# OSMANIA UNIVERSITY

# ROBOTIC VACUUM CLEANER

**PROJECT THESIS**

***Submitted***

***IN THE PARTIAL FULFILLMENT OF***

*the award of the degree of*

**Bachelor of Engineering**

**In**

## ELECTRONICS AND COMMUNICATION ENGINEERING

**BY**

**J. Arun Sai (2451-17-735-143)**

**Under the Guidance of**

**G. VISHNUKANTH M.Tech**

**Assistant Professor, Department of ECE**

**MATURI VENKATA SUBBA RAO (MVSR) ENGINEERING COLLEGE**

(Sponsored by Matrusri Education Society, Estd 1981)
(Approved by AICTE & Affiliated to OU)
(Accredited by NBA & NAAC)

**May, 2022**

# CERTIFICATE

This is to certify that the Project Thesis titled **"ROBOTIC VACUUM CLEANER"** that is being submitted by **J. Arun Sai (2451-17-735-143)** in partial fulfillment for the award of Bachelor of Engineering (BE) in Electronics and Communication Engineering, *MATURI VENKATA SUBBA RAO (MVSR) ENGINEERING COLLEGE* affiliated to *OSMANIA UNIVERSITY*, *Hyderabad*, is a record of the bonafide work carried out by him/her under my guidance and supervision.

Signature of the Guide

G. Vishnukanth
Assistant Professor

Head of the Department

Dr. G. Kanaka Durga

# DECLARATION

I hereby declare that the contents presented in the Project Thesis titled **"ROBOTIC VACUUM CLEANER"** submitted in partial fulfillment for the award of Degree of Bachelor of Engineering in *ELECTRONICS AND COMMUNICATION ENGINEERING (ECE), MATURI VENKATA SUBBA RAO (MVSR) ENGINEERING COLLEGE* affiliated to *OSMANIA UNIVERSITY, Hyderabad* is a record of the original work carried out by me under the supervision of **G. Vishnukanth.** Further this is to state that the results embodied in this project report have not been submitted to any University or institution for the award of any Degree or Diploma.

Signature of the Student

J. Arun Sai (2451-18-735-143)

# ACKNOWLEDGEMENTS

I express my deep sense of gratitude to my guide **G. Vishnukanth, Assistant Professor, Department of ECE** for his/her constant guidance, untiring help and sparing their valuable inputs throughout my project work. The discussions I had with them enriched my understanding of the project and helped in achieving the goal

I also express my deep sense of gratitude to our Head of the Department, **Dr. G. Kanaka Durga**, Professor and Principal for her constant support, encouragement and providing necessary facilities to carry out the project.

I also express our sense of gratitude to all the faculty and staff of the Department for their direct and indirect help in making the project a success.

I express my deep sense of gratitude to my parents for their constant support, encouragement and blessings.

Signature of the Student

J. Arun Sai (2451-18-735-143)

# CHAPTER 1
# INTRODUCTION

## 1.1 INTRODUCTION TO VACUUM CLEANERS:

Cleanliness is very important in our life. So we need to make sure our house and surroundings are clean from rubbish and dust. In order to do that, people have used a vacuum cleaner because it makes cleaning job become easier.

Vacuum cleaners were developed in mid-1920 and then we came up with different designs and features, in 1996 the 1st robotic vacuum cleaner was developed using programming.

Now a days, everything that we use in our daily life is getting automated and vacuum cleaner is one of those things, if they get automated it can make our lives a little easier.

A robotic vacuum cleaner which cleans your home even when you are out of your home, reduces work and saves more time for the household. There are many companies which develop vacuum cleaners using different methods with very high cost.

## 1.2 OVERVIEW OF DOMAIN:

In recent times, smart homes have moved into the focus of scientific and technological research and development. Most everyday-life technical devices are controlled by microprocessors. Home automation integrates such devices into a network. The network allows the devices to coordinate their behavior in order to fulfil complex tasks without human intervention.

Robot vacuum cleaners are amazing pieces of technology. These cordless, battery-operated devices can complete their mission autonomously by moving along the floor, detecting obstacles and surface variations, and returning to the initial position. Not only must they provide powerful cleaning quietly and efficiently, they need to be equipped with various sensors and enough computational power to handle complex algorithms.

## 1.3 ORGANIZATION OF REPORT:

- **Chapter 1** explains the aim and the introduction part of the project.
- **Chapter 2** gives the glimpse of present systems which are being practiced, the drawbacks they face and the methods which have been proposed.
- **Chapter 3** explains the scope and aim of the project.
- **Chapter 4** explains the theoretical concepts required to design the project.
- **Chapter 5** gives the complete overview on the software and hardware tools that are part of the project.
- **Chapter 6** gives the glimpse of the codes responsible for scanning and cleaning.
- **Chapter 7** explains the experimentation results/simulation and analysis.
- **Chapter 8** discusses about the conclusion.
- **Chapter 9** discusses the contribution of project and future scope.

## 2.1 EXISTING METHODS:

1. **Robotic Vacuum Cleaner using Arduino with Wi-Fi**

   **Authors:** P.B. Jarande, S.P. Murakar, N.S. Vast, N.P. Ubale, S.S Saraf
   **Published:** IEEE - 2018
   **Description:**
   This paper discussed about a fully Working Robotic Vacuum Cleaner was developed using Arduino UNO.

2. **Path Planning algorithm development for autonomous vacuum cleaner robots**
   **Authors:** Kazi Mahmud Hasan, Adullah-Al-Nahid, Khondker Jahid Reza
   **Published:** IEEE - 2014
   **Description:**
   An Algorithm has been developed for Path Planning and Navigation that prevents the Robot from bumping into Walls and Obstacles by reversing or changing path accordingly.

3. **Development of a vacuum cleaner robot**
   **Authors:** T.B. Asafa, T.M. Afonja, E.A. Olaniyan, H.O. Alade
   **Published:** AEJ - 2018
   **Description:**
   The robotic vacuum cleaner is fabricated from computer scraps, ultrasonic sensors and an Arduino mega2560 microcontroller.

4. **Human avoidance function for robotic vacuum cleaner through use of environmental sensors: Roomba making way for humans**
   **Authors:** Hiroshi Hisahara, Yuki Ishii, Masahito Ota, Takeki Ogitsu
   **Published:** IEEE - 2014

**Description:**

The robotic vacuum cleaner can automatically clean a room, and it can communicate with a computer through an ROS communication interface

## 2.2 COMPARISION TABLE:

| **Project Paper** | **Merits** | **Demerits** |
|---|---|---|
| Robotic Vacuum Cleaner using Arduino with Wi-Fi | Easy access and wider range due to Wi-Fi Module. | Excess power consumption due to no mapping. |
| Path Planning algorithm development for autonomous vacuum cleaner robots | Sensors are used for accurate path planning. | Mapping technology, automatic charging algorithm and virtual walls are absent. |
| Development of a vacuum cleaner robot | Ultrasonic sensors are used for accurate obstacle avoidance. | Mapping Technologies are absent. |
| Human avoidance function for robotic vacuum cleaner through use of environmental sensors: Roomba making way for humans | Detects positions of human and updates the location for proper mapping. | Not feasible as cameras are required to be placed everywhere in the area. |

**Table – 2.2**

## 2.3 MOTIVATION:

Robotic vacuum cleaners have attained a fair degree of success in the domestic robot market. The iRobot Corporation (one of the main players in this market) claims to have sold 6 million units of its ‘‘Roomba” robot between 2002 (its first release) and 2010. According to the statistics of the International Federation of Robotics, about 2.5 million personal and service robots were sold in 2011, an increase of 15% in numbers (19% in value) compared to 2010. The forecast for the period 2012–2015 exceeds 10 million units. This trend clearly emphasizes the growing impact of domestic robots in our homes, which creates new interaction paradigms. In parallel, the energy demand for the operation of millions of new cleaning robots will follow the same tendency.

Moreover, with the evolution of technologies, domestic robots shifted from the simple ‘‘random-walk’’ approach towards more evolved navigation schemes, involving a localization technology at an affordable price. Up to now, no scientific study has analyzed the potential impact of these newer robots in terms of user acceptance or energy consumption.

## 2.4 CONVENTIONAL METHOD:

- The Robotic vacuum cleaner consists of 7 ultrasonic sensors for scanning border of the location that needs to be cleaned
- It contains a compass sensor fixed facing the north direction.
- Line tracking sensor fixed beside a wheel, along with relay module all the sensors are connected to raspberry pi.

### 2.4.1 DIMENSIONS:

- Length and width is 30x30 cm
- Wheel radius is 1.5 cm
- Weight 1.7 - 2.0 kg

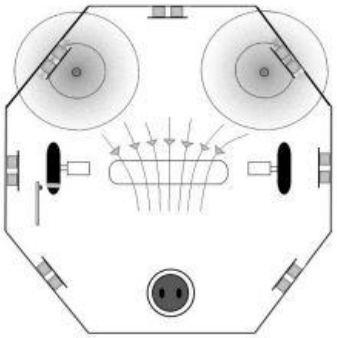

**Fig – 2.1**

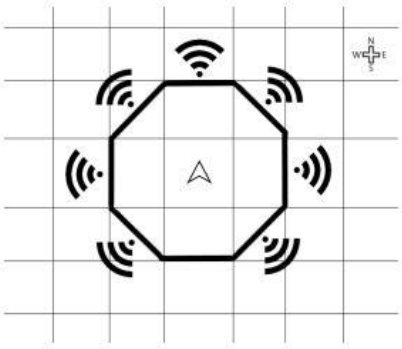

**Fig – 2.2**

## 2.4.2 ADVANTAGES:

- Ease of Use
- Hands-Free Operation
- Saves Time and Energy
- Improved Hygiene
- Low Maintenance
- Can handle tight space

## 3.1 SCOPE OF THE PROJECT:

The robot vacuum cleaners market is expected to register a CAGR of 13% during the forecast period, 2021 - 2026. The International Federation of Robotics (IFR) estimates that 31 million household robots to be sold between 2016 and 2019, and out of which 96% of which will be a vacuum and floor cleaning robots. This is encouraging companies to invest in the market.

- One of the major factors driving the deployment of the robot vacuum cleaner is that the highest number of occupational injuries being recorded in the janitorial industry. Statistics from the Bureau of Labour Statistics indicate that there are 2,384,600 building janitors and cleaners.
- Many companies are spending about USD 60 billion on average annually. The factors like these are boosting the demand for robotic vacuum cleaners.
- Another factor driving the market is the rapid advancement in the industry. For instance, in December 2018, Neato Robotics launched its on-demand zone cleaning robot vacuum, Botvac D7 connected which uses LIDAR technology to scan and map the efficient course of cleaning the designated room.


