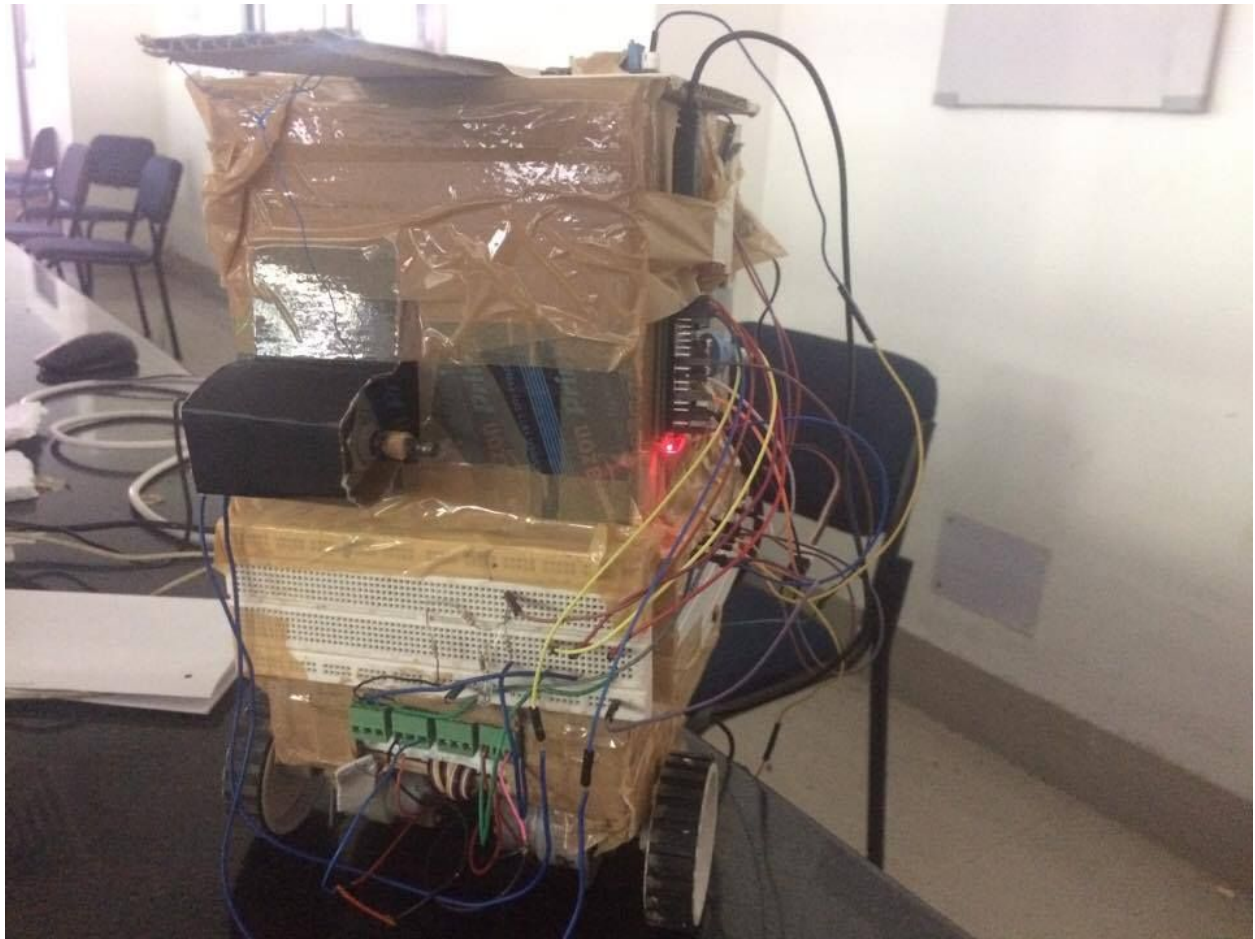


Smart Trash Can

14.11.2017



Guided By

Prof. S. B. Nair

Group Members

Neel Mittal (150101042)

Prashansi Kamdar (150101047)

Suhas Kantekar (150101077)

Tamilselvan S (150101079)

Index

Introduction.....	2
Materials and sensors used.....	2
Setup.....	4
Locomotion.....	4
Trash Can.....	4
Pairing bluetooth module.....	4
Connections for sensors.....	6
RSSI values vs Distance.....	9
IR sensor characteristics curve.....	10
Salient Features.....	11
Locomotion.....	11
Trash Can.....	11
Timeline.....	12
Week 1.....	15
Week 2.....	15
Final Week.....	16
Problems Faced.....	14
Alternate Uses.....	14
Code Snippets.....	15
Arduino.....	15
RaspBerry Pi.....	22

Introduction

The aim of this project is to create a smart mobile trash can. This trash-can can be summoned using a mobile phone with bluetooth. The trash can automatically reaches the position closest to the bluetooth signal origin by measuring the signal strength at a few critical locations. The trash-can's door automatically opens when it reaches the desired location. This can also has functionality of notifying the user by means of a buzzer when it detects that it is full.

Materials and sensors used

1. 4x Grove Line Finder IR sensors functioning as a line following unit

This IR sensors have been used for line following unit for the trash can. The readings of the 4 IR sensors along with the Arduino algorithm allows for line following functionality.

2. 3x DC Motors

Two DC motors have been used for locomotion of the trash can and the remaining one functions as a lid opener.

3. 1x Buzzer

Buzzer has been used to alarm the user when the trash can is full.

4. 1x IR Sensor

This IR sensor detects proximity to the lid of the trash present in the box. When it reaches a certain threshold, the automated buzzer is turned on.

5. 1x Arduino

This Arduino functions as locomotive agent to the RaspBerry Pi. The RaspBerry Pi works as a master and sends signals to arduino to carry out locomotion.

6. 1x RaspBerry Pi

RaspBerry Pi is the central functioning part of the trash can, it controls locomotion, bluetooth module and various sensors as mentioned above.

7. Wooden Board, Mounting brackets for base, Cardboard

These materials have been used for building the basic structure of trash can.

8. 1x Castor Wheel

This free moving wheel aids in balancing the trash can.

9. 2x Wheel

These wheels help in the motion of the trash can.

10. 1x Relay

This relay has been used to give large voltage source to dc motors.

11. 2x 9V batteries, 5V DC power source

The 9V batteries function as power supplies for the DC motors . The 5V power source functions as a power source for Arduino & RaspBerry Pi.

12. Resistors, Jumper wires, LEDs, breadboard

Resistors have been used to decrease current input to dc motors and hence, slow their speeds to allow smoother motion. Jumper wires and breadboard for connection. LEDs used for observing signal lines while operating.

13. Bluetooth Module for RaspBerry Pi

Bluetooth module which can pair with bluetooth devices and can get the RSSI value of connected device. Module uses Bluetooth 2.0 technology.

Setup

Locomotion

- 4x IR sensor were placed at the bottom of the base so as to sense black lines on the floor, their inputs were connected to the Arduino.
- 2x motors with wheels, a castor wheel and the relay were fixed to the base.
- The motors were connected to the relay and relay's input pins were connected to Arduino. The Arduino was fixed to base and its input pins were connected to the RaspBerry Pi.

