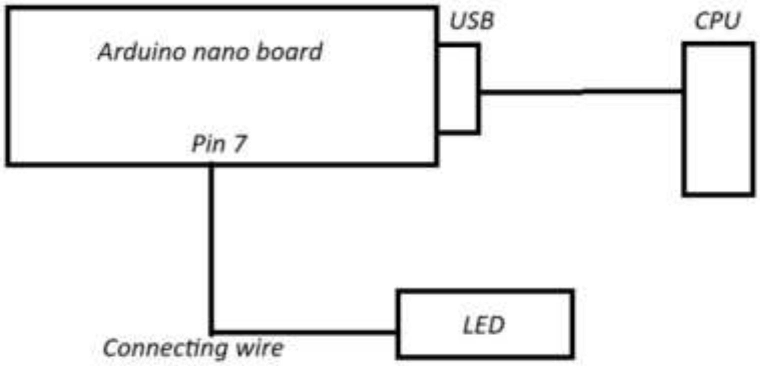
WEEK 1

1. Blinking single LED

SOURCE CODE:

```
void setup () {

   pinMode (7, OUTPUT);
}
void loop () {
 digitalWrite (7, HIGH);
 delay (1000);
 digitalWrite (7, LOW);
 delay (1000);
}
```

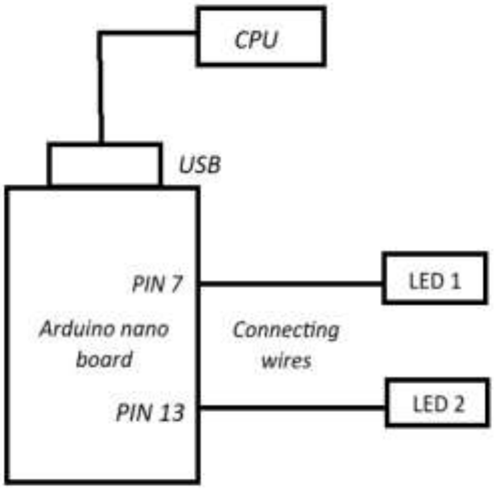


2. Blinking two LEDs

SOURCE CODE:

```
void setup () {

    pinMode (7, OUTPUT);
    pinMode (13, OUTPUT);
}
void loop () {
    digitalWrite (13, HIGH);
    digitalWrite (7, LOW);
    delay (1500);
    digitalWrite (13, LOW);
    digitalWrite (7, HIGH);
    delay (1500);
}
```



APPLICATIONS:

a. Traffic light systems

b. Notification indicators for devices

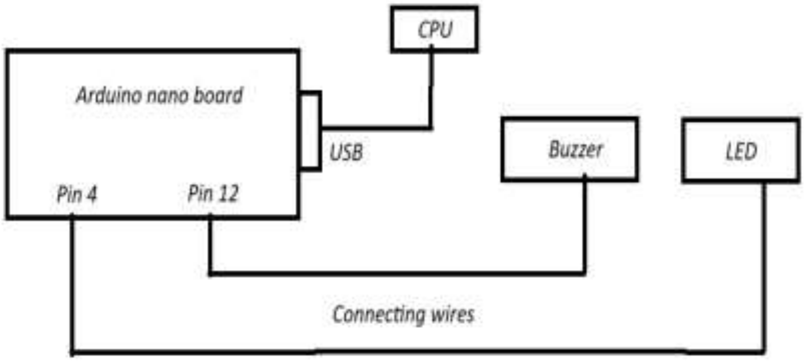
c. Simple debugging/testing tool for circuits

WEEK 2

1. LED and Buzzer work simultaneously.

SOURCE CODE:

```
void setup () {
    pinMode (4, OUTPUT);
    pinMode (12, OUTPUT);
}
void loop () {
    digitalWrite (4, HIGH);
    digitalWrite (12, HIGH);
    delay (1000);
    digitalWrite (4, LOW);
    digitalWrite (12, LOW);
    delay (1000);
}
```



2. LED and buzzer work alternatively.

SOURCE CODE:

```
void setup () {

    pinMode (4, OUTPUT);
    pinMode (12, OUTPUT);
}
void loop () {
    digitalWrite (4, HIGH);
    digitalWrite (12, LOW);
    delay (1000);
    digitalWrite (4, LOW);
    digitalWrite (12, HIGH);
    delay (1000);
}
```

APPLICATIONS:

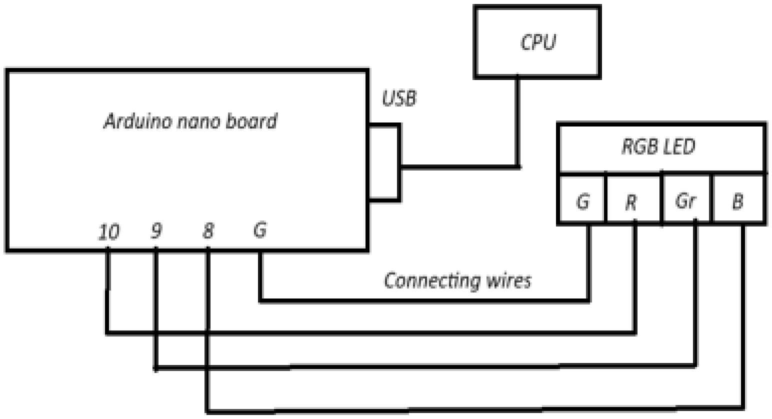
a. Alarm systems

b. Timer/countdown indicators

c. Simple alert mechanisms

WEEK 3

1. RGB LED to obtain different colours



SOURCE CODE:

```
int redPin =10;
int greenPin =9;
int bluePin = 8;

void setup () {
    pinMode (redPin, OUTPUT);
    pinMode (greenPin,
OUTPUT);
    pinMode (bluePin, OUTPUT);
}
void loop () {
    setColor(255,0,0); //Red color
    delay(1000);
    setColor(0,255,0); // Green color
    delay(1000);
    setColor(0,0,255); // Blue color
    delay(1000);
    setColor(255,255,255); //White color
    delay(1000);
    setColor(170,0,255); // Purple color
    delay(1000);
}
Void setColor( int redValue , int greenValue , int blueValue) {
    analogWrite ( redPin , redValue );
    analogWrite ( greenPin , greenValue );
    analogWrite ( bluePin , blueValue );
}
```

Applications:

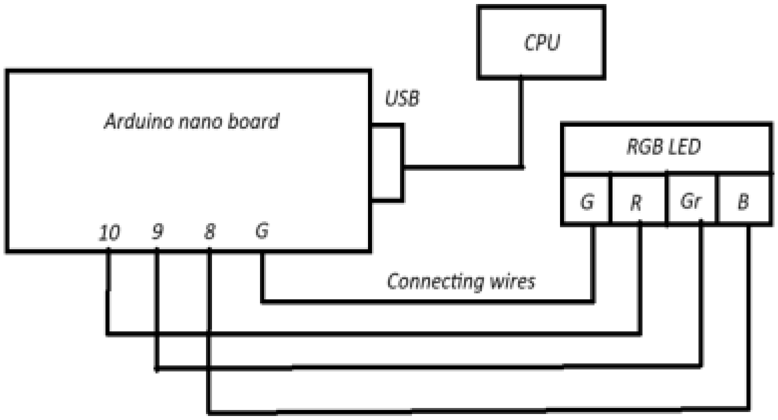
a. Ambient lighting systems

b. Status indicators

c. Simple data visualization

2. To obtain different brightnesses of colors using RGB LED.

SOURCE CODE:



```
void setup () {
    pinMode (10, OUTPUT);
    pinMode (9, OUTPUT);
    pinMode (8, OUTPUT);
}

Void loop ( ) {
    for ( int i=0 ; i<=255 ; i ++
) {
    analogWrite (10 , i );
    delay ( 10 );
    }
    for ( int i=255 ; i>=0 ; i -- ) {
    analogWrite (10 , i );
    delay ( 10 );
    }
    for ( int i=0 ; i<=255 ; i ++ ) {
    analogWrite ( 9 , i );
    delay ( 20 );
    }
    for ( int i= 255 ; i>= 0 ; i ++ ) {
    analogWrite ( 9 , i );
    delay ( 20 );
    }
    for ( int i=0 ; i<=255 ; i ++ ) {
    analogWrite ( 8 , i );
    delay ( 10 );
    }
    for ( int i= 255 ; i>=0 ; i -- ) {
    analogWrite ( 8 , i );
    delay ( 10 );
    }
}
```

