Lab: EDGE COMPUTING LAB (23A39605)

College: Andhra

Engineering College

5. MQTT-Based Edge Communication o Setup publisher/subscriber model for edge-to-cloud communication

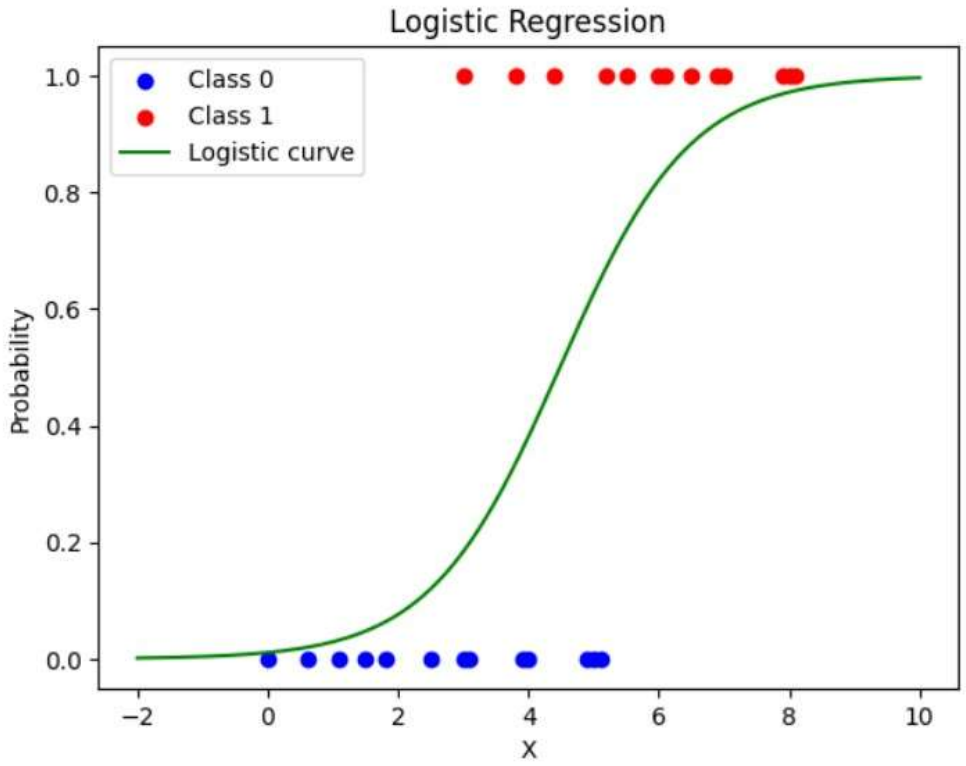
Test Case 01 Using Local Inference From Cloud

Step 01 (Take Any ML Model)

```
import numpy as np
from matplotlib import pyplot as plt
from sklearn.linear_model import LogisticRegression

# Data points
x1 = np.array([0,0.6,1.1,1.5,1.8,2.5,3,3.1,3.9,4,4.9,5,5.1])
y1 = np.zeros_like(x1)

x2 = np.array([3,3.8,4.4,5.2,5.5,6.5,6,6.1,6.9,7,7.9,8,8.1])
y2 = np.ones_like(x2)

# Combine data
X = np.concatenate([x1, x2]).reshape(-1,1)
y = np.concatenate([y1, y2])

# Scatter plot
plt.scatter(x1, y1, color='blue', label='Class 0')
plt.scatter(x2, y2, color='red', label='Class 1')

# Train logistic regression
model = LogisticRegression()
model.fit(X, y)

print("Intercept (b0):", model.intercept_)
print("Coefficient (b1):", model.coef_)

# Logistic function
def logistic(classifier, x):
    return 1 / (1 + np.exp(-(classifier.intercept_ + classifier.coef_ * x)))

# Plot logistic curve
x_vals = np.linspace(-2, 10, 200).reshape(-1,1)
y_vals = logistic(model, x_vals)
plt.plot(x_vals, y_vals, color='green', label='Logistic curve')
```

```
plt.xlabel("X")
plt.ylabel("Probability")
plt.title("Logistic Regression")
plt.legend()
plt.show()

# Predict probability for a new value
pred = model.predict_proba(np.array([[8]]))
print("Prediction for x=8:", pred)
```

```
Intercept (b0): [-4.50163542]
Coefficient (b1): [[1.00401882]]
```




```
Prediction for x=8: [[0.02845634 0.97154366]]
```

```
pr01=model.predict([[8]])
print(pr01)
pr02=model.predict([[2]])
print(pr02)
```