Trash can

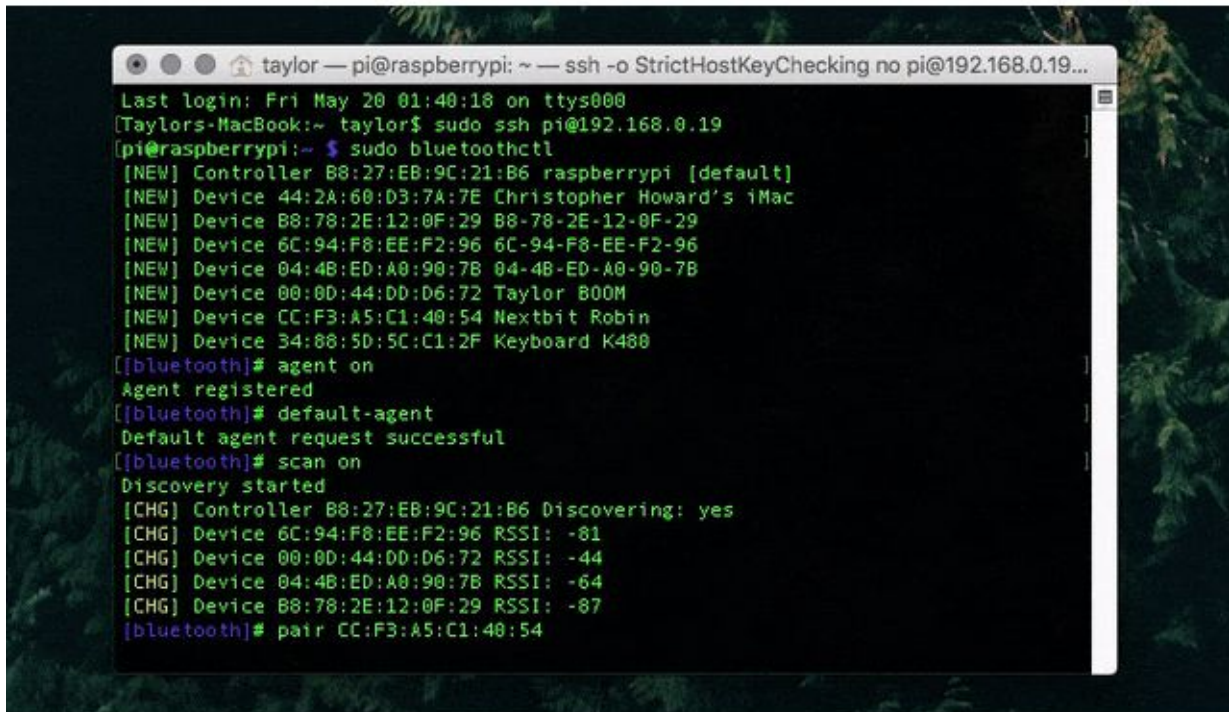
- A custom made cardboard box was fixed at top of the base to function as a can.
- Sensors, breadboards and motor were fixed at desirable positions on the trash can itself.
- The motor was connected to the lid of the trash can.
- RaspBerry Pi's control pins (GPIO pins) were connected to Arduino.
- The bluetooth module was connected to RaspBerry Pi's usb port.
- The sensors input and output pins were connected to RaspBerry's GPIO pins using the breadboards available.

Pairing Bluetooth Module with another Bluetooth device:

Steps to pair

1. Open terminal
2. Enter command - `$sudo bluetoothctl`
3. Now terminal shows all the bluetooth enabled devices near it.
4. Enter command - `$agent on`
5. Now the agent is registered.
6. Enter command - `$default-agent`
7. Default agent request is successful.

8. Enter command- \$scan on
9. Now terminal shows MAC address of nearby bluetooth devices.
10. Enter Command (XX:XX:XX:XX:XX:XX is the address of desired device) -
\$pair XX:XX:XX:XX:XX:XX
11. Enter Command - \$connect XX:XX:XX:XX:XX:XX
12. Now enter the same pin on both the devices and wait for devices to connect.
13. Bluetooth symbol changes with two dots to left and right of it implying a successful connection.

A screenshot of a terminal window titled 'taylor — pi@raspberrypi: ~ — ssh -o StrictHostKeyChecking no pi@192.168.0.19...'. The terminal shows the following commands and output:

```
Last login: Fri May 20 01:40:18 on ttys000
[Taylor's-MacBook:~ taylor$ sudo ssh pi@192.168.0.19
pi@raspberrypi:~$ sudo bluetoothctl
[NEW] Controller B8:27:EB:9C:21:B6 raspberrypi [default]
[NEW] Device 44:2A:60:D3:7A:7E Christopher Howard's iMac
[NEW] Device B8:78:2E:12:0F:29 B8-78-2E-12-0F-29
[NEW] Device 6C:94:F8:EE:F2:96 6C-94-F8-EE-F2-96
[NEW] Device 04:4B:ED:A0:90:7B 04-4B-ED-A0-90-7B
[NEW] Device 00:00:44:0D:D6:72 Taylor BOOM
[NEW] Device CC:F3:A5:C1:40:54 Nextbit Robin
[NEW] Device 34:88:5D:5C:C1:2F Keyboard K480
[[bluetooth]# agent on
Agent registered
[[bluetooth]# default-agent
Default agent request successful
[[bluetooth]# scan on
Discovery started
[CHG] Controller B8:27:EB:9C:21:B6 Discovering: yes
[CHG] Device 6C:94:F8:EE:F2:96 RSSI: -81
[CHG] Device 00:00:44:0D:D6:72 RSSI: -44
[CHG] Device 04:4B:ED:A0:90:7B RSSI: -64
[CHG] Device B8:78:2E:12:0F:29 RSSI: -87
[[bluetooth]# pair CC:F3:A5:C1:40:54
```

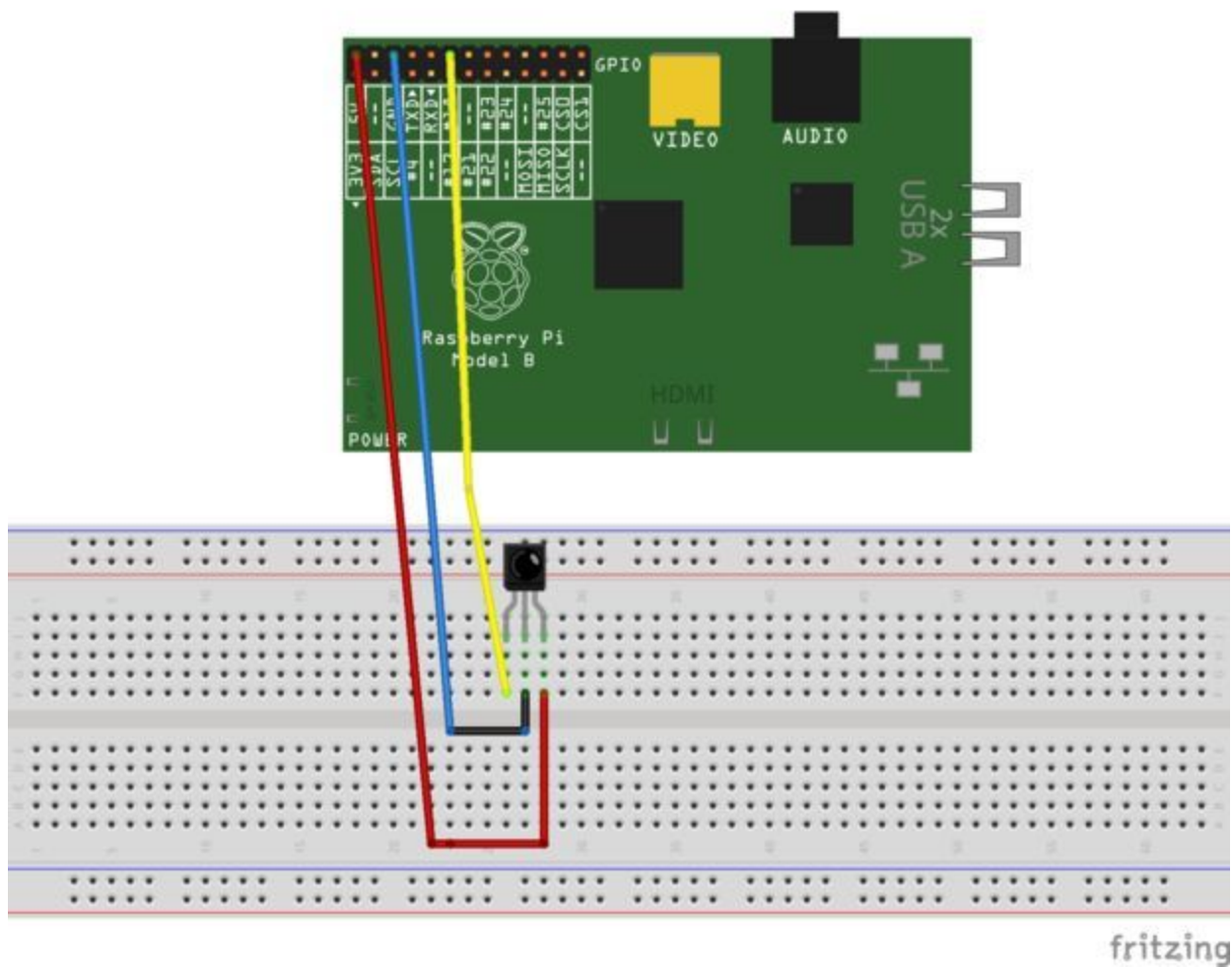
Screenshot of Terminal output

Connections for sensors:

1. IR sensor :

output: GPIO x3

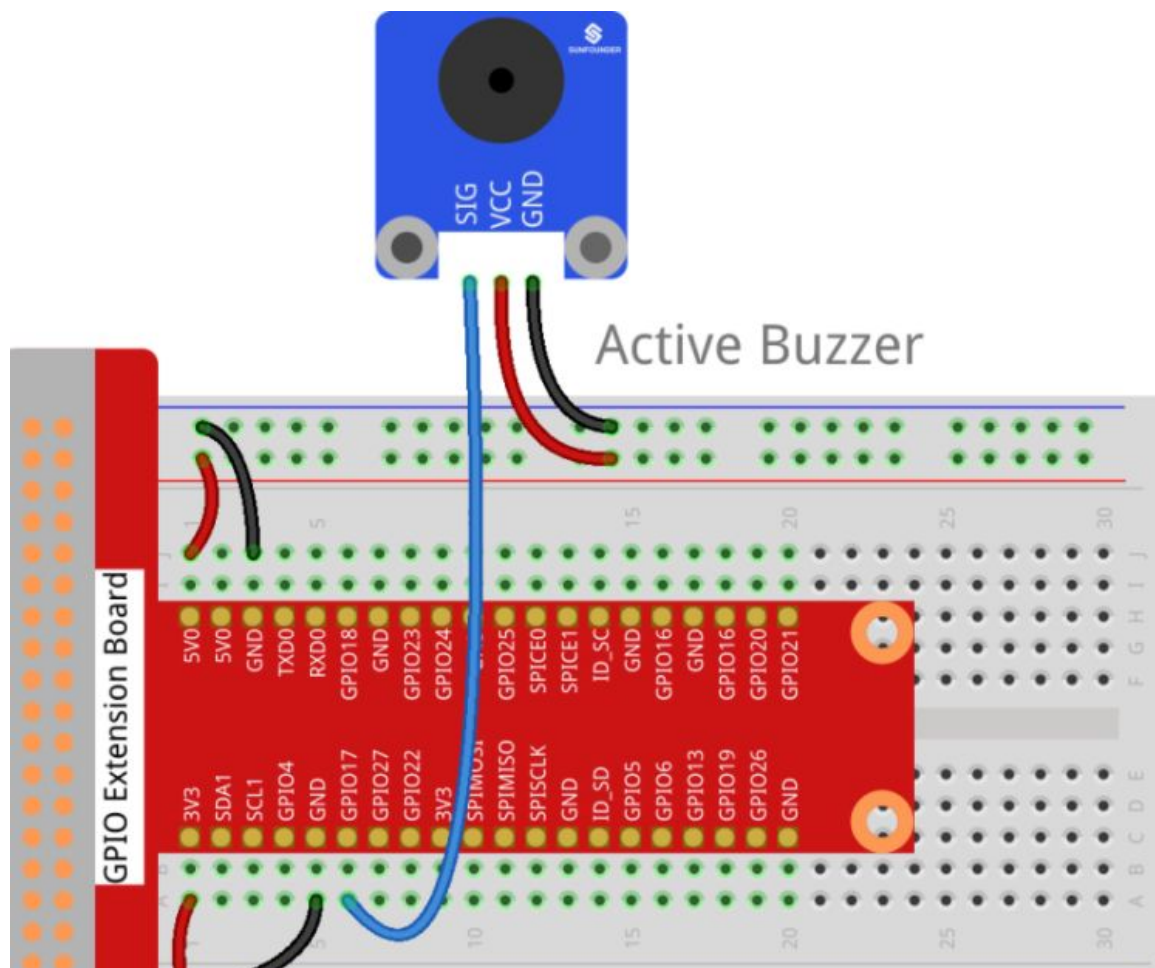
Connect Vcc and Ground to respective pins in RPi.



2. Buzzer :

Input: input GPIO x4

Connect Vcc and Ground to respective pins in RPi.

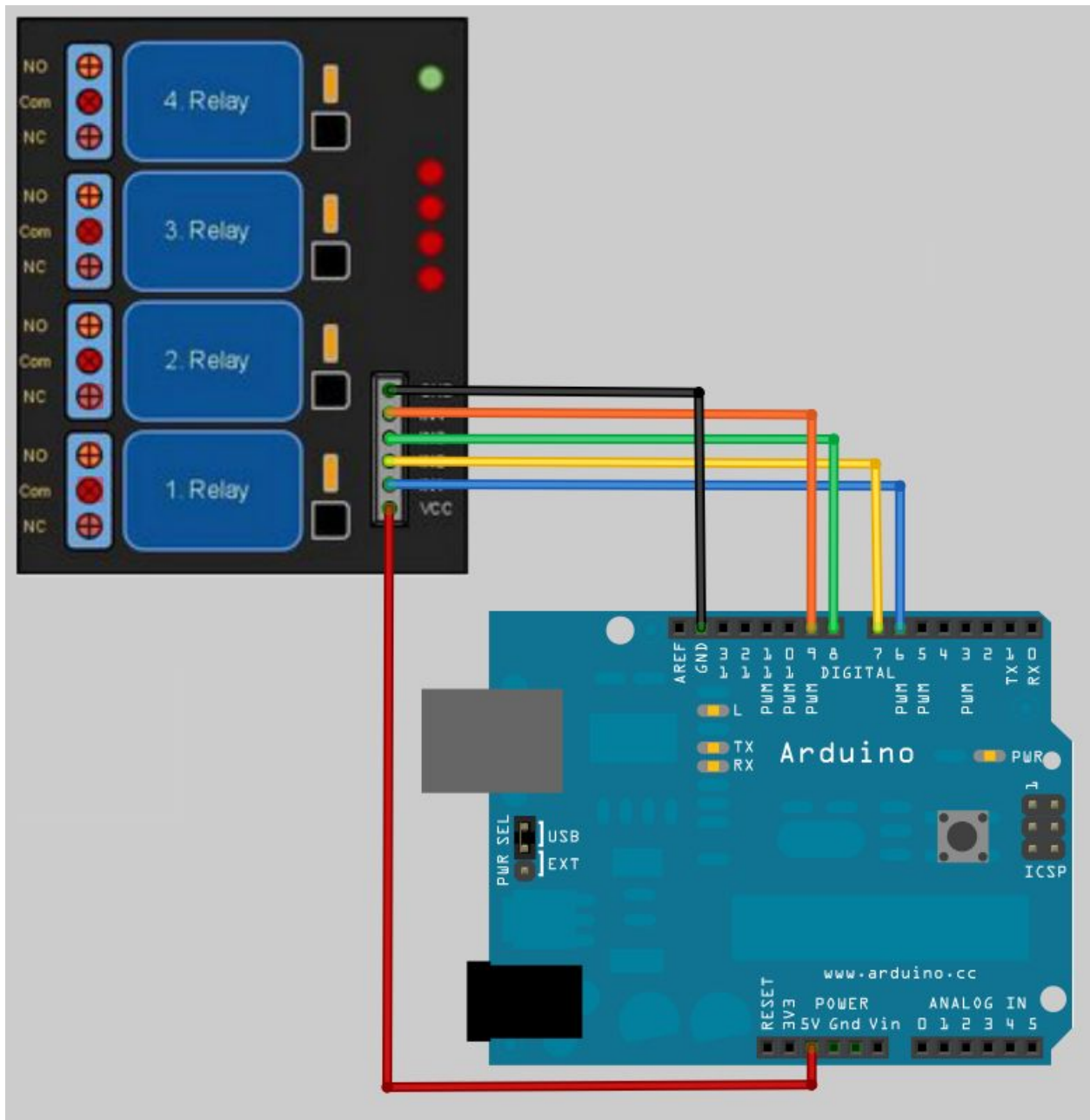


4. DC motor :

The + and - pins are connected to the RaspBerry Pi's GPIO out pins as to rotate the motor clockwise or anticlockwise.