**Fig – 3.1**

- Such technological advancements have a positive outlook on the market.
- However, the high cost of installation, coupled with the high cost of maintenance of the robot vacuum cleaners, is hindering the market growth.

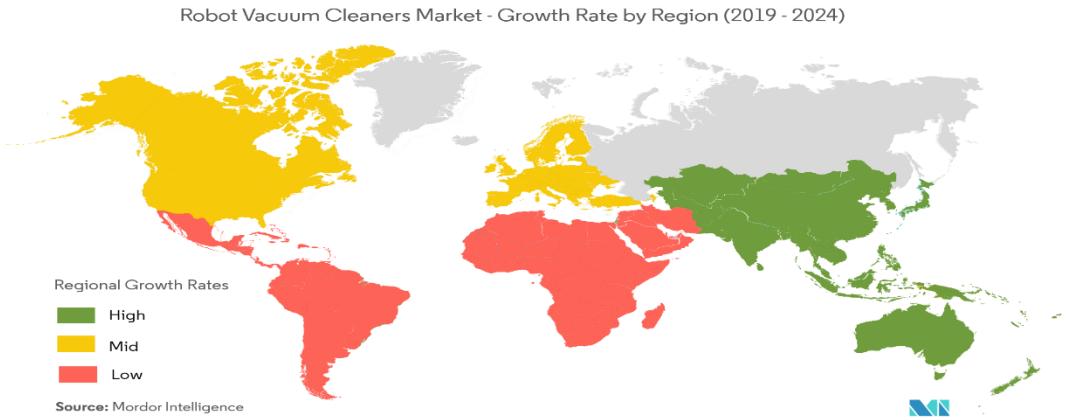


**Fig – 3.2**

## 3.2 OBJECTIVE OF THE PROJECT:

The objective of this project is to implement an octagon shaped robotic vacuum cleaner equipped with efficient scanning and vacuuming technologies which will automatically scan the area and vacuums in a time scheduled by the user.

## 4.1 DISTANCE ESTIMATION:

At its core, the HC-SR04 Ultrasonic distance sensor consists of two ultrasonic transducers. The one acts as a transmitter which converts electrical signal into 40 KHz ultrasonic sound pulses. The receiver listens for the transmitted pulses. If it receives them, it produces an output pulse whose width can be used to determine the distance the pulse travelled.

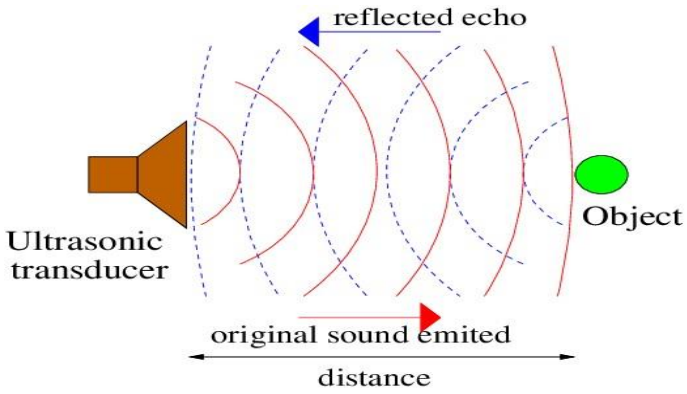


**Fig - 4.1**

## 4.2 ALGORITHM:

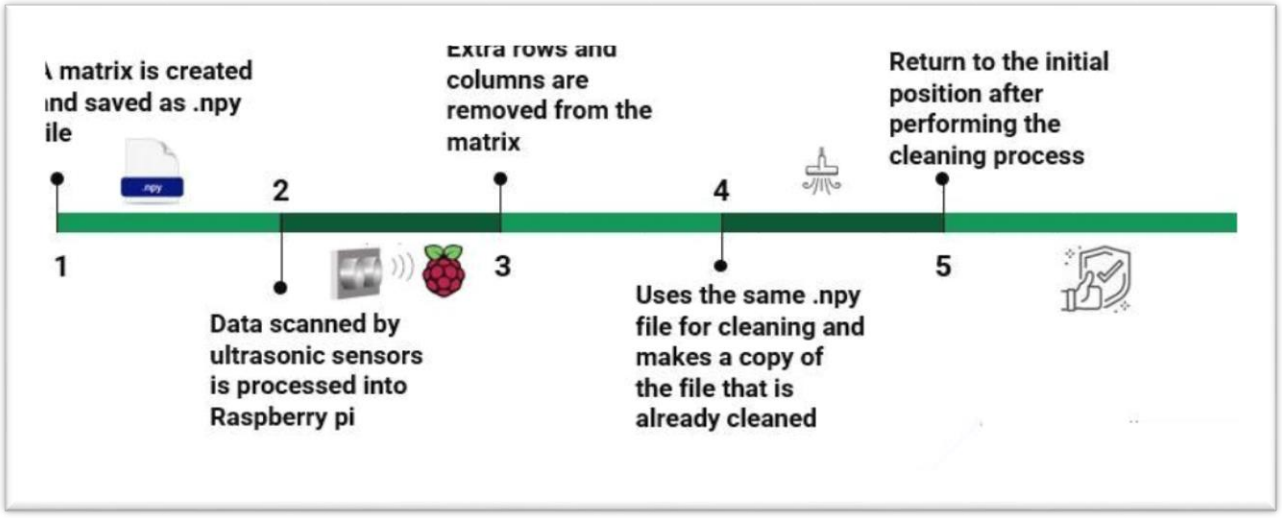


**Fig - 4.2**

## 4.3 STEPS INVOLVED:

### 4.3.1 SCANNING:

1. A matrix is created and saved as .npy extension file before the beginning of scanning process. The main purpose of this is to record the area of the location that needs to be cleaned.
2. It begins scanning using the ultrasonic sensors then, it processes the data in raspberry pi and updates the matrix file. The algorithm checks if any area which is not scanned if it finds any, then it moves towards the location and if not the scanning process stops.
3. Now all the extra rows and columns in the matrix which are not containing the location data are removed from the matrix file.

### 4.3.2 CLEANING:

4. Cleaning algorithm makes use of the same .npy extension file saved by the scanning algorithm. When the vacuum equipment is turned ON it starts moving according to the scanned co-ordinates from the scanned .npy file. After cleaning a grid, it makes the copy of the file to note that place is already cleaned.
5. After cleaning the entire area, the device is returns to stationary position.

.

## 4.4 FLOWCHART:

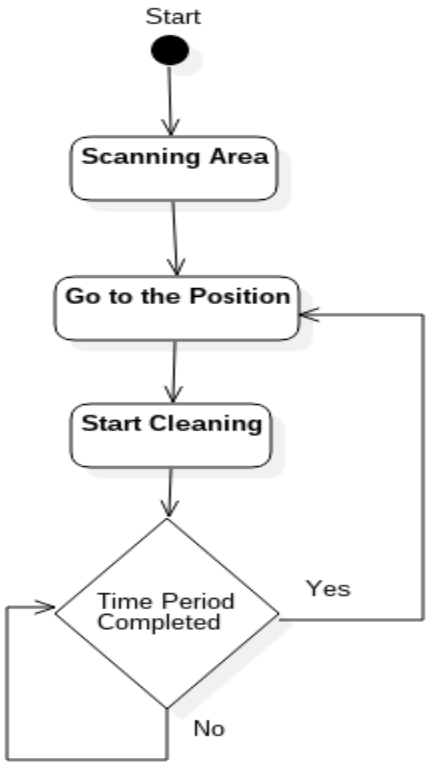


**Fig – 4.3**

## 4.5 INTERFACING DIAGRAM:

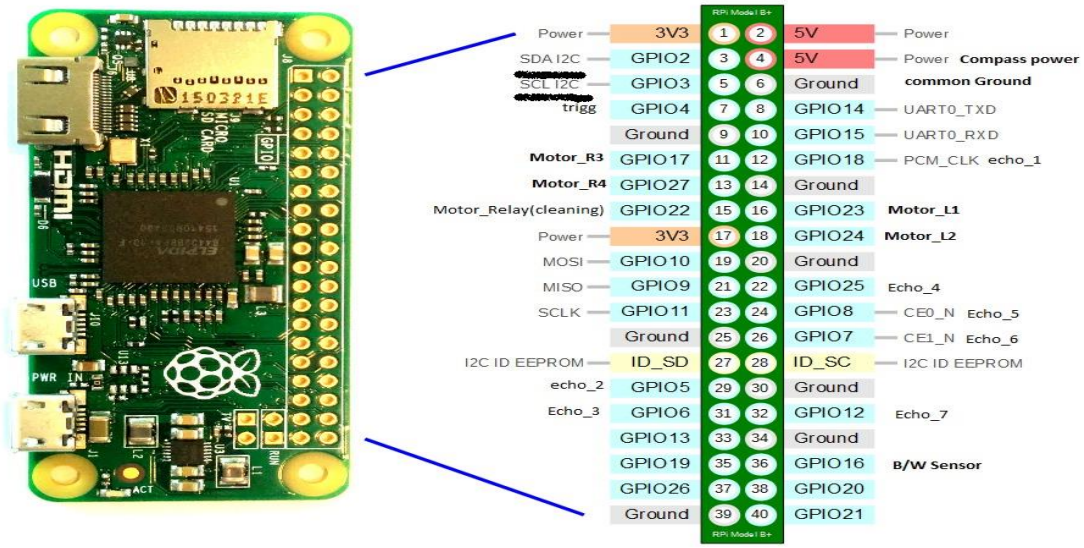


**Fig – 4.4**

## 5.1 HARDWARE TOOLS:

### 5.1.1 RASPBERRY PI ZERO W:

The Raspberry Pi is a popular Single Board Computer (SBC) in that it is a full computer packed into a single board. Many may already be familiar with the Raspberry Pi 3 and its predecessors, which comes in a form factor that has become as highly recognizable. The Raspberry Pi comes in an even smaller form factor. The introduction of the Raspberry Pi Zero allowed one to embed an entire computer in even smaller projects. The Raspberry Pi Zero - Wireless, which has an onboard Wi-Fi module.

## SPECIFICATIONS:

- 1GHz BCM 2835 single-core processor
- 512 MB RAM
- Bluetooth 4.1 and Bluetooth Low Energy (BLE)
- Flexible and compact with mini connectors and
- Consists 40-pin HAT compatible GPIO
- Mini HDMI ports and USB on-the-go ports.