Output

```
[1.]
[0.]
```

Step 02 (Save Model Inference)

```
import joblib

# Save model to file
joblib.dump(model, "logistic_model.pkl")
print("Model saved successfully")
```

output
Model loaded successfully

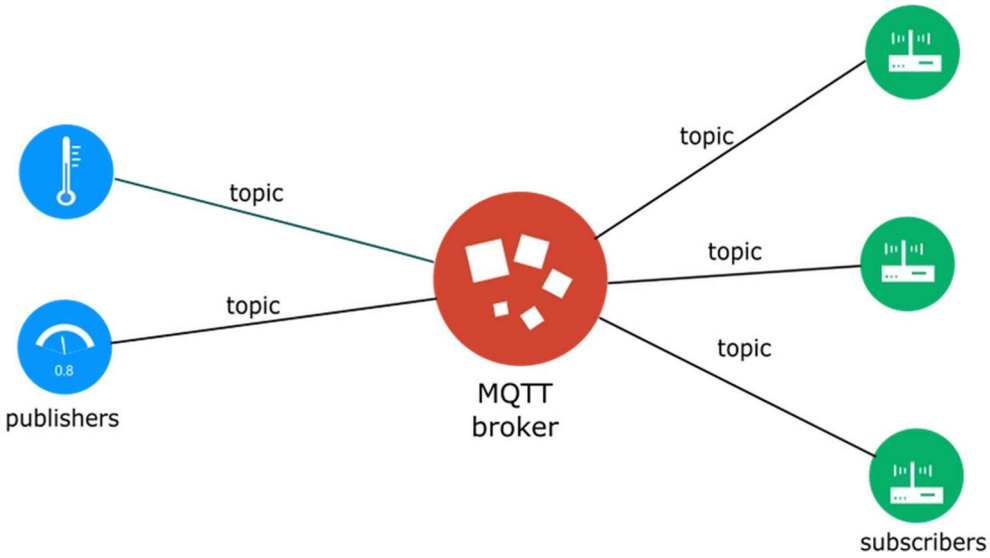
Using That Inference Test Predict Result

```
# Predict probability
x_new = np.array([[8]])
# proba = model.predict_proba(x_new)
prediction = model01.predict(x_new)

# print("Probabilities [Class 0, Class 1]:", proba)
print("Predicted class:", prediction)
```

Output :

Predicted class: [1.]

Step 03 (Using MQTT topic to predict do Actionable Events At Edge Device or Frog Lavel)



Server Side (Edge Lavel like Pi or TPU or GPU Boards)

```
import paho.mqtt.client as mqtt
import joblib
import numpy as np
```

```
# Load model
model01 = joblib.load("logistic_model.pkl")
print("Model loaded")

client01 = mqtt.Client()

def on_message(client01, userdata, message):
    payload = message.payload.decode().strip()
    print("Received:", payload)

    # ---------- COMMAND HANDLING ----------
    if payload.lower() == 'on':
        client01.publish('test/topic/result', 'led is on')
        client01.publish('test/topic','led is on')

    elif payload.lower() == 'off':
        client01.publish('test/topic/result', 'led is off')
        client01.publish('test/topic','led is off')

    elif payload.lower() == 'dh11':
        client01.publish('test/topic/result', 'DHT11 command received')

    # ---------- ML HANDLING ----------
    else:
        try:
            # Assume numeric input for ML
            value = float(payload)
            x_new = np.array([[value]])

            prediction = model01.predict(x_new)[0]

            response = f"ML Prediction for {value} = {int(prediction)}"
            print(response)

            client01.publish('test/topic/result', response)
            client01.publish('test/topic', response)


        except ValueError:
            client01.publish('test/topic/result', 'Invalid input')

# MQTT setup
client01.on_message = on_message
client01.connect('127.0.0.1', 1883)
client01.subscribe('test/topic')

print("Waiting for MQTT messages...")
```

```
client01.loop_forever()
```