5 . Relay:

The input pins of the relay are connected to the Arduino. The two DC motors are connected to the relay output and two 9V batteries are connected to high voltage input for the relay.



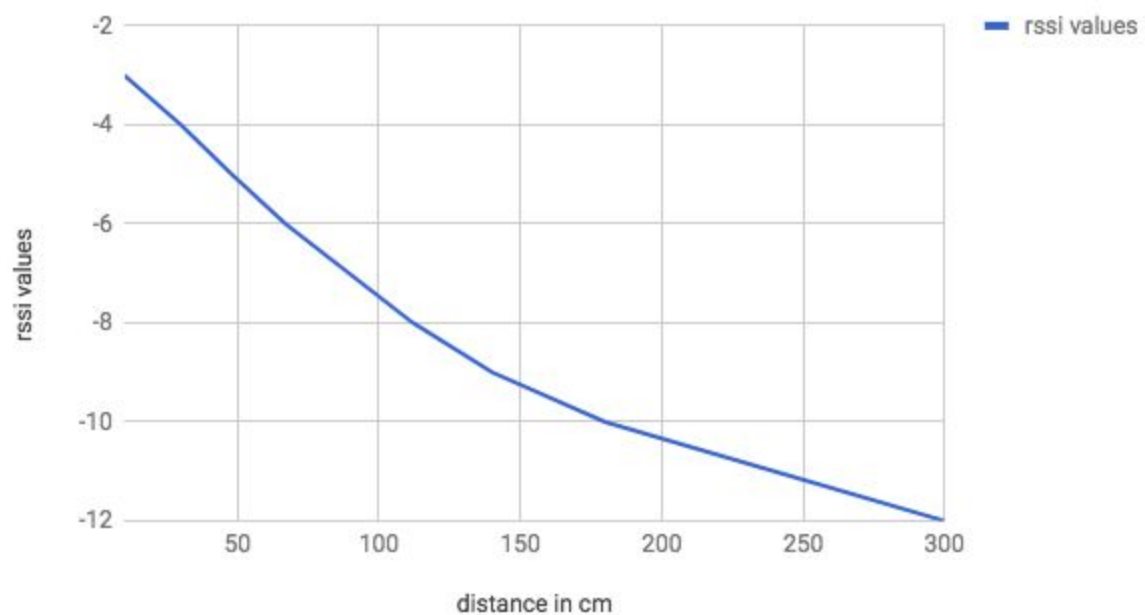
6. 4 IR sensors:

Theses sensors were connected to arduino to detect black tapes stuck on floor.

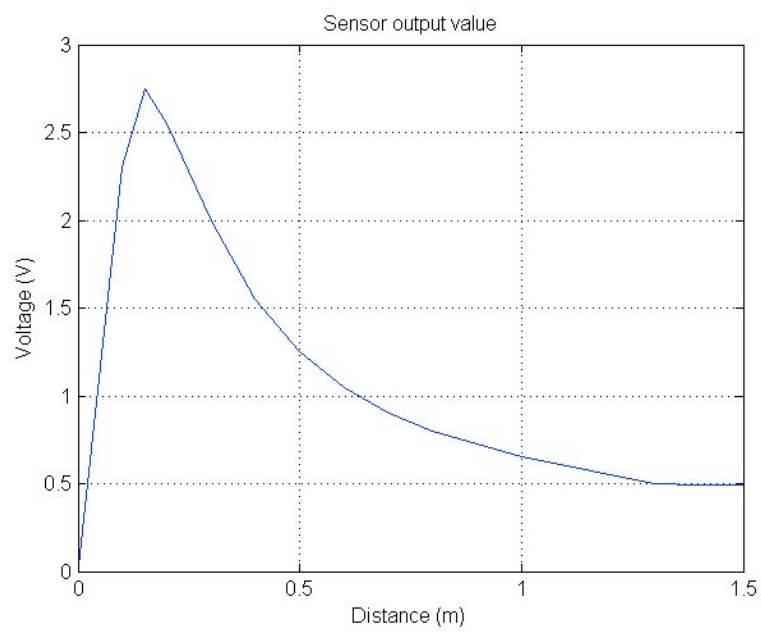
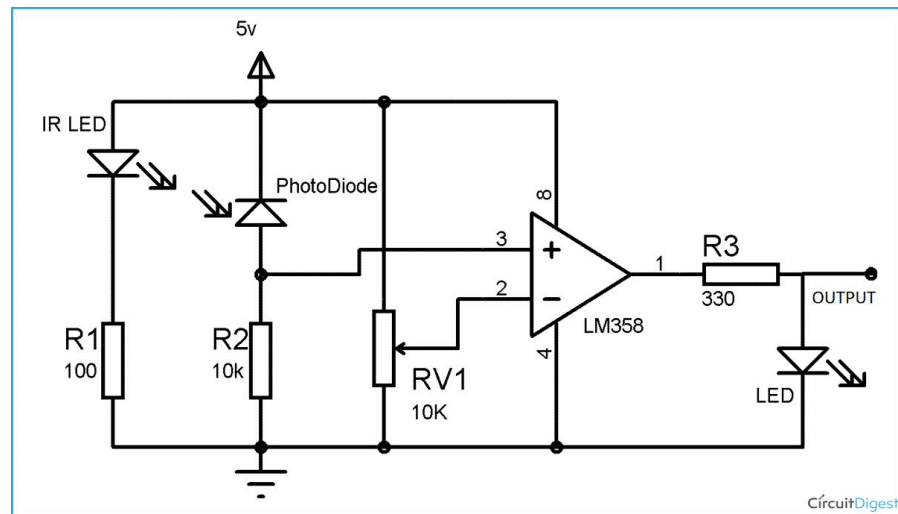
RSSI Values vs. Distance

Distance (cm)	RSSI Value
10	-3
30	-4
48	-5
67	-6
112	-8
140	-9
180	-10
240	-11
300	-12

rssl values vs. distance in cm



IR Sensor circuit diagram and its Characteristics Curve



Salient Features

Locomotion

- Whenever the trash can's bluetooth module is paired with the user's phone the trash can is summoned and locomotion phase of the trash can starts.
- The trash can follows black lined path and pauses at a few predefined nodes.
- It takes the RSSI value readings at these nodes, and using these values, is able to approximate where the user is, relative to itself. It then directly goes to that location, deciding where to turn whenever faced with a choice intelligently.
- While taking the readings at the predefined nodes, if the trash can's bluetooth module detects an RSSI value greater lesser than -4 it stops to collect the garbage, since the user is closest to that particular node.
- The trash can is accurate to one node within the user's range.
- After the disposal of garbage the user can switch off its bluetooth connection and the trash-can returns to its home node.

Trash can

- Whenever the trash can is summoned and reaches the destination it opens the can.
- The trash can stays at its destination till the user switches off the bluetooth.
- When the can is closed the trash can checks whether the can is full or not, if so the buzzer starts alarming the user to dump the trash.

Timeline

The duration of this project was effectively 3 weeks after a series of discussion with professors and teaching assistants.

