**WEEK – 09 MQTT - Publish**

**PRELAB QUESTIONS - 09**

1. What are the applications of MQTT protocol?  
   Ans: MQTT is primarily used in lightweight communication environments like IoT devices, for telemetry in vehicles, real-time updates to apps, and remote sensor networks. It enables efficient and reliable data transmission even with limited network bandwidth and power resources.
2. What is meant by Topic in MQTT?  
   Ans: In MQTT, a topic is a UTF-8 string that the broker uses to filter messages for each connected client. Topics allow clients to publish messages to specific channels and subscribe to receive updates from those channels, enabling organized and targeted data distribution.
3. What is MQTT-SN? Can we use this over a Zigbee based network?  
   Ans: MQTT-SN (MQTT for Sensor Networks) is a variation of MQTT designed for wireless sensor networks where the TCP/IP stack might be too cumbersome. MQTT-SN is suitable for non-TCP/IP networks like Zigbee, as it can operate over other transport protocols, making it highly useful in constrained environments.
4. Illustrate the MQTT message format.  
   Ans: An MQTT message consists of a fixed header (2 bytes), a variable header (which may include things like Packet Identifier and Topic Name), and a payload section. The fixed header includes control flags, the message type, and remaining length, making MQTT messages compact and efficient for network transmission.
5. What is the significance of Mosquitto? Give the list of some popular IoT platforms that support MQTT.  
   Ans: Mosquitto is an open-source MQTT broker that facilitates the lightweight MQTT protocol for IoT applications, providing both a broker for networking and tools for generating MQTT messages. Popular IoT platforms that support MQTT include AWS IoT, Google Cloud IoT Core, Microsoft Azure IoT Hub, and IBM Watson IoT.

**Lab programs:**

Write a program to implement MQTT-publish Protocol.

Code:

import paho.mqtt.publish as publish

publish.single("iot/home", "iotlab", hostname="test.mosquitto.org")

print("Done")

**OUTPUT:**

Mqtt-publish

Done

**WEEK – 10 MQTT - Subscribe**

**PRELAB QUESTIONS - 10**

1. What are the functions of MQTT brokers?

A: MQTT brokers facilitate communication between MQTT clients by routing messages, managing subscriptions, ensuring QoS, handling authentication and authorization, maintaining session state, retaining messages, and supporting scalability and load balancing.

2. How to subscribe to a topic in MQTT?

A: To subscribe to a topic in MQTT, a client sends a SUBSCRIBE message to the MQTT broker, specifying the topic it wants to subscribe to along with the desired Quality of Service (QoS) level. Upon receiving the SUBSCRIBE message, the broker adds the client to the list of subscribers for that topic and starts delivering messages published on that topic to the client.

3. What is the naming pattern for the MQTT topic?  
A: MQTT topics are hierarchical and structured like a path using forward slashes (/) to separate levels, similar to a filesystem path. This structure allows for precise and flexible topic filtering. Examples include "home/livingroom/temperature" or "device/status/updates".

4. What is the purpose of RabbitMQ?.

A: RabbitMQ is a message broker that facilitates communication between distributed systems by enabling asynchronous messaging. Its purpose is to efficiently route, buffer, and process messages between applications and services, aiding in building scalable and decoupled architectures.

**Lab Programs:**

1. Write a program to implement MQTT-subscribe Protocol.

Code:

import paho.mqtt.client as mqtt

def on\_connect(client, userdata, flags, rc):

print("Connected")

client.subscribe("iot/home")

def on\_message(client, userdata, msg):

print(msg.topic+""+str(msg.payload))

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_message = on\_message

client.connect("test.mosquitto.org", 1883, 60)

client.loop\_forever()

**OUTPUT:**

Mqtt-subscribe

Connected

iot/home iotlab

iot/home iotlab

iot/home iotlab

**WEEK – 11 LORA**

**PRELAB QUESTIONS – 11**

1.What is the range of LoRa?

A: In optimal conditions, LoRa technology can achieve ranges of up to 10 kilometers or more in rural or suburban environments. However, in urban environments with obstacles and interference, the range may be limited to a few kilometers or less.

2. Write a few applications of LoRa.

A: LoRa technology finds applications in various fields:

1. Smart Agriculture: Monitoring soil moisture, temperature, and crop health in vast agricultural fields.

2. Smart Cities: Tracking assets, monitoring parking spaces, and controlling street lighting.

3. Industrial IoT: Collecting data from sensors for predictive maintenance in factories and industrial equipment.

4. Environmental Monitoring: Measuring air quality, water levels, and pollution in remote areas.

5. Asset Tracking: Tracking vehicles, containers, and goods in logistics and supply chain management.

6. Smart Metering: Remote monitoring of utility meters for electricity, water, and gas consumption.

7. Building Automation: Controlling and monitoring heating, ventilation, and air conditioning systems in commercial and residential buildings.

3. How is LoRa different from ZigBee?

A: LoRa provides long-range, low-power communication suited for applications like smart agriculture and asset tracking. ZigBee offers shorter-range, higher-data-rate communication, ideal for home automation and industrial control.

4. What is the approved frequency band for LoRa in India?

A: The approved frequency band for LoRa in India is 865-867 MHz.

5.Can we use LoRa for Geolocation?

A: Yes, LoRa can be used for geolocation by employing techniques such as time difference of arrival (TDoA) or signal strength-based triangulation.

**Lab Programs:**

**Lab Programs**

**Lora\_Node1\_sending\_data\_to\_gateway**

#include <dht11.h> #include <SPI.h>

#include <LoRa.h>

int P = 6; //Powersensor - 6 int R = 5; //Reed Switch - 5 int X = 7; //PIR sensor - 7

int T = A2; // TEMPERATURE - A2 has already been defined as Analog Pin 2 in arduino lib int H = A5; //HUMIDITY

int L = A4; //LIGHT

float CELSIUS, HUM, LIGHT;

String data, DOOR, MOT, POW; dht11 DHT11;

#define Lora\_kit 001

#define Lora\_Node 001

void setup()

{

pinMode(P, INPUT); pinMode(R, INPUT); pinMode(X, INPUT); Serial.begin(9600);

Serial.println("-------Lora Kit ");

if (!LoRa.begin(433E6))

{

Serial.println("Starting LoRa failed!");

Serial.println("Please check the Lora Connections"); while (1);

}

LoRa.setSyncWord(0xA1); // The sync word assures you don't get LoRa messages from other LoRa transceivers

}

void TEMPERATURE()

{

int value\_temp = analogRead(T); //Read analog value of temperature sensor output from pin A2 delay(10);

value\_temp = analogRead(T); delay(10);

float millivolts\_temp = (value\_temp / 1023.0) \* 5000; //convert it to milli volts output ([actual temperature output from sensor] \* [Input voltage (5V = 5000mV)] / [Resolution of ADC 2^10 = 1024])

CELSIUS = millivolts\_temp / 10; Serial.print("Temperature : "); Serial.println(CELSIUS);

}

void HUMIDITY()

{

int chk = DHT11.read(H); HUM = DHT11.humidity; Serial.print("Humidity : "); Serial.println(HUM);

}

void LIG()

{

int value\_lig = analogRead(L); delay(10);

value\_lig = analogRead(L);

float volts\_lig = (value\_lig / 1023.0) \* 5;

LIGHT = 500/(4\*((5-volts\_lig)/volts\_lig)); // calculate the Lux = 500/[R1 \* ((Vin - Vsense)/Vsense)] Serial.print("Light : ");

Serial.print(LIGHT); Serial.println(" Lux");

}

void POWER()

{

if (digitalRead(P) == LOW) // if output form sensor is '0' then print NO power

{

POW = "OFF";

Serial.print("Power : "); Serial.println(POW);

}

else

{

POW = "ON";

Serial.print("Power : "); Serial.println(POW)

}

}

Void REED()

{

if (digitalRead(R) == LOW)

{

DOOR = "OPEN";

Serial.print("Reed/Door : "); Serial.println(DOOR);

}

else

{

DOOR = "CLOSE";

Serial.print("Reed/Door : "); Serial.println(DOOR);

}

}

void PIR()

{

if (digitalRead(X) == LOW)

{

MOT = "YES";

Serial.print("PIR : "); Serial.println(MOT);

}

else

{

MOT = "NO";

Serial.print("PIR : "); Serial.println(MOT);

}

Serial.println();

}

void loop()

{

TEMPERATURE(); HUMIDITY(); LIG();

REED();

POWER();

PIR();

delay(500); LoRa.beginPacket(); LoRa.print("Lora kit: "); LoRa.print(Lora\_kit); LoRa.print(" Lora Node: "); LoRa.println(Lora\_Node); LoRa.print("Temperature: "); LoRa.println(CELSIUS);

LoRa.print("Humidity: "); LoRa.println(HUM); LoRa.print("Light: "); LoRa.println(LIGHT); LoRa.print("POWER: "); LoRa.println(POW); LoRa.print("DOOR: "); LoRa.println(DOOR); LoRa.print("PIR: "); LoRa.println(MOT); LoRa.endPacket(); delay(5000);

}

**LoRa\_Node2\_sending\_data\_to\_gateway**

#include <dht11.h> #include <SPI.h> #include <LoRa.h>

int P = 6; //Powersensor - 6 int R = 5; //Reed Switch - 5 int X = 7; //PIR sensor - 7

int T = A2; // TEMPERATURE - A2 has already been defined as Analog Pin 2 in arduino lib int H = A5; //HUMIDITY

int L = A4; //LIGHT

float CELSIUS, HUM, LIGHT;

String data, DOOR, MOT, POW; dht11 DHT11;

#define Lora\_kit 001 // Enter Your Lora kit #define Lora\_Node 002 // Enter Your Lora Node

void setup()

{

pinMode(P, INPUT); pinMode(R, INPUT); pinMode(X, INPUT); Serial.begin(9600);

Serial.println("-------Lora Kit ");

if (!LoRa.begin(433E6))

{

Serial.println("Starting LoRa failed!"); Serial.println("Please check the Lora Connections"); while (1);

}

LoRa.setSyncWord(0xA1); // The sync word assures you don't get LoRa messages from other LoRa transceivers

}

void TEMPERATURE()

{

int value\_temp = analogRead(T); //Read analog value of temperature sensor output from pin A2 delay(10);

value\_temp = analogRead(T); delay(10);

float millivolts\_temp = (value\_temp / 1023.0) \* 5000; //convert it to milli volts output ([actual temperature output from sensor] \* [Input voltage (5V = 5000mV)] / [Resolution of ADC 2^10 = 1024])

CELSIUS = millivolts\_temp / 10; Serial.print("Temperature : "); Serial.println(CELSIUS);

}

void HUMIDITY()

{

int chk = DHT11.read(H); HUM = DHT11.humidity; Serial.print("Humidity : "); Serial.println(HUM);

}

void LIG()

{

int value\_lig = analogRead(L); delay(10);

value\_lig = analogRead(L);

float volts\_lig = (value\_lig / 1023.0) \* 5;

LIGHT = 500/(4\*((5-volts\_lig)/volts\_lig)); // calculate the Lux = 500/[R1 \* ((Vin - Vsense)/Vsense)] Serial.print("Light : ");

Serial.print(LIGHT); Serial.println(" Lux");

}

void POWER()

{

if (digitalRead(P) == LOW) // if output form sensor is '0' then print NO power

{

POW = "OFF";

Serial.print("Power : "); Serial.println(POW);

}

else

{

POW = "ON";

Serial.print("Power : "); Serial.println(POW);

}

}

void REED()

{

if (digitalRead(R) == LOW)

{

DOOR = "OPEN";

Serial.print("Reed/Door : "); Serial.println(DOOR);

}

else

{

DOOR = "CLOSE";

Serial.print("Reed/Door : "); Serial.println(DOOR);

}

}

void PIR()

{

if (digitalRead(X) == LOW)

{

MOT = "YES";

Serial.print("PIR : "); Serial.println(MOT);

}

else

{

MOT = "NO";

Serial.print("PIR : "); Serial.println(MOT);

}

Serial.println();

}

void loop()

{

TEMPERATURE(); HUMIDITY();

LIG();

REED();

POWER();

PIR();

delay(500); LoRa.beginPacket(); LoRa.print("Lora kit: "); LoRa.print(Lora\_kit); LoRa.print(" Lora Node: "); LoRa.println(Lora\_Node); LoRa.print("Temperature: "); LoRa.println(CELSIUS); LoRa.print("Humidity: "); LoRa.println(HUM); LoRa.print("Light: "); LoRa.println(LIGHT); LoRa.print("POWER: "); LoRa.println(POW); LoRa.print("DOOR: "); LoRa.println(DOOR); LoRa.print("PIR: "); LoRa.println(MOT); LoRa.endPacket(); delay(5000);

}

**LoRa\_Node3\_sending\_data\_to\_gateway**

#include <dht11.h> #include <SPI.h> #include <LoRa.h>

int P = 6; //Powersensor - 6 int R = 5; //Reed Switch - 5 int X = 7; //PIR sensor - 7

int T = A2; // TEMPERATURE - A2 has already been defined as Analog Pin 2 in arduino lib int H = A5; //HUMIDITY

int L = A4; //LIGHT

float CELSIUS, HUM, LIGHT;

String data, DOOR, MOT, POW; dht11 DHT11;

#define Lora\_kit 001

#define Lora\_Node 003

void setup()

{

pinMode(P, INPUT); pinMode(R, INPUT); pinMode(X, INPUT); Serial.begin(9600);

Serial.println("-------Lora Kit ");

if (!LoRa.begin(433E6))

{

Serial.println("Starting LoRa failed!"); Serial.println("Please check the Lora Connections"); while (1);

}

LoRa.setSyncWord(0xA1); // The sync word assures you don't get LoRa messages from other LoRa transceivers

}

void TEMPERATURE()

{

int value\_temp = analogRead(T); //Read analog value of temperature sensor output from pin A2 delay(10);

value\_temp = analogRead(T); delay(10);

float millivolts\_temp = (value\_temp / 1023.0) \* 5000; //convert it to milli volts output ([actual temperature output from sensor] \* [Input voltage (5V = 5000mV)] / [Resolution of ADC 2^10 = 1024])

CELSIUS = millivolts\_temp / 10; Serial.print("Temperature : "); Serial.println(CELSIUS);

}

void HUMIDITY()

{

int chk = DHT11.read(H); HUM = DHT11.humidity; Serial.print("Humidity : "); Serial.println(HUM);

}

void LIG()

{

int value\_lig = analogRead(L); delay(10);

value\_lig = analogRead(L);

float volts\_lig = (value\_lig / 1023.0) \* 5;

LIGHT = 500/(4\*((5-volts\_lig)/volts\_lig)); // calculate the Lux = 500/[R1 \* ((Vin - Vsense)/Vsense)] Serial.print("Light : ");

Serial.print(LIGHT); Serial.println(" Lux");

}

void POWER()

{

if (digitalRead(P) == LOW) // if output form sensor is '0' then print NO power

{

POW = "OFF";

Serial.print("Power : "); Serial.println(POW);

}

else

{

POW = "ON";

Serial.print("Power : "); Serial.println(POW);

}

}

void REED()

{

if (digitalRead(R) == LOW)

{

DOOR = "OPEN";

Serial.print("Reed/Door : "); Serial.println(DOOR);

}

else

{

DOOR = "CLOSE";

Serial.print("Reed/Door : "); Serial.println(DOOR);

}

}

void PIR()

{

if (digitalRead(X) == LOW)

{

MOT = "YES";

Serial.print("PIR : "); Serial.println(MOT);

}

else

{

MOT = "NO";

Serial.print("PIR : "); Serial.println(MOT);

}

Serial.println();

}

void loop()

{

TEMPERATURE(); HUMIDITY(); LIG();

REED();

POWER();

PIR();

delay(500); LoRa.beginPacket(); LoRa.print("Lora kit: "); LoRa.print(Lora\_kit); LoRa.print(" Lora Node: "); LoRa.println(Lora\_Node); LoRa.print("Temperature: "); LoRa.println(CELSIUS); LoRa.print("Humidity: "); LoRa.println(HUM); LoRa.print("Light: "); LoRa.println(LIGHT); LoRa.print("POWER: "); LoRa.println(POW); LoRa.print("DOOR: "); LoRa.println(DOOR); LoRa.print("PIR: "); LoRa.println(MOT); LoRa.endPacket(); delay(5000);

}

**Receiving\_the\_data\_from\_LoRa\_nodes**

#include <SPI.h> #include <LoRa.h>

#define ss 15

#define rst 16

#define dio0 2

void setup()

{

Serial.begin(115200);

Serial.println("-------Lora Gateway ");

LoRa.setPins(ss, rst, dio0); if (!LoRa.begin(433E6))

{

Serial.println("Starting LoRa failed!"); Serial.println("Please check the Lora Connections"); while (1);

}

LoRa.setSyncWord(0xA1); // The sync word assures you don't get LoRa messages from other LoRa transceivers

}

void loop()

{

int packetSize = LoRa.parsePacket(); if (packetSize)

{

// read packet

while (LoRa.available())

{

Serial.print((char)LoRa.read());

}

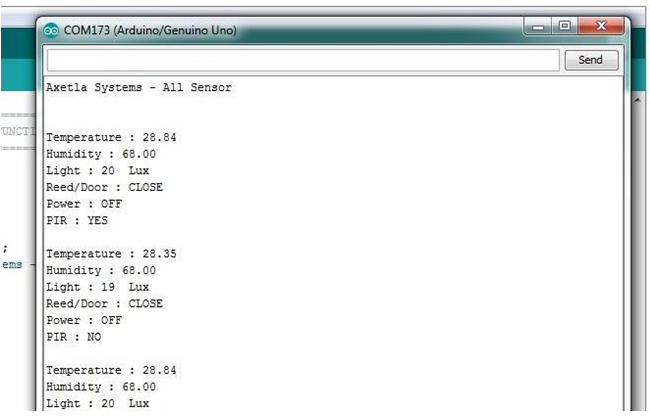
Serial.print("Signal Strength: "); Serial.println(LoRa.packetRssi()); Serial.println();

Serial.println(); Serial.println();

}

}

**Output:**

****

**WEEK – 12 Industrial IOT**

**PRELAB QUESTIONS – 12**

1. Define the purpose of using Node-Red.

A: The purpose of using Node-RED is to provide a visual programming tool for wiring together hardware devices, APIs, and online services in new and interesting ways. It simplifies the development of IoT applications by offering a flow-based programming environment where users can easily create, deploy, and manage event-driven applications.

2. List the different types of nodes present in Node-Red.

A: Node-RED provides various types of nodes including Input Nodes, Output Nodes, Function Nodes, Debug Nodes, Trigger Nodes, Template Nodes, Dashboard Nodes, Social Nodes, Storage Nodes, and Analysis Nodes.

3. What do you mean by IIoT and what is its relevance to Industry 4.0?

A: IIoT (Industrial Internet of Things) involves connecting industrial devices and equipment to collect data and enable automation. It's relevant to Industry 4.0 as it drives the digitization and optimization of industrial processes, fostering smarter factories and supply chains.

4. What are vibration sensors?

A: Vibration sensors, also known as accelerometers, are devices used to measure vibrations or accelerations in various industrial, automotive, and structural applications. They detect and quantify vibrations in machinery, structures, or environments and provide valuable data for condition monitoring, predictive maintenance, and fault diagnosis. Vibration sensors come in different types, including piezoelectric, piezoresistive, and capacitive, and they can measure vibrations across a range of frequencies and amplitudes.

**Lab Programs:**

Client-Side sensor code:

import time

import random

import requests

# Simulate sensor data collection

def collect\_sensor\_data():

# Simulate vibration sensor data

vibration\_data = random.uniform(0, 1)

return vibration\_data

# Transmit data to server

def transmit\_data\_to\_server(data):

server\_url = 'http://localhost:5000/data' # Assuming server is running locally

payload = {'vibration\_data': data}

try:

response = requests.post(server\_url, json=payload)

if response.status\_code == 200:

print("Data transmitted successfully!")

else:

print("Error transmitting data:", response.status\_code)

except Exception as e:

print("Exception occurred while transmitting data:", str(e))

# Main loop

while True:

# Collect sensor data

sensor\_data = collect\_sensor\_data()

# Transmit data to server

transmit\_data\_to\_server(sensor\_data)

# Sleep for a specified interval before collecting more data

time.sleep(5) # Sleep for 5 seconds (adjust as needed)

Server-side Flask code:

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route('/data', methods=['POST'])

def receive\_data():

data = request.get\_json()

vibration\_data = data.get('vibration\_data')

print("Received sensor data:", vibration\_data)

return 'Data received successfully!', 200

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

app.run(debug=True)

**Output:**

Client-side (Sensor) output:

Data transmitted successfully!

Data transmitted successfully!

Data transmitted successfully!

...

Server-side output (Flask server console):

Running on http://127.0.0.1:5000/

Received sensor data: 0.789123

Received sensor data: 0.456789

Received sensor data: 0.123456

...