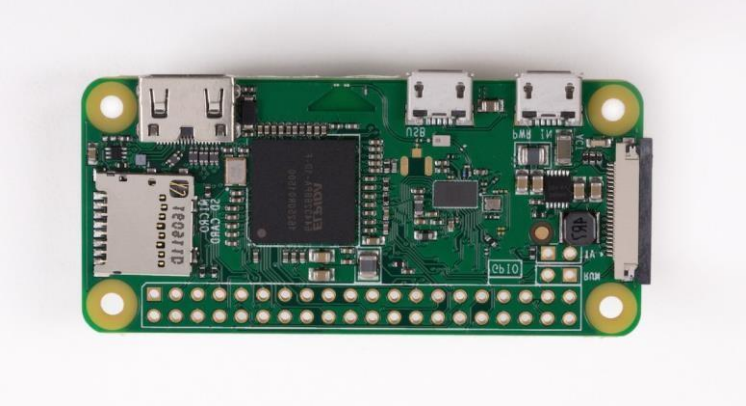

**Fig – 5.1**

## 5.1.2 ULTRASONIC SENSOR (HC-SR04):

The HCSR04 ultrasonic sensor uses sonar to determine distance to an object like bats or dolphins do. It offers excellent noncontact range detection with high accuracy and stable readings in an easy to use package. From 2cm to 400 cm or 1" to 13 feet. It operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with ultrasonic transmitter and receiver module.

## SPECIFICATIONS:

- Power Supply: +5V DC
- Quiescent Current : <2mA
- Working Current: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1" 13ft
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm


**Fig – 5.2**

## 5.1.3 MICRO GEAR MOTOR (N20-460 RPM):

The micro gear motors are incredibly tough and feature full metal gears. They have a gear ratio of 100:1 and operate up to 12 volts and have a stall torque of 25 oz-in. and a max speed of 270 RPM. Each micro gear motor sports a 3mm D-shaft.

The N20-6V-400 Rpm Micro Metal Gear Motor has small volume, torsion big, all metal gear, durable, not easy to wear. Great replacement for the rusty or damaged DC geared speed reduce motor on the machine.

The N20 Micro Gear 6V 400 RPM DC Motor (High Torque) is lightweight, high torque, and low RPM motor. It is equipped with gearbox assembly so as to increase the torque of the motor. It has a cross-section of 10 × 12 mm, and the D-shaped gearbox output shaft is 9 mm long and 3 mm in diameter. It has a very small size so as fit in complex spaces of small-scale application. One can connect this Micro Gear Motor to wheels to drive them from one place to other while carrying high loads.

**Fig - 5.3**

## 5.1.4 COMPASS SENSOR (HMC 5883L):

3-Axis Compass module, a member of grove family uses I²C based Honeywell HMC5883L digital compass. This ASIC is equipped with high resolution HMC118X magneto-resistive sensors and a 12-bit ADC. It provides compass heading accuracy up to

1° to 2°. Signal conditioning like amplification, automatic degaussing strap drivers and offset cancellation are inbuilt. This grove module also includes a MIC5205-3.3 for power supply requirement. Hence user can connect any 3.3V to 6V DC power supply.

## SPECIFICATIONS:

- Power Requirements: 2.7 to 6.5 VDC
- Communication Interface: I²C (up to 400 kHz)
- Operating temperature: (-30 to +85 °C)
- Dimensions: 0.73 x .65 in ( 1.8 x 1.7 cm)


**Fig – 5.4**

## 5.1.5 RELAY MODULE:

Relay is an electromechanical device that uses an electric current to open or close the contacts of a switch. The single-channel relay module is much more than just a plain relay, it comprises of components that make switching and connection easier and act as indicators to show if the module is powered and if the relay is active or not.

## SPECIFICATIONS:

- Supply voltage – 3.75V to 6V
- Quiescent current: 2mA
- Current when the relay is active: ~70mA
- Relay maximum contact voltage – 250VAC or 30VDC
- Relay maximum current – 10A


**Fig – 5.5**

### 5.1.6 LINE TRACKING SENSOR (TCRT 5000):

IR Line Tracking Module consists of a IR reflex sensor, it is a sensor with optoelectronic transmitter and receiver in a package. It functions by illuminating a surface with infrared light; the sensor then picks up the reflected infrared radiation and, based on its intensity, determines the reflectivity of the surface in question. Lightly colored surfaces will reflect more light than dark surfaces; therefore, lightly colored surfaces will appear brighter to the sensor. This allows the sensor to detect a dark line on a pale surface, or a pale line on a dark surface. This module enables a robot to autonomously navigate a line-marked path

### SPECIFICATIONS:

- Operating Power Supply DC 3.3V ~ 5V
- Detection Range 0 ~ 3cm
- Output (i) TTL logic ('H' on Black or no obstacle; 'L' on White or obstacle)
  (ii) Analog voltage output (0 ~ 5V)
- Interface 4 wires
- PCB Dimension 35mm x 15mm

**Fig - 5.6**

### 5.1.7 MOTOR DRIVER (LP298 2A Dual):

The **L298 Motor Driver Module** is a high power motor driver module for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M05 5V regulator. **L298 Module** can control up to 4 DC motors, or 2 DC motors with directional and speed control.

## SPECIFICATIONS:

- Driver Model: L298N 2A
- Driver Chip: Double H Bridge L298N
- Motor Supply Voltage (Maximum): 46V
- Motor Supply Current (Maximum): 2A
- Logic Voltage: 5V
- Driver Voltage: 5-35V
- Driver Current: 2A
- Logical Current: 0-36mA
- Maximum Power (W): 25W
- Current Sense for each motor
- Heatsink for better performance
- Power-On LED indicator

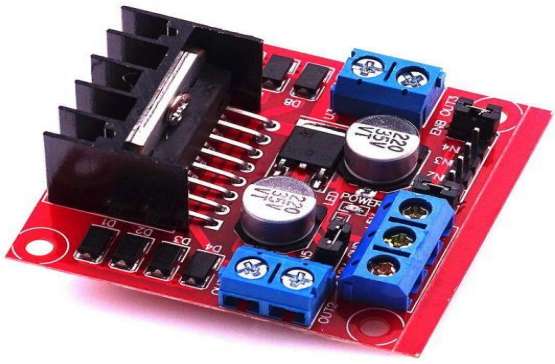

**Fig - 5.7**

### 5.1.8 DC MOTOR:

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.

**Fig - 5.8**

### 5.1.9 LITHIUM-ION BATTERY:

These round high capacity cells have been mainly used in flashlight type applications but with its capability to be used as a drop-in rechargeable cell at 3.7V with a capacity of 2600mAh. This is a great battery option for those of you who need a simple to install and replace cell with a lot of juice. These 18650 Cells have a standard discharge current of 0.2C to a maximum of 1C and can handle about 300 charge cycles.


**Fig - 5.9**

## 5.2 SOFTWARE TOOLS:

### 5.2.1 RASPBERRY PI OS:

Raspberry Pi OS is a free operating system based on Debian, optimized for the Raspberry Pi hardware, and is the recommended operating system for normal use on a Raspberry Pi. The OS comes with over 35,000 packages: precompiled software bundled in a nice format for easy installation on your Raspberry Pi.