Given below is detailed explanation of work done during weeks 1, 2 and 3.

Week 1

- Initial idea for the project was to make trash can navigate based on clap given by the user. The idea behind this clap navigation was based on trilateration done by three microphones.
- The problem with the idea was that the signals from microphone was not getting sampled properly and signals detected were mostly noises from unknown sources.
- Prof . Pradeep Kumar Das was very much interested and helpful over this idea and assisted us to experiment on the microphones.
- Parallely the structure of the trash can was developed and initial code was written so as to make record the data from the sensors properly.
- By the end of first week, the opening and closing functions based on user proximity to the trash-can and capacity detector of the trash can were functioning properly.

Week 2

- Halfway into the week, the progress on microphones did not match with our deadline for the project submission. Thus, the idea of microphones was dropped and alternatives were explored until we resorted to using bluetooth for locating the user.
- The idea was now that the trash can would navigate to the user by measuring the intensity of the bluetooth signal at various nodes along a circuit laid out on the floor.
- The week saw the bluetooth module being interfaced properly and the locomotive part being completed along with the functioning line following unit.
- The locomotive part comprised of a castor wheel and two wheels connected to one dc motor each. The dc motors were powered by 9V supply through a relay.
- The relay was controlled by an Arduino, which was responsible for the movement of the wheels. The entire algorithm for line following was implemented in the Arduino. The algorithm in the Arduino allowed for movement along black lines and decision-making at nodes.

- Thus, at nodes, the Arduino waited for some input, and based on that input, it either moved forward, took a left turn, a right turn, or a U-turn. This input would be given by the RaspBerry Pi.

Week 3 (Final Submission)

- This week saw the completion of user localization using the bluetooth signals, redesigning the line following module as advised by the TA, making it more robust, and setting up the master slave connections between RaspBerry Pi and the Arduino.
- The RaspBerry Pi now could control the locomotion of the trash-can based on the user location that it computed, and function as the 'brain' to the trash-can, while the Arduino functioned like the 'legs' of the trash-can.
- The trash-can was fixed atop the locomotion module, and rewiring was done to ensure stability of the circuits, buzzer was added to notify full status.
- The sensors were tested during operation and rectified if errors were found during the testing phase. At this point, the ultrasonic sensor started giving catastrophic readings, but due to lack of time, the issue could not be resolved and the proximity based opening and closing was removed in favour of opening on reaching the user, and closing when the user stops the bluetooth signal transmission.

Problems Faced

- Ultrasonic and PIR sensors even though working individually, started behaving abnormally when attached to the rest of the circuit, thus, dropped from the project due to lack of time. Cause might be lack of current to drive the sensors from RaspBerry Pi.
- The Sound Localisation turned out to be a wild goose chase when we failed to get any meaningful output from either the headphone mics (tried in the first week), or the condenser mics (tried in the second week). We ended up wasting a substantial amount of our time, with no visible output. Cause still unknown.
- The DC motors did not rotate equally given the same power supply. To control the speed given a relay, elaborate circuits were required, and the components were not readily available, so resistors were used to smoothen out the motion by giving lesser power to the motor which rotated faster than the other, thereby controlling the extent of drift and stabilising the motion in the most time and cost effective manner.
- The IR sensors were tuned by varying their resistance (using the potentiometer screw provided on the sensor), so as to meet the expectancy.

Alternative Uses

- Offices : One can use the model above to send documents to individual desks in offices. The desks may have their own bluetooth connection to the robot that will allow it to carry the documents to the right user. Model can then be configured to be locked until it reaches the destination, so that confidentiality of crucial documents is maintained.
- Warehouses : The Robot can be used to manage a warehouse. An army of these robots can be used to completely automate the warehouse when a scanning functionality is added like a barcode/RFID reader which allows it to differentiate between various items, thereby keeping track of their whereabouts in the warehouse. They can carry these items in warehouse to from storage location to counter when counters and storage areas are given appropriate bluetooth capabilities.
- Restaurants : This can be used carry food items to deliver orders to respective tables in a restaurant. The tables might've their own bluetooth signatures allowing the restaurant manager to deliver food to appropriate tables.

- The robot can be used as it is at places like a garage to take items to and from between working stations, libraries to send books to their particular shelves etc. This bot can be used for any sort of inventory management.

Code Snippets:

Arduino code for locomotion

```
#include <Ultrasonic.h>
#define lc1p 3
#define lc1n 2
#define lc2p 5
#define lc2n 4
#define leftend 6
#define leftmid 7
#define rightmid 8
#define rightend 9
#define isatnode 10
#define piin1 11
#define piin2 12
#define piin3 13
#define Black 1
#define White 0
Ultrasonic ultrasonic(50, 42);
bool state;
void setup() {
    // put your setup code here, to run once:
    pinMode(lc1p, OUTPUT);
    pinMode(lc1n, OUTPUT);
    pinMode(lc2p, OUTPUT);
    pinMode(lc2n, OUTPUT);
    pinMode(leftend, INPUT);
    pinMode(leftmid, INPUT);
    pinMode(rightmid, INPUT);
    pinMode(rightend, INPUT);
    pinMode(piin1, INPUT);
    pinMode(piin2, INPUT);
    pinMode(isatnode, OUTPUT);
    state = 0;
    digitalWrite(isatnode, LOW);
    Serial.begin(9600);
    int le = digitalRead(leftend);
    int re = digitalRead(rightend);
    while(le!=Black || re!=Black)
    {
        delay(100);
        le = digitalRead(leftend);
        re = digitalRead(rightend);
    }
}
```

```

digitalWrite(isatnode,HIGH);
}

// A busy wait if RaspBery Pi gives a signal of 00X
void test_and_wait()
{
    while(digitalRead(piin1)==0 && digitalRead(piin2)==0);
}

//stops the motors
void reset()
{
    digitalWrite(lc1p,LOW);
    digitalWrite(lc1n,LOW);
    digitalWrite(lc2p,LOW);
    digitalWrite(lc2p,LOW);
    //delay(100);
}

//code to move forward on the black line taking input from the Grove Line Sensor from the variables le, lm,
void move_forward(int x)
{
    reset();
    test_and_wait();
    int prev=0;
    digitalWrite(isatnode,LOW);
    //read input from the sensors
    int le = digitalRead(leftend);
    int lm = digitalRead(leftmid);
    int rm = digitalRead(rightmid);
    int re = digitalRead(rightend);

    //if detected a node
    while(le==Black && re == Black)
    {
        reset();
        // while(ultrasonic.distanceRead(<10)
        // {
        //     delay(10);
        // }
        test_and_wait();
        // if(le == Black && re == Black)

```

```
if(lm==Black && rm==Black)
{
    prev=0;
}
if(lm==Black && rm==White)
{
    prev=-1;
}
if(lm==White && rm==Black)
{
    prev=1;
}
if(prev==1)
{
    digitalWrite(lc2p,HIGH);
}
if(prev==1)
{
    digitalWrite(lc1p,HIGH);
}
if(prev==0)
{
    digitalWrite(lc1p,HIGH);
    digitalWrite(lc2p,HIGH);
}
//put delay to get the motors going before changing the signal
delay(100);
//out_of_path();
le = digitalRead(leftend);
lm = digitalRead(leftmid);
rm = digitalRead(rightmid);
re = digitalRead(rightend);
}
if(x==0)
{
    if(le==White)
        prev=-1;
    else if(re==White)
        prev=1;
    else
        prev=0;
}
```

```

}
else
  prev = x;
while(le!=Black || re!=Black || lm!=Black || rm!=Black)
{
  reset();
  // while(ultrasonic.distanceRead()<10)
  // {
  //   delay(10);
  // }
  test_and_wait();
  if(lm==Black && rm==Black)
  {
    prev=0;
  }
  if(lm==Black && rm==White)
  {
    prev=-1;
  }
  if(lm==White && rm==Black)
  {
    prev=1;
  }
  if(prev==1)
  {
    digitalWrite(lc2p,HIGH);
  }
  if(prev==1)
  {
    digitalWrite(lc1p,HIGH);
  }
  if(prev==0)
  {
    digitalWrite(lc1p,HIGH);
    digitalWrite(lc2p,HIGH);
  }
  //delay(100);
  //out_of_path();
  le = digitalRead(leftend);
  lm = digitalRead(leftmid);
  rm = digitalRead(rightmid);
}

