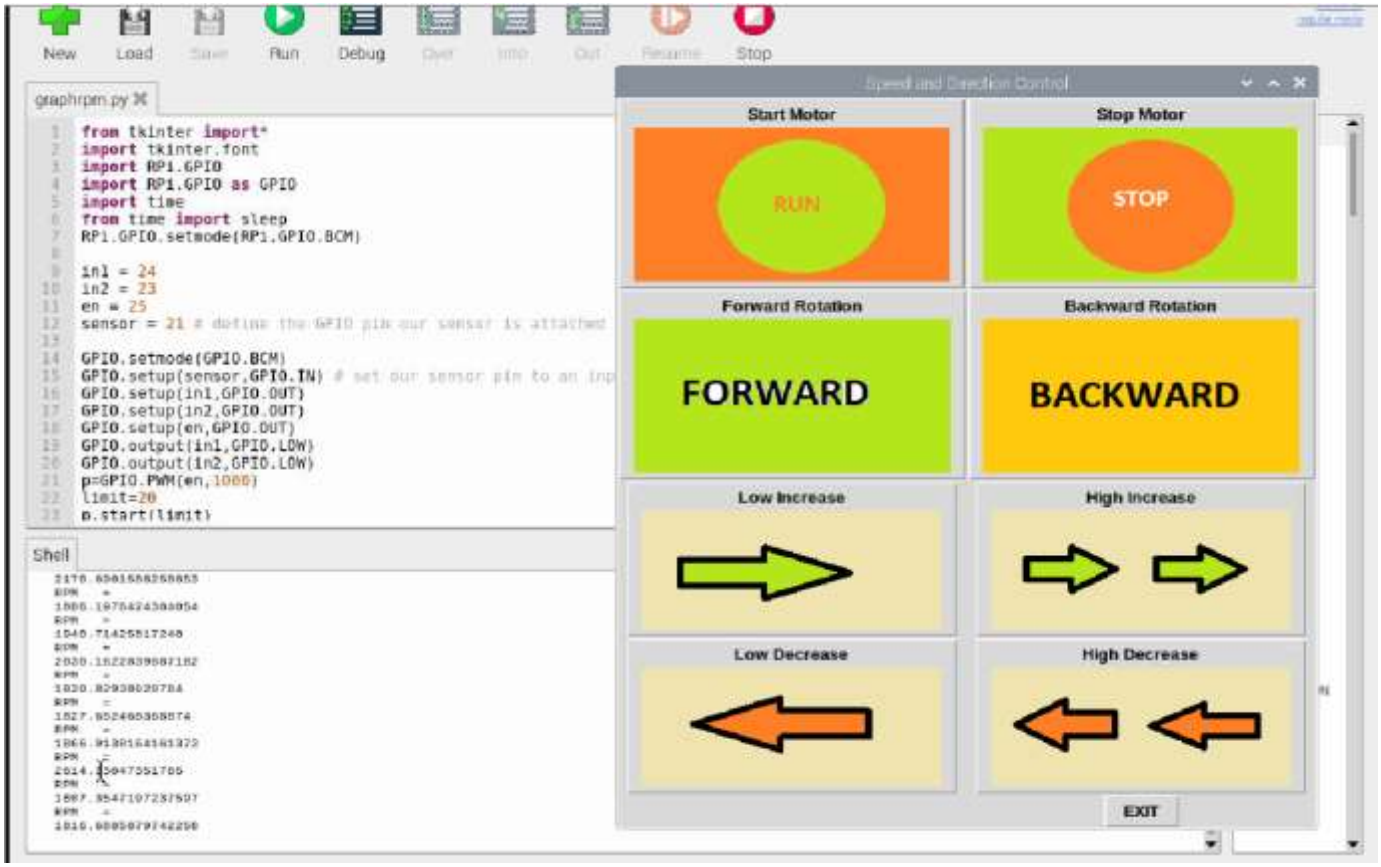


Fig 21: *Manual GUI interface along with rpm of the proposed project*

10.1 Automatic Speed control displayed in GUI Observation and Results:

In automatic control, the DHT 11 is used to get the ambient temperature and humidity .As soon as this is received, it is displayed as a label in Tkinter window as shown below. It shows the temperature in centigrade and humidity in percentage. With increase in temperature, speed of the motor increases. However, this is not rapid as the DHT 11 updates data after every 2 seconds. The rpm is displayed in the shell UI. The pictorial view is shown in the following figures

