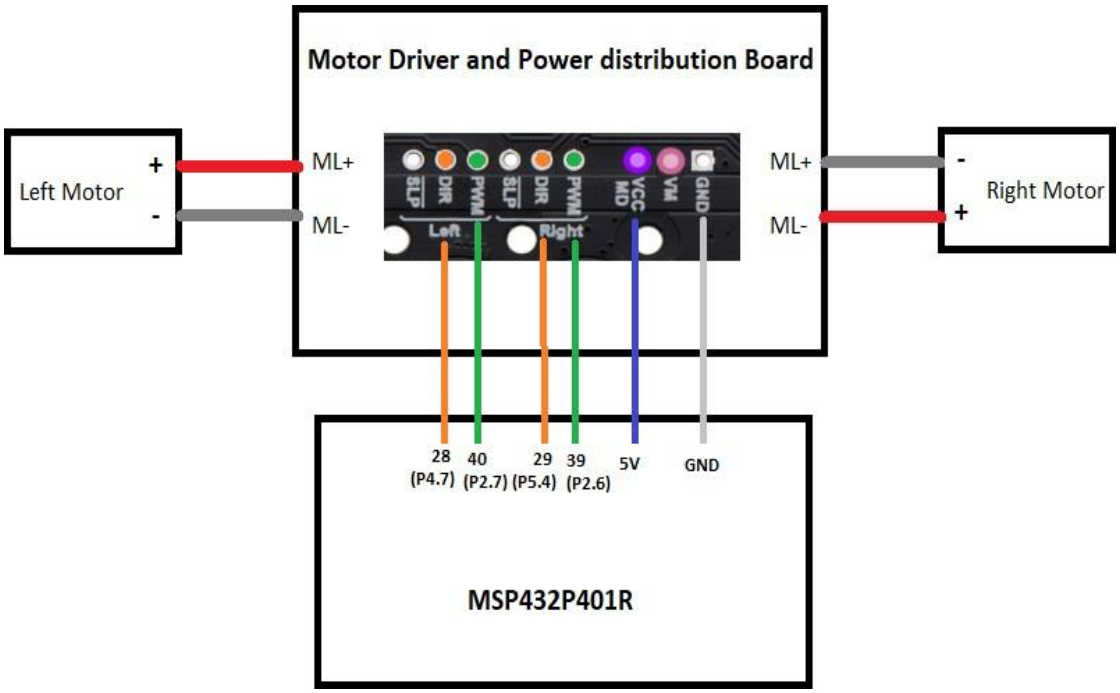
Expt.1 RSLK Motion Control with Energia and Delay Implementation.

```
#define LS 40    //Left motor
#define RS 39    //Right motor
#define LD 28    //Left motor direction pin
#define RD 29    //Right motor direction pin
int speedr = 0,speedl = 0; //sets the speed of motor using PWM
int ldir = 0,rdir = 0;  // 0□ forward // 1□ backward
void setup() {
Serial.begin(9600);
Serial1.begin(9600);
pinMode(LD,OUTPUT);
pinMode(RD,OUTPUT);
}
void loop() {
ldir = rdir = 0;  //Forward Movement
speedr = 50;
speedl = 50;
digitalWrite(LD, ldir);
digitalWrite(RD, rdir);
analogWrite(LS, speedl);
analogWrite(RS, speedr);
delay(5000);
ldir = rdir = 1;  //Backward Movement
speedr = 50;
speedl = 50;
digitalWrite(LD, ldir);
digitalWrite(RD, rdir);
analogWrite(LS, speedl);
analogWrite(RS, speedr);
delay(5000);
}
```


Motor Driver and Power distribution Board
Left Motor
+
-
ML+
ML-
SLP
DIR
PWM
SLP
DIR
PWM
VCC
MD
VM
GND
Left
Right
ML+
ML-
-
+
Right Motor
28
(P4.7)
40
(P2.7)
29
(P5.4)
39
(P2.6)
5V
GND
MSP432P401R

Expt. 2 BLUETOOTH AND VOICE-CONTROLLED NAVIGATION OF RSLK

2.1 BLUETOOTH CONTROLLED NAVIGATION OF RSLK

```
#define LS 40    //Left motor
#define RS 39    //Right motor
#define LD 28    //Left motor direction pin
#define RD 29    //Right motor direction pin
int speedr = 0,speedl = 0; //sets the speed of motor using PWM
int ldir = 0,rdir = 0;  // 0□ forward // 1□ backward
void setup() {
Serial.begin(9600);
Serial1.begin(9600);
pinMode(LD,OUTPUT);
pinMode(RD,OUTPUT);
}
void loop() {
char c = 0;
  // put your main code here, to run repeatedly:
if(Serial1.available()){
    c = Serial1.read();
Serial.println(c);
  }
switch(c)
  {
case 'F' :ldir = rdir = 0;  //Forward Movement
speedr = 50;
speedl = 50;
break;
case 'B' :ldir = rdir = 1;  //Backward Movement
speedr = 50;
speedl = 50;
break;
```