```

```

    re = digitalRead(rightend);
}
reset();
digitalWrite(isatnode,HIGH);
delay(100);
}
//code to turn left on a node
void turn_left()
{
    reset();
    int le = digitalRead(leftend);
    int re = digitalRead(rightend);
    test_and_wait();
    //A left turn will see black then move to a white area for the right sensor and then again to black area,
    while(re == Black)
    {
        reset();
        test_and_wait();
        digitalWrite(lc2p,HIGH);//turn the right motor
        delay(100);
        re = digitalRead(rightend);
    }
    while(re == White)
    {
        reset();
        test_and_wait();
        digitalWrite(lc2p,HIGH);//turn the right motor
        delay(100);
        re = digitalRead(rightend);
    }
    move_forward(-1);
}
void turn_right()
{
    reset();
    int le = digitalRead(leftend);
    int re = digitalRead(rightend);
    test_and_wait();
    //A right turn will see black then move to a white area for the left sensor and then again to black area,
    while(le == Black)
    {

```

```

    reset();
    test_and_wait();
    digitalWrite(lc1p,HIGH);//turn the left motor
    delay(100);
    le = digitalRead(leftend);
}
while(le == White)
{
    reset();
    test_and_wait();
    digitalWrite(lc1p,HIGH);//turn the left motor
    delay(100);
    le = digitalRead(leftend);
}

move_forward(1);
}
//function to take a U-turn
void go_back()
{
    reset();
    int le = digitalRead(leftend);
    int re = digitalRead(rightend);
    test_and_wait();
    while(re == Black)
    {
        reset();
        test_and_wait();
        digitalWrite(lc2p,HIGH);
        delay(100);
        re = digitalRead(rightend);
    }
    while(re == White)
    {
        reset();
        test_and_wait();
        digitalWrite(lc2p,HIGH);
        delay(100);
        re = digitalRead(rightend);
    }
}

```

```

while(re == Black)
{
    reset();
    test_and_wait();
    digitalWrite(lc2p,HIGH);
    delay(100);
    re = digitalRead(rightend);
}
while(re == White)
{
    reset();
    test_and_wait();
    digitalWrite(lc2p,HIGH);
    delay(100);
    re = digitalRead(rightend);
}
while(re == Black)
{
    reset();
    test_and_wait();
    digitalWrite(lc2p,HIGH);
    delay(100);
    re = digitalRead(rightend);
}
move_forward(-1);
}

//this is the main function for the locomotion of motor
void move()
{
    reset();
    int pi1 = digitalRead(piin1);
    int pi2 = digitalRead(piin2);
    if(pi1==0 && pi2==0) delay(500);
    if(pi1==1 && pi2==0) turn_left();
    if(pi1==0 && pi2==1) turn_right();
    if(pi1==1 && pi2==1)
    {
        if(digitalRead(piin3)==LOW) move_forward(0);
        else go_back();
    }
}

```

```

}

//checks whether the motor connections are working
void test_motor()
{
    digitalWrite(lc1p,LOW);
    digitalWrite(lc1n,HIGH);
    digitalWrite(lc2p,HIGH);
    digitalWrite(lc2n,LOW);
}

void loop() {
    // put your main code here, to run repeatedly:
    move();
    //test_motor();
}

```


Main python script for running the trash can

```

from __future__ import print_function
import os,sys
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

log = open('/home/pi/Desktop/log.txt','a')
print('\n\n\n\n new entry \n\n\n',file = log)

#dec
ultr_trig = 18
ultr_echo = 24
ir_res = 25
moto_plus = 16
moto_minus = 19
moto_plus2 = 22
moto_minus2 = 23
loco1_plus = 17
loco1_minus = 27
loco2_plus = 5
loco2_minus = 6
buzzer = 4
pir = 9
flag = 0
piout1 = 13
piout2 = 26
piout3 = 20
piin = 21
chanlist = (piout1,piout2,piout3)

#setup
GPIO.setup(moto_plus,GPIO.OUT)
GPIO.setup(moto_minus,GPIO.OUT)
GPIO.setup(ir_res,GPIO.IN)
GPIO.setup(buzzer,GPIO.OUT)
GPIO.setup(pir,GPIO.IN)

GPIO.setup(ultr_trig, GPIO.OUT)
GPIO.setup(ultr_echo, GPIO.IN)
GPIO.setup(moto_minus2, GPIO.OUT)
GPIO.setup(moto_plus2, GPIO.OUT)

GPIO.setup(loco1_minus,GPIO.OUT)
GPIO.setup(loco1_plus,GPIO.OUT)
GPIO.setup(loco2_plus,GPIO.OUT)
GPIO.setup(loco2_minus,GPIO.OUT)

GPIO.setup(piout1,GPIO.OUT)
GPIO.setup(piout2,GPIO.OUT)
GPIO.setup(piout3,GPIO.OUT)
GPIO.setup(piin,GPIO.IN)

#graph declaration
curnode = 1

```

```

file.py - /home/pi/Desktop/file.py (2.7.9)
File Edit Format Run Options Windows Help

currentnode = 1
intensity = [0,0,0,0]
line = 0
finalnode = 0

def distance():
    GPIO.output(ultr_trig, False)
    time.sleep(1)

    GPIO.output(ultr_trig, True)
    time.sleep(0.00001)
    GPIO.output(ultr_trig, False)

    StartTime = time.time()
    StopTime = time.time()
    TTime = time.time()
    while GPIO.input(ultr_echo) == 0:
        #print("asr")
        StartTime = time.time()
        if StartTime - TTime >= 1:
            return distance()

    TTime = time.time()
    while GPIO.input(ultr_echo) == 1:
        StopTime = time.time()
        if StopTime - TTime >= 1:
            return distance()

    TimeElapsed = StopTime - StartTime
    dist = (TimeElapsed * 34300) / 2
    print(dist)
    return dist

def open_car_dist():
    stop_here()
    GPIO.output(moto_plus, GPIO.HIGH)
    GPIO.output(moto_minus, GPIO.LOW)
    time.sleep(10)
    GPIO.output(moto_plus, GPIO.LOW)

    while (distance() + distance()) <= 20:
        print("Car Open Dist")

    GPIO.output(moto_minus, GPIO.HIGH)
    time.sleep(8)
    GPIO.output(moto_minus, GPIO.LOW)
    return

def open_car():
    GPIO.output(moto_plus, GPIO.HIGH)
    GPIO.output(moto_minus, GPIO.LOW)
    time.sleep(10)
    GPIO.output(moto_plus, GPIO.LOW)

```



```

while (get_intensity() != 50):
    print ("Can Open")

GPIO.output(moto_minus, GPIO.HIGH)
time.sleep(8)
GPIO.output(moto_minus, GPIO.LOW)
return

def check_full():
    GPIO.output(buzzer, GPIO.LOW)
    while GPIO.input(ir_res) == 1:
        print ("Can FULL")
        GPIO.output(buzzer, GPIO.HIGH)
        time.sleep(0.1)
        GPIO.output(buzzer, GPIO.LOW)
        time.sleep(0.1)
        GPIO.output(buzzer, GPIO.HIGH)
        time.sleep(0.1)
        GPIO.output(buzzer, GPIO.LOW)
        time.sleep(0.7)
    return

def check_proximity():
    #print('checking proximity\n', file = log)
    global curnode
    print ("Checking Proximity");
    print (curnode)

    #if (distance() + distance()) <= 20:
    #    open_can_dist()

    #put check for proximity and handle

def stop_here():
    print('stopped\n', file = log)
    print('Stopping')
    GPIO.output(chanlist, (GPIO.LOW, GPIO.LOW, GPIO.LOW))

def move_forward():
    print('moving forward\n', file = log)
    while GPIO.input(piin) == 0:
        continue

    while GPIO.input(piin) == 1:
        print('Moving Forward')
        check_proximity()
        GPIO.output(chanlist, (GPIO.HIGH, GPIO.HIGH, GPIO.LOW))

    while GPIO.input(piin) == 0:
        check_proximity()
        GPIO.output(chanlist, (GPIO.HIGH, GPIO.HIGH, GPIO.LOW))

    stop_here()

def move_left():
    print('moving left\n', file = log)

