

# PRACTICAL MQTT-SN EDGE GATEWAY INTEGRATION WITH CLOUD SERVER AND END TO END SENSOR DATA TRANSMISSION FOR IOT APPLICATIONS

M.OBULA REDDY<sup>1\*</sup>, J.B. SEVENTLINE<sup>2</sup>

<sup>1</sup>Research Scholar, Department of Electronics and Communication Engineering, GITAM (Deemed to be University), Vishakhapatnam, India

<sup>1</sup>moreddygitam@gmail.com

<sup>2</sup>Professor, Department of Electronics and Communication Engineering, GITAM (Deemed to be University), Vishakhapatnam, India

<sup>2</sup>seventline.joseph@gitam.edu

## ABSTRACT

In traditional two tier MQTT IoT Network, Sensors and related IoT platform directly connected to the Cloud MQTT Server. In this architecture sending sensor data to the cloud MQTT-Server is not efficient due to frequently sending of data to the cloud server, in the cloud storage server memory related issues will occur due to large volume of sensor data. To reduce the data volume traffic local edge MQTT-SN gateway incorporated into the network to filter out some of the sensor data before sending it to the Cloud MQTT Server. MQTT-SN is MQ Telemetry Transport Interface protocol is used in the sensor devices and local edge gateway due to low power and low message overhead compared to other protocols. In this paper we addressed practical MQTT-SN gateway integration with private secure Cloud MQTT Server, Sensor data transmission to the Internet of Thing (IOT) application through the EDGE MQTT-SN gateway, MQTT Server. MQTT-SN gateway to sensor device, Gateway to MQTT-Server and MQTT-Server to IoT Application Signaling analysis. Real time sensor data transmission to the IoT application through the Edge gateway and MQTT-Server. Finally, MQTT based IoT application received publish message from the MQTT serve, decode the publish msg and stored sensor data in the SQL Lite database for further data analytics.

**Keywords:** *IOT, MQTT-SN, CLOUD MQTT, WIRESHARK*

## 1. INTRODUCTION

Current wireless sensor networks are design and development of various applications like environmental monitoring, structural health monitoring and home automation etc due to low-cost nature. Current wireless sensor networks (WSN) are incompatible with wide spread existence of Internet Network and related applications. Wireless sensor networks need to implement internet application messaging protocol to integrate with enterprise communication servers. Currently, various types of application messaging protocols are available likes XMPP, HTTP, COAP, MQTT, DDS etc. These protocols designed and developed considering high computational power and bandwidth, but these protocols are not suitable for sensor devices due to low power and bandwidth.

Message Queue Telemetry Transport application messaging (MQTT) designed and developed for machine to machine, mobile devices and enterprise systems. MQTT works on Publish/Subscribe architecture. In this architecture Publish/Subscribe

are not directly communicative, but MQTT server receives the publish messages from clients and delivers to the subscribers. MQTT client publishes the sensor data to the MQTT server. MQTT subscriber subscribes to required message through the topic. MQTT Server delivers the message to subscribed topics. MQTT protocol was extensively developed and deployed in enterprise systems due to highly scalable and available feature. Almost all big industrial organizations like an IBM IOT Platform, Google IOT Platform, Amazon IoT and Cloud MQTT implemented MQTT messaging protocol to back-end networks and servers due to simple and low message overhead. MQTT is implemented on top of TCP transport protocol, still MQTT is not suitable for sensor devices due to complex implementation of TCP.

Solution to this problem is Message Queue Telemetry Transport -Sensor Network protocol (MQTT-SN) designed considering lossy wireless network characteristics. MQTT-SN is less complex and efficient protocol for next generation sensor

devices. MQTT-SN protocol is developed based on UDP/IP. UDP is simple compared to complex byte-oriented TCP. In the MQTT-SN based network consists of sensor devices/ actuators, MQTT-SN clients, MQTT-SN Gateways, MQTT server.

### 1.1 Proposed work

In this research work design, development of End-to-End MQTT-SN Qos sequence flows and Integration with Cloud MQTT server, IoT application, End (Sensor) to End (IOT Application) sensor data transmission, Performed each Node Log analysis.

Remaining paper is organized as follows. In Section 2 Literature survey, In Section 3 MQTT-SN based IoT System architecture, IOT system components, and IoT protocol stack are discussed. In Section 4, MQTT-SN Qos message sequence flows explanation and MQTT-SN Gateway software process. In Section 5 related work, experimental setup, Hardware and Software, sensor database schema is mentioned for end-to-end sensor data Transmission. In section 6 Sensor Node Data Logs, local MQTT-SN Edge Gateway Logs, Cloud MQTT Server Logs, IoT Application Logs analysis, wire shark messages analysis of MQTT-SN and MQTT. In Section 6 Using Practical ESP8266(IOT), DHT22 devices and open-source Arduino platform, MQTT-SN protocol stack for sensor data transmission to the IoT application. Device controlling using remote application. In section 7 conclusion and future scope of this paper.

## 2. LITERATURE SURVEY:

As per [1] MQTT protocol is developed pub/sub model and is extensively deployed in enterprise networks server. Due to success of MQTT protocol for back-end server and Mobile devices, MQTT-SN pub/sub are specified to WSN networks and sensor devices with some variations considering wireless sensor characteristics. MQTT-SN is low energy protocol and less complex compared with MQTT protocol. Using MQTT-SN protocol in the device side and MQTT protocol in the server-side interoperability is easily achieved compared to other protocols. The limitation of this paper is implementation aspects of message retry and sleeping clients not specified.