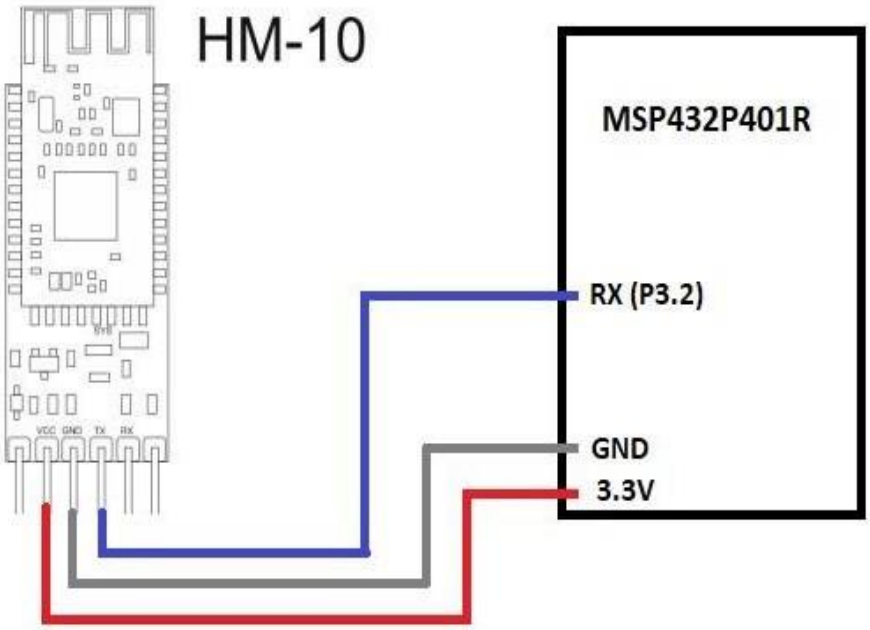
```
case 'R' :ldir = 0;  //Move towards left
speedl = 0;
speedr = 50;
break;
case 'L' :rdir = 0;  //Move towards right
speedl =50;
speedr = 0;
break;
  }
digitalWrite(LD, ldir);
digitalWrite(RD, rdir);
analogWrite(LS, speedl);
analogWrite(RS, speedr);
}
```

2.2 VOICE CONTROLLED NAVIGATION OF RSLK

```
#include<String.h>
//Predefined commands to compare the string received from the voice control app
#define START "start"
#define FORWARD "forward"
#define BACKWARD "backward"
#define RIGHT "right"
#define LEFT "left"
#define STOP "stop"
//macro section
#define LM 40
#define RM 39
#define LD 28
#define RD 29
//global variables declaration
int k = 0;
```

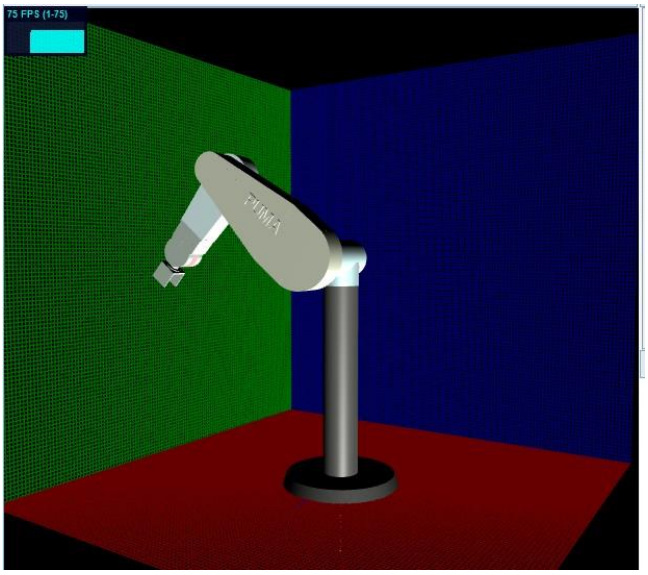
```
int lmspeed = 0,speedl = 0,ldir = 0,rdir = 0;
int rmspeed = 0,speedr = 0;
char cmd[10] = {"\0"};
char precmd[9][10] = {START,STOP,FORWARD,BACKWARD,LEFT,RIGHT};
char ctl[9] = {'1','2','3','4','5','6'};
voidsetup() {
  // put your setup code here, to run once:
Serial.begin(9600);
Serial1.begin(9600);
pinMode(LD, OUTPUT);
pinMode(RD, OUTPUT);
}

void loop() {
int i = 0;
// put your main code here, to run repeatedly:
if(Serial1.available())
Serial1.readBytes(cmd,10);
Serial.println(cmd);
   //To identify the command
for(i = 0;i<9;i++ )
   {
   if((strcmp(cmd,precmd[i])) == 0)
   {
      k = i;
break;
   }
}
//decision and parameters config part
switch(ctl[i])
{
```

```
case '1' :lmspeed = 50;
rmspeed = 50;
digitalWrite(LD,0);
digitalWrite(RD,0);
break;
case '2' :speedl = 0;
speedr = 0;
break;
case '3' :speedl = lmspeed;
speedr = rmspeed;
digitalWrite(LD,0);
digitalWrite(RD,0);
break;
case '4' :speedl = lmspeed;
speedr = rmspeed;
digitalWrite(LD,1);
digitalWrite(RD,1);
break;
case '5' :speedl = 0;
speedr = rmspeed;
break;
case '6' :speedr = 0;
speedl = lmspeed;
break;
default  :speedl = 0;
speedr = 0;
break;
  }
analogWrite(LM,speedl);
analogWrite(RM,speedr);
}
```