Applications:

a. Ambient lighting systems

b. Status indicators

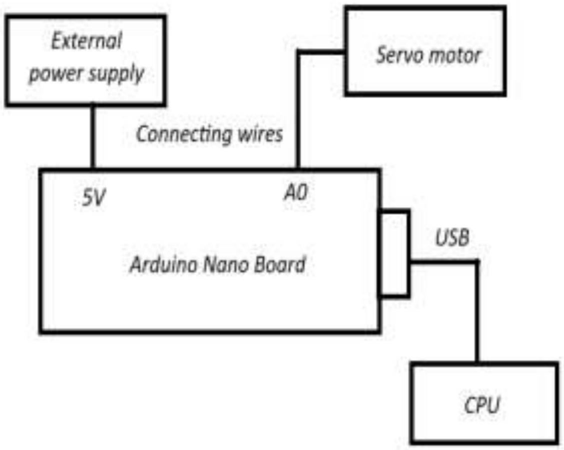
c. Simple data visualization

WEEK 4

1. Servo motor

SOURCE CODE:

```
# include < Servo.h >
Servo S;
void setup () {
 S.attach (A0);
}
void loop () {
  S.write (0);
  delay (500);
  S.write (80);
  delay (500);
  S.write (120);
  delay (500);
  S.write (170);
  delay (500);
}
```



APPLICATIONS:

a. Robotics/animatronics

b. Automated mechanisms

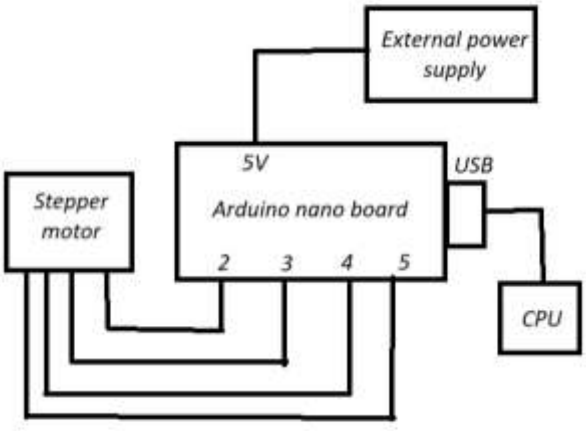
c. Simple positioning systems

2. Stepper motor

SOURCE CODE:

```
# include < Stepper.h >
Const int S=200;
Stepper muStepper(S,2,4,3,5);
void setup () {
 myStepper.setSpeed(60);
Serial.begin(9600);
}

void loop () {
 Serial.println(“Clockwise”);
 myStepper.steps(S);
 delay(500);
 Serial.println(“Counter Clockwise”);
 myStepper.step(-S);
 delay(500);
}
```



APPLICATIONS:

a. Precision motion control

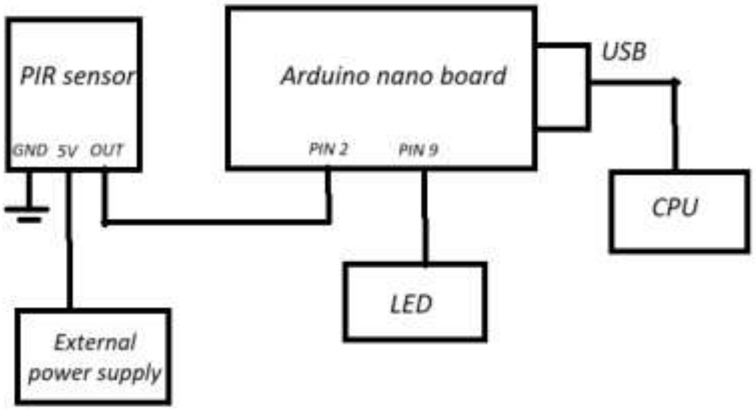
b. Robotics

c. Instrumentation

WEEK 5

1. PIR Sensor

```
int Val;
int led = 9;
int Sensor = 2;
Void setup () {
  pinMode (led, OUTPUT);
  pinMode (Sensor, INPUT);
  Serial.begin(9600);
}
Void loop () {
  Val = digitalRead(Sensor);
  if (Val== HIGH){
        digitalWrite(led, HIGH);
        delay(100);
        Serial.prinntln(“Motion detected”);
        Serial.print(“Time: “);
        Serial.println(millis());
}
else {
   digitalWrite (led, LOW);
   delay(200);
   Serial.println(“Motion not detected”);
}}
```