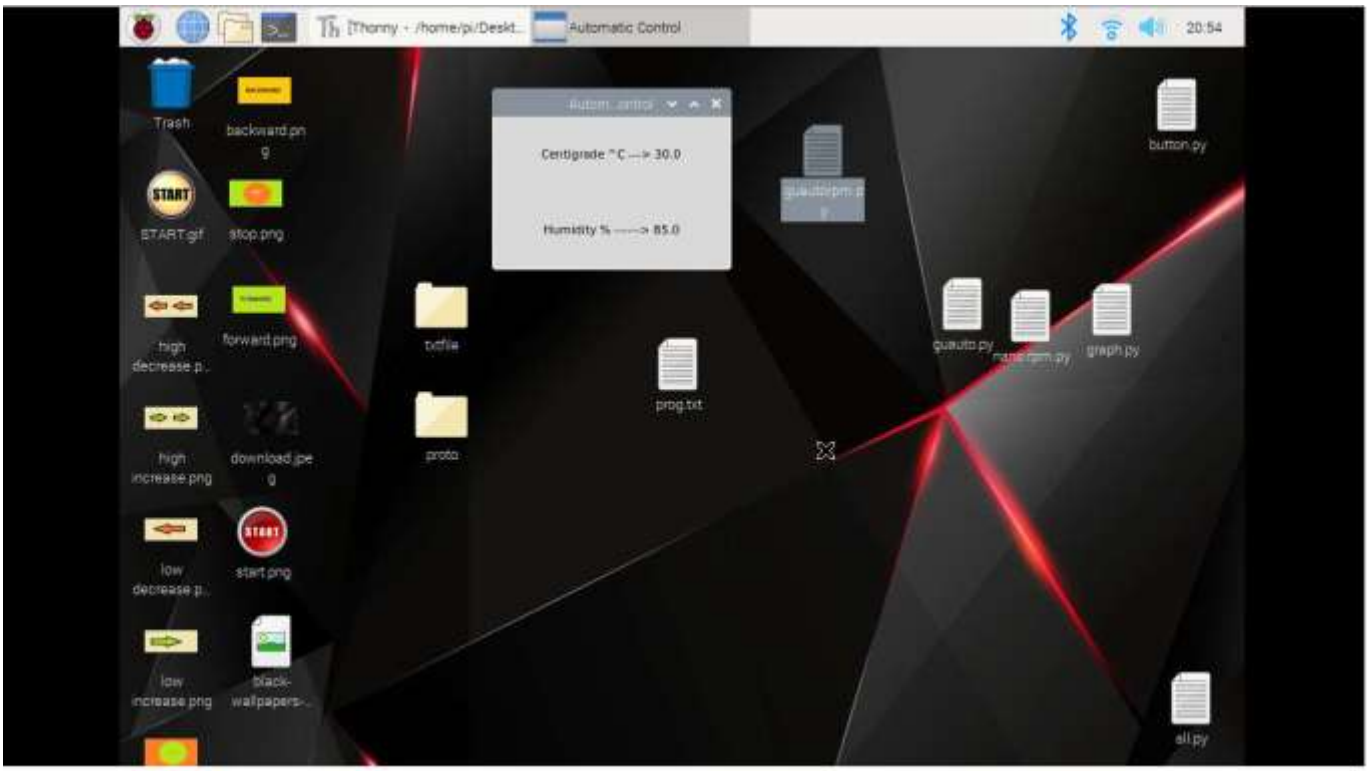


Fig 22: *Automatic GUI interface of the proposed project*

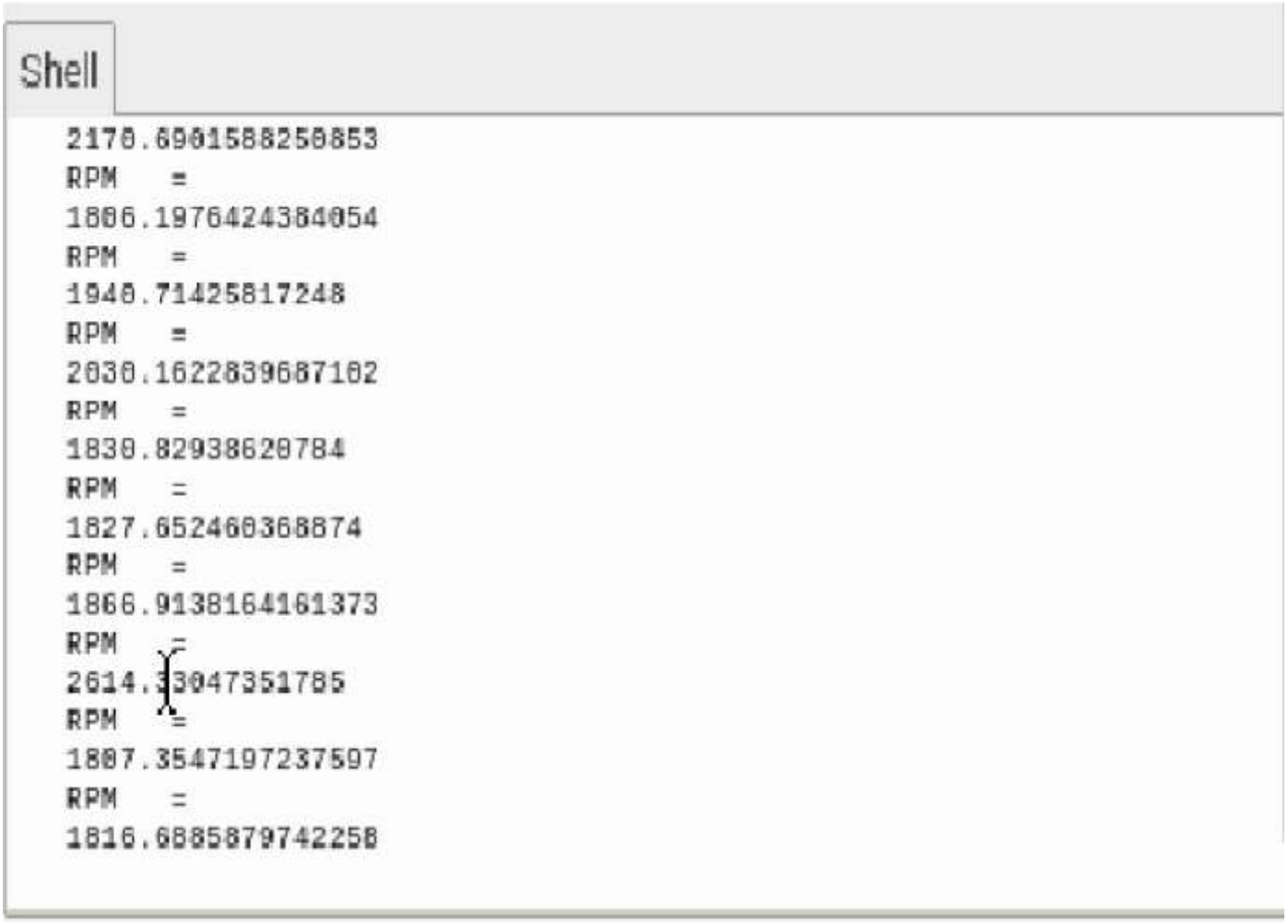


Fig 23: *Rpm display of the proposed project*

11 Conclusion:

1. In this project, a prototype all-inclusive DC motor sped control using GUI is implemented.
2. The proposed system includes control and regulation of speed using the GUI. Using the DHT sensor, automatically speed of motor is controlled based on ambient temperature.
3. The speed measuring system displays the speed in rpm in both manual GUI mode and Automatic mode.
4. Raspberry Pi proves to be a smart, economic and efficient platform for implementing this project.
5. The automatic system is slow to react as the DHT11 sensor has a time delay of 2 seconds before displaying last data.
6. The system can be controlled by any remote pc as long as a LAN connection is present with the raspberry PI3B+

11.1 FUTURE SCOPE:

1. Our project is a prototype, hence there is a lot or room for improvement.
2. Two power supplies are used in our project. One to power the Raspberry PI3b+ and another to power the motor driver. It would be better if one universal varying power supply was made to provide power to both.
3. The Rpm meter is not very accurate. Since there is huge time delay between sensing signal and transmitting it in the IR sensor, the actual rpm may vary. Also, the transition from black to white on the rotating disc is not ideal which further adds to the error.
4. Using the rpm meter, the whole system can be made closed loop system where the RPM value can be taken as feedback and we can fix the rpm of motor to a fixed desired value using control system techniques.
5. The project can easily be made wireless by connecting the Raspberry Pi wirelessly to a PC using Wifi and the whole system would become wireless.

12 SPECIFICATION OF HARDWARE COMPONENTS:

Raspberry Pi 3B+

- SoC: Broadcom BCM2837B0 quad-core A53 (ARMv8) 64-bit @ 1.4GHz
- GPU: Broadcom Videocore-IV
- RAM: 1GB LPDDR2 SDRAM
- Networking: Gigabit Ethernet (via USB channel), 2.4GHz and 5GHz 802.11b/g/n/ac Wi-Fi
- Bluetooth: Bluetooth 4.2, Bluetooth Low Energy (BLE)
- Storage: Micro-SD 8 GB class 10
- GPIO: 40-pin GPIO header, populated
- Ports: HDMI, 3.5mm analogue audio-video jack, 4x USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)
- Dimensions: 82mm x 56mm x 19.5mm, 50***g***

***DC Motor*:**

- **Motor Type :** DC motor
- **Maximum Torque:** ~0.4 Kg-cm at 12V
- **RPM :** 8000 RPM at 12V
- **Weight** :145 Gms
- **No load current :** 0.13A
- **Max Load Current:** ~5A at 12V

***IR sensor* :**

- 5VDC Operating voltage
- I/O pins are 5V and 3.3V compliant
- Range: Up to 20cm
- Adjustable Sensing range
- Built-in Ambient Light Sensor
- 20mA supply current
- Mounting hole

DHT 11 **:**

• CB size	22.0mm X 20.5mm X 1.6mm
• Working voltage	3.3 or 5V DC
• Operating voltage	3.3 or 5V DC
• Measurement range	20-95%RH ; 0-50℃
• Resolution	8bit（temperature）， 8bit（humidity）
• Compatible interfaces	2.54 3-pin interface and 4-pin Grove interface(1)

L298n **:**

- Dual H Bridge Motor Driver
- L298N motor driver IC
- Drives up to 2 bidirectional DC motors
- Integrated 5V power regulator
- 5V – 35V drive voltage
- 2A max drive current

12V DC Adapter

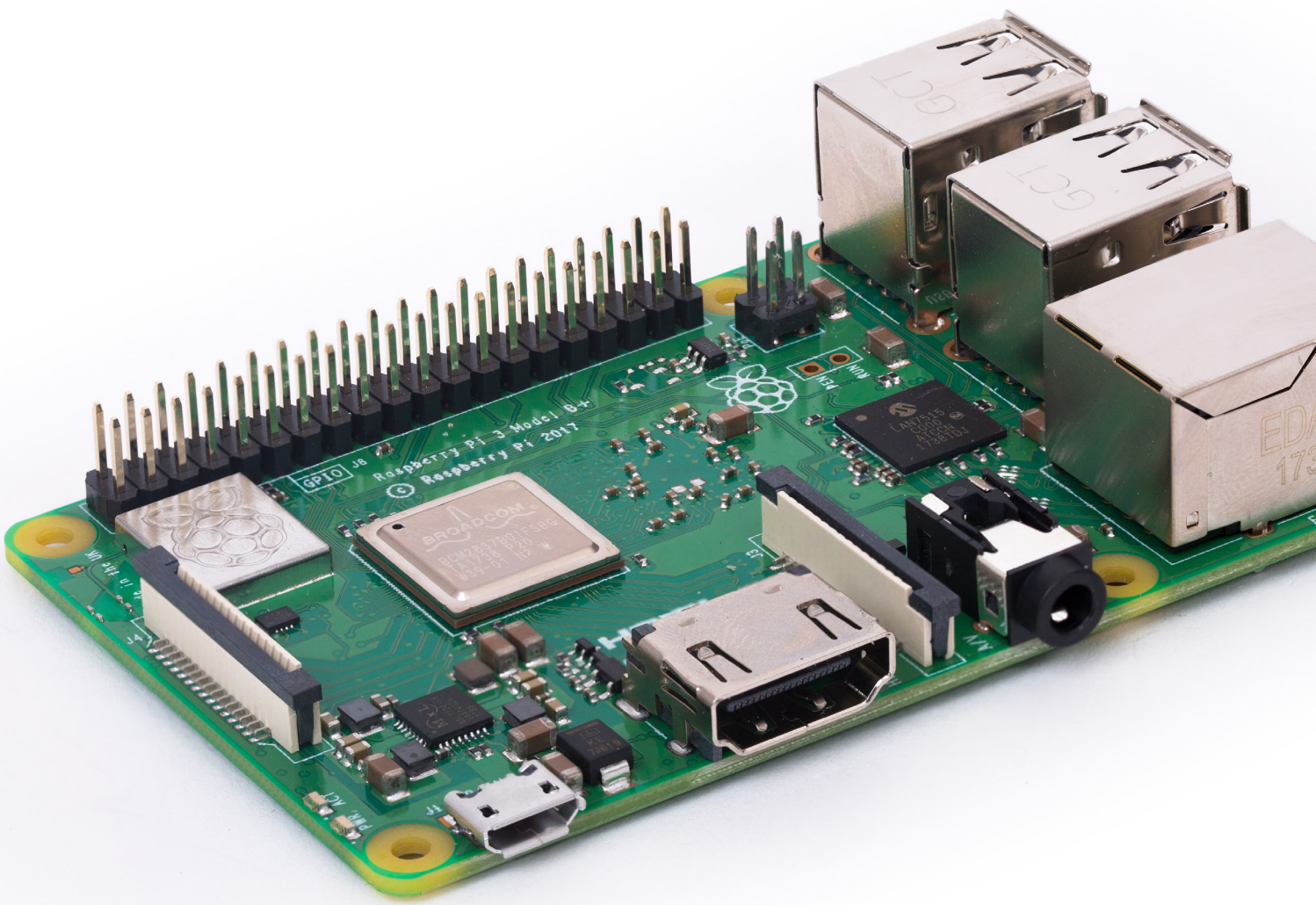
- Operating voltage : 100-240V AC
- Input : AC(100-240)V 50/60Hz 0.3A
- Output: 12V 2A