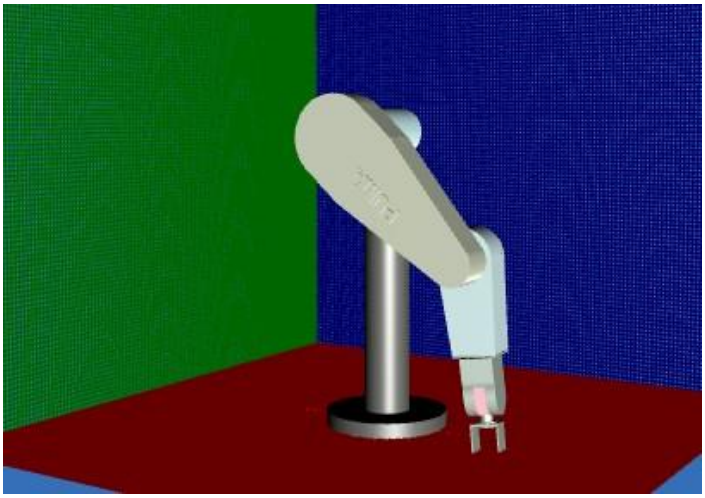

HM-10
MSP432P401R
RX (P3.2)
GND
3.3V
VCC
GND
TX
RX

Expt. 3 & 4 VIRTUAL LAB IMPLEMENTATION OF FORWARD KINEMATICS

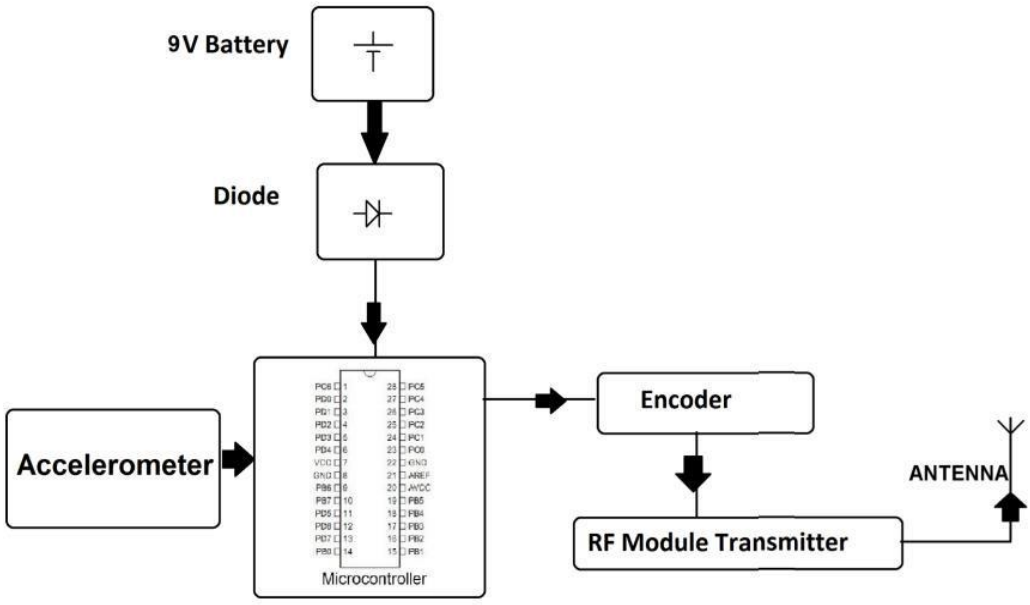
3. FORWARD KINEMATICS



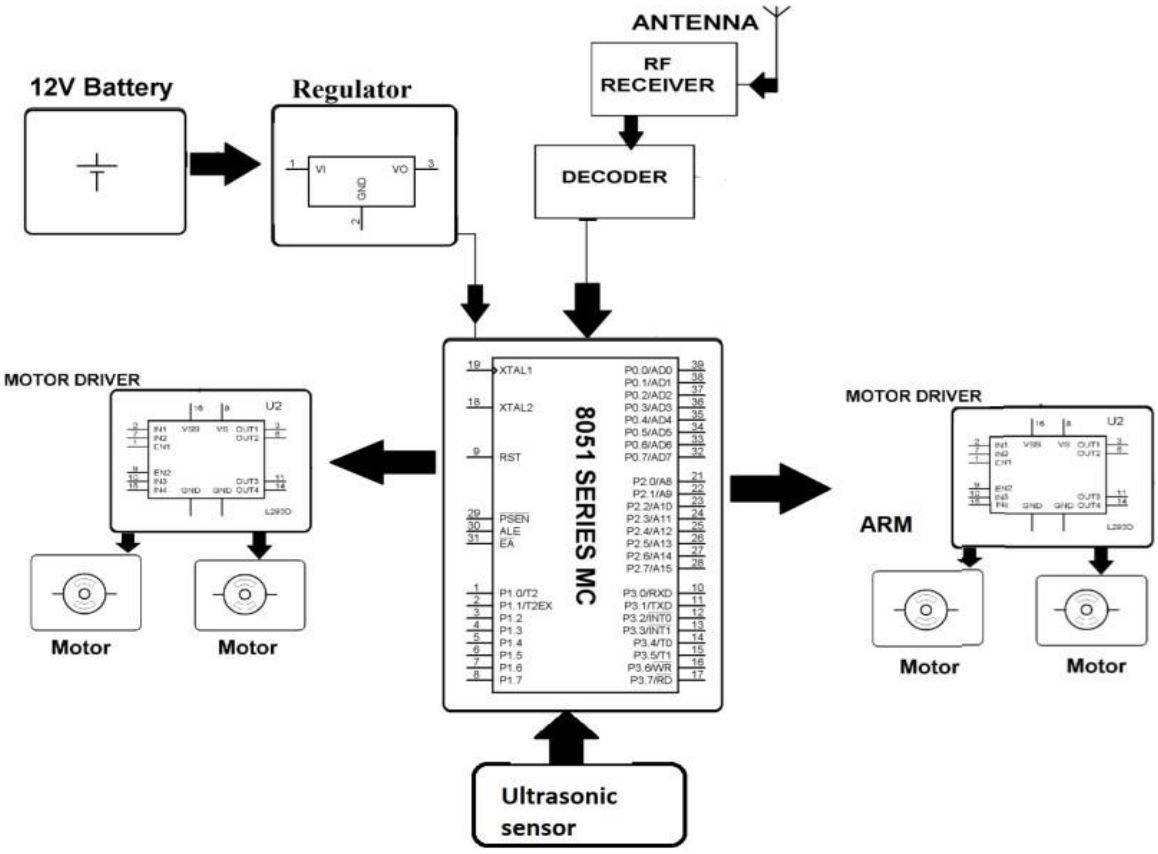
4. REVERSE KINEMATICS



EX no:5



Transmitter Section of Pick and Place Robot using 8051



Expt 6. COLOR AND SHAPE IDENTIFICATION USING RASPBERRY PI

6.1 COLOR IDENTIFICATION

```
import time
import board
import adafruit_tcs34725
import RPi.GPIO as GPIO
i2c = board.I2C()
sensor = adafruit_tcs34725.TCS34725(i2c)
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
while True:
    try:
        color = sensor.color
        color_rgb = sensor.color_rgb_bytes
        (Red,Green,Blue,CL) = sensor.color_raw
        if Green > Blue and Green > Red :
            print("Green Color Detected ")
        if Red > Blue and Red > Green :
             print("Red color is Detected ")
         if Blue > Red and Blue > Green :
            print("Blue color is Detected ")
        else:
            print("Specified color not Detected")
        time.sleep(2)
    except :
        print("Check Connection!")
        i2c = board.I2C()
        sensor = adafruit_tcs34725.TCS34725(i2c)
```