APPLICATIONS:

a. Environmental monitoring

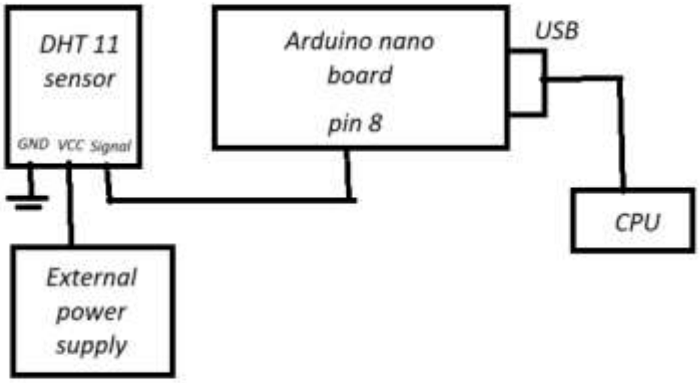
b. Industrial process monitoring

c. Data logging systems

2. DHT11 SENSOR

SOURCE CODE:

```
#include< DHT.h>
DHT dht (8,DHT11)
float h,t;
void setup () {
  Serial.begin(9600);
  dht.begin();
}
Void loop () {
  h = dht.readHumidity( );
  t = dht.readTemparature( );
  Serial.print(“Humidity: “);
  Serial.println(h);
  Serial.print(“Temparature: “);
  Serial.println(t);
}
```



APPLICATIONS:

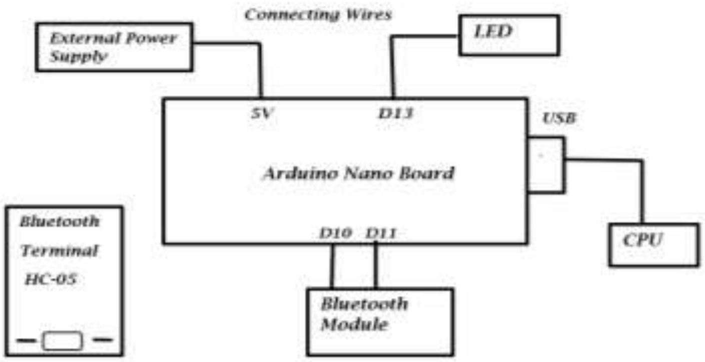
a. Environmental monitoring

b. Industrial process monitoring

c. Data logging systems

WEEK 6

1. **Control any two actuators connected to the Arduino using Bluetooth/ Wi-Fi. LED blinks when message is sent and received.**



SOURCE CODE:

```
#include <SoftwareSerial.h>
SoftwareSerial EEBLUE(10,11);
void setup(){
 Serial.begin(9600);
 EEBLUE.begin(9600);
 Serial.println("Bluetooth devices activated ");
 pinMode (13, OUTPUT);
}
void loop(){
 if(EEBLUE.available()) {
  Serial.write(EEBLUE.read());
  digitalWrite(13, HIGH);
}
 if(Serial.available()){
 EEBLUE.write(Serial.read());
} }
```

APPLICATIONS:

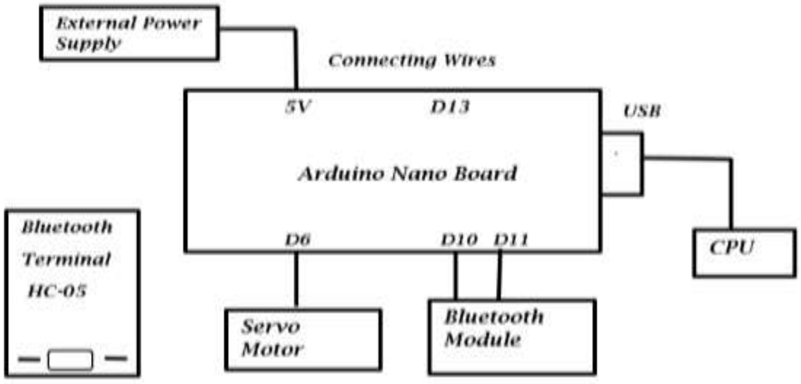
a. Home/industrial automation

b. Remote control systems

c. Internet of Things (IoT) device

2. Controlling servo motor connected to Arduino using Bluetooth.

SOURCE CODE:



```
#include<Servo.h>
#include <SoftwareSerial.h>
SoftwareSerial EEBLUE(10,11);
Servo s1;
int Servopin = 6;
void setup() {
 s1.attach(Servopin);
 Serial.begin(9600);
 EEBLUE.begin(9600);
 Serial.println("Bluetooth gates open");
}
void loop(){
if(EEBLUE.available()){
 Servopin =EEBLUE.parseInt();
 for(int i=0;i<=Servopin;i+=20){
 s1.write(i);
 delay(500);
}}}
```