13 References:

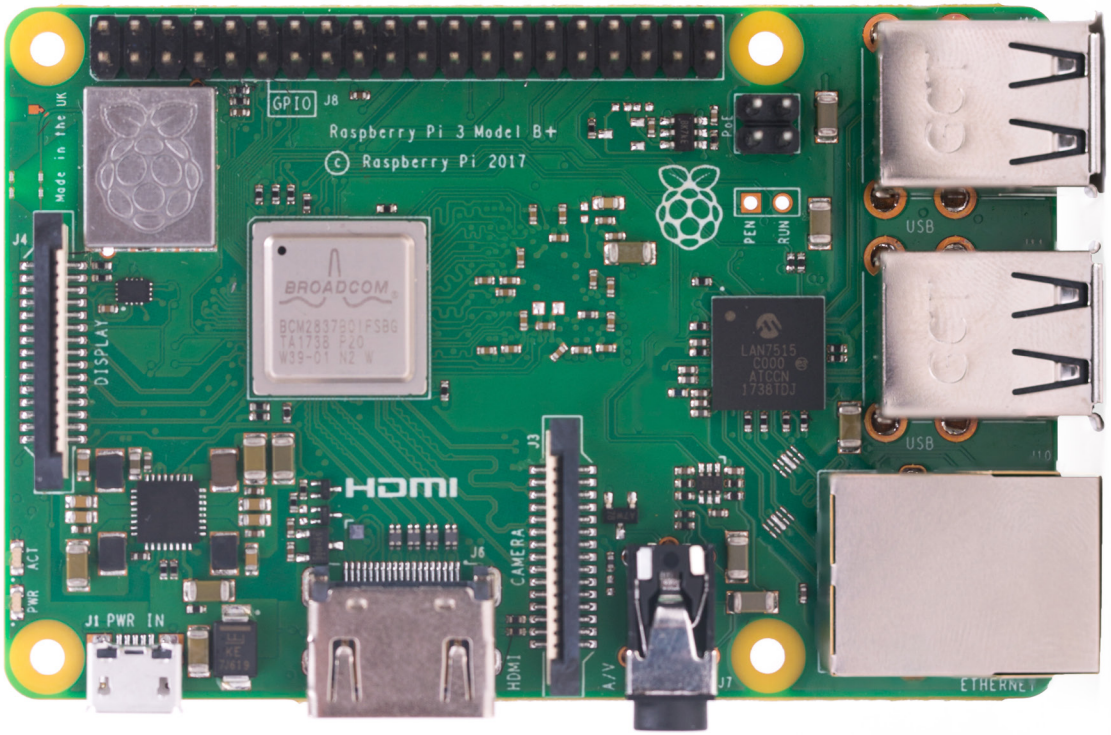
1. https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
2. https://www.raspberrypi.org/help/quick-start-guide/2/
3. https://www.electronicshub.org/raspberry-pi-l298n-interface-tutorial-control-dc-motor-l298n-raspberry-pi/
4. https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
5. Python Programming: An Introduction to Computer Science, by John Zelle.
6. Python Crash Course: A Hands-On, Project-Based Introduction to Programming, by Eric Matthes.
7. https://docs.python.org/2/library/tkinter.html
8. https://www.instructables.com/id/Tutorial-for-MD-L298-Motor-Driver-Module/
9. https://www.instructables.com/id/Controlling-Direction-and-Speed-of-DC-Motor-Using-/
10. https://howchoo.com/g/mjg5ytzmnjh/controlling-dc-motors-using-your-raspberry-pi
11. https://www.geeksforgeeks.org/python-gui-tkinter/
12. Head-First Python (2nd edition)by Paul Barry
13. Learn Python the Hard Way (3rd Edition) by Zed A.

DATA SHEET

Raspberry Pi 3 Model B+



Overview



The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4 GHz, dual-band 2.4 GHz and 5 GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT

The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

Specifications

Processor:
Broadcom BCM2837B0, Cortex-A53
64-bit SoC @ 1.4 GHz

Memory:
1GB LPDDR2 SDRAM

Connectivity:
- 2.4 GHz and 5 GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- 4 × USB 2.0 ports

Access:
Extended 40-pin GPIO header

Video & sound:
- 1 × full size HDMI
- MIPI DSI display port
- MIPI CSI camera port
- 4 pole stereo output and composite video port

Multimedia:
H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics

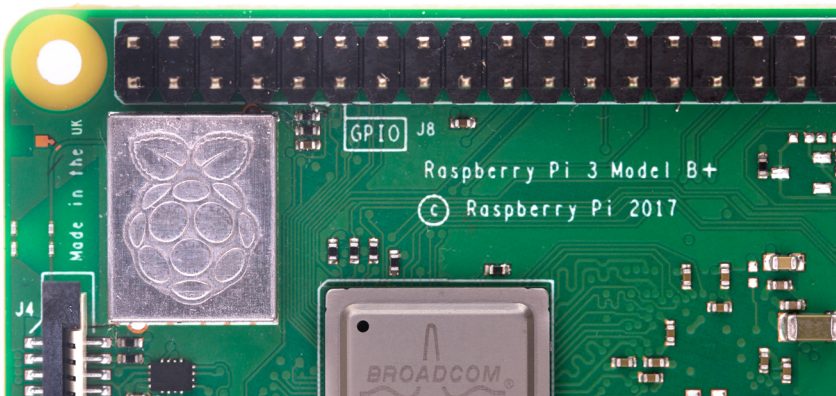
SD card support:
Micro SD format for loading operating system and data storage

Input power:
- 5 V/2.5 A DC via micro USB connector
- 5 V DC via GPIO header
- Power over Ethernet (PoE)–enabled (requires separate PoE HAT)

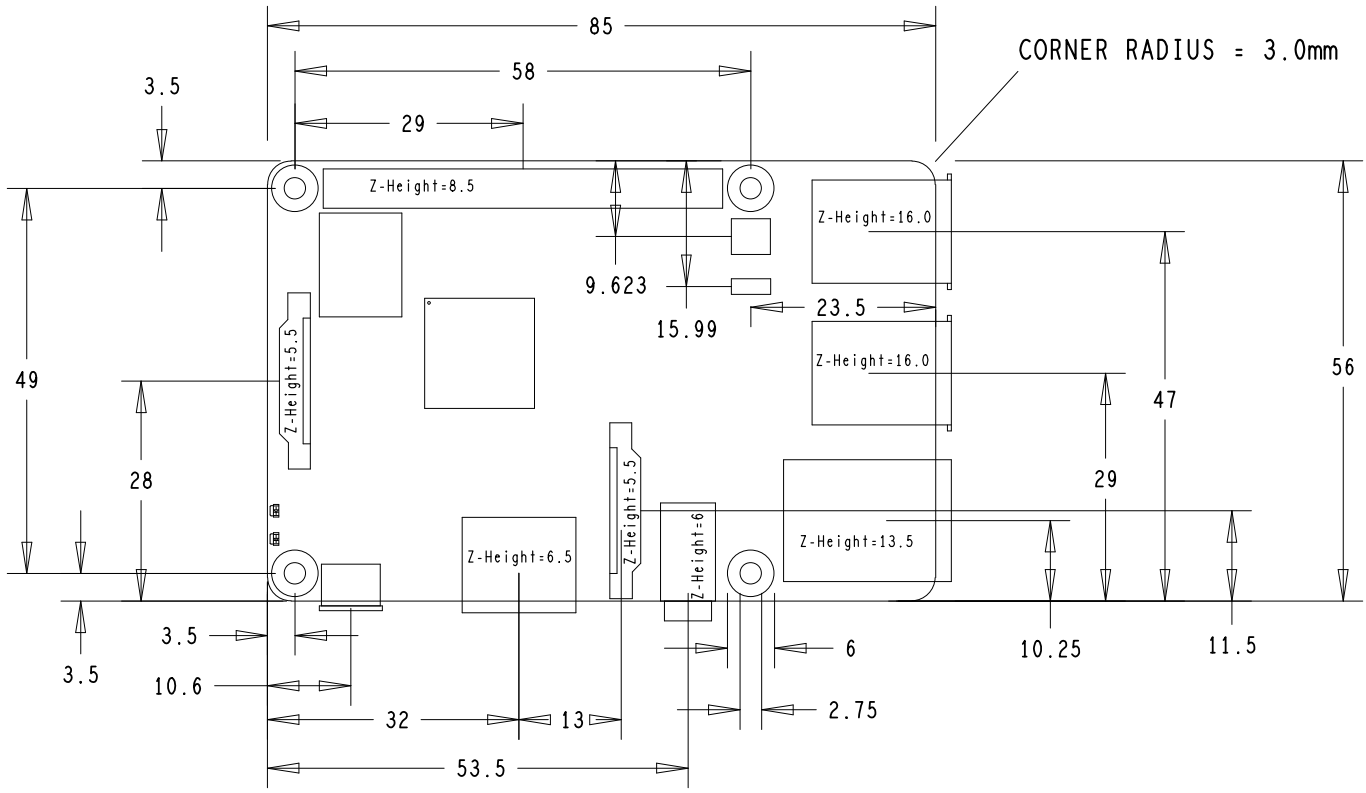
Environment:
Operating temperature, 0–50 °C

Compliance:
For a full list of local and regional product approvals, please visit www.raspberrypi.org/products/raspberry-pi-3-model-b+

Production lifetime:
The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.



Physical specifications



Warnings

- This product should only be connected to an external power supply rated at 5 V/2.5 A DC. Any external power supply used with the Raspberry Pi 3 Model B+ shall comply with relevant regulations and standards applicable in the country of intended use.
- This product should be operated in a well-ventilated environment and, if used inside a case, the case should not be covered.
- Whilst in use, this product should be placed on a stable, flat, non-conductive surface and should not be contacted by conductive items.
- The connection of incompatible devices to the GPIO connection may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met. These articles include but are not limited to keyboards, monitors, and mice when used in conjunction with the Raspberry Pi.
- The cables and connectors of all peripherals used with this product must have adequate insulation so that relevant safety requirements are met.

Safety instructions

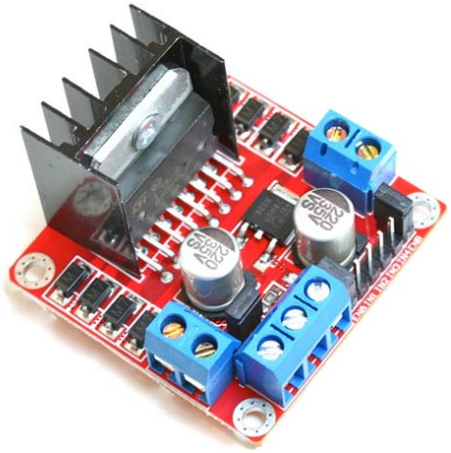
To avoid malfunction of or damage to this product, please observe the following:

- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; the Raspberry Pi 3 Model B+ is designed for reliable operation at normal ambient temperatures.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or only handle it by the edges to minimise the risk of electrostatic discharge damage.

User Guide

L298N Dual H-Bridge Motor Driver

This dual bidirectional motor driver, is based on the very popular L298 Dual H-Bridge Motor Driver Integrated Circuit. The circuit will allow you to easily and independently control two motors of up to 2A each in both directions.It is ideal for robotic applications and well suited for connection to a microcontroller requiring just a couple of control lines per motor. It can also be interfaced with simple manual switches, TTL logic gates, relays, etc. This board equipped with power LED indicators, on-board +5V regulator and protection diodes.