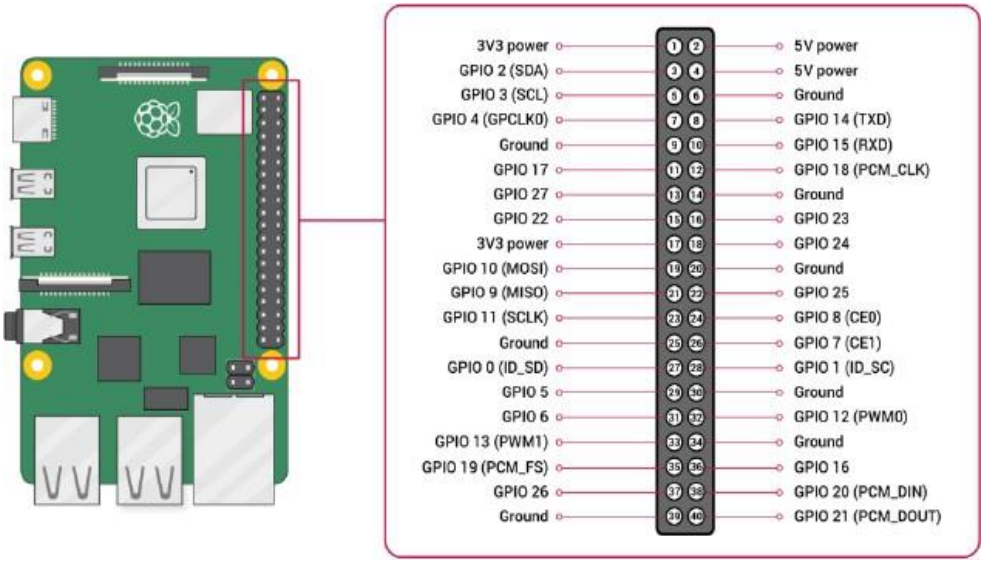
6.2 SHAPE IDENTIFICATION

```
from picamera.array import PiRGBArray # Generates a 3D RGB array
from picamera import PiCamera # Provides a Python interface for the RPi Camera Module
import time # Provides time-related functions
import cv2 # OpenCV library
import numpy as np
# Initialize the camera
camera = PiCamera()
 # Set the camera resolution
camera.resolution = (640, 480)
 # Set the number of frames per second
camera.framerate = 32
# Generates a 3D RGB array and stores it in rawCapture
raw_capture = PiRGBArray(camera, size=(640, 480))
# Wait a certain number of seconds to allow the camera time to warmup
time.sleep(0.1)
def nothing(x):
    # any operation
    pass
cv2.namedWindow("Trackbars")
cv2.createTrackbar("L-H", "Trackbars", 0, 180, nothing)
cv2.createTrackbar("L-S", "Trackbars", 66, 255, nothing)
cv2.createTrackbar("L-V", "Trackbars", 134, 255, nothing)
cv2.createTrackbar("U-H", "Trackbars", 180, 180, nothing)
cv2.createTrackbar("U-S", "Trackbars", 255, 255, nothing)
cv2.createTrackbar("U-V", "Trackbars", 243, 255, nothing)
font = cv2.FONT_HERSHEY_COMPLEX
# Capture frames continuously from the camera
for frame in camera.capture_continuous(raw_capture, format="bgr", use_video_port=True):
```

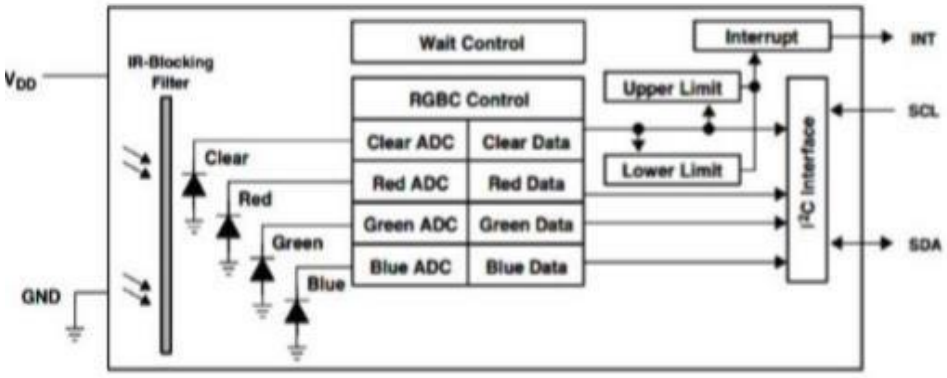
```
    # Grab the raw NumPy array representing the image
    image = frame.array
        # Display the frame using OpenCV
    cv2.imshow("Frame", image)
    #                                                                                #
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    l_h = cv2.getTrackbarPos("L-H", "Trackbars")
    l_s = cv2.getTrackbarPos("L-S", "Trackbars")
    l_v = cv2.getTrackbarPos("L-V", "Trackbars")
    u_h = cv2.getTrackbarPos("U-H", "Trackbars")
    u_s = cv2.getTrackbarPos("U-S", "Trackbars")
    u_v = cv2.getTrackbarPos("U-V", "Trackbars")
    lower_red = np.array([l_h, l_s, l_v])
    upper_red = np.array([u_h, u_s, u_v])
    mask = cv2.inRange(hsv, lower_red, upper_red)
    kernel = np.ones((5, 5), np.uint8)
    mask = cv2.erode(mask, kernel)
    # Contours detection
    if int(cv2.__version__[0]) > 3:
        # Opencv 4.x.x
        contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    else:
        # Opencv 3.x.x
        _, contours, _ = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    for cnt in contours:
        area = cv2.contourArea(cnt)
        approx = cv2.approxPolyDP(cnt, 0.02*cv2.arcLength(cnt, True), True)
        x = approx.ravel()[0]
        y = approx.ravel()[1]
```