APPLICATIONS:

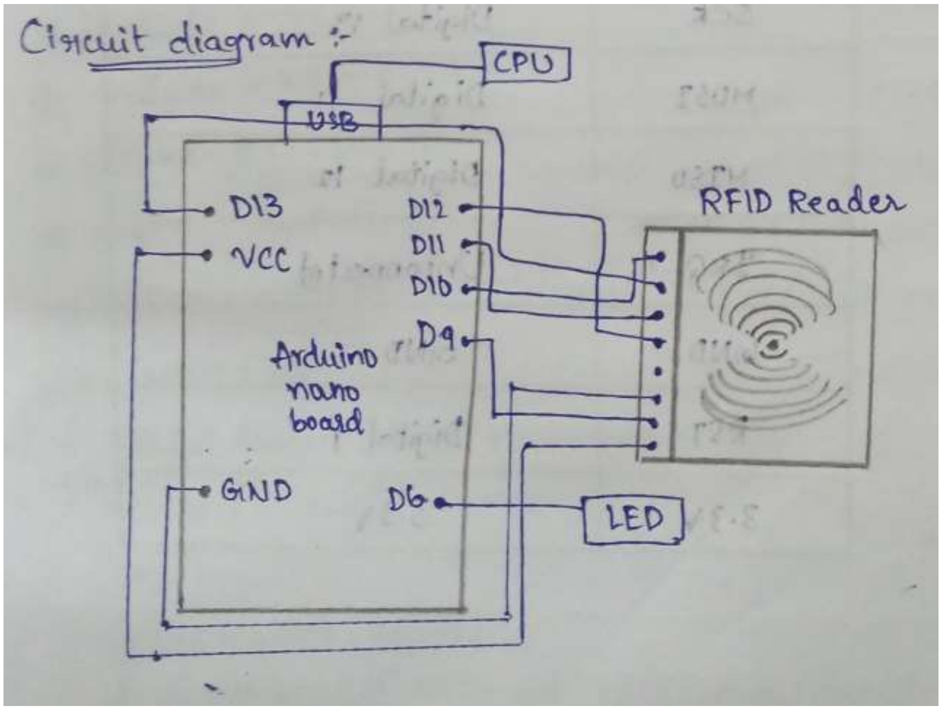
a. Home/industrial automation

b. Remote control systems

c. Internet of Things (IoT) device

WEEK 7

A program to read data from RFID tag and display information on display board using Arduino and control LED.



SOURCE CODE:

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN 9
#define SS_PIN 10
MFRCS22 mfrc522 (SS_PIN,
RST_PIN);
void setup() {
    Serial.begin(9600)
    while (!Serial);
    SPI begin();
    mfrc522.PCD_Init();
    delay(4)
    mfrc522. PCD_Dump Version To Serial ();
    pinMode (6, OUTPUT);
}
void loop(){
    if (! mfrc 522. PICC_IsNewCardPresent ()) {
    digitalWrite(6, Low);
    }
    if (!mfrc 522. PICC_ReadCardSerial ()) {
    return;
    }
    digitalWrite(6, HIGH);
    delay (500);
    mfrc522.PICC_DumpToSerial(&(mfrc522.vid));
}
```

APPLICATIONS:

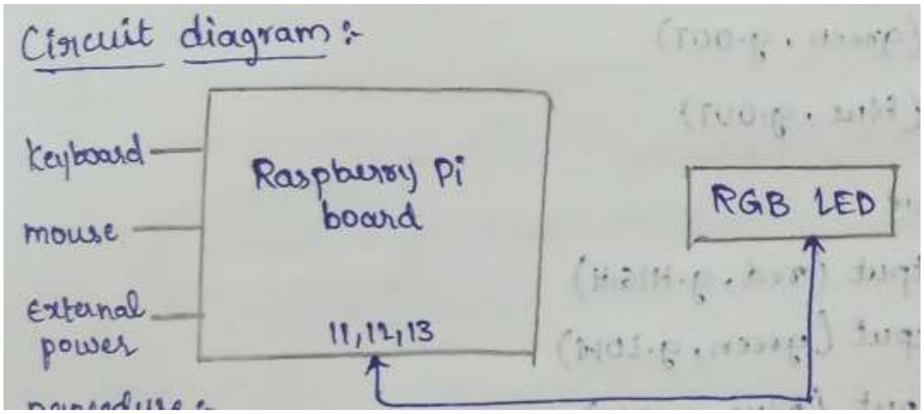
a. Access control systems

b. Asset tracking

c. Attendance monitoring

WEEK9

Interface RGB LED with Raspberry Pi to obtain different colours.



SOURCE CODE:

```
import RP. GPID as g
from time import sleep
g. setwarnings (False)
g. setmode (9.BCM)
red = 11
green = 12
blue = 13
g.setup (red, g.OUT)
g. setup (green, g.OUT)
g. setup (blue, g.OUT)
while True:
  g. output (red, g.HIGH)
  g.output (green, g.LOW)
  g.output (blue, g. LOW)
  sleep(2)
  g.output (red, g.LOW)
  g.output (green, g.HIGH)
  g.output (blue, g. LOW)
  sleep (2)
  g.output (red, g.LOW)
  g.output (green, g.HIGH)
  g.output (blue, g. LOW)
  sleep (2)
```

APPLICATIONS:

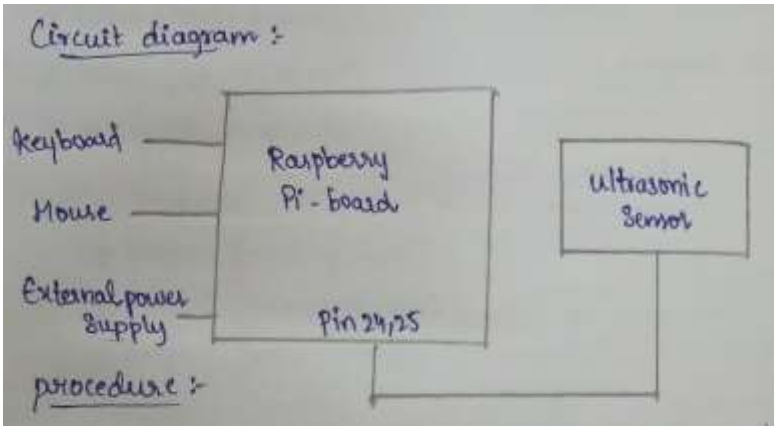
a. Smart lighting systems

b. Visual feedback indicators

c. Interactive art installations

WEEK 10

Interface an ultrasonic sensor with Raspberry pi to print distance readings on the monitor when the sensor changes its position.



SOURCE CODE:

```
import RP. GPID as g
import time
g. setwarnings (False)
g. setmode (9.BCM)
trigpin = 24
echo = 25
g.setup (trigpin, g.OUT)
g. setup (echopin, g.OUT)
def distance(trigpin, echopin):
  g.output(trigpin, True)
  time.sleep(0.0001)
  g.output(trigpin, False)
  while g.input(echopin)==0:
    start == time.time()
  while g.input(echopin)==1:
    end == time.time()
  try:
    duration = end - start
  except:
    print("Calibrating")
    return -1
  distance = duration*17150
  distance = round(distance + 1.15,2)
  return distance
while True:
  dist=distance(trigpin, echopin)
  print('Measured distance=()cm'.format(dist))
  time.sleep(1)
```

APPLICATIONS:

a. Obstacle detection/avoidance

b. Level monitoring

c. Proximity sensing

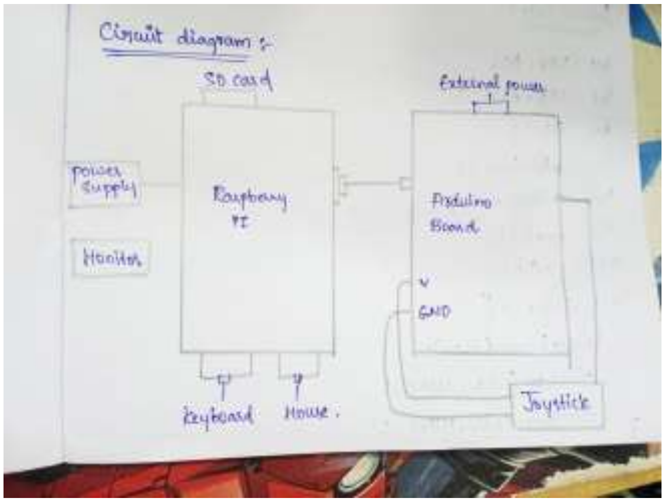
OUTPUT:

We can see the distance measured in the message in the Shell.