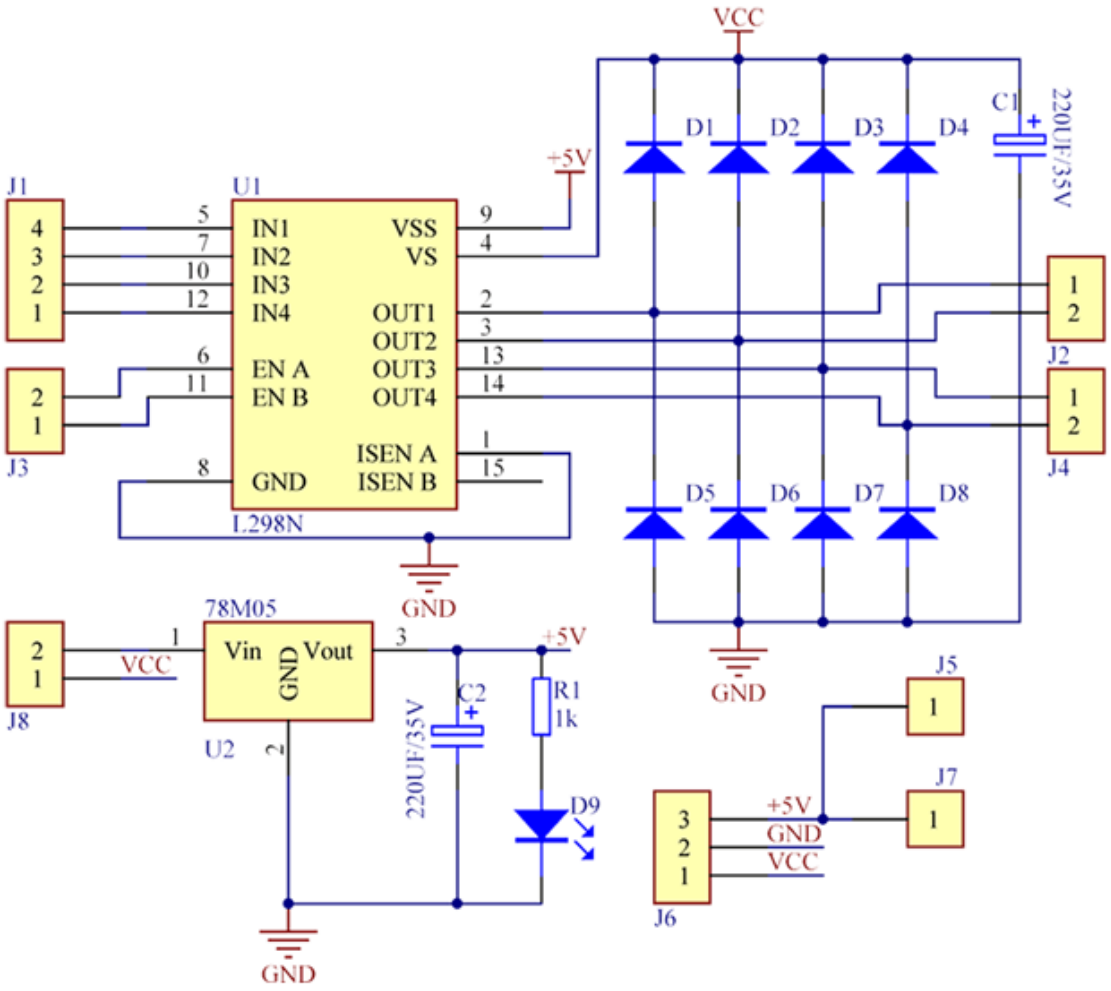


SKU: MDU-1049

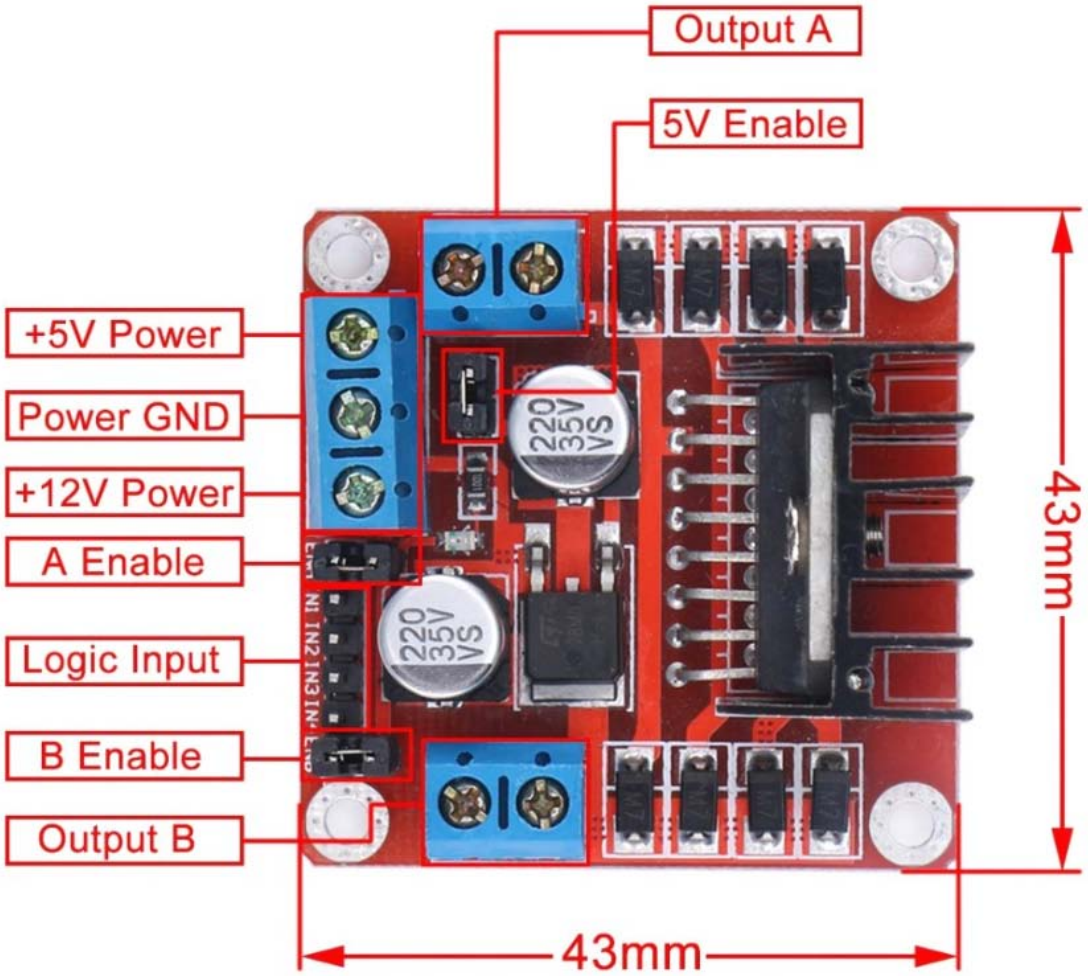
Brief Data:

- Input Voltage: 3.2V~40Vdc.
- Driver: L298N Dual H Bridge DC Motor Driver
- Power Supply: DC 5 V - 35 V
- Peak current: 2 Amp
- Operating current range: 0 ~ 36mA
- Control signal input voltage range :
- Low: -0.3V ⩽ Vin ⩽ 1.5V.
- High: 2.3V ⩽ Vin ⩽ Vss.
- Enable signal input voltage range :
    - Low: -0.3 ⩽ Vin ⩽ 1.5V (control signal is invalid).
    - High: 2.3V ⩽ Vin ⩽ Vss (control signal active).
- Maximum power consumption: 20W (when the temperature T = 75 ℃).
- Storage temperature: -25 ℃ ~ +130 ℃.
- On-board +5V regulated Output supply (supply to controller board i.e. Arduino).
- Size: 3.4cm x 4.3cm x 2.7cm

Schematic Diagram:



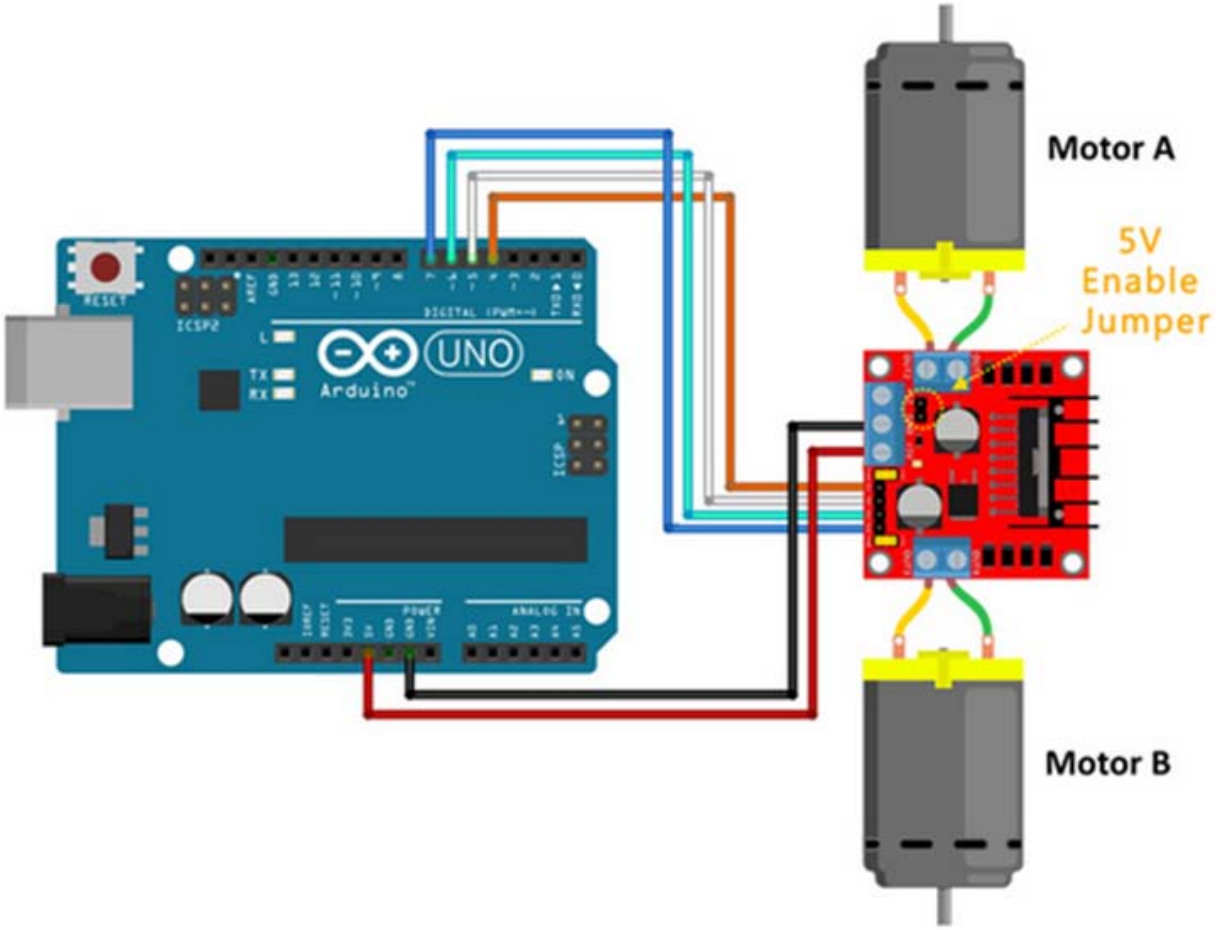
Board Dimension & Pins Function:



Connection Examples:

Controlling 2-DC Motor with +5V Arduino onboard Power Supply:

Below is the circuit connection use the on-board +5V power supply from Arduino board, and should be done without the 5V Enable Jumper on (Active 5V). This connection can drive two 5V DC motors simultaneously.



Sketch Listing:

Copy and paste the sketch below to Arduino IDE and upload to Arduino Uno/Mega board.

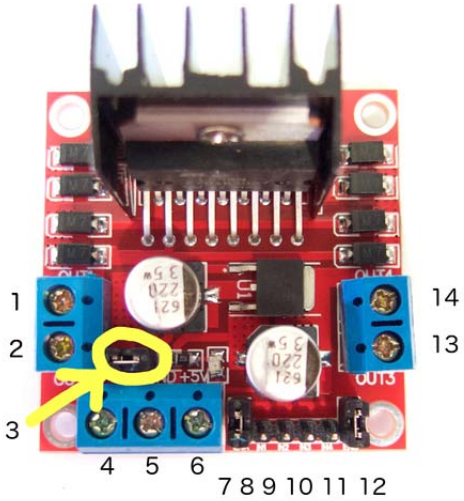
```
/*=====================================================================
//  Author      : Handson Technology
//  Project     : Arduino Uno
//  Description : L298N Motor Driver
//  Source-Code : L298N_Motor.ino
//  Program: Control 2 DC motors using L298N H Bridge Driver
//=====================================================================
*/

// Definitions Arduino pins connected to input H Bridge
int IN1 = 4;
int IN2 = 5;
int IN3 = 6;
int IN4 = 7;

void setup()
{
 // Set the output pins
```

```
 pinMode(IN1, OUTPUT);
 pinMode(IN2, OUTPUT);
 pinMode(IN3, OUTPUT);
 pinMode(IN4, OUTPUT);
}

void loop()
{
 // Rotate the Motor A clockwise
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, LOW);
 delay(2000);
 // Motor A
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, HIGH);
 delay(500);

 // Rotate the Motor B clockwise
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, LOW);
 delay(2000);
 // Motor B
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, HIGH);
 delay(500);

 // Rotates the Motor A counter-clockwise
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, HIGH);
 delay(2000);
 // Motor A
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, HIGH);
 delay(500);

 // Rotates the Motor B counter-clockwise
 digitalWrite(IN3, LOW);
 digitalWrite(IN4, HIGH);
 delay(2000);
 // Motor B
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, HIGH);
 delay(500);
}
```

Controlling Stepper Motor

In this example we have a typical NEMA-17 stepper motor with four wires:

The key to successful stepper motor control is identifying the wires - that is which one is which. You will need to determine the A+, A-, B+ and B- wires. With our example motor these are red, green, yellow and blue. Now let's get the wiring done.



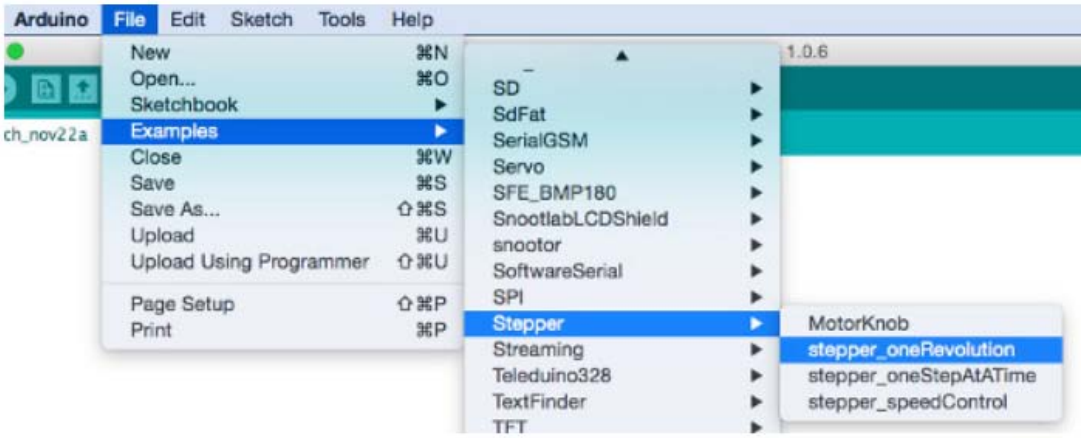
Connect the A+, A-, B+ and B- wires from the stepper motor to the module connections 1, 2, 13 and 14 respectively. Place the jumpers included with the L298N module over the pairs at module points 7 and 12. Then connect the power supply as required to points 4 (positive) and 5 (negative/GND).

Once again if your stepper motor's power supply is less than 12V, fit the jumper to the module at point 3 which gives you a neat 5V power supply for your Arduino.

Next, connect L298N module pins IN1, IN2, IN3 and IN4 to Arduino digital pins D8, D9, D10 and D11 respectively. Finally, connect Arduino GND to point 5 on the module, and Arduino 5V to point 6 if sourcing 5V from the module.

Controlling the stepper motor from your sketches is very simple, thanks to the *Stepper* Arduino library included with the Arduino IDE as standard.

To demonstrate your motor, simply load the *"stepper_oneRevolution"* sketch that is included with the *Stepper* library, for example:



Finally, check the value for

```
const int stepsPerRevolution = 200;
```

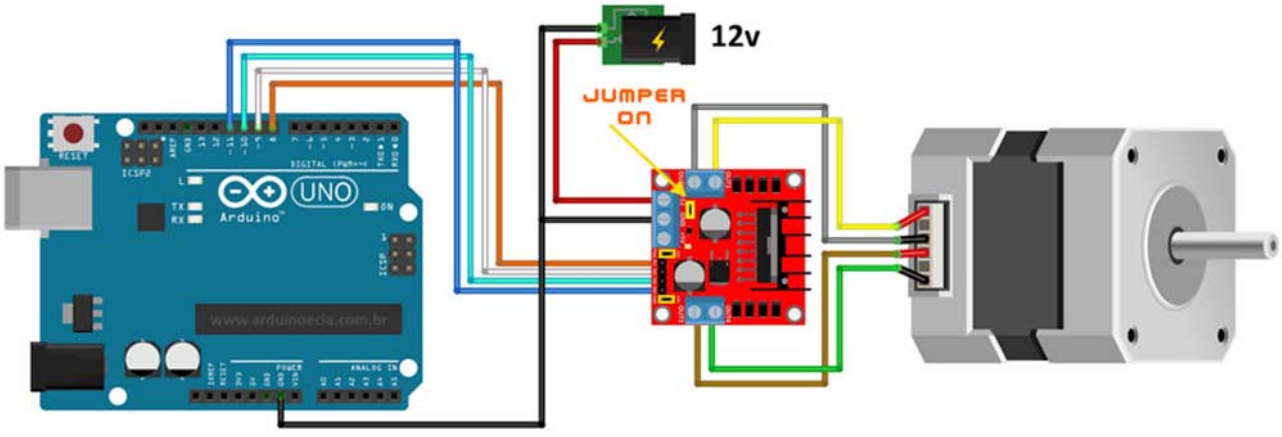
in the sketch and change the 200 to the number of steps per revolution for your stepper motor, and also the speed which is preset to 60 RPM in the following line:

```
myStepper.setSpeed(60);
```

Now you can save and upload the sketch, which will send your stepper motor around one revolution, then back again. This is achieved with the function

```
myStepper.step(stepsPerRevolution); // for clockwise
myStepper.step(-stepsPerRevolution); // for anti-clockwise
```

Connection for the sketch "*stepper_oneRevolution*":



Web Resources:

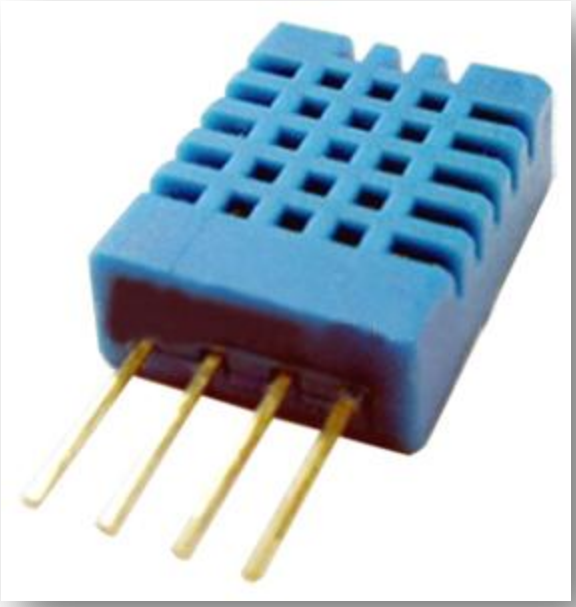
DHT11 Humidity & Temperature Sensor

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output.