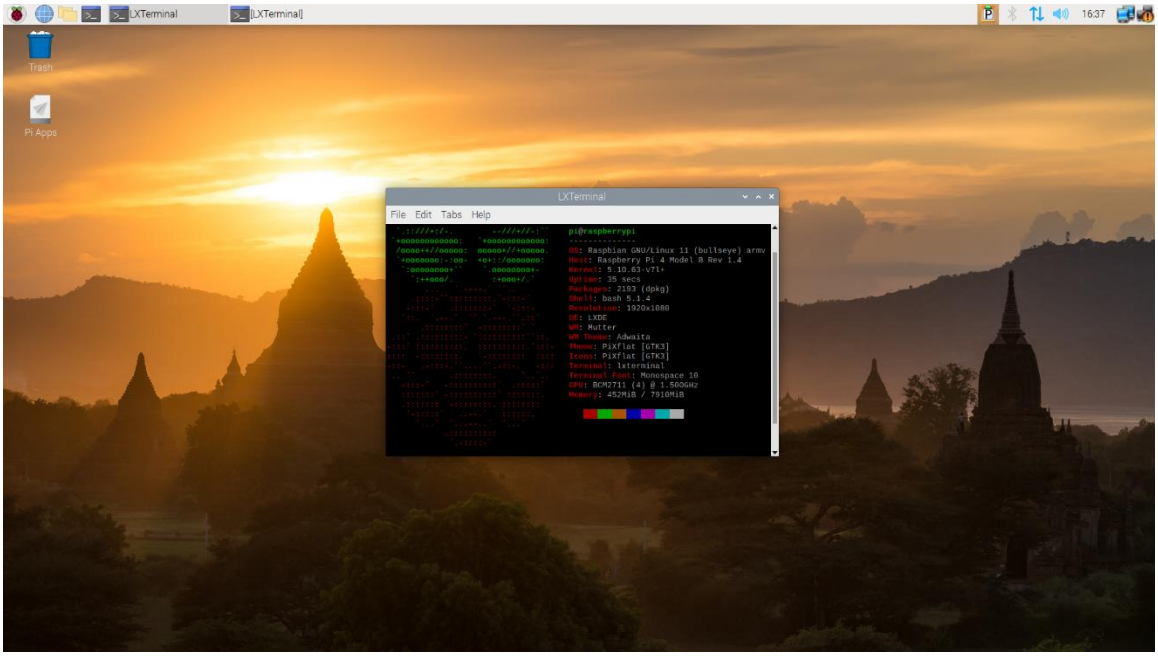


**Fig - 5.10**

### 5.2.2 PYTHON PROGRAMMING LANGUAGE:

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects

**Fig – 5.11**

## 6.1 CODE FOR SCANNING:

```
import ultrasonic as us
import com_dc as cd
import black_white as bw
import numpy as np
import RPi.GPIO as GPIO
from time import sleep
import methods as meth
import compass as cmp
import angles as a
N=a.N
S=a.S
E=a.E
W=a.W

n=a.n
s=a.s
e=a.e
w=a.w


#print(W)
#ang=0
size=25
p=[5,int(size/2)]
rem_r=0
try:
   a=np.load('area50.npy')
except:
   a=np.zeros((size,size),dtype=int)
   np.save('area50',a)

def start():
   ang=bw.moveD(n)
   cd.position(n)
   #ang=s
   while(1):
      sleep(0.1)
      cd.position(ang)
      print('angle:')
      print(ang)
      print(p[0])
      print(p[1])
```

```
    for i in a:
        print(i)

    l1=us.sense()
    l2=us.sense()
    while(sum(l2) > 2000):
        l2=us.sense()
    print(str(l1[4]) +' -  '+str(l2[4]))
    print(l2)
    if(p[0]==0 and (ang in N)):
        print('turning towards west')
        meth.turnL()
        p[0]=p[0]+1
        p[1]=p[1]-1
        ang=readW()
    # bw.move1()



    elif((ang in S)  and p[1]>4 and (((l1[4]+l2[4])/2) > 30) and l2[2]>25 and (a[p[0]-
1,p[1]-3]==0)and (a[p[0]-1,p[1]-2]!=2) and(a[p[0]-2,p[1]-4]!=0 or (a[p[0]-1,p[1]-4]!=0)
or (a[p[0]-2,p[1]-2]!=0))and((a[p[0]-2,p[1]-3] !=0) or (a[p[0]-2,p[1]-2]!=0))):
        print('Turning towards Right ie. West')
        bw.move1()
        meth.turnR()
        p[1]=p[1]-1
        ang=readW()

    elif( (ang in E) and p[0]<(size-4) and ((l1[4]+l2[4])/2)>30  and l2[2]>25 and
a[p[0]+2,p[1]-1]!=2 and a[p[0]+3,p[1]-1]==0 and (a[p[0]+4,p[1]-1]!=0 or a[p[0]+4,p[1]-
2]!=0 or a[p[0]+2,p[1]-2]!=0 ) and (a[p[0]+2,p[1]-2]!=0 or a[p[0]+3,p[1]-2]!=0) ):
        print(' Turning towards Right ie. south')
        bw.move1()
        meth.turnR()
        p[0]=p[0]+1
        ang=readS()

    elif((ang in W)   and  ((l1[4]+l2[4])/2) >30  and a[p[0]-3,p[1]+1]==0 and l2[2]>25
and a[p[0]-2,p[1]+1]!=2 and (a[p[0]-4,p[1]+1]!=0 or a[p[0]-4,p[1]+2]!=0 or a[p[0]-
2,p[1]+2]!=0 ) and (a[p[0]-2,p[1]+2]!=0 or a[p[0]-3,p[1]+2]!=0)):
        print('Turning towards Right ie. North')
        bw.move1()
        cd.position(n)
        p[0]=p[0]-1
```

```
        ang=readN()

    elif((ang in N) and p[1]<(size-4) and ((l1[4]+l2[4])/2)>30 and l2[2]>25 and
a[p[0]+1,p[1]+2]!=2 and a[p[0]+1,p[1]+3]==0 and ( a[p[0]+1,p[1]+4]!=0 or
a[p[0]+2,p[1]+4]!=0 or a[p[0]+2,p[1]+2]!=0) and (a[p[0]+2,p[1]+2]!=0  or
a[p[0]+2,p[1]+3]!=0)):
        print('Turning Right ie. East')
        bw.move1()
        cd.position(e)
        p[1]=p[1]+1
        ang=readE()

    elif((ang in W) and (a[p[0]-1,p[1]]==0 or a[p[0]+1,p[1]]==0)  and l2[1]>=5 and
l2[2]>=5  and a[int(p[0]),int(p[1])-1]!=2  and a[p[0]-1,p[1]-1]!=2 and ( a[p[0],p[1]-2]==0
or a[p[0]-1,p[1]-1]==0)and a[p[0]-1,p[1]-1]!=2  and(int(p[1])!=0) and
(a[int(p[0]),int(p[1])]==1)):
        print('reading west')
        ang=readW()

    elif((ang in W) and ((a[p[0]+3,p[1]+1]==0 and (a[int(p[0]),int(p[1])-1]==2 or
a[p[0]-1,p[1]-1]==2 or l2[0]<10 or l2[2]<=9) and a[p[0]+3,int(p[1])+1]==0) or (
(a[p[0],p[1]-3]!=0) and (a[p[0],p[1]-2]!=0)))):
        print('turning south')
        meth.turnL()
        cd.position(s)
        p[0]=p[0]+1
        p[1]=p[1]+1
        ang=readS()

    elif((ang in S) and ( a[p[0],p[1]-1]!=2  and  a[p[0],p[1]+1]!=2 ) and (l2[1] >= 5) and
(l2[2]>=5) and  (a[int(p[0]+1),int(p[1])]!=2) and (a[p[0]+1,p[1]-1]!=2) and
(int(p[0]!=size-1)) and (a[int(p[0]),int(p[1])]==1) and ((a[p[0]+2,p[1]]==0  or
a[p[0]+2,p[1]+1]==0 or a[p[0]-1,p[1]-3]!=0))):
        print('reading south')
        ang=readS()

    elif((ang in S) and (a[int(p[0]+1),int(p[1])]==2 or a[p[0]+1,p[1]-1]==2 or l2[0]<10
or l2[2]<=9)):
        print('turning towards east')
        meth.turnL()
        cd.position(e)
        p[0]=p[0]-1
        p[1]=p[1]+1
        ang=readE()
```

```
    elif((ang in E) and (a[p[0]+1,p[1]]==0 or a[p[0]-1,p[1]]==0  ) and (l2[1]>=5) and
(l2[2]>=5) and (a[int(p[0]),int(p[1])+1]!=2  and  a[p[0]+1,p[1]+1]!=2 )
and(int(p[1]!=size-1)) and (a[int(p[0]),int(p[1])]==1)):
        print('reading East')
        ang=readE()

    elif((ang in E) and (a[int(p[0]),int(p[1]+1)]==2 or a[p[0]+1,p[1]+1]  or l2[0]<10 or
l2[2]<=9)):
        print('turning towards North')
        cd.position(n)
        p[0]=p[0]-1
        p[1]=p[1]-1
        ang=readN()

    elif((ang in N) and p[0]>1 and (a[p[0],p[1]+1]!=2  and a[p[0],p[1]-1]!=2 ) and
(l2[1]>=5) and (l2[2]>=5)  and (a[int(p[0])-1,int(p[1])]!=2  and a[p[0]-1,p[1]+1]!=2) and
(int(p[0])!=0) and (a[int(p[0]),int(p[1])]==1)):
        print('Reading North')
        ang=readN()

    elif((ang in N) and (l2[0]<8  or(p[0]==1) or (a[p[0]-1,p[1]] == 2 or a[p[0]-
1,p[1]+1]==2  or l2[0]<10 or l2[2]<=9))):
        print('Turning towards West')
        meth.turnL()
        cd.position(w)
        p[0]=p[0]+1
        p[1]=p[1]-1
        ang=readW()

    else:

        rem= findrem()
        if(rem!='none'):
            print('Moving to area not sensed')
            moveto(rem)
        else:
            print('Done Mapping')
            minimize()
            for i in a:
                print(i)
            break

    print('\n\n')
```

```
def minimize():

    b=a.tolist()
    nl=[]
    for i in b:
        if(sum(i)!=0):
            nl.append(i)
    arr=np.array(nl)
    d=arr.transpose()
    nl=[]
    b=d.tolist()
    for i in b:
        if(sum(i)!=0):
            nl.append(i)
    arr=np.array(nl)
    arr=arr.transpose()
    np.save('area50',arr)


def findrem():
    rc=-1
    print('in findrem()')
    for r in a:
       # print(r)
        rc=rc+1
        if(rc<=rem_r):
            continue
        v1=0
        b=-1
        cc=-1
        d=0
        for c in r:
            cc=cc+1
            if((b==1 and c==0 and v1==0) or (b==0 and c==1 and v1==1)):
                v1=v1+1
            if(v1==1 and d==0):
                s=cc
                d=d+1
            if(v1==2):
                e=cc
                break
```

```
            b=c
        if(v1==2):
            return([rc,s])
    return('none')


def moveto(rem):

    print('Moving to the positon of '+ str(rem))
    if(rem[1]<p[1]):
        cm=w
    else:
        cm=e

    if(rem[0]<p[0]):
        rm=n
    else:
        rm=s
    trymis=0
    while(rem!=p):
        #print('MOVING COLUMN')
        trymis=trymis+1
        tryc=0
        while(rem[1]!=p[1]):
            print('MOVING COLUMN')
            cd.position(cm)
            sleep(0.01)
            l=us.sense()
            while(sum(l) > 2000):
                l=us.sense()
            if(l[0]<=12):
                tryc=tryc+1
            print(l)
            print(tryc)
            if(cm==w and l[0]>12):
                bw.move1()
                p[1]=p[1]-1
            elif(cm==w and rm==s and l[3]>12):
                a[p[0],p[1]-1]=2
                meth.turnL()
                cd.position(rm)
                bw.move1()
                meth.turnR()
                cd.position(cm)
```

```
        p[0]=p[0]+1
    elif(cm==w and rm==n and l[4]>12):
        a[p[0],p[1]-1]=2
        meth.turnR()
        cd.position(rm)