In the paper [2] modeling of MQTT-SN protocol end to service delay calculations, content delivery calculation theoretically and simulated NS-2 tool. But practical MQTT-SN gateway is not considered.

In paper [3] authors are discussed an overview of MQTT-SN protocol functionality, drawback of the paper is practical results are not included.

In the Paper [4] authors are discussed sensor data transmission to IoT applications using the MQTT server and MQTT-SN Gateway. MQTT-SN protocol functionality, MQTT-SN protocol Quality of services-0, and Quality of service-1 call flows are discussed. MQTT-SN gate way integration with Mosquitto MQTT server and MQTT-SN clients developed by the C-language tool for sending data. Drawback of this paper is JSON packet format and real time sensor integration is too complex.

In the Paper [5] authored discussed use cases of Internet of things for smart city applications. Constrained Application Protocol (COAP) used for field sensor/actuator devices. In WSN gateway one side is COAP protocol and other side well know Hypertext transport protocol used for back-end server communication. COAP and HTTP protocols are different, COAP protocol converter to HTTP protocol needed in the WSN gateway. WSN gateway complexity increases and not easily achievable interoperability due to a different protocol used.

In paper [6] discussed MQTT protocol principles and working. MQTT-based IOT protocol works on pub/sub mechanism. MQTT clients publish data to the MQTT server. MQTT server delivers data to the register subscriber. MQTT-Clients and MQTT-subscribers are not connected directly, but through the MQTT-Server. Drawback of the paper is practical results are not included.

In the paper [7] authors discussed performance analysis of various IoT protocols like MQTT (Message Queue Telemetry Transport), COAP (Constrained Application protocol), and DDS (Data distribution service) protocols. Performance analysis various parameters are latency, packet loss and bandwidth discussed.

In the paper [8] authors discussed MQTT-SN protocol for smart city application-based wireless network. MQTT-SN protocol performance parameter calculations like packet loss, and throughput.

In this research, we contributed to sensor data transmission to the IoT application using a practical real-time IOT device, MQTT-SN Gateway and MQTT server. Controlling of the actuator from remote application Cloud and integration of Eclipse MQTT-SN gateway with cloud MQTT Server. IoT application developed for receiving of sensor data and storage of sensor data for further processing and discussion of various node logs as per MQTT-SN Qos-0, Qos-1, Qos-2 message sequence.

### 3. IOT SYSTEM ARCHITECTURE AND COMPONENTS DESCRIPTION

Generic IoT System as shown in Figure 1, consists of sensor devices, actuators and IOT plot form, local edge gateway (MQTT-SN), MQTT-Server, IoT application, and database

#### 3.1 MQTT-SN IoT Sensor Node Platform

IoT device platform consists of sensors, IoT hardware, and related software. Sensor devices monitor the physical parameters and convert them into digital form through A/D converters. Digital sensor data is packed using JSON format and the

MQTT-SN client publishes the sensor data to the MQTT-SN Gateway. In this project python based MQTT-SN clients enhanced for this requirement and also used a real-time Arduino-based platform for sensor data transmission.

IoT device platform consists of sensors, IoT hardware and related software. Sensor devices monitor the physical parameters and convert them into digital form through A/D converters. Digital sensor data is packed using JSON format and the MQTT-SN client publishes the sensor data to the MQTT-SN

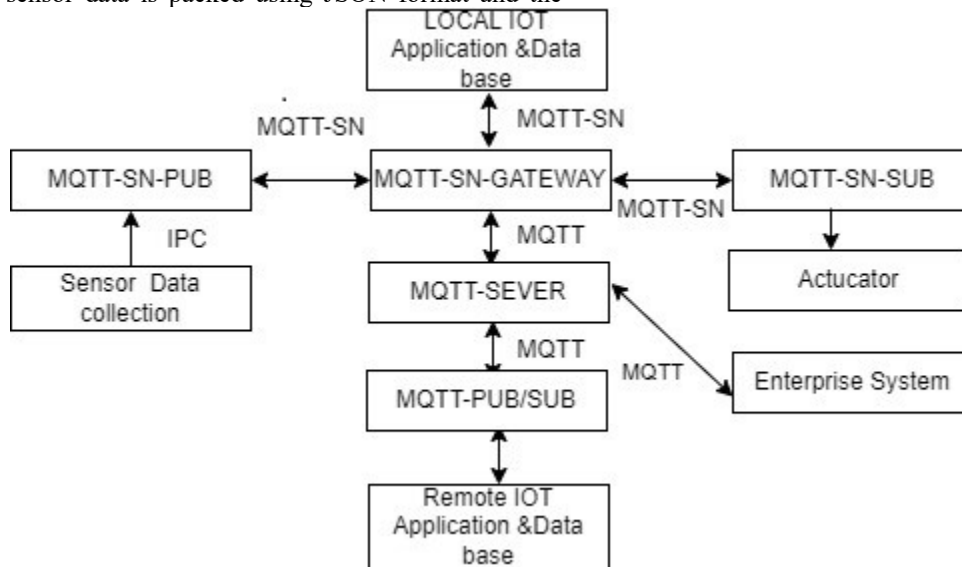


Figure-1: MQTT-SN Based IOT System Block diagram

Gateway. In this project python based MQTT-SN clients enhanced for this requirement and also used real-time Arduino based platform for sensor data transmission

#### 3.2 MQTT-SN Gateway

Main function of Message Queue Telemetry Transport sensor network (MQTT-SN) Gateway is translation of MQTT-SN protocol to Message Queue Telemetry Transport (MQTT) protocol. In the MQTT-SN Gateway, MQTT-SN protocol software and MQTT-protocol software and related state machines are implemented.