DHT 11 Humidity & Temperature Sensor

1. Introduction

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high-performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness.



Each DHT11 element is strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as programmes in the OTP memory, which are used by the sensor's internal signal detecting process. The single-wire serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20 meter signal transmission making it the best choice for various applications, including those most demanding ones. The component is 4-pin single row pin package. It is convenient to connect and special packages can be provided according to users' request.

2. Technical Specifications:

Overview:

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Detailed Specifications:

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S

3. Typical Application (Figure 1)

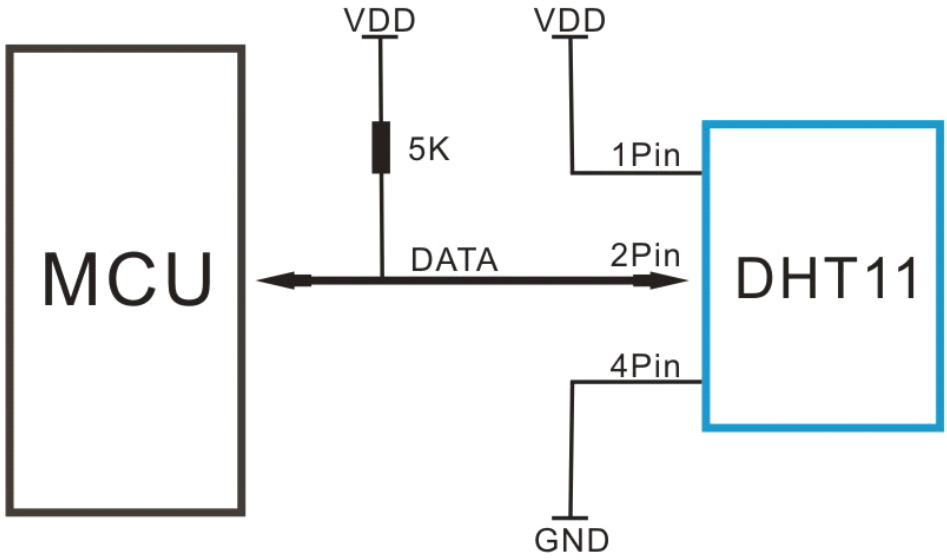


Figure 1 Typical Application

Note: 3Pin – Null; MCU = Micro-computer Unite or single chip Computer

When the connecting cable is shorter than 20 metres, a 5K pull-up resistor is recommended; when the connecting cable is longer than 20 metres, choose a appropriate pull-up resistor as needed.

4. Power and Pin

DHT11's power supply is 3-5.5V DC. When power is supplied to the sensor, do not send any instruction to the sensor in within one second in order to pass the unstable status. One capacitor valued 100nF can be added between VDD and GND for power filtering.

5. Communication Process: Serial Interface (Single-Wire Two-Way)

Single-bus data format is used for communication and synchronization between MCU and DHT11 sensor. One communication process is about 4ms.

Data consists of decimal and integral parts. A complete data transmission is **40bit**, and the sensor sends **higher data bit** first.

Data format: 8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum. If the data transmission is right, the check-sum should be the last 8bit of "8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data".

5.1 Overall Communication Process (Figure 2, below)

When MCU sends a start signal, DHT11 changes from the low-power-consumption mode to the running-mode, waiting for MCU completing the start signal. Once it is completed, DHT11 sends a response signal of 40-bit data that include the relative humidity and temperature information to MCU. Users can choose to collect (read) some data. Without the start signal from MCU, DHT11 will not give the response signal to MCU. Once data is collected, DHT11 will change to the low-power-consumption mode until it receives a start signal from MCU again.

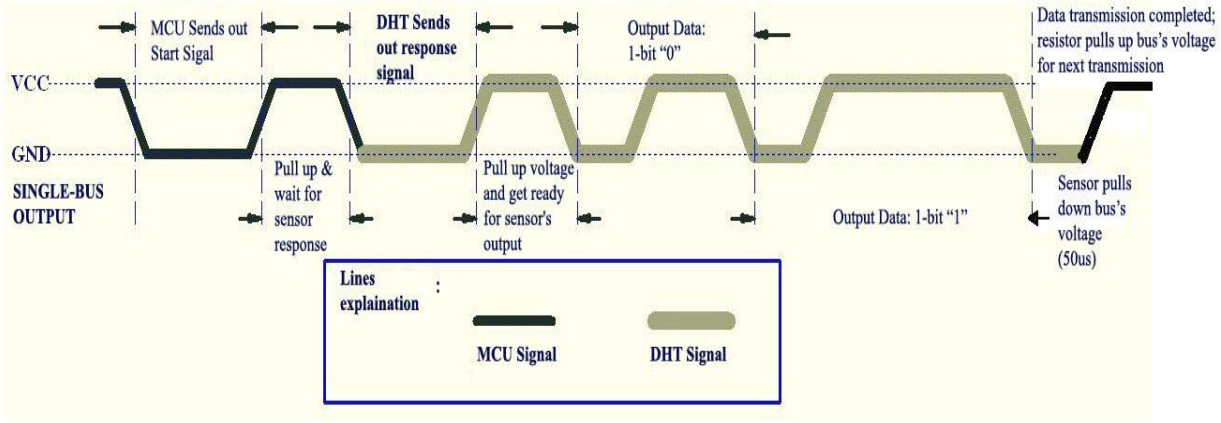


Figure 2 Overall Communication Process

5.2 MCU Sends out Start Signal to DHT (Figure 3, below)

Data Single-bus free status is at high voltage level. When the communication between MCU and DHT11 begins, the programme of MCU will set Data Single-bus voltage level from high to low and this process must take at least 18ms to ensure DHT's detection of MCU's signal, then MCU will pull up voltage and wait 20-40us for DHT's response.

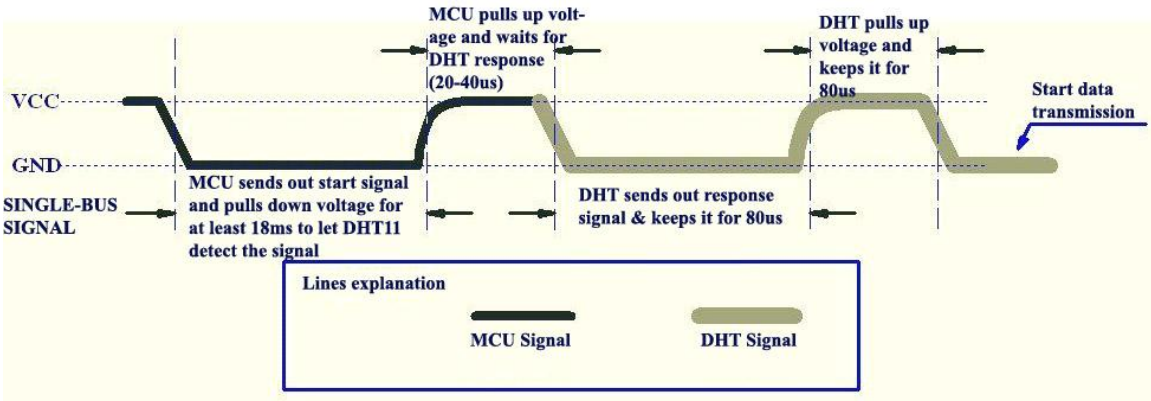


Figure 3 MCU Sends out Start Signal & DHT Responses

5.3 DHT Responses to MCU (Figure 3, above)

Once DHT detects the start signal, it will send out a low-voltage-level response signal, which lasts 80us. Then the programme of DHT sets Data Single-bus voltage level from low to high and keeps it for 80us for DHT's preparation for sending data.

When DATA Single-Bus is at the low voltage level, this means that DHT is sending the response signal. Once DHT sent out the response signal, it pulls up voltage and keeps it for 80us and prepares for data transmission.

When DHT is sending data to MCU, every bit of data begins with the 50us low-voltage-level and the length of the following high-voltage-level signal determines whether data bit is "0" or "1" (see Figures 4 and 5 below).