```
Calibrating
Measure distance = -1 cm
Calibrating
Measure distance = -1 cm
```

WEEK 11

Reading the data from an analog sensor(joy stick) with Raspberry using Arduino serial port or ADC CP3208 using SPI.



SOURCE CODE:

```
int VRx = A0;
int VRy = A1;
int SW = 2;
int xposition = 0;
int yposition = 0;
int mapX = 0;
int mapY = 0;
void setup() {
Serial.begin(9600);
pinMode(VRx, INPUT);
pinMode(VRy, INPUT);
}
void loop()
{
  xposition = analogRead(VRx);
  yposition = analogRead(VRy);
  mapX = map(xposition, 0, 1023, -512, 512);
  mapY = map(yposition, 0, 1023, -512, 512);
  Serial.print("X:");
  Serial.print(mapX),
  Serial.print("Y:");
  Serial.print(mapY);
  delay(100);
}
```

APPLICATIONS:

a. Data acquisition systems

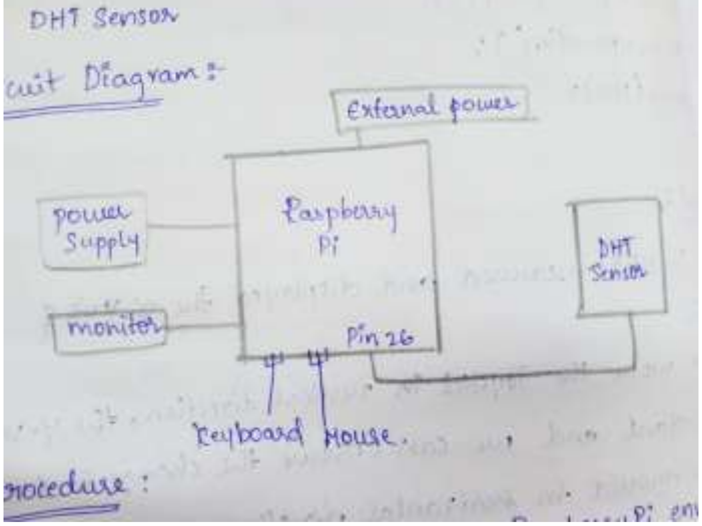
b. Environmental monitoring stations

c. Instrumentation/test equipment

WEEK 12

**Post/read the data to/from the cloud
via MQTT broker with a Raspberry Pi.**

PROCEDURE:



a. Open Chromium browser in Raspberry Pi environment.
b. Search for ThingSpeak and go to first URL.
c. Click on Get Started and create a new account if account is not created.
d. Verify account and the account is successfully verified.
e. Enter all the details.
f. Create new channel and name it as DHT.
g. Name:DHT Sensor
h. Description: Reading temperature and humidity using DHT
i. Field1 : Temperature
j. Field2: Humidity
k. Click show status and save the channel.
l. Go to API Keys and copy “Write the API Key”
m. Create a new python file and write the below code.
n. Connect DHT Sensor to pin 26 of Raspberry pi.

SOURCE CODE:

```
import urllib.request

import time
import RPi.GPIO as GPIO
import Adafruit_DHT

writeAPIKey = "8CMZMVX6CQELK7KG"            # enter write API key here
```

```
baseURL =
"https://api.thingspeak.com/update?api_key={}".format(writeAPIKey)
sensor = Adafruit_DHT.DHT11
sensor_Pin = 26
GPIO.setmode(GPIO.BCM) # Using BCM numbering

try:
while True:
        humidity, temperature = Adafruit_DHT.read_retry(sensor, sensorPin)
        if humidity is not None and temperature is not None:
        # Format the readings to two decimal places
        humidity = '%.2f' % humidity
        temperature = '%.2f' % temperature
        # Sending the readings to thingspeak
        conn = urllib.request.urlopen(baseURL + '&field1={}&field2={}'.format(humidity,
        temperature))  print(conn.read())

        # Closing the connection
        conn.close()

        time.sleep(20) # wait 20 seconds before uploading next reading
        except KeyboardInterrupt:
        GPIO.cleanup()
        exit()
```

OUTPUT:

The DATA is visualized in cloud.

APPLICATIONS:

a. IoT device communication

b. Remote monitoring systems

c. Data logging to cloud