#### 3.3 Cloud MQTT-Server

Cloud MQTT-Server provides a highly available and scalable system for handling large sensor data. It is used Cloud Based MQTT-Server for testing and local MQTT-Server for full pledged functionality. Main advantages of the Cloud platform are from anywhere access data using the MQTT IOT application

#### 3.4 Internet of things Application (IoT)

Internet of things application is any type of application as per project requirements. In this research project MQTT based IOT application was developed for sensor data collection and analysis.

#### 3.5 IOT Protocol stack

IoT Protocol stack consists of the application, Packing layer, application messaging layer, Transport, and Network layers, data link layers and Physical layer. In this project sensor side consists of an application layer, JSON packing layer, UDP transport layer, IPV4 or IPV6 network layer, data link layer, and physical layer process. In the Gateway device-side communication with MQTT-SN protocol stack and MQTT protocol stack for communication of MQTT-Server. MQTT server is the main server for handling published messages from MQTT-SN clients through the Gateway and transmission of MQTT published messages to the registered subscriber. MQTT-based IOT application

receives sensor data from the MQTT server, decodes the application sensor data, and stores it in the SQL lite database for further processing. In the results section, we discussed MQTT-SN messages and MQTT message details.

#### 4. MQTT-SN QOS MESSAGE FLOWS

MQTT-SN protocol provides four types of QoS mechanisms for the application designer. which are QoS-0, QoS-1, QoS-2, and QoS-3. In this paper QoS-0, QoS-1, and QoS-2 mechanisms are discussed.

##### 4.1 MQTT-SN QoS-0 Message Sequence flow

QoS-0 Message sequence flow as shown in Figure2. In QoS-0 Model, Messages are sent to the Gateway without expecting any acknowledgment. Implementation is simple but messages are may be lost in the QoS-0 model. This model useful for the frequent sensor data transmission

##### 4.2 MQTT-SN QoS-1 Message Sequence flow

QoS-1 Message sequence flow as shown in Figure3. In QoS-1 Model, Messages are sent to the Gateway with acknowledgment. Messages received may be duplicated in the QoS-1 model. This model useful for reliable sensor data transmission

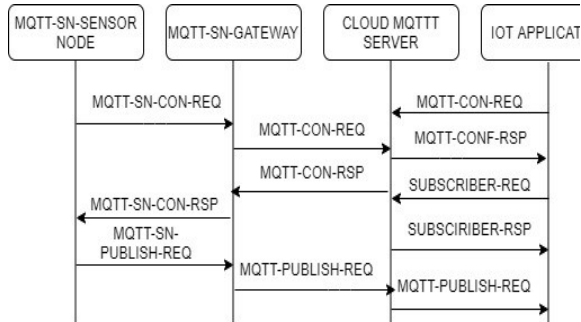


Figure-2: QoS-0 Message Sequence flow

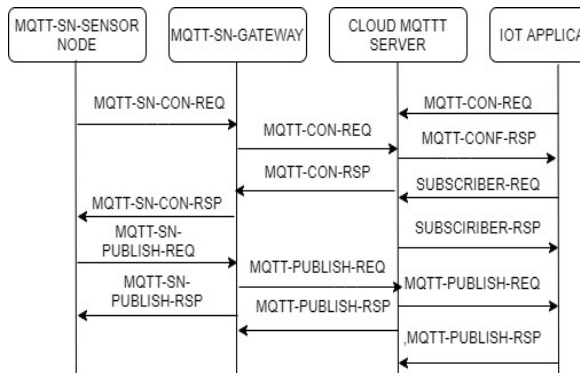


Figure-3: QoS-1 Message Sequence flow

##### 4.3 MQTT-SN QoS-2 Message Sequence flow

QoS-2 Message Sequence flow is shown in Figure 4. In QoS-2 Model, Messages are sent to the Gateway with acknowledgment. Messages are received exactly once. No duplicates and message are lost. This model useful for real time sensor data transmission like a telemedical application.

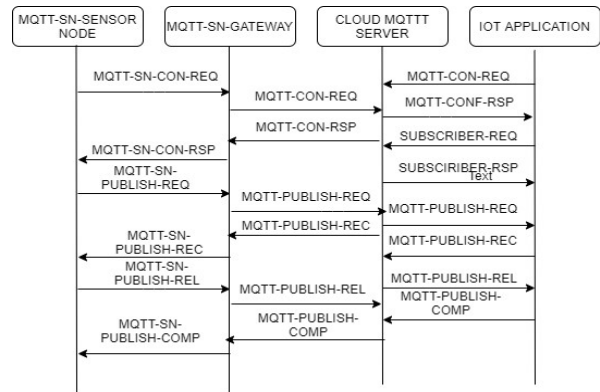


Figure-4: QoS-2 Message Sequence flow

#### 4.4 Gateway Software flow chart

In the MQTT-SN Gateway consists of MQTT-SN Receiver, MQTT-SN Sender, MQTT Sender, MQTT Receiver and Event Queues are crucial software processes. Eclipse MQTT-SN gateway was used in this project. It is designed and developed based on the Multithreading concept. MQTT-SN receiver process consists of the state machine for receiving of MQTT-SN messages from the MQTT-SN clients, decodes the MQTT-SN messages and crucial parameters queue to transmit MQTT server. MQTT-SN sender process takes the parameters form the Queue and transmits to the MQTT-SN clients. MQTT sender process take the parameters from the Queue process and transmits in the form of MQTT message format to the MQTT Server. MQTT receiver process receives the MQTT messages from the MQTT server and decodes the MQTT messages and keeps the parameters in the Queue to transmit MQTT-SN clients.

#### 5. RELEVANT WORK AND EXPERIMENTAL SETUP

##### 6.

##### 5.1 Relevant work

- Design and Development of MQTT-SN Message flow for QoS-0, QoS-1, and QoS-2.
- MQTT-SN Python scripts are improved for this project.

- MQTT IOT Application development
- Design and Development of Sensor database.
- MQTT-SN Gateway Integration with practical cloud MQTT Server.
- MQTT-SN and MQTT call flow log analysis
- Real-time sensor data transmission using ESP 8266 IoT hardware platform and MQTT-SN protocol software.

## 5.2 Experimental Setup and practical nodes discussion

### 5.2.1 Sensor IOT Platform

NodeMCU (ESP8266) is a low-cost IoT hardware platform, it supports WIFI capabilities and TCP, UDP/IP protocol stack. Temperature and pressure sensor devices (DHT22), Node MCU, and LED hardware are used in this research project. Arduino development platform, related software packages, and MQTT-SN protocol software are used MQTT-SN IoT platform is used for collection of sensor data, JSON encoding of sensor data, and Publishes to the IoT Gateway through the MQTT-SN communication software. simple LED used for the actuator. The practical setup is shown in Figure 5.



Figure-5 Node MCU IOT Platform And DHT22 Sensor

### 5.2.2 MQTT-SN Gateway

Multi-threaded and multi process MQTT-SN Gateway protocol software deployed in the Raspberry PI0-B Hardware platform. Configured it to communicate with IoT device, Cloud MQTT Server, and local MQTT Server. In this research ECLIPSE MQTT-SN GW Software used. It is deployed in the Raspberry PI Hardware, customized Linux OS, and other dependent software. Gateway software starting process is shown in Figure 6.



Figure-6 MQTT-SN Gateway Starting Process

### 5.2.3 MQTT-Server

Highly available and scalable private Cloud MQTT server and local MQTT server used for this research project. Local MQTT server starting process as shown in Figure 7.

```
C:\Windows\system32\cmd.exe - mosquitto -d -v
C:\Program Files\mosquitto>mosquitto -d -v
1627836889: Warning: Can't start in daemon mode in Windows.
1627836889: mosquitto version 1.6.9 starting
1627836889: Using default config.
1627836889: Opening ipv6 listen socket on port 1883.
1627836889: Opening ipv4 listen socket on port 1883.
```

Figure-7 MQTT Server Starting Process

### 5.2.4 IOT MQTT pub/Sub Systems

MQTT IoT pub/sub python scripts are designed and developed based on the MQTT-SN protocol and MQTT protocol framework for this IoT project. For sensor databases, SQLite database is used for storage of sensor data. It may be used with other databases as per project requirements.

### 5.2.5 Complete experimental setup

Complete experimental setup as shown in Figure 8.

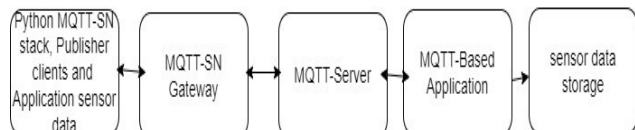


Figure-8 End To End Hardware Setup And Software Deployment

## 7. RESULTS AND DISCUSSION

Practical simulated logs for the Sensor Data Node, MQTT-SN gateway, Cloud MQTT Server and IoT Application Logs are discussed in the following section.

**6.1 Experimental Logs for the MQTT-SN QoS-02**

**6.1.1 MQTT-SN client Publish message to the MQTT-SN Gateway-with Qos-02**

MQTT-SN clients gather the temperature sensor data, Json packing format and Publish to the Gateway. The following log as shown in Figure 9.

```

Connected to gateway...
Topic registered. 1
now = 2022-03-13 18:07:39.348579
{"Device_id": "SENSORID_TMP36",
"Device_Type": "TMP36", "temperature": 26.67,
"curr_date_time": "13/03/2022 18:07:39", "Info":
"TMPL36Data"}
Message is publishing with Qos-2
    
```

Figure-9 MQTT-SN Client Publish Sensor Data With Qos-2

**6.1.2 MQTT Based-Application Received Sensor Data from the Cloud MQTT server-with-Qos-2**

MQTT based IOT application receives sensor data from the MQTT server with Qos-2, decodes it and storage of sensor data in the database. The following log is shown in Figure10 receiving data, processing, and storage in the database.

```

Gitam_PhD_IOT Application is starting
Subscribing topic
Connected with result code 0
Subscribing topic
MQTT Data Received...
MQTT Topic: TEMP_MQTT_SN_DATA
data Received {"Device_id": "SENSORID_TMP36",
"Device_Type": "TMP36", "temperature": 26.67,
"curr_date_time": "13/03/2022 18:07:39", "Info":
"TMPL36Data"}
Received Device Type TMP36

=====
RECEIVED_TMP36_DATA
Device_id : SENSORID_TMP36
Device Type: TMP36
temperature: 26.67
received_date_time: 13/03/2022 18:07:39
Info : TMPL36Data
Inserted TMP36_DATA into Database.
    
```

Figure-10 Received Data -With Qos-2 And Decode Sensor Data

**6.1.3 MQTT-SN IOT Gateway Log with Cloud MQTT connection for Qos-2**

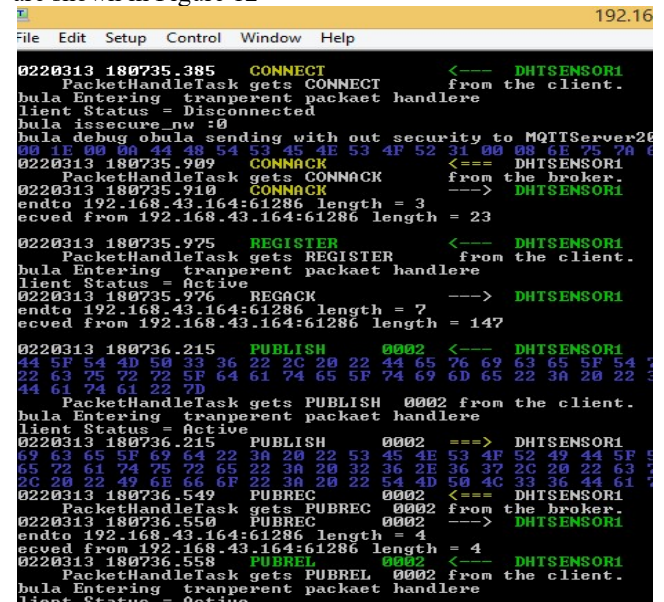
MQTT-SN gateway receives messages from IoT devices, acknowledges to the IoT device, and sends to the MQTT-Server, receiving acknowledges from the MQTT Server. The following message flow happens for Qos-2 at the Gateway. It message flow as shown in Figure 11.

**Cloud MQTT-Server logs-with-Qos-02**

Assume that subscriber connect id PHD\_TEST\_SUB

**6.1.4 Process involved at the MQTT server whenever subscriber message received-QoS-2**

whenever the subscriber is received at MQTT server, if new subscriber MQTTT server creates the connection, and subscriber parameters are verified by the MQTT server and replies subscriber connection acceptance to the subscriber. The following events will happen and are shown in Figure 12



```

192.16
File Edit Setup Control Window Help
0220313 180735.385 CONNECT <--- DHTSENSOR1
PacketHandleTask gets CONNECT from the client.
bula Entering tranparent packet handler
lient Status = Disconnected
bula issecure_nw :0
bula debug obula sending with out security to MQTTServer26
00 1E 00 00 44 48 24 53 45 4E 53 4F 52 31 00 00 6E 75 7A 6
0220313 180735.909 CONNACK <=== DHTSENSOR1
PacketHandleTask gets CONNACK from the broker.
0220313 180735.910 CONNACK ---> DHTSENSOR1
entdo 192.168.43.164:61286 length = 3
eved from 192.168.43.164:61286 length = 23
0220313 180735.975 REGISTER <--- DHTSENSOR1
PacketHandleTask gets REGISTER from the client.
bula Entering tranparent packet handler
lient Status = Active
0220313 180735.976 REGACK ---> DHTSENSOR1
entdo 192.168.43.164:61286 length = 7
eved from 192.168.43.164:61286 length = 147
0220313 180736.215 PUBLISH 0002 <--- DHTSENSOR1
44 5F 54 4D 50 33 36 22 2C 20 22 44 65 76 69 63 65 5F 54 2
22 63 75 72 72 5F 64 61 74 65 5F 74 69 6D 65 22 3A 20 22 5
44 61 74 61 22 7D
PacketHandleTask gets PUBLISH 0002 from the client.
bula Entering tranparent packet handler
lient Status = Active
0220313 180736.215 PUBLISH 0002 ==> DHTSENSOR1
69 63 65 5F 69 64 22 3A 20 22 53 45 4E 53 4F 52 49 44 5F 5
65 72 61 74 75 72 65 22 3A 20 32 36 2E 36 37 2C 20 22 63 5
2C 20 22 49 6E 66 6F 22 3A 20 22 54 4D 50 4C 33 36 44 61 7
0220313 180736.549 PUBREC 0002 <=== DHTSENSOR1
PacketHandleTask gets PUBREC 0002 from the broker.
0220313 180736.550 PUBREL 0002 ---> DHTSENSOR1
entdo 192.168.43.164:61286 length = 4
eved from 192.168.43.164:61286 length = 4
0220313 180736.558 PUBREL 0002 <--- DHTSENSOR1
PacketHandleTask gets PUBREL 0002 from the client.
lient Status = Active
bula Entering tranparent packet handler
    
```

Figure- 11 MQTT-SN And MQTT Protocol Message At Gateway-For Qos-2

```

Sending CONNACK to PHD_TEST_SUB(0, 0)
Received SUBSCRIBE from PHD_TEST_SUB
TEMP_MQTT_SN_DATA (QoS 2)
Sending SUBACK to PHD_TEST_SUB
    
```

Fig-12 Subscriber Subscription Process For Qos-2

### 6.1.5 Process involved whenever publishers comes from the gateway to the MQTT Server-with-Qos-2

Whenever publish message is received from MQTT-Clients through the Gateway to the MQTT server for Qos-2, processing events are shown in Figure 13

No will message specified.  
 Sending CONNACK to DHTSENSOR1 (0, 0)  
 Received PUBLISH from DHTSENSOR1 (d0, q2, r0, m2, 'TEMP\_MQTT\_SN\_DATA', ... (140 bytes))  
 Sending PUBREC to DHTSENSOR1 (Mid: 2)  
 Received PUBREL from DHTSENSOR1 (Mid: 2)  
 Sending PUBCOMP to DHTSENSOR1 (Mid: 2)

Fig-13 Publish message received from the Gateway with Qos-2

### 6.1.6 Process involved at the MQTT Server whenever message sending for subscribed subscriber -with-Qos-2

Here PHD\_TEST\_SUB and Cinet\_id1 are two subscribers. Cinet\_id1 subscribes with Qos-1 and PHD\_TEST\_SUB subscribes with QoS-2. Whenever Publish message with Qos-2 sending to the subscribed subscriber, processing events will happen and as shown in Figure-14

Sending PUBLISH to PHD\_TEST\_SUB (d0, q2, r0, m1, 'TEMP\_MQTT\_SN\_DATA', ... (140 bytes))  
 Sending PUBLISH to Cinet\_id1(d0, q1, r0, m1, 'TEMP\_MQTT\_SN\_DATA', ... (140 bytes))  
 Received PUBACK from Cinet\_id1 (Mid: 1)  
 Received PUBREC from PHD\_TEST\_SUB (Mid: 1)  
 Sending PUBREL to PHD\_TEST\_SUB (Mid: 1)  
 Received PUBCOMP from PHD\_TEST\_SUB (Mid: 1)  
 Received DISCONNECT from DHTSENSOR1

Figure -14 MQTT Server Sending Publish Message To The Subscriber

### 6.1.7 MQTT-SN Wire shark with Qos-2

Wire shark is an open-source network monitoring tool to capture a variety of protocol messages. In this project, a wire shark is used for capturing MQTT and MQTT-SN protocol messages. MQTT-SN wire shark messages for Qos-2 as shown in Figure 15.

Time	Source	Destination	Protocol	Length	Info
1/43 400.11151/	192.168.43.220	192.168.43.104	MQTT-SN	44	DISCONNECT Req
4642 1081.249120	192.168.43.164	192.168.43.220	MQTT-SN	58	Connect Command
4656 1081.783684	192.168.43.220	192.168.43.164	MQTT-SN	45	Connect Ack
4658 1081.838659	192.168.43.164	192.168.43.220	MQTT-SN	65	Register
4663 1081.848270	192.168.43.220	192.168.43.164	MQTT-SN	49	Register Ack
4668 1082.075817	192.168.43.164	192.168.43.220	MQTT-SN	189	Publish Message
4676 1082.422381	192.168.43.220	192.168.43.164	MQTT-SN	46	Publish Received
4679 1082.422696	192.168.43.164	192.168.43.220	MQTT-SN	46	Publish Release
4690 1082.618734	192.168.43.220	192.168.43.164	MQTT-SN	46	Publish Complete

Figure -15 MQTT-SN Message Sequence With Qos-2

### 6.1.8 MQTT Wire shark with Qos-2 wire shark MQTT message sequence for Qos-2 as shown in Figure 16.

No.	Time	Source	Destination	Protocol	Length	Info
1650	121.493293	192.168.43.220	192.168.43.164	MQTT	90	Connect Command
1651	121.494445	192.168.43.164	192.168.43.220	MQTT	70	Connect Ack
1675	121.982660	192.168.43.220	192.168.43.164	MQTT	249	Publish Message (id=2)
1676	121.985318	192.168.43.164	192.168.43.220	MQTT	70	Publish Received (id=2)
1695	122.014365	192.168.43.220	192.168.43.164	MQTT	70	Publish Release (id=2)
1696	122.016853	192.168.43.164	192.168.43.220	MQTT	70	Publish Complete (id=2)
1720	122.988184	192.168.43.220	192.168.43.164	MQTT	68	Disconnect Req

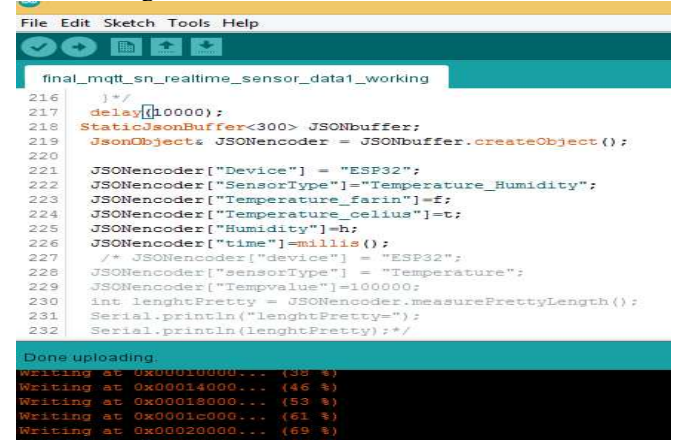
Figure-16 MQTT Message Sequence For Qos-2

## 6.2 Real time sensor data transfer and actuator device controlling from the remote application

Arduino IDE is an open-source IDE platform for developing embedded IOT software and Loading software to the device, serial monitoring of real-time data sensor. MQTT-SN protocol enhanced for sending of sensor data and controlling of the device from the remote command.

### 6.2.1 Arduino Development IDE Environment

Portion of Arduino develop environment as shown in Figure-17



```

File Edit Sketch Tools Help
final_mqtt_sn_realtime_sensor_data1_working
216 }*/
217 delay(10000);
218 StaticJsonBuffer<300> JSONbuffer;
219 JsonObject* JSONencoder = JSONbuffer.createObject();
220
221 JSONencoder["Device"] = "ESP32";
222 JSONencoder["SensorType"] = "Temperature_Humidity";
223 JSONencoder["Temperature_farin"] = f;
224 JSONencoder["Temperature_celsius"] = t;
225 JSONencoder["Humidity"] = h;
226 JSONencoder["time"] = millis();
227 /* JSONencoder["device"] = "ESP32";
228 JSONencoder["sensorType"] = "Temperature";
229 JSONencoder["Tempvalue"] = 10000;
230 int lenghtPretty = JSONencoder.measurePrettyLength();
231 Serial.println("lenghtPretty=");
232 Serial.println(lenghtPretty);*/
Done uploading.
Writing at 0x00010000... (38 %)
Writing at 0x00014000... (46 %)
Writing at 0x00018000... (53 %)
Writing at 0x0001c000... (61 %)
Writing at 0x00020000... (69 %)
Writing at 0x00024000... (76 %)
    
```

Figure-17 Arduino Develop Environment

### 6.2.2 Actuator controlling from remote application

In combination of MQTT-SN and MQTT protocol, we can control devices remotely. In a sensor, terminology actuator is a controlling device. Here we used a simple LED for on/off conditions. with ESP8266 and LED hardware are used for actuator function implementation. Practical results as shown in Figure 18 and Figure18-A

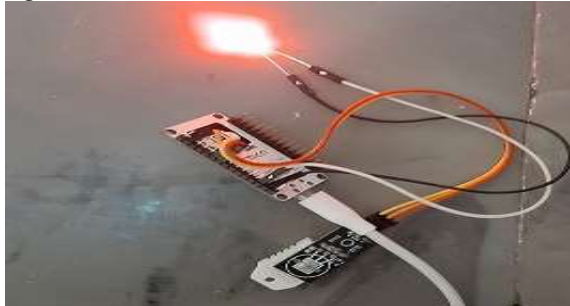


Figure-18 Actuator On Condition



Figure-18-A Actuator Off Condition

Remote command received from MQTT device controlling application. In this case, led on/off commands are received from the MQTT application device. Arduino serial command experiment as shown in Figure-18-B

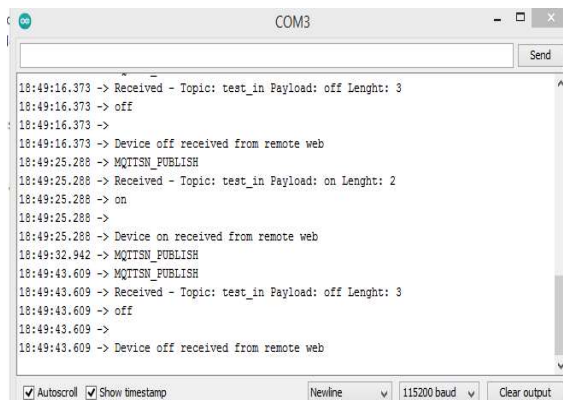


Figure-18-B Received Message To Actuator Device Controlling On/Off

### 6.3 MQTT based IoT application experimental collected sensor data

#### 6.3.1 MQTT-Based Application received BME280 Sensor data

MQTT based application receives the BME280 (temperature, Humidity, and Pressure) sensor data from the MQTT Server publish message, decode it and store in the data base for further analysis. Its measurements are shown **Table 1**

Id	Device_Type	Tempera- - Ture	Humidit y	Pressur e
1	D_T_H_P	36.9	0.43	81.85
2	D_T_H_P	42.74	31.17	76.81
3	D_T_H_P	35.28	15.18	59.14
4	D_T_H_P	19.41	17.22	4.88
5	D_T_H_P	31.42	18.73	1.33
6	D_T_H_P	27.25	15.28	22.31
7	D_T_H_P	26.76	74.02	37.41
8	D_T_H_P	45.36	19.31	39.65
9	D_T_H_P	18.4	26.35	48.93
10	D_T_H_P	15.43	24.49	34.2

Bme280 Sensor-Data-Table-01

#### 6.3.2 MQTT-Based Application received TEMP\_HUMIDITY\_DHT22 sensor data

MQTT based application receives the publish message (DHT22\_Sensor data) from the MQTT server, decode it and stores in the database. Its measurements are shown in Table 2

Id	Device_Type	temperature	Pressure
1	D_T_P	50.36	88.68
2	D_T_P	41.24	80.12
3	D_T_P	49.36	33.84
4	D_T_P	19.32	62.18
5	D_T_P	27.33	67.34
6	D_T_P	48.11	69.27
7	D_T_P	27.04	67.64
8	D_T_P	26.14	30.77
9	D_T_P	32.24	73.80
10	D_T_P	42.13	56.77

Dht22\_Temp\_Humidity Sensor-Data- Table-02



## 7. CONCLUSION AND FUTURE SCOPE

In this research paper we discussed low energy protocol, it is Message Queue Telemetry Transport-sensor network (MQTT-SN) protocol. practical MQTT-SN Gateway integration with a highly available and scalable Cloud MQTT server. In present private Cloud MQTT server limited connections, but in practical Cloud MQTT server increase the number of connections as per application requirement. Field Sensor device data transmission to the IoT application through MQTT-SN gateway and MQTT server. MQTT-SN protocol stack enhanced using python and real-time IoT platform Node MCU. Actuator Device controlling (on/off) performed from the remote web. All the important logs are discussed. In future porting of python based MQTT-SN stack to real time micro python and this practical setup real time application integration is required.

## REFERENCES

- [1]. Urs Hunkeler, IHongLinh Truong, Andy Stanford-Clark "MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks"-3rd International Conferences on Communication Systems Software and Middleware and workshops, 2008
- [2]. Govindan, K., & Azad, A. P. "End-to-end service assurance in IoT MQTT-SN.", 12<sup>th</sup> Annual IEEE Consumer Communications and Networking Conference, 2015
- [3]. Maria K Tom;"MQTT-SN Protocol - A Review", International Journal of Research in Engineering, Science and Management" , October-201
- [4]. M Obula Reddy, J B Seventlin," Real Time Sensor Data transmission to the IoT Applications using MQTT-SN and MQTT", International Journal of Recent Technology and Engineering (IJRTE) - November 2019
- [5]. Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. "Internet of Things for Smart Cities.", IEEE Internet of Things Journal, Vol 1, pp. 22–32, 2014.
- [6]. Suvam Mohanty, Sagar Sharma, Vaibhav Vishal;" MQTT – Messaging Queue Telemetry Transport IOT based Messaging Protocol" INTERNATIONAL RESEARCH JOURNAL OF ENGINEERING AND TECHNOLOGY (IRJET)"- SEP-2016
- [7]. Yuang Chen , Thomas Kunz ;"Performance Evaluation of IoT Protocols under a Constrained Wireless Access Network"- 2016 International Conference on Selected Topics in Mobile & Wireless Networking- IEEE
- [8]. ESP8266 Arduino Core Documentation Release 2.4.0
- [9] Dave Locke. (2010, Aug.) MQTT V3.1 protocol specification. [Online]. Available: { <http://www.ibm.com/developerworks/webservices/library/ws-mqttlws-mqtt-pdf.pdf> }
- [10]. MQTT-SN protocol specification ([https://www.oasis-open.org/committees/download.php/66091/MQTT-SN\\_spec\\_v1.2.pdf](https://www.oasis-open.org/committees/download.php/66091/MQTT-SN_spec_v1.2.pdf))
- [11]. CoAP protocol specification (<http://coap.technology>)
- [10]. MQTT-SN protocol specification ([https://www.oasis-open.org/committees/download.php/66091/MQTT-SN\\_spec\\_v1.2.pdf](https://www.oasis-open.org/committees/download.php/66091/MQTT-SN_spec_v1.2.pdf))
- [11]. CoAP protocol specification (<http://coap.technology>)
- [12]. Mukherjee, Amartya, Nilanjan Dey, and Debashis De. "EdgeDrone: QoS aware MQTT middleware for mobile edge computing in opportunistic Internet of Drone Things." *Computer Communications* 152 (2020)
- [13]. B. Wukkadada, K. Wankhede, R. Nambiar, A. Nair,0 "Comparison with HTTP and MQTT In Internet of Things (IoT)," International Conference on Inventive Research in Computing Applications (ICIRCA), 11-12 July 2018
- [14]. R. Light, "Mosquitto: server and client implementation of the mqtt protocol," *Journal of Open Source Software*, vol. 2, no. 13, p. 265, 2017
- [15]. Muhammad Harith Amaran, Nazmin Arif Mohd Noh, Mohd Saufy Rohmad, Habibah Hashim:"A Comparison of Lightweight Communication Protocols in Robotic Applications "-IEEE-2015
- [16]. Yogendra Singh Parihar;" Internet of Things and Nodemcu"- 2019 JETIR
- [17]. Monika Kashyap, Vidushi Sharma, Neeti Gupta;"Taking MQTT and NodeMcu to IOT: Communication in Internet of Things"- International Conference on Computational

- Intelligence and Data Science (ICCIDS 2018)
- [18]. Ravi Kishore Kodali , Kopulwar Shishir Mahesh:" A low cost implementation of MQTT using ESP8266"-IEEE-2016
- [19]. Bormann, C., Hartke, K., Shelby, Z.: The constrained application protocol (CoAP). RFC 7252  
(2015). <https://doi.org/10.17487/rfc7252>. <https://www.rfc-editor.org/rfc/rfc7252.txt>
- [20] Kevin Ashton,"That 'Internet of Things' Thing", Jun, 2009, <https://www.rfidjournal.com/articles/view>
- [21]. M. Udin Harun Al Rasyid; Ferry Astika; Fadlul Fikri:" Implementation MQTT-SN Protocol on Smart City Application based Wireless Sensor Network"-IEEE-2019
- [22] A. Stanford-Clark and H. L. Truong, "MQTT for sensor networks (MQTTs)" [http://www.mqtt.org /MQTTs Specification V1.0.pdf](http://www.mqtt.org/MQTTs%20Specification%20V1.0.pdf), Oct. 2007.