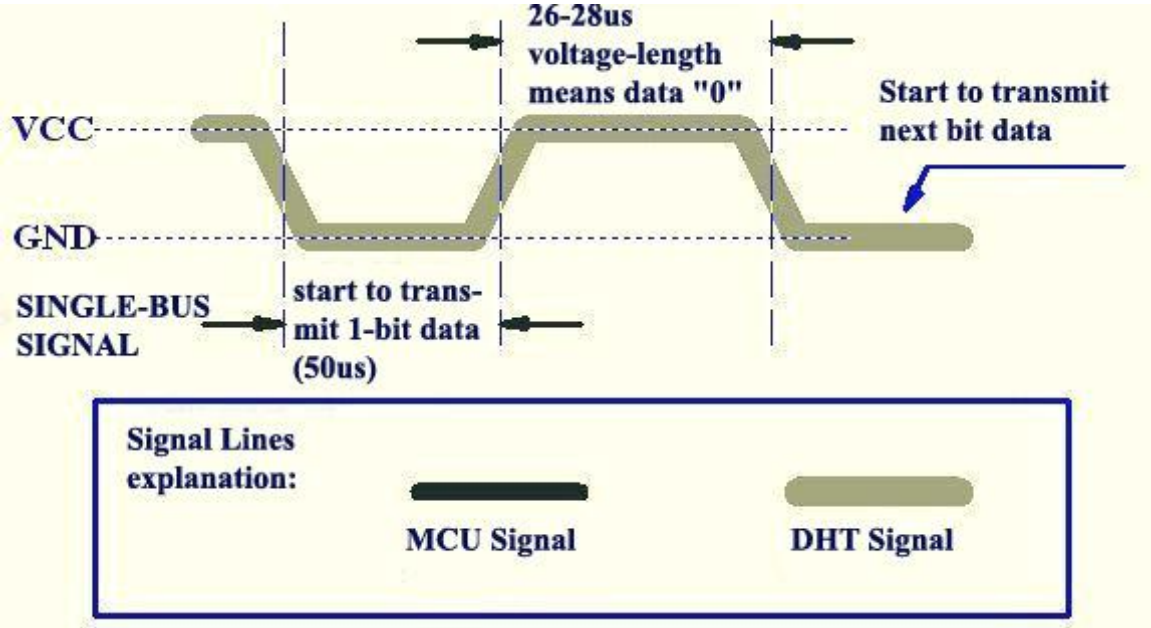


Figure 4 Data "0" Indication

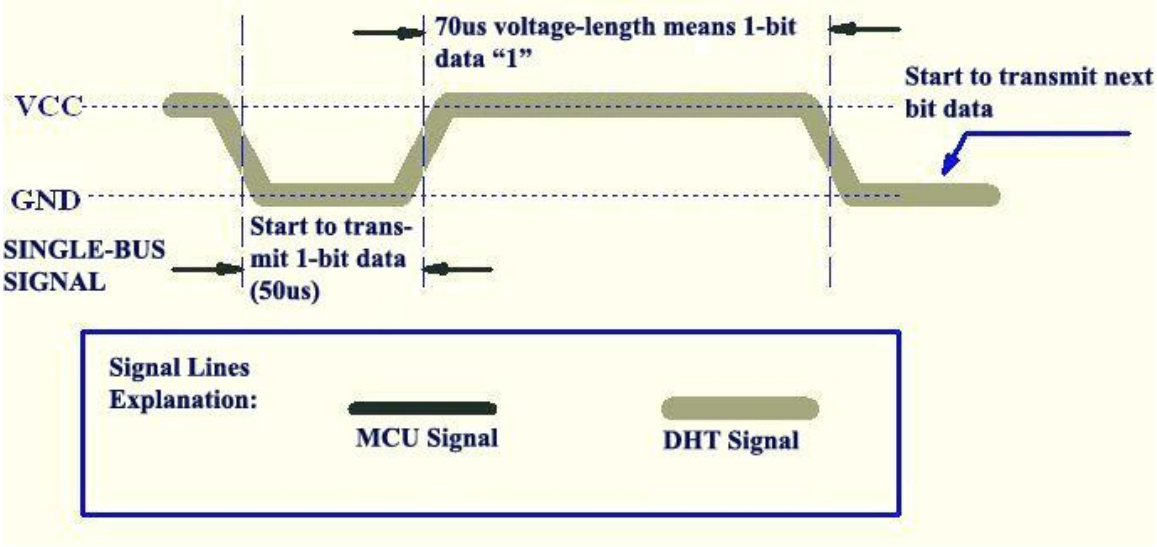


Figure 5 Data "1" Indication

If the response signal from DHT is always at high-voltage-level, it suggests that DHT is not responding properly and please check the connection. When the last bit data is transmitted, DHT11 pulls down the voltage level and keeps it for 50us. Then the Single-Bus voltage will be pulled up by the resistor to set it back to the free status.

6. Electrical Characteristics

VDD=5V, T = 25℃ (unless otherwise stated)

	Conditions	Minimum	Typical	Maximum
Power Supply	DC	3V	5V	5.5V
Current Supply	Measuring	0.5mA		2.5mA
	Average	0.2mA		1mA
	Standby	100uA		150uA
Sampling period	Second	1		

Note: Sampling period at intervals should be no less than 1 second.

7. Attentions of application

(1) Operating conditions

Applying the DHT11 sensor beyond its working range stated in this datasheet can result in 3%RH signal shift/discrepancy. The DHT11 sensor can recover to the calibrated status gradually when it gets back to the normal operating condition and works within its range. Please refer to (3) of

this section to accelerate its recovery. Please be aware that operating the DHT11 sensor in the non-normal working conditions will accelerate sensor's aging process.

(2) Attention to chemical materials

Vapor from chemical materials may interfere with DHT's sensitive-elements and debase its sensitivity. A high degree of chemical contamination can permanently damage the sensor.

(3) Restoration process when (1) & (2) happen

Step one: Keep the DHT sensor at the condition of Temperature 50~60Celsius, humidity <10%RH for 2 hours;
Step two:K keep the DHT sensor at the condition of Temperature 20~30Celsius, humidity >70%RH for 5 hours.

(4) Temperature Affect

Relative humidity largely depends on temperature. Although temperature compensation technology is used to ensure accurate measurement of RH, it is still strongly advised to keep the humidity and temperature sensors working under the same temperature. DHT11 should be mounted at the place as far as possible from parts that may generate heat.

(5) Ligh t Affect

Long time exposure to strong sunlight and ultraviolet may debase DHT's performance.

(6) Connection wires

The quality of connection wires will affect the quality and distance of communication and high quality shielding-wire is recommended.

(7) Other attentions

* Welding temperature should be bellow 260Celsius and contact should take less than 10 seconds.
* Avoid using the sensor under dew condition.
* Do not use this product in safety or emergency stop devices or any other occasion that failure of DHT11 may cause personal injury.
* Storage: Keep the sensor at temperature 10-40℃, humidity <60%RH.

Disclaimer

This is a translated version of the manufacturer's data sheet. OSEPP is not responsible for the accuracy of the translated information.

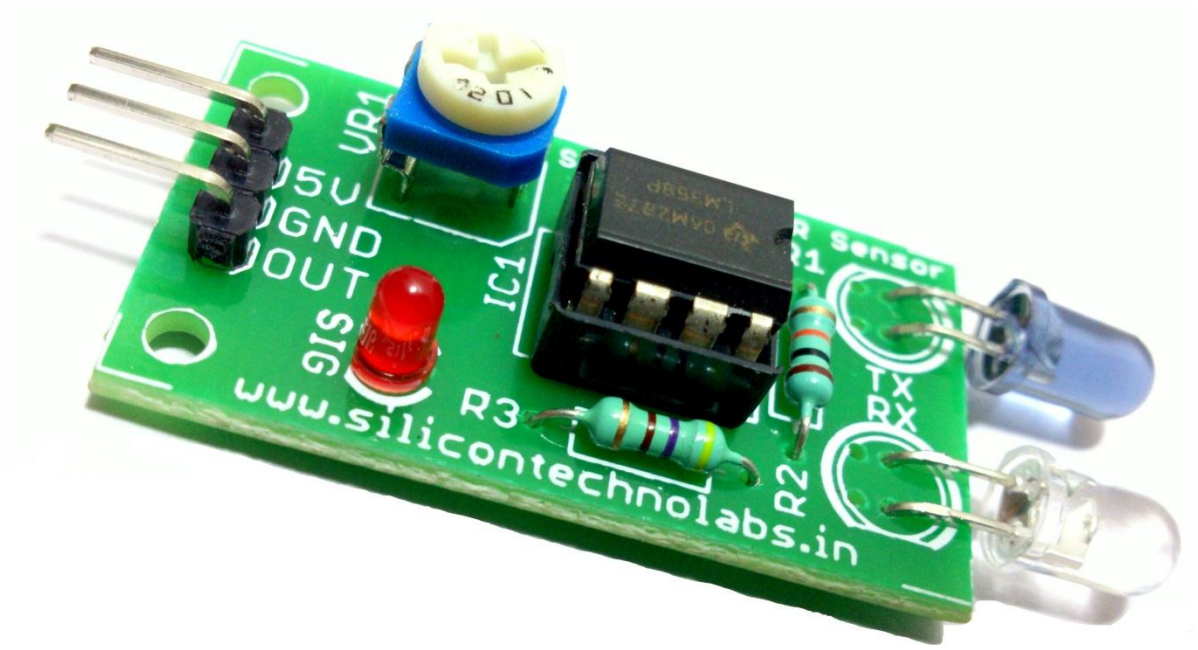
Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

OSEPP Electronics:

HUMI-01



IR Proximity Sensor

1. Descriptions

The Multipurpose Infrared Sensor is an add-on for your line follower robot and obstacle avoiding robot that gives your robot the ability to detect lines or nearby objects. The sensor works by detecting reflected light coming from its own infrared LED. By measuring the amount of reflected infrared light, it can detect light or dark (lines) or even objects directly in front of it. An onboard RED LED is used to indicate the presence of an object or detect line. Sensing range is adjustable with inbuilt variable resistor.

The sensor has a 3-pin header which connects to the microcontroller board or Arduino board via female to female or female to male jumper wires. A mounting hole for easily connect one or more sensor to the front or back of your robot chassis.

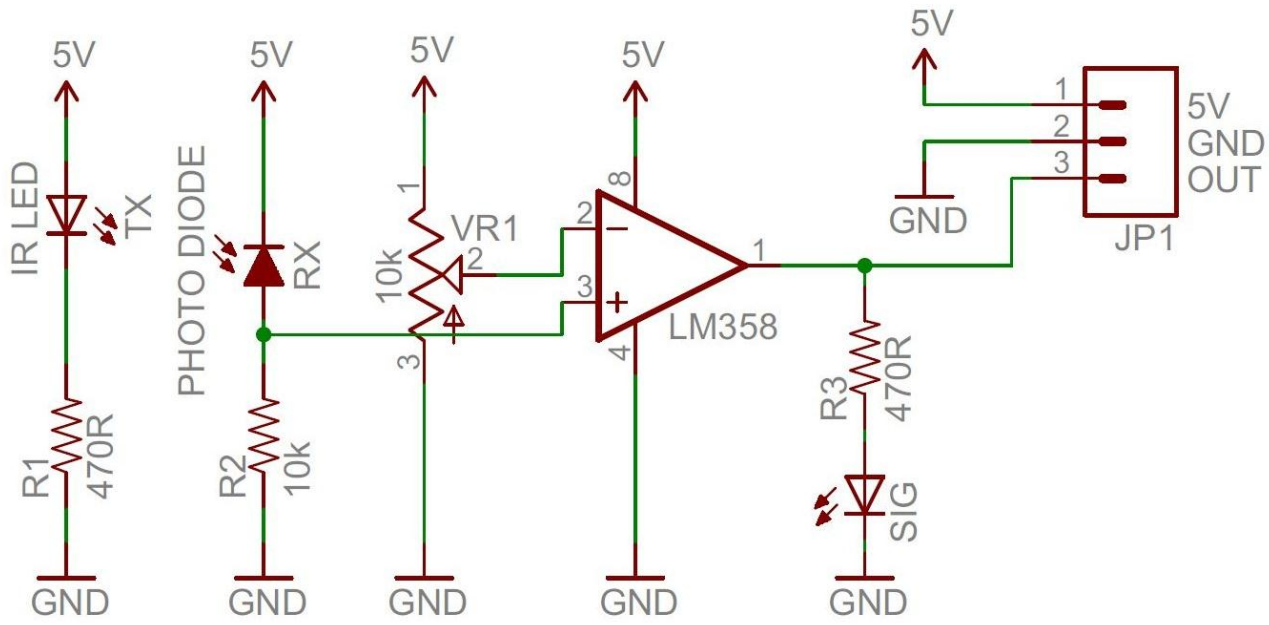
2. Features

- 5VDC operating voltage.
- I/O pins are 5V and 3.3V compliant.
- Range: Up to 20cm.
- Adjustable Sensing range.
- Built-in Ambient Light Sensor.
- 20mA supply current.
- Mounting hole.

