**DHT11**

import Adafruit\_DHT

import time

sensor = Adafruit\_DHT.DHT11

pin = 4

try:

while True:

humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin)

if humidity is not None and temperature is not None:

print(f'Temperature: {temperature:.2f}°C, Humidity: {humidity:.2f}%')

else:

print('Failed to get reading. Retrying...')

time.sleep(2)

except KeyboardInterrupt:

print('Exiting program.')

finally:

GPIO.cleanup()

**LED(normal)**

import RPi.GPIO as GPIO

from time import sleep

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

GPIO.setup(18, GPIO.OUT)

try:

while True:

GPIO.output(18, GPIO.HIGH)

print("LED ON")

sleep(1)

GPIO.output(18, GPIO.LOW)

print("LED OFF")

sleep(1)

except KeyboardInterrupt:

# Cleanup GPIO on keyboard interrupt

GPIO.cleanup()

**LED(5sec)**

import RPi.GPIO as GPIO

import random

import BlynkLib

from time import sleep

# Set up GPIO pin for the LED

LED\_PIN = 17

GPIO.setmode(GPIO.BCM)

GPIO.setup(LED\_PIN, GPIO.OUT)

# Blynk app authentication token

AUTH\_TOKEN = 'pJHE2cqG8ekVcu6uqV-5p06Fh-OJZjH1'

# Initialize the Blynk client

blynk = BlynkLib.Blynk(AUTH\_TOKEN)

# Virtual pin for the LED

# Function to handle virtual pin changes

@blynk.on("V0") # Replace '1' with your Virtual Pin number

def v0\_write\_handler(value):

if int(value[0]) == 1:

GPIO.output(LED\_PIN, GPIO.HIGH)

print('LED is ON')

sleep(5)

GPIO.output(LED\_PIN, GPIO.LOW)

else:

GPIO.output(LED\_PIN, GPIO.LOW)

print('LED is OFF')

while True:

blynk.run()

**LED(BLYNK)**

import BlynkLib

import RPi.GPIO as GPIO

from BlynkTimer import BlynkTimer

import time

# Blynk authentication token

BLYNK\_AUTH = 'pJHE2cqG8ekVcu6uqV-5p06Fh-OJZjH1'

# Set up GPIO

led\_pin = 17 # Replace with the actual GPIO pin number

GPIO.setmode(GPIO.BCM)

GPIO.setup(led\_pin, GPIO.OUT)

GPIO.output(led\_pin, GPIO.LOW) # Initialize LED as OFF

# Initialize Blynk

blynk = BlynkLib.Blynk(BLYNK\_AUTH)

# Define the Virtual Pin handler

@blynk.on(0) # Replace '1' with your Virtual Pin number

def v1\_write\_handler(value):

if int(value[0]) == 1:

GPIO.output(led\_pin, GPIO.HIGH) # Turn on the LED

print("LED ON")

else:

GPIO.output(led\_pin, GPIO.LOW) # Turn off the LED

print("LED OFF")

try:

while True:

blynk.run()

time.sleep(0.1)

except KeyboardInterrupt:

GPIO.cleanup()

# Create BlynkTimer Instance

timer = BlynkTimer()

# function to sync the data from virtual pins

@blynk.VIRTUAL\_WRITE("connected")

def blynk\_connected():

print("Hi, You have Connected to New Blynk2.O")

print(".............................")

print("...........MITADT Edge Computing Lab........")

time.sleep(2);

**DHT11 MQTT**

import Adafruit\_DHT

import time

import paho.mqtt.client as mqtt

# Define Variables

MQTT\_HOST = "192.168.249.12"

MQTT\_PORT = 1883

MQTT\_KEEPALIVE\_INTERVAL = 5

MQTT\_TOPIC = "hello/world"

MQTT\_MSG = ""

# Define on\_connect event Handler

def on\_connect(mosq, obj, rc):

print ("Connected to MQTT Broker")

# Define on\_publish event Handler

def on\_publish(client, userdata, mid):

print ("Message Published...")

# Initiate MQTT Client

mqttc = mqtt.Client()

# Register Event Handlers

mqttc.on\_publish = on\_publish

mqttc.on\_connect = on\_connect

# Connect with MQTT Broker

mqttc.connect(MQTT\_HOST, MQTT\_PORT, MQTT\_KEEPALIVE\_INTERVAL)

# Publish message to MQTT Topic

# Set up the sensor

sensor = Adafruit\_DHT.DHT11

pin = 4 # GPIO pin where the DHT11 sensor is connected

try:

while True:

# Attempt to get sensor reading

humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin)

# Check if reading was successful

if humidity is not None and temperature is not None:

MQTT\_MSG = "Temperature: "+ str(temperature) + "°C, Humidity: " + str(humidity) + "%"

mqttc.publish(MQTT\_TOPIC,MQTT\_MSG)

else:

MQTT\_MSG = "Failed to get reading. Retrying..."

mqttc.publish(MQTT\_TOPIC,MQTT\_MSG)

# Wait for a short interval before the next reading

time.sleep(2)

except KeyboardInterrupt:

print('Exiting program.')

finally:

# Cleanup GPIO on exit

GPIO.cleanup()

# Disconnect from MQTT\_Broker

mqttc.disconnect()