        bw.move1()
        meth.turnL()
        cd.position(cm)
        p[0]=p[0]-1
    elif(cm==e and l[0]>12):
        bw.move1()
        p[1]=p[1]+1
    elif(cm==e and rm==s and l[4]>12):
        a[p[0],p[1]+1]=2
        meth.turnL()
        cd.position(rm)
        bw.move1()
        meth.turnR()
        cd.position(cm)
        p[0]=p[0]+1
    elif(cm==e and rm==n and l[3]>12):
        a[p[0],p[1]+1]=2
        meth.turnR()
        cd.position(rm)
        bw.move1()
        meth.turnL()
        cd.position(cm)
        p[0]=p[0]-1
    else:
        print('stable')
    if(tryc > 5):
        break

print('MOVING ROW')
tryr=0
while(rem[0]!=p[0]):
    print('MOVING ROW')
    cd.position(rm)
    sleep(0.01)
    l=us.sense()
    while(sum(l) > 2000):
        l=us.sense()
    if(l[0]<=12):
        tryr=tryr+1
```

```
print(l)
print(tryr)
if(rm==n and l[0]>12):
   bw.move1()
   p[0]=p[0]-1
elif(rm==n and cm==w and l[3]>12):
   a[p[0]-1,p[1]]=2
   meth.turnL()
   cd.position(cm)
   bw.move1()
   meth.turnR()
   cd.position(rm)
   p[1]=p[1]-1
elif(rm==n and cm==e and l[4]>12):
   a[p[0]-1,p[1]]=2
   meth.turnR()
   cd.position(cm)
   bw.move1()
   meth.turnL()
   cd.position(rm)
   p[1]=p[1]+1

elif(rm==s and l[0]>12):
   bw.move1()
   p[0]=p[0]+1

elif(rm==s and cm==w and l[4]>12):
   a[p[0]+1,p[1]]=2
   meth.turnR()
   cd.position(cm)
   bw.move1()
   meth.turnL()
   cd.position(rm)
   p[1]=p[1]-1

elif(rm==s and cm==e and l[3]>12):
   a[p[0]+1,p[1]]=2
   meth.turnL()
   cd.position(cm)
   bw.move1()
   meth.turnR()
   cd.position(rm)
   p[1]=p[1]+1
else:
```

```
            print('stable')

          if(tryr>5):
            break

       if(trymis==1):
          print('quicting the row')
          rem_r=rem[0]
          break
   if(rem==p):
      a[p[0],p[1]]=1



def readW():
  # cd.position(w)
   r=int(p[0])
   c=int(p[1])
   print('['+str(r)+','+str(c)+']')
   a[r,c]=1
   a[r,c+1]=1
   a[r-1,c+1]=1
   a[r,c+2]=1
   a[r+1,c+1]=1

   print('before sense in readW')
   l=us.sense()
   while(sum(l) > 2000):
      l=us.sense()
   print(l)
   if(l[0]>10):
      a[r,c-1]=1
   else:
      d=us.sense()
      if(d[0]<=10):
         a[r,c-1]=2
      else:
         a[r,c-1]=1
   if(l[1]>5):
      a[r+1,c]=1
   else:
      a[r+1,c]=2

   if(l[2]>4):
```

```
    a[r-1,c]=1
else:
    a[r-1,c]=2

if(l[3]>10):
    a[r-2,c+1]=1
else:
    a[r-2,c+1]=2

if(r!=1):
    if(l[4]>15):
        a[r+1,c-2]==1
    else:
        a[r+1,c-2]==2


if(l[5]>8):
    a[r+1,c+2]=1
else:
    a[r+1,c+2]=2

if(l[6]>8):
    a[r-1,c+2]=1
else:
    a[r-1,c+2]=2
if(a[r,c-1]==1 and a[r+1,c]==1):
    cd.position(w)
    if(l[0]>18):
        print('Moving forward')
        p[1]=c-1
        bw.move1()
        cd.position(w)
    else:
        print('AUTO TURN LEFT')
        bw.move1()
        meth.turnL()
        p[0]=r+1
        cd.position(s)
else:
    print(a[r,c-1])
    print(a[r-1,c])
    print(a[r+1,c])
   # sleep(10)
np.save('area50',a)
```

```
  #  for i in a:
   #     print(i)
   # cd.position(270)
    print('read west')
    return(cmp.angle())


def readS():
   # cd.position(s)
    print(cmp.angle())
    r=int(p[0])
    c=int(p[1])
    print('['+str(r)+','+str(c)+']')
    a[r,c]=1
    a[r-1,c]=1
    a[r-2,c]=1
    a[r-1,c+1]=1
    a[r-1,c-1]=1
    print('Before sense')
    l=us.sense()
    while(sum(l) > 2000):
        l=us.sense()
    print(l)
    if(l[0]>10):
        a[r+1,c]=1
    else:
        d=us.sense()
        if(d[0]<=10):
            a[r+1,c]=2
        else:
            a[r+1,c]=1
    if(l[1]>5):
        a[r,c+1]=1
    else:
        a[r,c+1]=2

    if(l[2]>4):
        a[r,c-1]=1
    else:
        a[r,c-1]=2

    if(l[3]>10):
        a[r-1,c+2]=1
    else:
```

```
        a[r-1,c+2]=2

    if(l[4]>15):
        a[r-1,c-2]=1
    else:
        a[r-1,c-2]=2

    if(l[5]>8):
        a[r-2,c+1]=1
    else:
        a[r-2,c+1]=2

    if(l[6]>8):
        a[r-2,c-1]=1
    else:
        a[r-2,c-1]=2

    print('cloumn ::'+str(c))

    if(a[r+1,c]==1 and a[r,c-1]==1  and a[r,c+1]==1):
        cd.position(s)
        if(l[0]>18):
            print('moving forward')
            p[0]=r+1
            bw.move1()
            cd.position(s)
        else:
            print('AUTO TURN LEFT')
            bw.move1()
            meth.turnL()
            p[1]=c+1
            cd.position(e)
    np.save('area50',a)
   # for i in a:
    #    print(i)

    print('read south')
    return(cmp.angle())



def readE():
   # cd.position(e)
    r=p[0]
```

```
c=p[1]
print('['+str(r)+','+str(c)+']')
a[r,c]=1
a[r,c-1]=1
a[r,c-2]=1
a[r-1,c-1]=1
a[r+1,c-1]=1
print('Before sense')
l=us.sense()
while(sum(l) > 2000):
   l=us.sense()
print(l)

if(l[0]>10):
   a[r,c+1]=1
else:
   d=us.sense()
   if(d[0]<=10):
      a[r,c+1]=2
   else:
      a[r,c+1]=1

if(l[1]>5):
   a[r-1,c]=1
else:
   a[r-1,c]=2

if(l[2]>4):
   a[r+1,c]=1
else:
   a[r+1,c]=2

if(l[3]>10):
   a[r-2,c-1]=1
else:
   a[r-2,c-1]=2

if(c!=(size-2)):
   if(l[4]>15):
      a[r+2,c-1]=1
   else:
      a[r+2,c-1]=2

if(l[5]>8):
```

```
        a[r-1,c-2]=1
    else:
        a[r-1,c-2]=2

    if(l[6]>8):
        a[r+1,c-2]=1
    else:
        a[r+1,c-2]=2

    if(a[r-1,c]==1 and a[r,c+1]==1 ):
        cd.position(e)
        if(l[0]>25):
            p[1]=c+1
            bw.move1()
            cd.position(e)
        else:
            print('AUTO TURN LEFT')
            bw.move1()
            cd.position(n)
            p[0]=r-1
            cd.position(n)
    np.save('area50',a)
   # for i in a:
    #    print(i)
    print('read east')
    return(cmp.angle())



def readN():
   # cd.position(n)
    r=p[0]
    c=p[1]
    print('['+str(r)+','+str(c)+']')
    a[r,c]=1
    a[r+1,c]=1
    a[r+1,c+1]=1
    a[r+1,c-1]=1
    a[r+2,c]=1
    l=us.sense()
    while(sum(l) > 2000):
        l=us.sense()
    print(l)
```

```
if(l[0]>10):
   a[r-1,c]=1
else:
   d=us.sense()
   if(d[0]<=10):
      a[r-1,c]=2
   else:
      a[r-1,c]=1

if(l[1]>5):
   a[r,c-1]=1
else:
   a[r,c-1]=2

if(l[2]>=3):
   a[r,c+1]=1
else:
   a[r,c+1]=2

if(l[3]>10):
   a[r,c-2]=1
else:
   a[r,c-2]=2

if(c!=(size-2)):
   if(l[4]>15):
      a[r+1,c+2]=1
   else:
      a[r+1,c+2]=2

if(l[5]>8):
   a[r+2,c-1]=1
else:
   a[r+2,c-1]=2

if(l[6]>8):
   a[r+2,c+1]=1
else:
   a[r+2,c+1]=2

if(a[r-1,c]==1 and a[r,c-1]==1 and a[r,c+1]==1):
   cd.position(n)
   if(l[0]>18):
      p[0]=r-1
```

```
            bw.move1()
            cd.position(n)
        else:
            print('AUTO TURN LEFT')
            bw.move1()
            meth.turnL()
            p[1]=c-1
            cd.position(w)
    np.save('area50',a)
   # for i in a:
    #    print(i)

    print('read North')
    return(cmp.angle())



def map(l,pr,pc):
    f=np.load('area1.npy')
    if(l[0]>10):
        f[int(pr-1),int(pc)]=1
    else:
        f[int(pr-1),int(pc)]=2
    if(l[1]>8):
        f[int(pr),int(pc-1)]=1
    else:
        f[int(pr),int(pc-1)]=2
    if(l[2]>8):
        f[int(pr),int(pc+1)]=1
    else:
        f[int(pr),int(pc+1)]=2
    if(l[4]>10):
        f[int(pr+1),int(pc-2)]=1
    else:
        f[int(pr+1),int(pc-2)]=2
    if(l[5]>10):
        f[int(pr+1),int(pc+2)]=1
    else:
        f[int(pr+1),int(pc+2)]=2
    np.save('area1',f)
    return( [pr,pc])
```