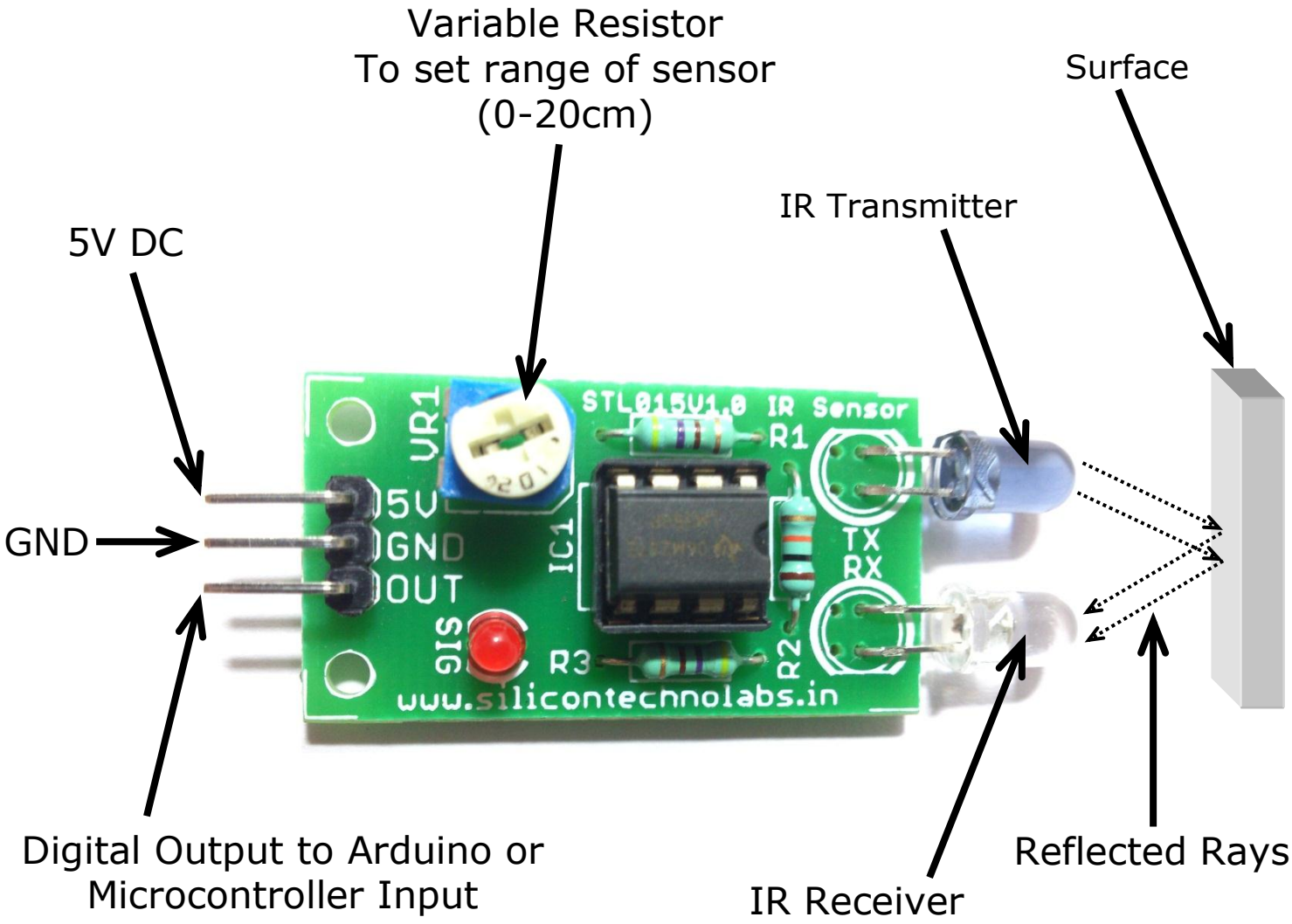
3. Specifications

- Size: 50 x 20 x 10 mm (L x B x H)
- Hole size: ϕ2.5mm

4. Schematics

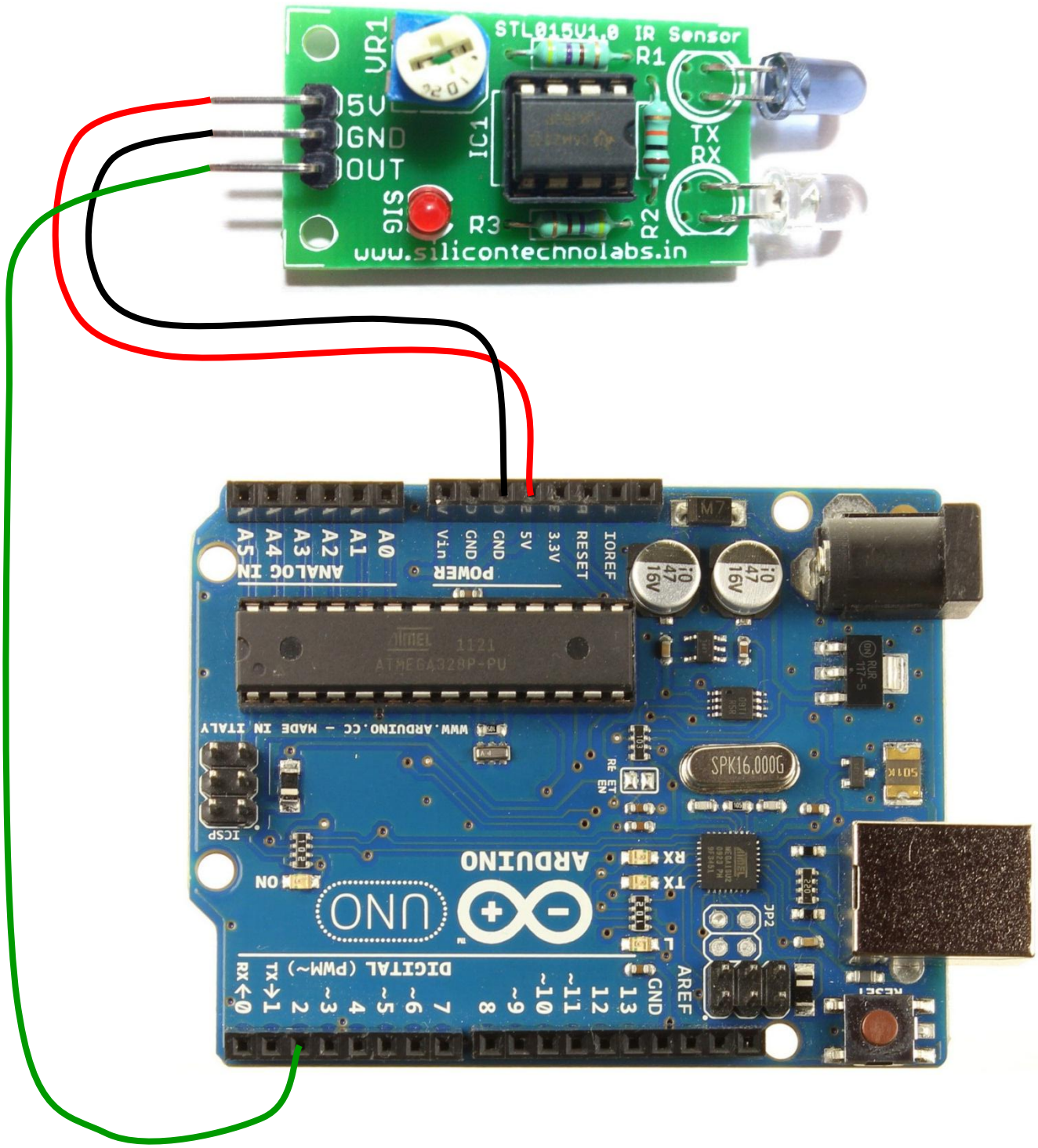


5.Hardware Details



6.Interface to Arduino

Now let's we build simple object counter using IR Proximity Sensor that's counts the Number of objects.Connect Silicon TechnoLabs IR Proximity Sensor to your arduino board as shown in below image.



7.Arduino Sample Code

```
/*
 Object counter
 Counts the number of objects and prints the results to the serial monitor.
 The circuit:
 * OUT attached to pin 2
 Created 2015
 by Harshit Borad <http://www.silicontechnolabs.in>
*/
// constants won't change. They're used here to
// set pin numbers:
const int OUT = 2; // the number of the IR Proximity Sensor pin
const int ledPin =  13;// the number of the LED pin
// variables will change:
int Number_of_Object = 0;// variable for reading the Number of Objects passing from sensor
int SensorState = 0;
void setup()
{
  Serial.begin(9600);  // initialize serial communications at 9600 bps:
  pinMode(ledPin, OUTPUT);  // initialize the LED pin as an output:
  pinMode(OUT, INPUT);   // initialize the IR Proximity Sensor pin as an input:
}
void loop()
{
  SensorState = digitalRead(OUT);// read the state of the Sensor Signal
  // check if the Sensor Signal is HIGH then there is object in front of sensor
  // so increment Number_of_Object variable by one.
  if (SensorState == HIGH)
  {
    digitalWrite(ledPin, HIGH);// turn LED on:
    Number_of_Object++;
    Serial.println(Number_of_Object);// print the results to the serial monitor:
  }
  else
  {
    digitalWrite(ledPin, LOW);// turn LED off:
  }
}
```

Thank you

"Happy Coding"