Output :

```
Model loaded
Waiting for MQTT messages...
/tmp/ipykernel_9171/2463887080.py:9: DeprecationWarning: Callback API version 1 is
deprecated, update to latest version
  client01 = mqtt.Client()
Received: dh11
Received: on
Received: led is on
Received: off
Received: led is off
Received: 8
ML Prediction for 8.0 = 1
Received: ML Prediction for 8.0 = 1
```

Client Side

```
import paho.mqtt.client as mqtt
from time import sleep

client01 = mqtt.Client()
def on_message(client01, userdata, message):
    msg=message.topic
    payload = message.payload.decode().strip()
    print(payload)
def on_publish(client01, userdata, mid):
    print("✓ Published | mid:", mid)

client01.on_publish=on_publish
client01.connect('127.0.0.1', 1883)

client01.on_message=on_message
client01.subscribe('test/topic')
client01.loop_start()
#client01.loop_forever()

# ---- Publish commands ----
client01.publish("test/topic", "dh11")
sleep(2)

client01.publish("test/topic", "on")
sleep(2)
```

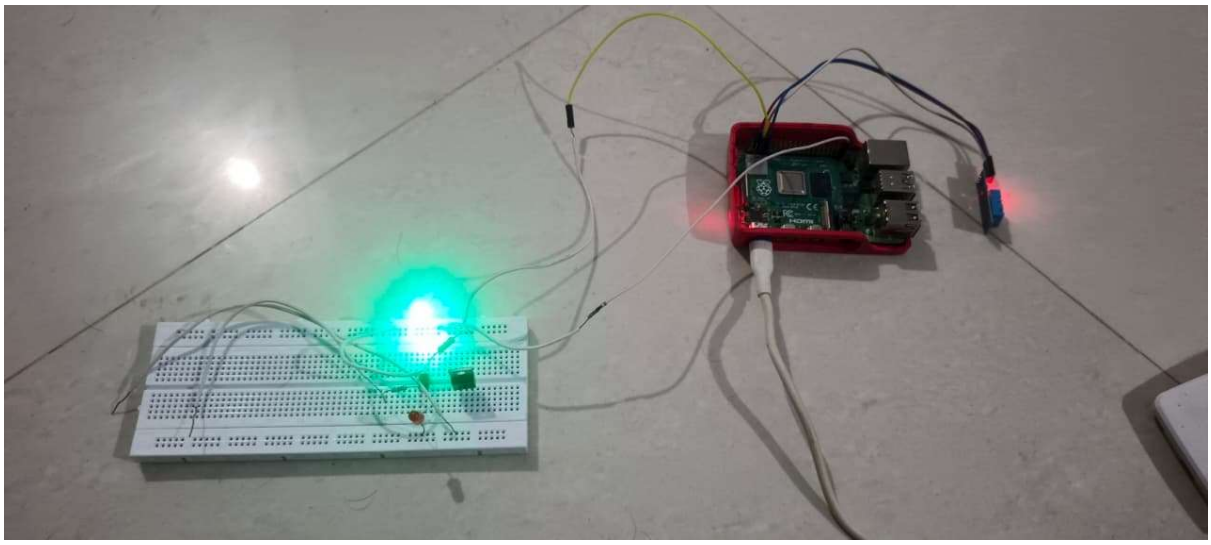
```
client01.publish("test/topic", "off")
sleep(2)

client01.publish("test/topic", "8")
sleep(2)

sleep(20)
client01.loop_stop()
client01.disconnect()
```

Output

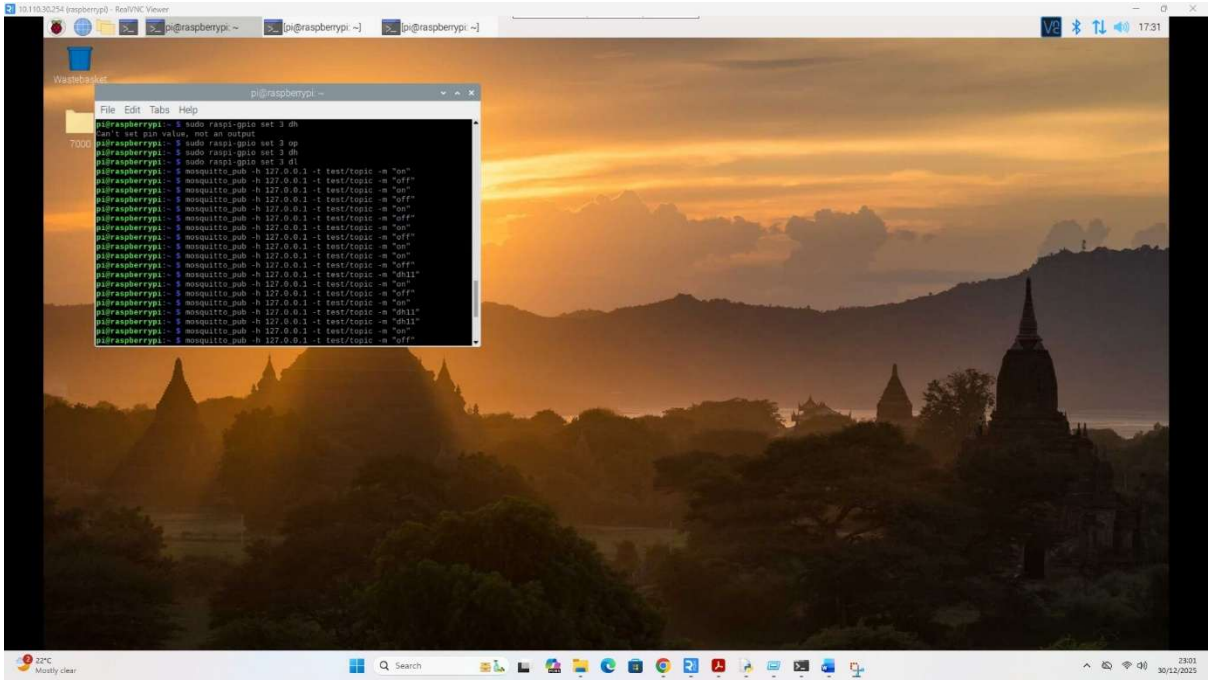
```
✓ Published | mid: 2
dh11
✓ Published | mid: 3
on
led is on
✓ Published | mid: 4
off
led is off
✓ Published | mid: 5
8
ML Prediction for 8.0 = 1
<MQTTErrorCode.MQTT_ERR_SUCCESS: 0>
```

Test Case 02 Using Edge to Actionable Taks

Connections View



Publish Topic without Python



Pi Side

```
import paho.mqtt.client as mqtt
from time import sleep
import RPi.GPIO as GPIO

import time
import board
import adafruit_dht

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
led=3
GPIO.setup(led,GPIO.OUT)

dhtDevice = adafruit_dht.DHT11(board.D4)
last_read_time = 0
READ_INTERVAL = 2   # seconds (VERY IMPORTANT)



client01 = mqtt.Client()
def on_message(client01, userdata, message):
    msg=message.topic
    payload = message.payload.decode().strip()

    if payload.lower()=='on':
```

```
        GPIO.output(led,1)
        sleep(READ_INTERVAL)
        print(payload)
        client01.publish('test/topic','led is on')
    elif payload.lower()=='off':
        print(payload)
        GPIO.output(led,0)
        client01.publish('test/topic','led is off')
        sleep(READ_INTERVAL)
    elif payload.lower()=='dh11':
#         dhtDevice = adafruit_dht.DHT11(board.D4)
        try:
            temperature = dhtDevice.temperature
            humidity = dhtDevice.humidity
            print(f"Temp={temperature}°C  Humidity={humidity}%")
            msg = f"Temp={temperature},Humidity={humidity}"
            client01.publish('test/topic',msg)
        except :
            print('Exception')

#     dir(message)
#     print("All attributes:", dir(message.payload.decode()))

# client.on_publish=on_publish
client01.connect('10.110.30.254', 1883)

client01.on_message=on_message
client01.subscribe('test/topic')
# client01.loop_start()
client01.loop_forever()
```

Output

```
Temp=27.7°C  Humidity=82%
on
off
```

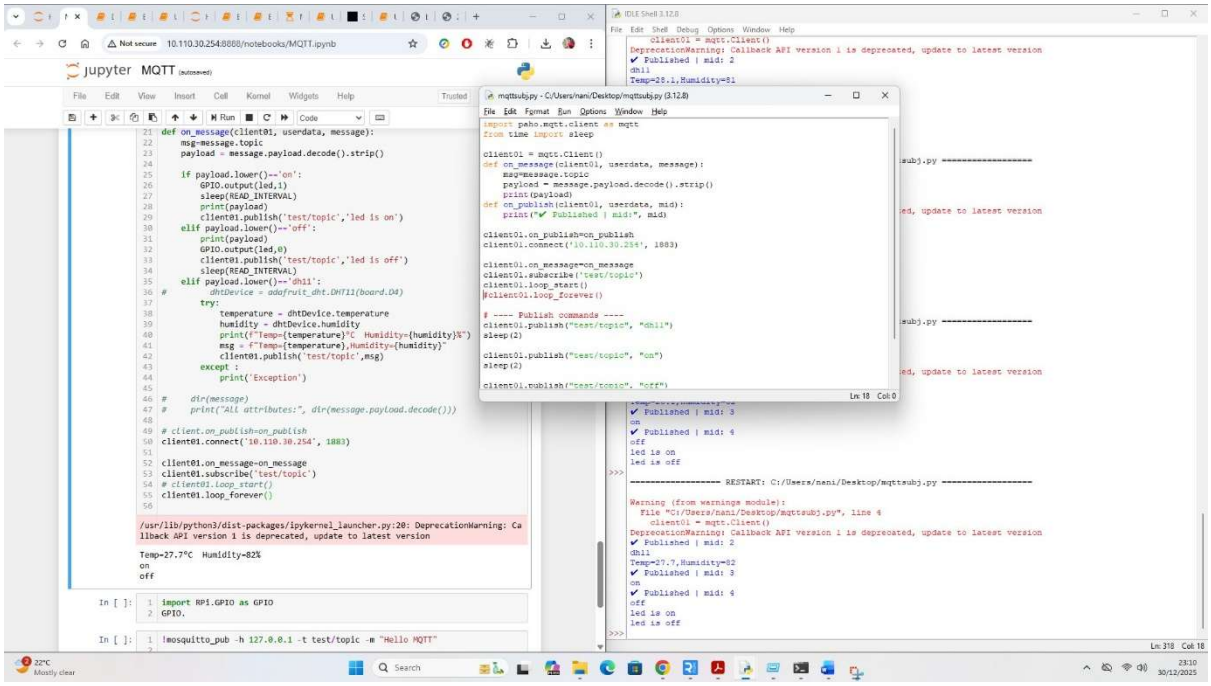
Client Side Over Network

```
import paho.mqtt.client as mqtt
from time import sleep

client01 = mqtt.Client()
def on_message(client01, userdata, message):
    msg=message.topic
    payload = message.payload.decode().strip()
    print(payload)
```

```
def on_publish(client01, userdata, mid):
    print("✓ Published | mid:", mid)

client01.on_publish=on_publish
client01.connect('10.110.30.254', 1883)

client01.on_message=on_message
client01.subscribe('test/topic')
client01.loop_start()
#client01.loop_forever()

# ---- Publish commands ----
client01.publish("test/topic", "dh11")
sleep(2)

client01.publish("test/topic", "on")
sleep(2)

client01.publish("test/topic", "off")

sleep(20)
client01.loop_stop()
client01.disconnect()
```

Output:

```
================== RESTART: C:/Users/nani/Desktop/mqttsubj.py ==================

Warning (from warnings module):
  File "C:/Users/nani/Desktop/mqttsubj.py", line 4
    client01 = mqtt.Client()
DeprecationWarning: Callback API version 1 is deprecated, update to latest version
✓ Published | mid: 2
dh11
Temp=27.7,Humidity=82
✓ Published | mid: 3
on
✓ Published | mid: 4
off
led is on
led is off
```



Use Cases :

Low latency Case under this bellow

Autonomous / Semi-Autonomous Vehicles