```

```

print('moving left\n', file = log)
while GPIO.input(piin) == 0:
    continue

while GPIO.input(piin) == 1 :
    check_proximity()
    GPIO.output(chanlist, (GPIO.HIGH, GPIO.LOW, GPIO.LOW))

while GPIO.input(piin) == 0 :
    check_proximity()
    GPIO.output(chanlist, (GPIO.HIGH, GPIO.LOW, GPIO.LOW))

stop_here()

def move_right():
    print('moving right\n', file = log)
    while GPIO.input(piin) == 0:
        continue

    while GPIO.input(piin) == 1 :
        check_proximity()
        GPIO.output(chanlist, (GPIO.LOW, GPIO.HIGH, GPIO.LOW))

    while GPIO.input(piin) == 0 :
        check_proximity()
        GPIO.output(chanlist, (GPIO.LOW, GPIO.HIGH, GPIO.LOW))

    stop_here()

def move_back():
    print('moving back\n', file = log)
    while GPIO.input(piin) == 0:
        continue

    while GPIO.input(piin) == 1 :
        check_proximity()
        GPIO.output(chanlist, (GPIO.HIGH, GPIO.HIGH, GPIO.HIGH))

    while GPIO.input(piin) == 0 :
        check_proximity()
        GPIO.output(chanlist, (GPIO.HIGH, GPIO.HIGH, GPIO.HIGH))

    stop_here()

def get_rssi():
    print('getting rssi\n', file = log)
    try:
        f = open('rssi_value.txt')
        sum = 0
        for line in f:
            sum = sum + int(line)

        f.close()
        os.remove("rssi_value.txt")
    except:
        sum=-500
    os.remove("/home/pi/Desktop/top")
    return -float(sum)/10

```



```

return - (float(sum)/10)

def start_move():
    print('starting to move\n', file = log)
    global line
    global curnode
    global finalnode
    global intensity
    line = 0
    while 1:
        rssi = get_intensity()
        if rssi == 50.0 :
            return
        if line == 0 :
            if rssi < 4.0 :
                print ("less than 4\n")
                #stop_here()
                #while get_intensity() != 50 :
                #    check_proximity()
                return
            else :
                intensity[curnode-1] = rssi
                curnode = curnode + 1
                if curnode != 5:
                    move_forward()
                else:
                    minnode = 0
                    intval = 100
                    for i in range(1,5):
                        if intensity[i-1] < intval :
                            intval = intensity[i-1]
                            minnode = i

                    finalnode = 9 - minnode
                    if intval >= 8.0 :
                        finalnode = finalnode + 5
                    move_left()
                    curnode = 5
                    line = 1
                    break
        while curnode != finalnode :
            rssi = get_intensity()
            if rssi == 50.0 :
                return
            if line == 1 :
                if finalnode - curnode < 5 :
                    move_left()
                    line = curnode/5 + 1
                    curnode = curnode + 1
                else :
                    move_forward()
                    curnode = curnode + 5
            continue
        else :
            move_forward()
            curnode = curnode + 1
    #while get_intensity() != 50:
    #    check_proximity()

```

```

return (float(sum)/10)

def start_move():
    print('starting to move\n',file = log)
    global line
    global curnode
    global finalnode
    global intensity
    line = 0
    while 1:
        rssi = get_intensity()
        if rssi == 50.0 :
            return
        if line == 0 :
            if rssi < 4.0 :
                print ("less than 4\n")
                #stop_here()
                #while get_intensity() != 50 :
                #    check_proximity()
                return
            else :
                intensity[curnode-1] = rssi
                curnode = curnode + 1
                if curnode != 5:
                    move_forward()
                else:
                    minnode = 0
                    intval = 100
                    for i in range(1,5):
                        if intensity[i-1]<intval :
                            intval = intensity[i-1]
                            minnode = i

                    finalnode = 9 - minnode
                    if intval >= 8.0 :
                        finalnode = finalnode + 5
                    move_left()
                    curnode = 5
                    line = 1
                    break
        while curnode != finalnode :
            rssi = get_intensity()
            if rssi == 50.0 :
                return
            if line == 1 :
                if finalnode - curnode < 5 :
                    move_left()
                    line = curnode/5 + 1
                    curnode = curnode + 1
                else :
                    move_forward()
                    curnode = curnode + 5
                continue
            else :
                move_forward()
                curnode = curnode + 1
        #while get_intensity() != 50:

```

```

    return

def go_back():
    print('go back called\n',file = log)
    print("go back called")
    global curnode
    while curnode < 4 :
        move_forward()
        curnode = curnode + 1
    if curnode == 4 :
        start_rev()
        return
    while curnode < 9 :
        if curnode == 5 :
            move_left()
        else :
            move_forward()
            curnode = curnode + 1
    if curnode == 9 :
        start_rev()
        return
    while curnode < 14 :
        if curnode == 10 :
            move_left()
        else :
            move_forward()
            curnode = curnode + 1
    start_rev()
    return

command = '~/Desktop/bt.sh >> /home/pi/Desktop/tnp'
def get_intensity():
    print('getting intensity\n',file = log)
    for i in range (0,10):
        os.system(command)
        time.sleep(0.2)
    rssi=get_rssi()
    print (rssi)
    return rssi

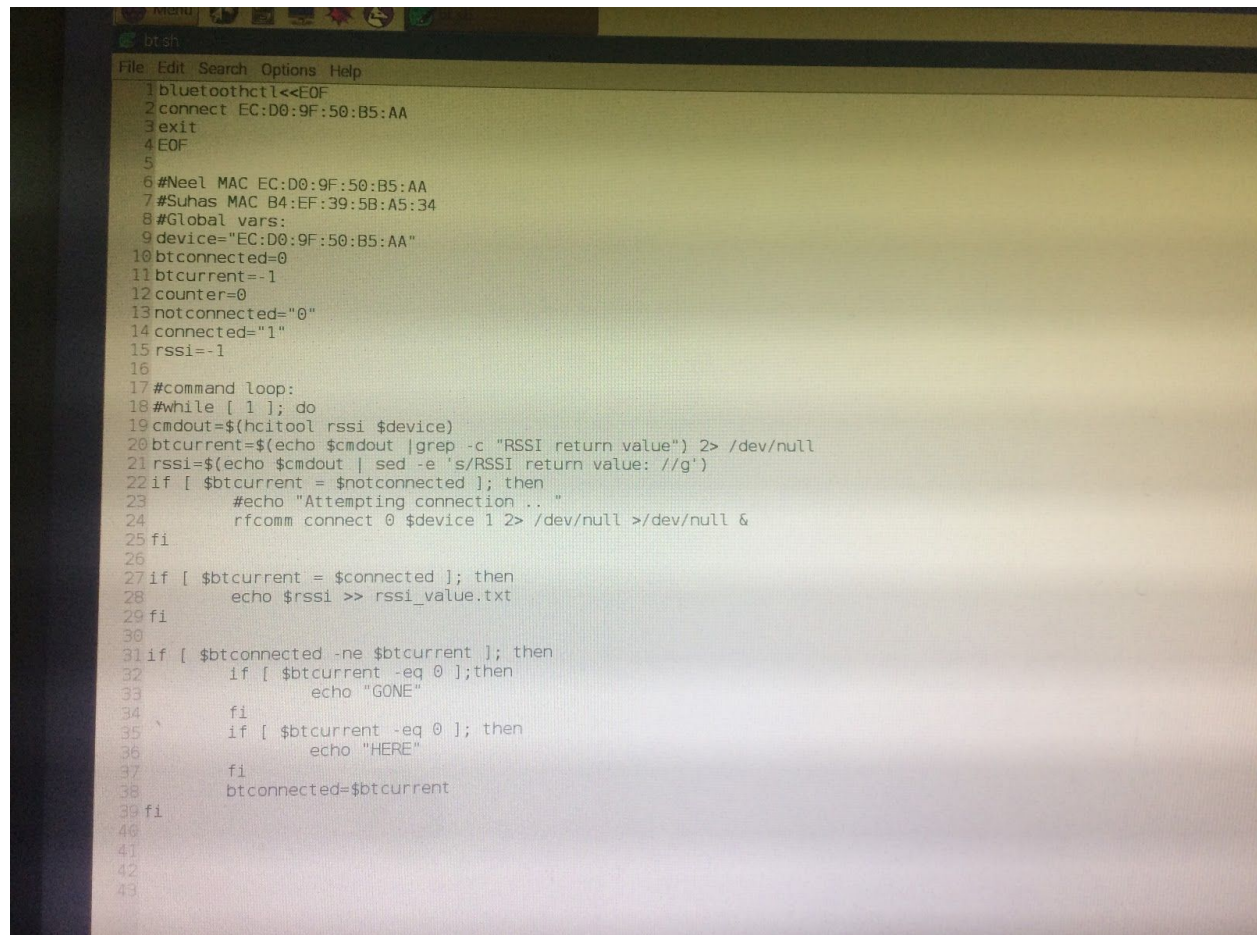
while 1:

    stop_here()
    check_full()
    time.sleep(0.5)
    check_proximity()
    rssi = get_intensity()
    if rssi == 50.0:
        continue

    start_move()
    open_can()
    go_back()