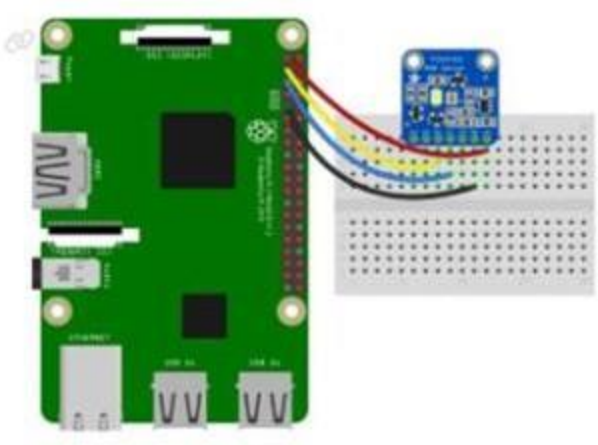
```
    if area > 400:
        cv2.drawContours(image, [approx], 0, (0, 0, 0), 5)
        if len(approx) == 3:
            cv2.putText(image, "Triangle / Pyramid", (x, y), font, 1, (0, 0, 0))
        elif len(approx) == 4:
            cv2.putText(image, "Rectangle / Cube", (x, y), font, 1, (0, 0, 0))
        elif len(approx) == 6:
            cv2.putText(image, "Polygon", (x, y), font, 1, (0, 0, 0))
        elif 6 < len(approx) < 20:
            cv2.putText(image, "Circle / Sphere", (x, y), font, 1, (0, 0, 0))
cv2.imshow("Frame", image)
cv2.imshow("Mask", mask)
#..........................................................................................................................................#
 # Wait for keyPress for 1 millisecond
key = cv2.waitKey(1) & 0xFF
 # Clear the stream in preparation for the next frame
raw_capture.truncate(0)
# If the `q` key was pressed, break from the loop
if key == ord("q"):
    break
```

Colour identification:



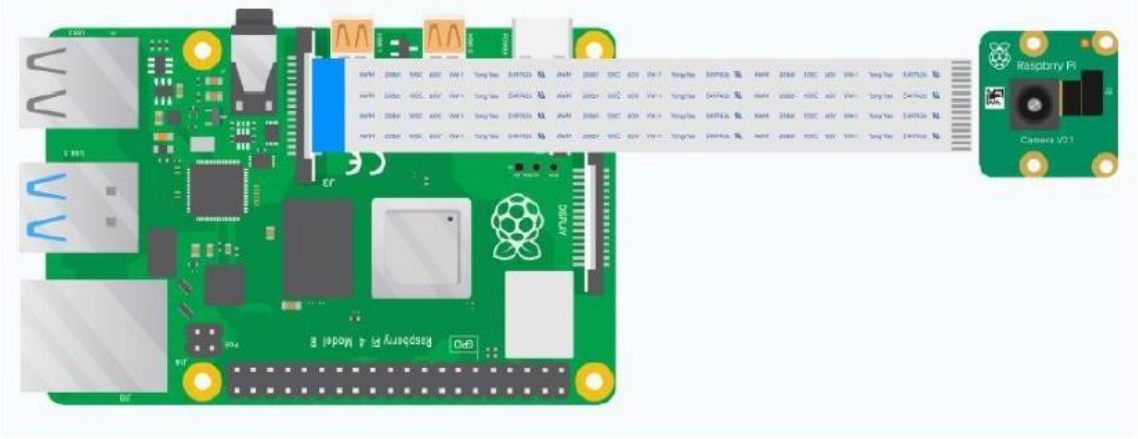
Colour sensor:





- Pi 3V3 to sensor VIN
- Pi GND to sensor GND
- Pi SCL to sensor SCL
- Pi SDA to sensor SDA

Shape identification:



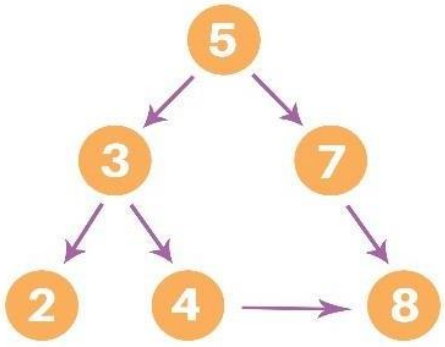
Expt.8 IMPLEMENTATION OF DEPTH FIRST AND BREADTH FIRST SEARCH ALGORITHMS

8.1 DEPTH FIRST SEARCH ALGORITHM

```
# Using a Python dictionary to act as an adjacency list
graph = {
  '5' : ['3','7'],
  '3' : ['2', '4'],
  '7' : ['8'],
  '2' : [],
  '4' : ['8'],
  '8' : []
} visited = set() # Set to keep track of visited nodes of graph.
def dfs(visited, graph, node): #function for dfs
    if node not in visited:
        print (node)
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited, graph, neighbour)
# Driver Code
print("Following is the Depth-First Search")
dfs(visited, graph, '5')
```

Following is the Depth-First Search

5 3 2 4 8 7



8.2 BREADTH FIRST SEARCH ALGORITHM

```
graph = {
  '5' : ['3','7'],
  '3' : ['2', '4'],
  '7' : ['8'],
  '2' : [],
  '4' : ['8'],
  '8' : []
}
visited = [] # List for visited nodes.
queue = []     #Initialize a queue
def bfs(visited, graph, node): #function for BFS
  visited.append(node)
  queue.append(node)
  while queue:          # Creating loop to visit each node
    m = queue.pop(0)
    print (m, end = " ")
    for neighbour in graph[m]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)
# Driver Code
print("Following is the Breadth-First Search")
bfs(visited, graph, '5')    # function calling
```

```
Following is the Breadth-First Search
 5 3 7 2 4 8
```

Expt. 9 REAL TIME SENSORS INTERFACE WITH RASPBERRY PI

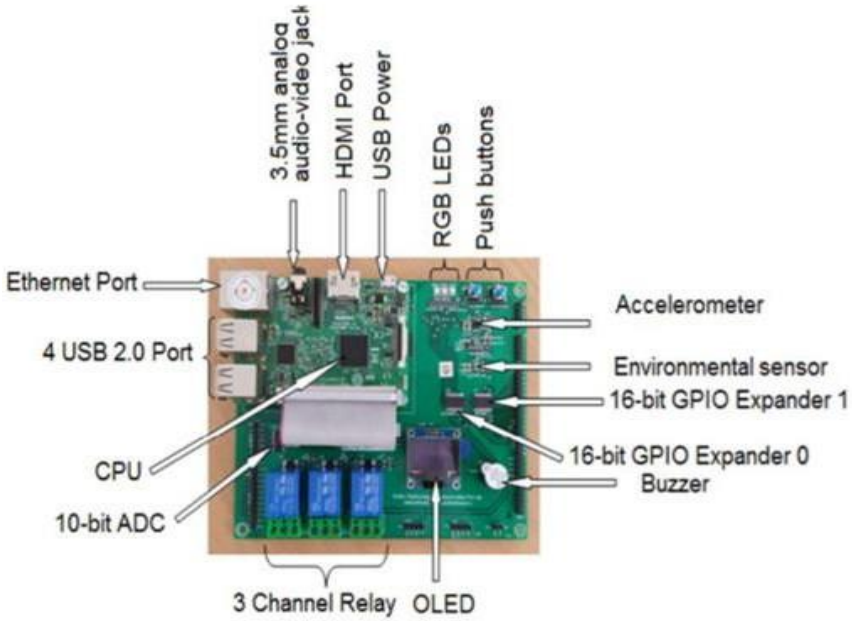
9.1 TEMPERATURE AND HUMIDITY SENSOR INTERFACE WITH RASPBERRY PI

```
#This program will display Temperature and humidity.

# Sensor Library : sudo pip3 install adafruit-circuitpython-sht31d

# Reference Link : https://learn.adafruit.com/adafruit-sht31-d-temperature-and-humidity-sensor-breakout/python-circuitpython

import sys

sys.path.append('/home/pi/Adafruit-Raspberry-Pi-Python-Code-legacy/Adafruit_CircuitPython_SHT31D-main')

import time

import board

import busio

import adafruit_sht31d

i2c =board.I2C()

sensor =adafruit_sht31d.SHT31D(i2c)

while True:

    print("Temperature:%0.1f C" %sensor.temperature)

    print("Humidity:%0.1f %%" %sensor.relative_humidity)

    time.sleep(2)
```