## 6.2 CODE FOR CLEANING:

```
import ultrasonic as us
import com_dc as cd
import black_white as bw
import numpy as np

import RPi.GPIO as GPIO
from time import sleep
import methods as meth
import compass as cmp
import ucdc
import time
p=[]
d=np.load('area50.npy')
np.save('area51clean',d)
np.save('area51',d)

b=np.load('area51clean.npy')
a=np.load('area51.npy')
corners={}
n=ucdc.n
s=ucdc.s
e=ucdc.e
w=ucdc.w
N=ucdc.N
S=ucdc.S
W=ucdc.W
E=ucdc.E

Rsize=len(a)
Csize=len(a[0])
print('starting')
print(Rsize)
print(Csize)

def southcor():
    print('started')
    r=0
    nor=len(b)
    noc=len(b[0])
    print(noc)
    c=0
```

```
    for i in b:
        c=0
        for j in i:
            if(c>0 and c < (noc-1)):
#                print(str(r)+' '+str(c))
                if(r<(nor-1) and  b[r,c-1]!=1 and b[r,c]==1 and b[r+1,c]!=1 and b[r,c+1]==1):
                    temp=[r,c+1]
                    print(temp)
                    R=r
                    C=c+1
                    des=0
                    while(b[R-1,C]==1 and (b[R-1,C-2]!=1 or b[R-1,C-3]!=1 or C==2)):
                        des=des+1
                        R=R-1
                    corners[des]=temp
            c=c+1
        r=r+1

    print(corners)

def initial():
    ang= bw.moveD(ucdc.w)
    meth.turnL()
    cd.position(ucdc.s)
    ang=bw.moveD(ucdc.s)
    cd.position(ucdc.n)

    len=0
    l=us.sense()
    while(sum(l) > 2000):
            l=us.sense()

    while(l[0]>10):
        cd.position(ucdc.n)
        bw.move1()
        len=len+1
        sleep(0.01)
        l=us.sense()
        while(sum(l) > 2000):
            l=us.sense()
    print(len)
    return(len)

def r_c(corners,len):
```

```
    l=[]
    for key in corners:
        l.append(key)
    l2=[]
    for i in l:
        if((i+5)-len>0):
            l2.append(i-len)
    val=min(l2)+len
    print(corners[val])
    return(corners[val])


def in_pos():
    cd.position(ucdc.s)
    bw.moveD(ucdc.s)
    cd.position(ucdc.w)
    bw.moveD(ucdc.w)
    cd.position(ucdc.n)
    print(p)
    p[0]=p[0]-2

def test(r,c):
    d=np.load('area51clean.npy')
    if(d[r,c]==1 and d[r,c+1]==1 and d[r,c+2]==1 and d[r+1,c]==1 and d[r+1,c+1]==1 and
d[r+1,c+2]==1 and d[r+2,c]==1 and d[r+2,c+1]==1 and d[r+2,c+2]==1   and a[r,c]==1
and a[r,c+1]==1 and a[r,c+2]==1 and a[r+1,c]==1 and a[r+1,c+1]==1 and a[r+1,c+2]==1
and a[r+2,c]==1 and a[r+2,c+1]==1 and a[r+2,c+2]==1   ):
        return(1)
    return(0)

def moveto(rem):

    print('Moving to the positon of '+ str(rem))
    if(rem[1]<p[1]):
        cm=w
    else:
        cm=e

    if(rem[0]<p[0]):
        rm=n
    else:
        rm=s
    trymis=0
    while(rem!=p):
```

```
#print('MOVING COLUMN')
trymis=trymis+1
tryc=0
while(rem[1]!=p[1]):
    print('MOVING COLUMN')
    cd.position(cm)
    sleep(0.01)
    l=us.sense()
    while(sum(l) > 2000):
        l=us.sense()
    if(l[0]<=12):
        tryc=tryc+1
    print(l)
    print(tryc)
    if(cm==w and l[0]>12):
        bw.move1()
        p[1]=p[1]-1
    elif(cm==w and rm==s and l[3]>12):
        a[p[0],p[1]-1]=2
        meth.turnL()
        cd.position(rm)
        bw.move1()
        meth.turnR()
        cd.position(cm)
        p[0]=p[0]+1
    elif(cm==w and rm==n and l[4]>12):
        a[p[0],p[1]-1]=2
        meth.turnR()
        cd.position(rm)
        bw.move1()
        meth.turnL()
        cd.position(cm)
        p[0]=p[0]-1
    elif(cm==e and l[0]>12):
        bw.move1()
        p[1]=p[1]+1
    elif(cm==e and rm==s and l[4]>12):
        a[p[0],p[1]+1]=2
        meth.turnL()
        cd.position(rm)
        bw.move1()
        meth.turnR()
        cd.position(cm)
        p[0]=p[0]+1
```

```
    elif(cm==e and rm==n and l[3]>12):
        a[p[0],p[1]+1]=2
        meth.turnR()
        cd.position(rm)
        bw.move1()
        meth.turnL()
        cd.position(cm)
        p[0]=p[0]-1
    else:
        print('stable')
    if(tryc > 5):
        break

print('MOVING ROW')
tryr=0
while(rem[0]!=p[0]):
    print('MOVING ROW')
    cd.position(rm)
    sleep(0.01)
    l=us.sense()
    while(sum(l) > 2000):
        l=us.sense()
    if(l[0]<=12):
        tryr=tryr+1
    print(l)
    print(tryr)
    if(rm==n and l[0]>12):
        bw.move1()
        p[0]=p[0]-1
    elif(rm==n and cm==w and l[3]>12):
        a[p[0]-1,p[1]]=2
        meth.turnL()
        cd.position(cm)
        bw.move1()
        meth.turnR()
        cd.position(rm)
        p[1]=p[1]-1
    elif(rm==n and cm==e and l[4]>12):
        a[p[0]-1,p[1]]=2
        meth.turnR()
        cd.position(cm)
        bw.move1()
        meth.turnL()
        cd.position(rm)
```

```
            p[1]=p[1]+1

        elif(rm==s and l[0]>12):
            bw.move1()
            p[0]=p[0]+1

        elif(rm==s and cm==w and l[4]>12):
            a[p[0]+1,p[1]]=2
            meth.turnR()
            cd.position(cm)
            bw.move1()
            meth.turnL()
            cd.position(rm)
            p[1]=p[1]-1

        elif(rm==s and cm==e and l[3]>12):
            a[p[0]+1,p[1]]=2
            meth.turnL()
            cd.position(cm)
            bw.move1()
            meth.turnR()
            cd.position(rm)
            p[1]=p[1]+1
        else:
            print('stable')

        if(tryr>5):
            break

    if(trymis==1):
        print('quicting the row')
        rem_r=rem[0]
        break
  if(rem==p):
    a[p[0],p[1]]=1


def notcleaned():
  d=np.load("area51clean.npy")
  r=0
  for i in d:
    count=0
    c=0
    if(r<(Rsize-2)):
```

```
            for j in i:
                if(j==1 and count<=2 ):
                    count=count+1
                if(count==3 and c<(Csize-2)):
                    print(str(r)+'-'+str(c-2))
                    q=test(r,c-2)
                    if(q==1):
                        #print('moving')
                        ucdc.moveto([r+1,c-2])
                        return(1)
                c=c+1
        r=r+1
    return(0)

def start():
    stp=1
    ang=n
    while(stp):
#        for i in a:
 #           print(i)
  #      print('cleaned area')
        for i in b:
            print(i)
        print("\n"+str(p)+str(ang))
        r=p[0]
        c=p[1]
        sleep(0.02)
        l=us.sense()
        while(sum(l) > 2000):
            l=us.sense()
        l1=us.sense()
        while(sum(l1) > 2000):
            l1=us.sense()
       # print(r)
       # print(Rsize)
        print(l)
       # print(l1)
        if((ang in N) and r>2 and ((l[0]+l1[0])/2)>=10 and ((l[1]+l1[1])/2)>=6 and
((l[2]+l1[2])/2)>=6 and (((l[3]+l1[3])/2)<18 or b[r+1,c-2]==3 or c<=2) ):
            print('clean north')
            ang=readN()

        elif(ang in N and c>2 and ((l[3]+l1[3])/2)>=18 and b[r+1,c-2]!=3):
            meth.turnL()
```

```
            bw.movec()
            meth.turnR()
            p[1]=p[1]-1

        elif(ang in N and (((l[0]+l1[0])/2)<10 or ((l[1]+l1[1])/2)<12 or ((l[2]+l1[2])/2)<12
or r<=2) and ((l[4]+l1[4])/2)>10 ):
            print('Turning Right ie. East')
            bw.movec()
            meth.turnR()
            p[1]=p[1]+1
            ang=readE()

        elif(ang in S and (((((l[0]+l1[0])/2)<10 or ((l[1]+l1[1])/2)<12 or ((l[2]+l1[2])/2)<12)
and ((l[3]+l1[3])/2)>10) or (r>=(Rsize-2) and ((l[3]+l1[3])/2)>10) )):
            print('turning east')
           # meth.turnL()
            cd.position(e)
            p[0]=p[0]-1
            p[1]=p[1]+1
            ang=readE()


        elif(ang in S and ((l[4]+l1[4])/2)>=18 and b[r-1,c-2]!=3):
            print('turning right')
            meth.turnR()
            bw.movec()
            meth.turnL()
            p[1]=p[1]-1


        elif( (ang in S) and r<(Rsize-2)  and ((l[0]+l1[0])/2)>=10 and ((l[1]+l1[1])/2)>=6
and ((l[2]+l1[2])/2)>=6 and (b[r+1,c]!=3 or b[r+1,c-1]!=3 or b[r+1,c+1]!=3)):
            print('clean south')
            ang=readS()


        elif(ang in E and c<(Csize-1)  and((l[0]+l1[0])/2)>12 and ((l[1]+l1[1])/2)>=6 and
((l[2]+l1[2])/2)>=6 and (((l[3]+l1[3])/2)<12 or ((l[4]+l1[4])/2)<12 or ((l[6]+l1[6])/2)<12
or ((l[2]+l1[2])/2)<12  or ((l[1]+l1[1])/2)<12)):
            print('clean east')
            ang=readE()

        elif(ang in E and r<(Rsize-3) and ( ((l[0]+l1[0])/2)>10 and ((l[1]+l1[1])/2)>6 and
((l[2]+l1[2])/2)>6) and  ((l[4]+l1[4])/2)>12  and b[r+2,c-1]!=3 and b[r-2,c-1]!=3 ):
```

```
            #meth.turnR()
            cd.position(s)
            p[0]=p[0]+1
            p[1]=p[1]-1
            ang=readS()

        elif(ang in E and r<(Rsize-3) and ( ( ((l[0]+l1[0])/2)<10 or ((l[1]+l1[1])/2)<6 or
((l[2]+l1[2])/2)<6) or (b[r-2,c-2]==3 and b[r+2,c-1]!=3 and  ((l[6]+l1[6])/2)>10 and
((l[4]+l1[4])/2)>10 and ((l[2]+l1[2])/2)>10) ) and (((l[3]+l1[3])/2)<10 or b[r-2,c-2]==3)
and ((l[4]+l1[4])/2)>12 ):
            meth.turnR()
            bw.movec()
            meth.turnL()
            p[0]=p[0]+1

        elif(ang in E and ( ((l[0]+l1[0])/2)<10 or ((l[1]+l1[1])/2)<6 or ((l[2]+l1[2])/2)<6)
and (((l[4]+l1[4])/2)<10 or b[r+2,c-1]==3) and ((l[3]+l1[3])/2)>12 ):
            cd.position(n)
            bw.movec()
            meth.turnR()
            p[0]=p[0]-1

        else:
            res=notcleaned()
            if(res==0):
                print('done cleaning')
                break

def readN():
   # cd.position(n)
    r=p[0]
    c=p[1]
    print('['+str(r)+','+str(c)+']')
    b[r,c]=3
    b[r+1,c]=3
    b[r+1,c+1]=3
    b[r+1,c-1]=3
    b[r,c+1]=3
    b[r,c-1]=3
    b[r+2,c]=3
    b[r+2,c-1]=3
    b[r+2,c+1]=3
    a[r,c]=1
    a[r+1,c]=1
```

```
a[r+1,c+1]=1
a[r+1,c-1]=1
a[r+2,c]=1

l=us.sense()
while(sum(l) > 2000):
   l=us.sense()
#print(l)
if(b[r-1,c]!=3):
   if(l[0]>10):
      a[r-1,c]=1
   else:
      d=us.sense()
      if(d[0]<=10):
         a[r-1,c]=2
      else:
         a[r-1,c]=1

if(b[r,c-1]!=3):
   if(l[1]>5):
      a[r,c-1]=1
   else:
      a[r,c-1]=2

if(b[r,c+1]!=3):
   if(l[2]>=3):
      a[r,c+1]=1
   else:
      a[r,c+1]=2

if(b[r,c-2]!=3):
   if(l[3]>10):
      a[r,c-2]=1
   else:
      a[r,c-2]=2

if(c<(Csize-2)):
   if(b[r+1,c+2]!=3):
      if(l[4]>15):
         a[r+1,c+2]=1
      else:
         a[r+1,c+2]=2

if(b[r+2,c-1]!=3):
```

```
        if(l[5]>8):
            a[r+2,c-1]=1
        else:
            a[r+2,c-1]=2

    if(b[r+2,c+1]!=3):
        if(l[6]>8):
            a[r+2,c+1]=1
        else:
            a[r+2,c+1]=2

    if(a[r-1,c]==1 and l[1]>10 and l[2]>10):
        cd.position(n)
        if(l[0]>18):
            p[0]=r-1
            bw.movec()
            cd.position(n)
        if(l[0]<=18 and r>=2):
            print('AUTO TURN RIGHT')
            bw.movec()
            #meth.turnR()
            p[1]=c+1
            cd.position(e)
            readE()
    np.save('area51',a)
    np.save('area51clean',b)
   # for i in a:
    #    print(i)

    print('cleaned North')
    return(cmp.angle())

def readS():
   # cd.position(s)
    print(cmp.angle())
    r=int(p[0])
    c=int(p[1])
    print('['+str(r)+','+str(c)+']')
    b[r,c]=3
    b[r-1,c]=3
    b[r-2,c]=3
    b[r-2,c-1]=3
    b[r-2,c+1]=3
    b[r-1,c+1]=3
```

```
 b[r-1,c-1]=3
 b[r,c-1]=3
 b[r,c+1]=3
 a[r,c]=1
 a[r-1,c]=1
 a[r-2,c]=1
 a[r-1,c+1]=1
 a[r-1,c-1]=1
 print('Before sense')
 l=us.sense()
 while(sum(l) > 2000):
    l=us.sense()
# print(l)

 if(b[r+1,c]!=3):
    if(l[0]>10):
       a[r+1,c]=1
    else:
       d=us.sense()
       if(d[0]<=10):
          a[r+1,c]=2
       else:
          a[r+1,c]=1

 if(b[r,c+1]!=3):
    if(l[1]>5):
       a[r,c+1]=1
    else:
       a[r,c+1]=2

 if(b[r,c-1]!=3):
    if(l[2]>4):
       a[r,c-1]=1
    else:
       a[r,c-1]=2

 if(p[1]<(Csize-2)):
    if(b[r-1,c+2]!=3):
       if(l[3]>10):
          a[r-1,c+2]=1
       else:
          a[r-1,c+2]=2
 if(c>1):
    if(b[r-1,c-2]!=3):
```

```
            if(l[4]>15):
                a[r-1,c-2]=1
            else:
                a[r-1,c-2]=2

    if(b[r-2,c+1]!=3):
        if(l[5]>8):
            a[r-2,c+1]=1
        else:
            a[r-2,c+1]=2

    if(b[r-2,c-1]!=3):
        if(l[6]>8):
            a[r-2,c-1]=1
        else:
            a[r-2,c-1]=2

    print('cloumn ::'+str(c))

    if(a[r+1,c]==1 and l[1]>10  and l[2]>10):
        cd.position(s)
        if(l[0]>18):
            print('moving forward')
            p[0]=r+1
            bw.movec()
            cd.position(s)
        else:
            print('AUTO TURN LEFT')
            bw.movec()
            cd.position(e)
            p[1]=c+1
            cd.position(e)
            readE()

    np.save('area51',a)
    np.save('area51clean',b)
   # for i in a:
    #    print(i)

    print('cleaned south')
    return(cmp.angle())

def readE():
   # cd.position(e)
```

```
r=p[0]
c=p[1]
print('['+str(r)+','+str(c)+']')
b[r,c]=3
b[r,c-1]=3
b[r,c-2]=3
b[r-1,c-2]=3
b[r+1,c-2]=3
b[r-1,c-1]=3
b[r+1,c-1]=3
b[r-1,c]=3
b[r+1,c]=3
a[r,c]=1
a[r,c-1]=1
a[r,c-2]=1
a[r-1,c-1]=1
a[r+1,c-1]=1
print('Before sense')
l=us.sense()
while(sum(l) > 2000):
   l=us.sense()
#print(l)

if(b[r,c+1]!=3):
   if(l[0]>10):
      a[r,c+1]=1
   else:
      d=us.sense()
      if(d[0]<=10):
         a[r,c+1]=2
      else:
         a[r,c+1]=1

if(b[r-1,c]!=3):
   if(l[1]>5):
      a[r-1,c]=1
   else:
      a[r-1,c]=2

if(b[r+1,c]!=3):
   if(l[2]>4):
      a[r+1,c]=1
   else:
      a[r+1,c]=2
```

```
if(b[r-2,c-1]!=3):
    if(l[3]>10):
        a[r-2,c-1]=1
    else:
        a[r-2,c-1]=2


if(c!=(Csize-2)):
    if(b[r+2,c-1]!=3):
        if(l[4]>15):
            a[r+2,c-1]=1
        else:
            a[r+2,c-1]=2

if(b[r-1,c-2]!=3):
    if(l[5]>8):
        a[r-1,c-2]=1
    else:
        a[r-1,c-2]=2

if(b[r-1,c-2]!=3):
    if(l[6]>8):
        a[r+1,c-2]=1
    else:
        a[r+1,c-2]=2

if(a[r-2,c-1]==2 ):
    cd.position(e)
j=0
for i in range(2):
    l=us.sense()
    while(sum(l) > 2000):
        l=us.sense()

    if(l[0]>12):
        j=j+1
        p[1]=p[1]+1
        b[p[0],p[1]]=3
        b[p[0],p[1]-1]=3
        b[p[0],p[1]-2]=3
        b[p[0]-1,p[1]-1]=3
        b[p[0]+1,p[1]-1]=3
        b[p[0]-1,p[1]]=3
```

```
            b[p[0]+1,p[1]]=3
            b[p[0]+1,p[1]-2]=3
            b[p[0]-1,p[1]-1]=3
            a[p[0],p[1]]=1
            a[p[0],p[1]-1]=1
            a[p[0],p[1]-2]=1
            a[p[0]-1,p[1]-1]=1
            a[p[0]+1,p[1]-1]=1
            bw.movec()
            cd.position(e)
            if(b[p[0]-2,p[1]-1]!=3):
               if(l[3]>10):
                  a[p[0]-2,p[1]-1]=1
               else:
                  a[p[0]-2,p[1]-1]=2
            r=p[0]
            c=p[1]
         else:
            print(' TURN RIGHT')
            meth.turnR()
            p[0]=r+1
            p[1]=c-1
            cd.position(s)
            break
      print(str(j)+'  '+str(l[3]))
      if(j==2 and l[3]<12):
         meth.turnR()
         p[0]=r+1
         p[1]=c-1
         cd.position(s)
      if(j==2 and (a[r+2,c-1]==2 or l[4]<=15 or r>=Rsize-3 or c>=(Csize-2))):
         cd.position(n)
         p[0]=r-1
         p[1]=c-1
         cd.position(n)
      np.save('area51',a)
      np.save('area51clean',b)
     # for i in a:
      #    print(i)
      print('read east')
      return(cmp.angle())

c=b
southcor()
```

```
len =initial()
p=r_c(corners,len)
print(p)
in_pos()
start()
```

## 7.1 RVC PROTOTYPE:

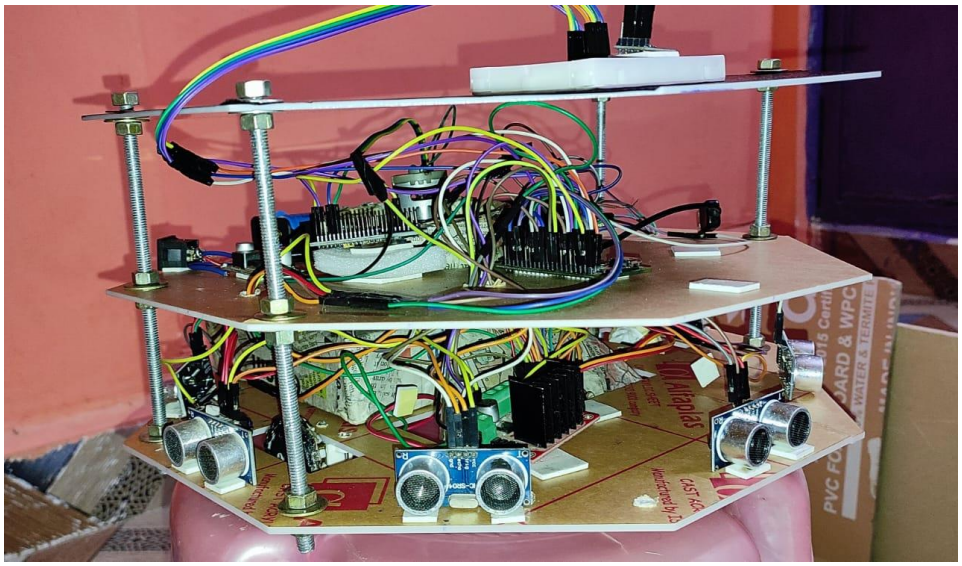

**Fig – 7.1 a**

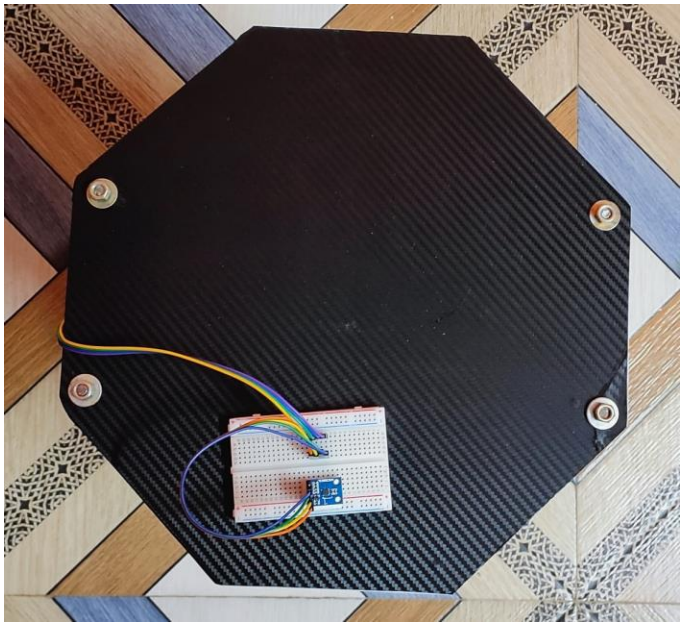

**Fig – 7.1 b**

## 7.2 SIMULATION RESULTS:

### 7.2.1 SCANNING RESULT:


**Fig - 7.2**

- **"Done Mapping"** statement appears on the console of raspberrypi os after the completion of proper scanning process.
- Also with the phrase - **"Done Mapping"**, a matrix with information will be displayed.
- **area.py** is the file that contains the location data and gets updated while scanning inside the perimeter that is supposed to get clean.
- **area.py** file interprets the location data in a form of matrix.
- **0 –** represents the area that is not scanned.
- **1** – represents the area is scanned.
- **2** – represents that corresponding location has obstacles.


**Fig – 7.3**

## 7.2.2 CLEANING RESULT:

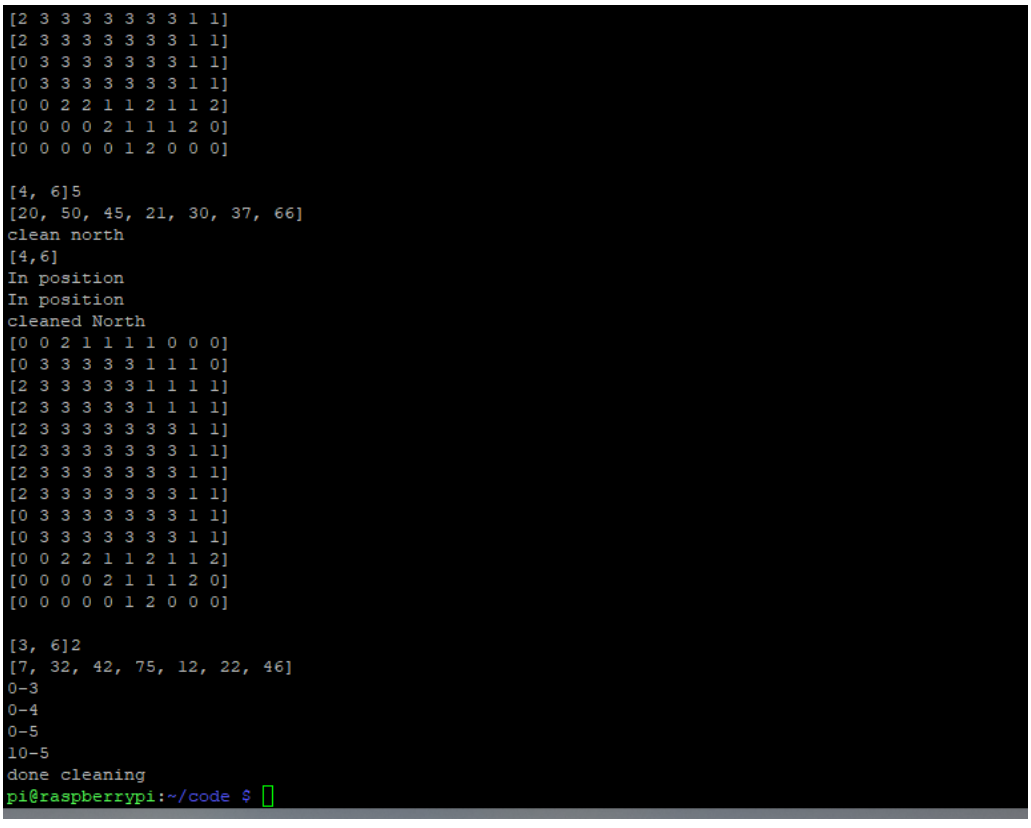


**Fig – 7.4**

- After the completion of scanning process, by executing the cleaning program the cleaning process will begin.
- It will make use of the same area50.npy file for cleaning.
- **"done cleaning"** phrase appears on the console indicating the completion of cleaning process and it also displays the matrix.
- **3** in the matrix indicates that cleaning has been done at the corresponding location.

## 7.3 ANALYSIS:

### 7.3.1 SEQUENCE OF OPERATIONS:

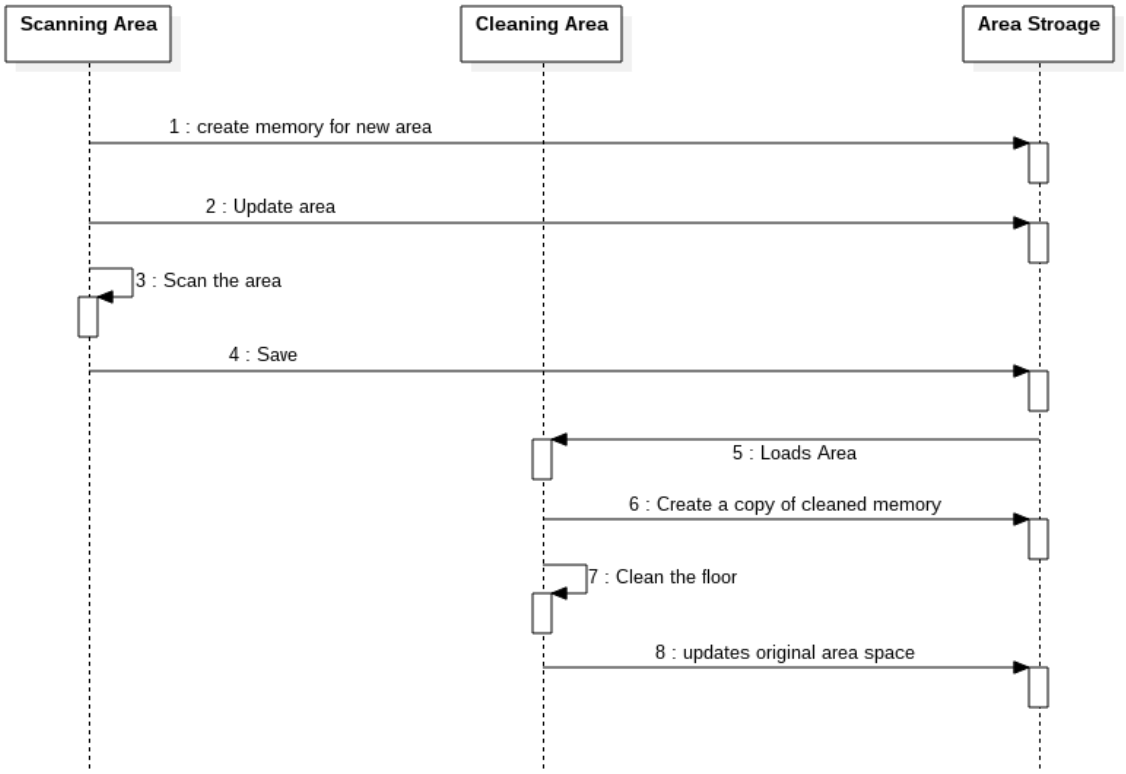


**Fig – 7.5**

## 8.1 CONCLUSION:

The developed product efficiently ensures:

1. Scanning of a new area which needs to be cleaned and stores scanned area as a .npy file in a memory present in raspberry pi.
2. Cleaning process executes after every time period completed, simultaneously updating the area file if any obstacles are found.
3. By using such method, one can have a cleaner home without breaking a sweat with the help of a robot vacuum. They’re great for the upkeep of floors, and it’ll reduce the need to deep-clean frequently. By using these sensors, it is not difficult to acquire these parameters with high accuracy in real time. Hence system is safe.

## 9.1 CONTRIBUTION OF THE PROJECT:

The ‘digital revolution’ of household life is underway, with technologies such as robotic vacuum cleaners (robovacs) increasingly common. Various other automated appliances are emerging and being adopted in pursuit of the ‘smart home’.

This Robotic vacuum cleaner which is in octagonal shape will help in cleaning your home for you. With India undergoing a major digital transformation as well as social and economic changes, the demand for robot vacuum cleaners is expected to grow further in the coming years

There are several benefits of using robotic vacuum cleaner and some of them are listed below:

- No need for manual operation
- Detect preset boundaries
- Low Maintenance
- Fits into tight space
- No cords to deal with
- Smart home connectivity

## 9.2 FUTURE SCOPE:

- By introducing a **UV-rays** source, Robotic Vacuum cleaner can also be used as a floor Disinfectant.
- **AI** and **Machine Language** can also be implemented to make mapping technology much more accurate.
- In future the vacuum cleaner can be added with many features like moping, carpet cleaning and an application can be designed for the robot to make track of it.

## REFERENCES:


1. P.B. Jarande, S.P. Murakar, N.S. Vast, N.P. Ubale, S.S Saraf, “Robotic Vaccum Cleaner using Arduino with WiFi ” published in 2018 Second International Conference on Inventive Communication and Computational Technologies.
   **Link:** https://ieeexplore.ieee.org/abstract/document/8473256

2. Kazi Mahmud Hasan, Adullah-Al-Nahid, Khondker Jahid Reza, “Path Planning algorithm development for autonomous vacuum cleaner robots” published in 2014 International Conference on Informatics, Electronics and Vision (ICIEV).
   **Link:** https://ieeexplore.ieee.org/abstract/document/6850799

3. T.B. Asafa, T.M. Afonja, E.A. Olaniyan, H.O. Alade, ”Development of a vacuum cleaner robot” published in 2018 Alexandria Engineering Journal [AEJ].
   **Link**: https://www.sciencedirect.com/science/article/pii/S111001681 8300899

4. Hiroshi Hisahara, Yuki Ishii, Masahito Ota, Takeki Ogitsu, “Human avoidance function for robotic vacuum cleaner through use of environmental sensors: Roomba making way for humans” published in 2014 IEEE

5. The Raspberry Pi Foundation
   **Link:** https://www.raspberrypi.com/documentation/

6. Ultrasonic sensor user Manual by Cytron Technologies
   **Link:**
   https://web.eece.maine.edu/~zhu/book/lab/HCSR04%20User%20Manual.pdf