```


Bluetooth scanner shell script:



```
1 bluetoothctl<<EOF
2 connect EC:D0:9F:50:B5:AA
3 exit
4 EOF
5
6 #Neel MAC EC:D0:9F:50:B5:AA
7 #Suhas MAC B4:EF:39:5B:A5:34
8 #Global vars:
9 device="EC:D0:9F:50:B5:AA"
10 btconnected=0
11 btcurrent=-1
12 counter=0
13 notconnected="0"
14 connected="1"
15 rssi=-1
16
17 #command loop:
18 #while { 1 }; do
19 cmdout=$(hcitool rssi $device)
20 btcurrent=$(echo $cmdout | grep -c "RSSI return value") 2> /dev/null
21 rssi=$(echo $cmdout | sed -e 's/RSSI return value: //g')
22 if [ $btcurrent = $notconnected ]; then
23     #echo "Attempting connection .."
24     rfcomm connect 0 $device 1 2> /dev/null >/dev/null &
25 fi
26
27 if [ $btcurrent = $connected ]; then
28     echo $rssi >> rssi_value.txt
29 fi
30
31 if [ $btconnected -ne $btcurrent ]; then
32     if [ $btcurrent -eq 0 ]; then
33         echo "GONE"
34     fi
35     if [ $btcurrent -eq 0 ]; then
36         echo "HERE"
37     fi
38     btconnected=$btcurrent
39 fi
40
41
42
43
```

Bluetooth energy scan shell script named as bt.sh

```

1 #!/bin/bash
2
3
4 if [[ $1 == "parse" ]]; then
5     packet=""
6     capturing=""
7     count=0
8     while read line
9     do
10         count=$((count + 1))
11         if [ "$capturing" ]; then
12             if [[ $line =~ ^[0-9a-fA-F]{2}\ [0-9a-fA-F] ]]; then
13                 packet="$packet $line"
14             else
15                 if [[ $packet =~ ^04\ 3E\ 2A\ 02\ 01\ ..{26}\ 02\ 01\ ..{14}\ 02\ 15 ]]; then
16                     UUID=$(echo $packet | sed 's/^.\{69\}\(.{47}\).*$/\1/')
17                     MAJOR=$(echo $packet | sed 's/^.\{117\}\(.{5}\).*$/\1/')
18                     MINOR=$(echo $packet | sed 's/^.\{123\}\(.{5}\).*$/\1/')
19                     POWER=$(echo $packet | sed 's/^.\{129\}\(.{2}\).*$/\1/')
20                     UUID=$(echo $UUID | sed -e 's/\ //g' -e 's/\(.{8}\)\(.{4}\)\(.{4}\)\(.{4}\)\(.{12}\)/\1-\2-\3-\4-\5/')
21                     MAJOR=$(echo $MAJOR | sed 's/\ //g')
22                     MINOR=$(echo $MINOR | sed 's/\ //g')
23                     MINOR=$(echo $MINOR | sed 's/\ //g')
24                     MINOR=$(echo $MINOR | sed 's/\ //g')
25                     POWER=$(echo $POWER | sed 's/\ //g')
26                     POWER=$((POWER - 256))
27                     if [[ $2 == "-b" ]]; then
28                         echo "$UUID $MAJOR $MINOR $POWER"
29                     else
30                         echo "UUID: $UUID MAJOR: $MAJOR MINOR: $MINOR POWER: $POWER"
31                     fi
32                 fi
33                 capturing=""
34                 packet=""
35             fi
36         fi
37     done
38 else
39     if [ ! "$capturing" ]; then
40         if [[ $line =~ ^> ]]; then
41             packet=$(echo $line | sed 's/^>.\{*\}/\1/')
42             capturing=1
43         fi
44     fi
45     sudo hcitool lescan --duplicates 1>/dev/null &
46     # I added this line to the script

```



```

6  capturing=""
7  count=0
8  while read line
9  do
10 count=$((count + 1))
11 if [ "$capturing" ]; then
12   if [[ $line =~ ^[0-9a-fA-F]{2}\ [0-9a-fA-F] ]]; then
13     packet="$packet $line"
14   else
15     if [[ $packet =~ ^04\ 3E\ 2A\ 02\ 01\ .{26}\ 02\ 01\ .{14}\ 02\ 15 ]]; then
16       UUID=`echo $packet | sed 's/^.\{69\}\(. \{47\}\)\.*/\1/'`
17       MAJOR=`echo $packet | sed 's/^.\{117\}\(. \{5\}\)\.*/\1/'`
18       MINOR=`echo $packet | sed 's/^.\{123\}\(. \{5\}\)\.*/\1/'`
19       POWER=`echo $packet | sed 's/^.\{129\}\(. \{2\}\)\.*/\1/'`
20       UUID=`echo $UUID | sed -e 's/\ //g' -e 's/^(\.[8\])\(\.[4\])\(\.[4\])\(\.[4\])\(\.[12\])\)/\1-\2-\3-\4-\5/'`
21       MAJOR=`echo $MAJOR | sed 's/\ //g'`
22       MINOR=`echo $MINOR | sed 's/\ //g'`
23       MINOR=`echo "ibase=16; $MINOR" | bc`
24       POWER=`echo "ibase=16; $POWER" | bc`
25       POWER=$((POWER - 256))
26       if [[ $2 == "-b" ]]; then
27         echo "$UUID $MAJOR $MINOR $POWER"
28       else
29         echo "UUID: $UUID MAJOR: $MAJOR MINOR: $MINOR POWER: $POWER"
30       fi
31     fi
32   fi
33   capturing=""
34   packet=""
35 fi
36 fi
37
38 if [ ! "$capturing" ]; then
39   if [[ $line =~ ^> ]]; then
40     packet=`echo $line | sed 's/^>.\(.*\)/\1/'`
41     capturing=1
42   fi
43 fi
44 done
45 else
46   sudo hcitool lescan --duplicates 1>/dev/null &
47   if [ "$(pidof hcitool)" ]; then
48     sudo hcidump --raw | ./$0 parse $1
49   fi
50 fi
51

```