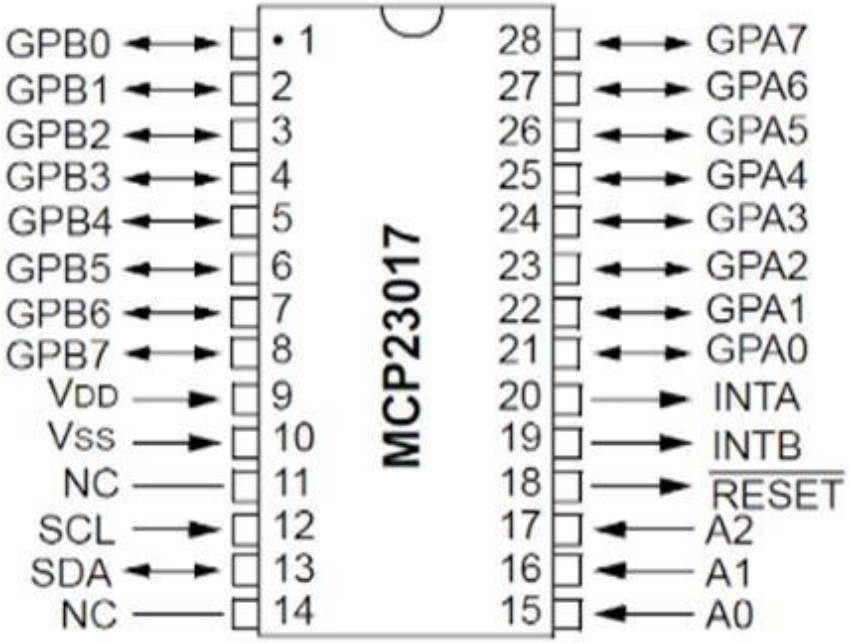
9.2 OLED INTERFACE

```
import sys
sys.path.append('/home/pi/Adafruit-Raspberry-Pi-Python-Code-legacy/Adafruit_CircuitPython_SSD1306-main')
import busio
import board
import time
import digitalio
from board import SCL, SDA
from digitalio import Direction, Pull
from PIL import Image, ImageDraw, ImageFont
import adafruit_ssd1306

RESET_PIN = digitalio.DigitalInOut(board.D4)

# Very important... This lets py-gaugette 'know' what pins to use in order to reset the display
i2c = board.I2C()
oled = adafruit_ssd1306.SSD1306_I2C(128, 64, i2c, addr=0x3C, reset=RESET_PIN)
#Clear Display
oled.fill(0)
oled.show()
# Create blank image for drawing.
image = Image.new("1", (oled.width, oled.height))
draw = ImageDraw.Draw(image)
`# Load a fonts in same size.
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 11)
font2 = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 18)
draw.text((0, 0), "*PSREC ECE*", font=font, fill=255) # Line 1
draw.text((0, 20), "mons ", font=font2, fill=255) # Line 2
draw.text((0, 40), "***************", font=font2, fill=255) # Line 3
oled.image(image)
oled.show()
print("PSREC ECE")
print("mons")
time.sleep(10)
```

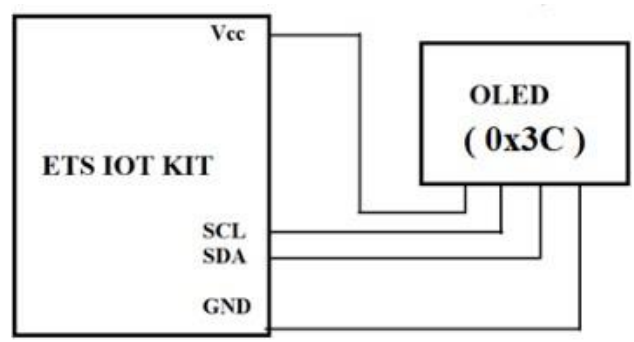
Block Diagram of IoT Kit:

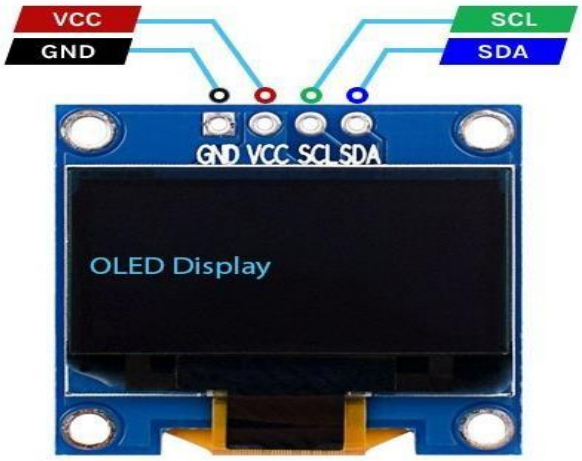


MCP23017 IC Pin diagram:



OLED INTERFACE:





TEMPERATURE AND HUMIDITY SENSOR INTERFACE: