**Expt 2 (LED Switch )**

**Code Arduino:**

const int buttonPin = 8; // Digital Pin 8 define as a push Button pin

const int ledPin = 10; // Digital Pin 13 define as LED Pin

int buttonState = 0; // variable for reading the pushbutton status

void setup() {

pinMode(ledPin, OUTPUT);

pinMode(buttonPin, INPUT);

Serial.begin(9600);

}

void loop() {

buttonState = digitalRead(buttonPin);

if (buttonState = HIGH)

{ digitalWrite(ledPin, HIGH);

Serial.println("Button Pressed");

}

else

}

{ digitalWrite(ledPin, LOW);

Serial.println("Button Not Pressed");

}

----------------------------------------------------------

int LEDpin = 13;

int delayT = 1000;

void setup() {

// put your setup code here, to run once:

pinMode(LEDpin, OUTPUT);

}

void loop() {

// put your main code here, to run repeatedly:

digitalWrite(LEDpin, HIGH);

delay(delayT);

digitalWrite(LEDpin, LOW);

delay(delayT);

}

**Code RPi: (LED Switch )**

import RPi.GPIO as GPIO

import time

LED = 5

GPIO.setmode (GPIO.BCM)

GPIO.setup(LED, GPIO.OUT)

GPIO.output(LED, GPIO.HIGH)

def blink():

GPIO.output(LED, GPIO.HIGH)

time.sleep(1)

GPIO.output(LED, GPIO.LOW)

time.sleep(1)

def destroy():

GPIO.output(LED, GPIO.LOW)

GPIO.cleanup()

if \_\_name\_\_ == '\_\_main\_\_':

try:

while True:

blink()

except KeyboardInterrupt:

destroy()

**Expt 3 (DTH11)**

**Code Arduino:**

#include < DHT.h>;

#define DHTPIN 2 // Define the pin where your DHT11 is connected

#define DHTTYPE DHT11 // Define the type of DHT sensor (DHT11 or DHT22)

DHT dht(DHTPIN, DHTTYPE);

void setup() {

  Serial.begin(9600);

  dht.begin();}

void loop() {

  delay(2000); // Delay for 2 seconds between measurements

  float temperature = dht.readTemperature();

  float humidity = dht.readHumidity();

  // Check if any reads failed and exit early (to try again).

  if (isnan(temperature) || isnan(humidity)) {

    Serial.println(”Failed to read from DHT sensor”);

    return;

  }

  Serial.print(“Temperature: “);

  Serial.print(temperature);

  Serial.println(“°C”);

  Serial.print(“Humidity: “);

  Serial.print(humidity);

  Serial.println(“%”);

  // float fahrenheit = (temperature \* 1.8) + 32;

  // Serial.print(“Temperature: “);

  // Serial.print(fahrenheit);

  // Serial.println(“°F”);}

**Code RPi: (DTH11)**

#sudo pip3 install adafruit-circuitpython-dht

#sudo apt-get install libgpiod2

import time

import board

import digitalio

import adafruit\_character\_lcd.character\_lcd as characterlcd

import adafruit\_dht

import RPi.GPIO as GPIO

# Initial the dht device, with data pin connected to:

dhtDevice = adafruit\_dht.DHT11(board.D19)

# Modify this if you have a different sized character LCD

lcd\_columns = 16

lcd\_rows = 2

# Raspberry Pi Pin Config:

lcd\_rs = digitalio.DigitalInOut(board.D5)

lcd\_en = digitalio.DigitalInOut(board.D6)

lcd\_d4 = digitalio.DigitalInOut(board.D12)

lcd\_d5 = digitalio.DigitalInOut(board.D13)

lcd\_d6 = digitalio.DigitalInOut(board.D16)

lcd\_d7 = digitalio.DigitalInOut(board.D17)

# Initialise the lcd class

lcd = characterlcd.Character\_LCD\_Mono(

lcd\_rs, lcd\_en, lcd\_d4, lcd\_d5, lcd\_d6, lcd\_d7, lcd\_columns, lcd\_rows)

if \_\_name\_\_ == '\_\_main\_\_':

while True:

try:

# Print the values to the serial port

temperature\_c = dhtDevice.temperature

temperature\_f = temperature\_c \* (9 / 5) + 32

humidity = dhtDevice.humidity

print("Temp: {:.1f} F / {:.1f} C Humidity: {}% "

.format(temperature\_f, temperature\_c, humidity))

lcd.clear()

#lcd\_line\_1 = "Temperature:" + str(temperature\_c) + " C"

#lcd\_line\_2 = "\nHumidity:"+ str(humidity) + " %"

#lcd.message = lcd\_line\_1 + lcd\_line\_2;

lcd.message = ("Temper:%.1f C " %temperature\_c)

lcd.message = ("\nHumidity:%.1F " %humidity)

time.sleep(2.0)

except RuntimeError as error:

# Errors happen fairly often, DHT's are hard to read, just keep going

print(error.args[0])

time.sleep(2.0)

continue

except KeyboardInterrupt:

GPIO.cleanup()

print ('Exiting Program')

exit()

**Ultrasonic**

**Code Arduino:**

#define TRIG\_PIN 9  // Define the digital output pin for the ultrasonic trigger

#define ECHO\_PIN 10 // Define the digital input pin for the ultrasonic echo

void setup() {

  Serial.begin(9600);

  pinMode(TRIG\_PIN, OUTPUT);

  pinMode(ECHO\_PIN, INPUT);

}

void loop() {

  // Send a pulse to the ultrasonic sensor to trigger a measurement

  digitalWrite(TRIG\_PIN, LOW);

  delayMicroseconds(2);

  digitalWrite(TRIG\_PIN, HIGH);

  delayMicroseconds(10);

  digitalWrite(TRIG\_PIN, LOW);

  // Read the duration of the echo signal (in microseconds)

  long duration = pulseIn(ECHO\_PIN, HIGH);

  // Calculate the distance in centimeters using the speed of sound (343 m/s)

  // and the formula: Distance = (Duration \* Speed of Sound) / 2

  float distance\_cm = (duration \* 0.0343) / 2;

  // Print the measured distance to the serial monitor

  Serial.print(“Distance: “);

  Serial.print(distance\_cm);

  Serial.println(“ cm”);

  delay(1000); // Wait for a second before taking the next measurement

}

**Code RPi: Ultrasonic**

#Libraries

import RPi.GPIO as GPIO #library for Raspberry Pi GPIOs

import time #library to use sleep function

import board

import digitalio

import adafruit\_character\_lcd.character\_lcd as characterlcd

#GPIO Mode (BOARD / BCM)

GPIO.setmode(GPIO.BCM)

# Modify this if you have a different sized character LCD

lcd\_columns = 16

lcd\_rows = 2

# Raspberry Pi Pin Config:

lcd\_rs = digitalio.DigitalInOut(board.D5)

lcd\_en = digitalio.DigitalInOut(board.D6)

lcd\_d4 = digitalio.DigitalInOut(board.D12)

lcd\_d5 = digitalio.DigitalInOut(board.D13)

lcd\_d6 = digitalio.DigitalInOut(board.D16)

lcd\_d7 = digitalio.DigitalInOut(board.D17)

# Initialise the lcd class

lcd = characterlcd.Character\_LCD\_Mono(

lcd\_rs, lcd\_en, lcd\_d4, lcd\_d5, lcd\_d6, lcd\_d7, lcd\_columns, lcd\_rows)

#set GPIO Pins

TRIGGER = 19 # board pin as trigger

ECHO = 20 # board pin as echo

#set GPIO direction (IN / OUT)

GPIO.setup(TRIGGER, GPIO.OUT)

GPIO.setup(ECHO, GPIO.IN)

lcd.clear()

#function distance will use 2 GPIOs to trigger and echo to calculate distance using the distance formula

def distance():

# set Trigger to HIGH

GPIO.output(TRIGGER, True)

# set Trigger after 0.01ms to LOW

time.sleep(0.00001)

GPIO.output(TRIGGER, False)

StartTime = time.time()

StopTime = time.time()

# save StartTime

while GPIO.input(ECHO) == 0:

StartTime = time.time()

# save time of arrival

while GPIO.input(ECHO) == 1:

StopTime = time.time()

# time difference between start and arrival

TimeElapsed = StopTime - StartTime

# multiply with the sonic speed (34300 cm/s)

# and divide by 2, because there and back

distance = (TimeElapsed \* 34300) / 2

return distance

#simple if statement

if \_\_name\_\_ == '\_\_main\_\_':

#simple try exception programming

try:

while True:

dist = distance()#we accept the value in a variable dist

print ("Measured Distance = %.1f cm" % dist)#display dist

lcd.clear()

lcd.message = ("Dist.:%.1f cm" % dist)

time.sleep(2)

# Reset by pressing CTRL + C

except KeyboardInterrupt:

print("Measurement stopped by User")

GPIO.cleanup()#finally GPIO cleanup to flush all the buffers of the GPIOs used in this code

**IR Sensors**

**Code Arduino:**

int SensorPin = 2;

int OutputPin = 13;

void setup() {

pinMode(OutputPin, OUTPUT);

pinMode(SensorPin, INPUT);

Serial.begin(9600);

}

void loop() {

int SensorValue = digitalRead(SensorPin);

Serial.print( “SensorPin Value: “);

Serial.println(SensorValue);

delay(1000);

if (SensorValue==LOW){ // LOW MEANS Object Detected

digitalWrite(OutputPin, HIGH);}

else

{ digitalWrite(OutputPin, LOW);

}}

**Code Arduino: IR Sensors**

int ledPin=13;

int inputPin=2;

int val=0;

void setup()

{

pinMode(13,OUTPUT);

pinMode(inputPin, INPUT);

Serial.begin(9600);

}

void loop()

{

val=digitalRead(inputPin); // check the pin status (High=1/Low=0) //Active Low output

if(val=HIGH)

{

Serial.print("Object Absent\n");

digitalWrite(13,LOW);

}

else

{

Serial.print("Object Present\n");

digitalWrite(13,HIGH);

}}

**Code RPi: IR Sensors**

#Raspberry Pi Libraries

import RPi.GPIO as GPIO #GPIO library

import time #library for sleep

import board

import digitalio

import adafruit\_character\_lcd.character\_lcd as characterlcd

#set mode as BCM

GPIO.setmode(GPIO.BCM)

# Modify this if you have a different sized character LCD

lcd\_columns = 16

lcd\_rows = 2

# Raspberry Pi Pin Config:

lcd\_rs = digitalio.DigitalInOut(board.D5)

lcd\_en = digitalio.DigitalInOut(board.D6)

lcd\_d4 = digitalio.DigitalInOut(board.D12)

lcd\_d5 = digitalio.DigitalInOut(board.D13)

lcd\_d6 = digitalio.DigitalInOut(board.D16)

lcd\_d7 = digitalio.DigitalInOut(board.D17)

# Initialise the lcd class

lcd = characterlcd.Character\_LCD\_Mono(

lcd\_rs, lcd\_en, lcd\_d4, lcd\_d5, lcd\_d6, lcd\_d7, lcd\_columns, lcd\_rows)

#set pins

IR\_OUT = 21

BUZ = 22

#setup pins at output

GPIO.setup(IR\_OUT, GPIO.IN)

GPIO.setup(BUZ, GPIO.OUT)

def destroy():

GPIO.output (BUZ, GPIO.LOW)

GPIO.cleanup()

if \_\_name\_\_ =='\_\_main\_\_':

try:

while True:

IR\_State = GPIO.input(IR\_OUT)

if (IR\_State == True):

print ("OBJECT DETECTED")

lcd.clear()

lcd.message ="OBJECT DETECTED"

GPIO.output (BUZ, GPIO.HIGH)

time.sleep(0.5)

GPIO.output (BUZ, GPIO.LOW)

else:

lcd.clear()

lcd.message ="NO OBJECT"

time.sleep(0.5)

print ("NO OBJECT")

except KeyboardInterrupt:

destroy()

**PIR**

**Code RPi:**

#Raspberry Pi Libraries

import RPi.GPIO as GPIO #GPIO library

import time #library for sleep

import board

import digitalio

import adafruit\_character\_lcd.character\_lcd as characterlcd

#set mode as BCM

GPIO.setmode(GPIO.BCM)

# Modify this if you have a different sized character LCD

lcd\_columns = 16

lcd\_rows = 2

# Raspberry Pi Pin Config:

lcd\_rs = digitalio.DigitalInOut(board.D5)

lcd\_en = digitalio.DigitalInOut(board.D6)

lcd\_d4 = digitalio.DigitalInOut(board.D12)

lcd\_d5 = digitalio.DigitalInOut(board.D13)

lcd\_d6 = digitalio.DigitalInOut(board.D16)

lcd\_d7 = digitalio.DigitalInOut(board.D17)

# Initialise the lcd class

lcd = characterlcd.Character\_LCD\_Mono(

lcd\_rs, lcd\_en, lcd\_d4, lcd\_d5, lcd\_d6, lcd\_d7, lcd\_columns, lcd\_rows)

#set pins

PIR = 21

BUZ = 22

#setup pins at output

GPIO.setup(PIR, GPIO.IN)

GPIO.setup(BUZ, GPIO.OUT)

if \_\_name\_\_ =='\_\_main\_\_':

try:

while True:

PIR\_State = GPIO.input(PIR)

if (PIR\_State == True):

print ("Motion Detected")

lcd.clear()

lcd.message = "Motion Detected"

GPIO.output (BUZ, GPIO.HIGH)

time.sleep(0.5)

GPIO.output (BUZ, GPIO.LOW)

time.sleep(0.5)

else:

lcd.clear()

lcd.message = "NO Motion"

print ("No Motion")

time.sleep(0.5)

except KeyboardInterrupt:

GPIO.cleanup()

**Expt 4(DC Motor)**

**Code Arduino:**

intmotorPin=9;

void setup(){

pinMode(motorPin,OUTPUT);

Serial.begin(9600);

While(!Serial);

Serial.println(“Speed 0 to 255”);

}

void loop(){

if(Serial.available()){  
int speed =Serial.parseInt();

if(speed>=0&&speed<=255){

analogWrite(motorPin,speed);

}  
}  
}

--------------------------------------------------------

int DCMOTOR=13;

intdelayT=1000;

void setup(){

//put your main code here ,to run repeatedly;

digitalWrite(DCMOTOR,HIGH);

delay(delayT);

digitalWrite(LEDpin,LOW);

delay(delayT); }

**Code with RPi: (DC Motor)**

import RPi.GPIO as GPIO # Importing RPi library to use the GPIO pins

import time

EN1 = 25 # Initializing the GPIO pin 25 for the enable 1

IN1 = 26 # Initializing the GPIO pin 26 for input 1 of the motor driver

IN2 = 27 # Initializing the GPIO pin 27 for input 2 of the motor driver

GPIO.setmode(GPIO.BCM) # We are using the BCM pin numbering

GPIO.setup(EN1,GPIO.OUT) ## Declaring as EN1 output pin

GPIO.setup(IN1,GPIO.OUT) ## Declaring as IN1 output pin

GPIO.setup(IN2, GPIO.OUT) ## Declaring as IN2 output pin

#clear GPIOs

def destroy():

GPIO.output(25, False)

GPIO.output(26, False)

GPIO.output(27, False)

GPIO.cleanup()

def Clockwise():

GPIO.output(25, True)

GPIO.output(26, True)

GPIO.output(27, False)

def AntiClockwise():

GPIO.output(25, True)

GPIO.output(26, False)

GPIO.output(27, True)

def Stop():

GPIO.output(25, False)

GPIO.output(26, False)

GPIO.output(27, False)

if \_\_name\_\_ == '\_\_main\_\_': # Program start from here

try:

while True: # Loop will run forever

Clockwise()

time.sleep(2)

Stop()

time.sleep(1)

AntiClockwise()

time.sleep(2)

Stop()

time.sleep(1)

# If keyboard Interrupt (CTRL-C) is pressed

except KeyboardInterrupt:

destroy()

**Expt 5( MQTT )**

**Code Arduino:**

#include<DTH.h>

#define DHTPIN 2 // Define the pin where your DHT11 is connected

#define DHTTYPE DHT11 // Define the type of DHT sensor (DHT11 or DHT22)

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600);

dht.begin();

}

void loop() {

delay(2000); // Delay for 2 seconds between measurements

// Read temperature and humidity from the sensor

float temperature = dht.readTemperature();

float humidity = dhtreadHumidity();

//Check if any reads failed and exit early (to try again).

if (isnan(temperature) || isnan(humidity)) {

Serial.println(“Failed to read from DHT sensor”);

return;

}

// Print the temperature and humidity values to the serial monitor

Serial.print(“Temperature: “);

Serial.print(temperature);

Serial.println(“°C”);

Serial.print(“Humidity: “);

Serial.print(humidity);

Serial.println(“%”);

// You can also convert to Fahrenheit if needed

// float fahrenheit (temperature 1.8) +32; //Serial.print(“Temperature: “);

// Serial.print(fahrenheit);

// Serial.println(“°F”);

}

**CODE RPi : ( MQTT )**

# Install the MQTT Publisher

# sudo pip3 install paho-mqtt

# open the MQTT browser client in web browser

# http://www.hivemq.com/demos/websocket-client/

import os

import sys

import time

import board

import adafruit\_dht

import paho.mqtt.client as mqtt

import json

# Initial the dht device, with data pin connected to:

dhtDevice = adafruit\_dht.DHT11(board.D19, use\_pulseio=False)

sensor\_data = {'temperature': 0, 'humidity': 0}

MQTTServer = 'broker.mqttdashboard.com'

client = mqtt.Client()

client.connect(MQTTServer, 1883, 8000)

client.loop\_start( )

if \_\_name\_\_ == '\_\_main\_\_':

while True:

try:

# Print the values to the serial port

temperature = dhtDevice.temperature

humidity = dhtDevice.humidity

print("Temp: {:.1f} C Humidity: {}% ".format( temperature, humidity))

time.sleep(2.0)

sensor\_data['temperature'] = temperature

sensor\_data['humidity'] = humidity

# Sending humidity and temperature data to HIVEMQ

client.publish('RPI4\_MQTT', json.dumps(sensor\_data), 1)

time.sleep(5)

except RuntimeError as error:

# Errors happen fairly often, DHT's are hard to read, just keep going

print(error.args[0])

time.sleep(2.0)

continue

except KeyboardInterrupt:

client.loop\_stop()

client.disconnect()

print ('Exiting Program')

exit()

**Expt 7( Bluetooth )**

**CODE Ardiuno :**

#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

void setup() {

// Open serial communications and wait for port to open:

Serial.begin(9600);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

Serial.println("Goodnight moon!");

// set the data rate for the SoftwareSerial port

mySerial.begin(9600);

mySerial.println("Hello, world?");

}

void loop() { // run over and over

if (mySerial.available()) {

Serial.write(mySerial.read());

}

if (Serial.available()) {

mySerial.write(Serial.read());

}

}

**Expt 8 ( Zig Bee)**

**CODE Ardiuno :**

#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

void setup() {

// Open serial communications and wait for port to open:

Serial.begin(9600);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

Serial.println("Goodnight moon!");

// set the data rate for the SoftwareSerial port

mySerial.begin(9600);

mySerial.println("Hello, world?");

}

void loop() { // run over and over

if (mySerial.available()) {

Serial.write(mySerial.read());

}

if (Serial.available()) {

mySerial.write(Serial.read());

}

}

**CODE RPi : ( Zig Bee)**

# Install Python Serial Package

# sudo pip3 install pyserial

# Check the COM PORT Number

# sudo dmesg|grep tty

# Use ttyS0 for on-board Serial

# Use ttyUSB0 or ttyUSB1 for USB to serial converter

# after checking the com port number

import time

import serial

ser = serial.Serial(

port='/dev/ttyUSB2',

baudrate = 9600,

parity=serial.PARITY\_NONE,

stopbits=serial.STOPBITS\_ONE,

bytesize=serial.EIGHTBITS,

timeout=1

)

counter=0

if \_\_name\_\_ == "\_\_main\_\_":

try:

while True:

ser.write(str.encode('Write counter: %d \n'%(counter)))

time.sleep(1)

counter += 1

x=ser.readline().strip()

if len(x) != 0 :

print(x)

except KeyboardInterrupt:

ser.close()

print ('Exiting Program')