Impact Factor (SJIF): 4.542

International Journal of Advance Research in Engineering, Science & Technology

e-ISSN: 2393-9877, p-ISSN: 2394-2444
Volume 4, Issue 4, April-2017

# Implementation of MQTT protocol in Context with Industry 4.0

Nishant M. Sonawala
Department of Electronics & Communication,
PIT, Parul University, Limda
Vadodara, Gujarat

Bharat Tank
Department of Electronics & Communication,
PIT, Parul University, Limda
Vadodara, Gujarat

Hardik Patel
Founder & CEO
(Epsilon Electronics)
Ahmedabad, Gujarat

***Abstract***- Day by day, the internet is unfolding and making various connectivity mechanism. The Internet of Things (IoT) is a mechanism, which provides communication between human to machine and machine to machine. Moreover, IoT connectivity mechanism is possible only by it's different protocols. This paper work contains previous comparison results of IoT protocols. However, the key aspects of this paper is to implement the MQ Telemetry Transport (MQTT) protocol on open source platform in context with industry 4.0, to get real time data monitoring and controlling. In addition to that, real time data for industrial environmental information of different plants, such as temperature, pressure, humidity need to be monitor with mobile device and control rooms. Industry 4.0 is the smart factory model, in which, large amount of data (from production to dispatch and its environment) can be monitor and controlled with the use of Wi-Fi, internet of things (IoT), cloud computing (CC) and cyber physical system (CPS).

***Index Terms* – Industry 4.0, Internet of Things, MQ Telemetry Transport protocol, Wi-Fi, ESP8266**

## I. INTRODUCTION

Now a day, Internet of Things become popular in every IT and Electronics field. Everyone wants to integrated their system with cloud and make it robust. Not only IT and Electronics field people, Industries also want to integrate their production line with IoT to monitor and control data in real time scenario from a remote place without any hurdle. However, we have implemented one of the IoT protocol, named as MQTT for the industry 4.0 make reality.

In this paper we have shown the previous comparison of different IoT protocols like Constrained Application Protocol(CoAP), MQ Telemetry Transport(MQTT), Advanced Message Queuing Protocol(AMQP), Data Distribution Service(DDS), Hypertext Transfer Protocol(HTTP) and custom UDP-based protocol. Every protocol has their pros and cons and based on that we have chosen the MQTT protocol as the finest solution towards the industry 4.0. Moreover, with the use of this MQTT protocol we are going to monitor and control the following sensors AM2302 (Humidity & Temperature), BMP180 (Pressure & Altitude), RTC, Fire Alert, RTD (PT-100) and MQ-135 (Air Quality Detection). Above mention all the sensors will be monitor with the use of ESP8266 Wi-Fi module. Also, we are implementing the MQTT protocol to ESP8266 module and ESP will send the data to broker.

The rest of the paper will have written as follows. Section II describe related work and give information related to component selection as well as brief reason to choose that component. Section III describe the comparison of IoT protocols as well as reason to choose the MQTT protocol in context with industry 4.0. Section IV describe our system architecture and information about proposed work that we have implemented. Section V describe the experimental results, monitoring and controlling all sensors on the mobile devices. Lastly the section VI concludes the paper and shows the future implementation.

## II. RELATED WORK

We are implementing the MQTT protocol to monitor the real time data of the industrial sensors. Sensors provides temperature, pressure, altitude, humidity, fire alert, real-time and air quality detection and these sensors data will be publishing to the broker with the use of MQTT protocol with internet connection as well as local connected devices.

Also, for monitoring the sensors data we are using android application for the mobile devices and for the computer we are using the MQTT Lens utility.

### A. AM2302 Sensor

The AM2302 sensor is for the humidity measurement. We are using this sensor for the monitoring the humidity inside industries. It is capacitive humidity sensing, which contains the compound has been calibrate digital signal output of humidity and temperature. It is small size, low power consumption, signal transmission distance up to 20 meters. Moreover, it's operating voltage range is 3.3V to 5.5V. Also, it's resolution is 0.1%RH, Accuracy at 25℃ is ±2%RH and measurement range is between 0 to 99.9 %RH [1].

This sensor is also providing the temperature data because atmospheric temperature on the industry needs to be monitor continuously for the safety of the workers as well as for the sensitive component used outside the industry, though we are interested in the humidity as well as temperature because it's temperature range is between -40 to 80 ℃. Also, it's resolution in is 0.1 ℃ with 16bit [1].

### B. BMP180 Sensor

For temperature, pressure, and altitude we are using BMP180 sensor, it is the new digital barometric pressure sensor of Bosch Sensor Tec, with a very high performance. It consists of a piezo-resistive sensor, an analog to digital converter and a control unit with E2PROM and a serial I2C interface. The E2PROM has stored 176 bit of individual calibration data. The pressure and temperature data has to be compensated by the calibration data of the E2PROM of the BMP180 [2].

After sending a start sequence to initiate a pressure and/or temperature measurement and its conversion, the result value (pressure and/or temperature) can be read via the I2C interface. Therefore, calculating temperature in °C and pressure in hPa, the calibration data has been used [2]. These constants can be read out from the BMP180 E2PROM via the I2C interface by software initialization. The sampling rate can be increased up to 128 samples per second (standard mode) for dynamic measurement. It is low cost, very tiny sensor device, with three industrial environmental data parameters capture like, temperature range between -40 ºC to +85 ºC, pressure range between 300 to 1100 hPa, and give accurate altitude changes from reference height. It also operates in low power supply range 1.8 V to 3.6 V DC as per specification manual of this product [3].

### C. ESP8266 Wi-Fi Module

For Wi-Fi module we are using ESP8266, which is SoC (system on a chip) based produced by Espressif Systems and it is a highly integrated chip designed to provide full internet connectivity in a small package [4]. It is also serial Wi-Fi wireless transceiver module for Internet of Things (IoT). Here we are using ESP 12-E which is latest module of ESP8266 series, and most suitable for our requirement because it have 22 different kind of pins including UART, GPIOs, I2C, MISO & MOSI of SPI interface, RST, power supply etc. [5], listed on specification manual of ESP 12-E. Moreover, it is small size Wi-Fi module and operate on 3.3 V DC low input power supply, so it is suitable for interface sensors' data and send that to connected MQTT broker, if internet connection present nor it send on TCP/IP stack so, local connected device can be monitoring that data. ESP 12-E version of ESP8266 most suitable for our requirement because, it support 802.11 b/g/n, additionally built-in TCP/IP protocol stack, TR switch, power amplifier and matching network, voltage regulator and power management components, and low-power 32-bit CPU, it can double as an application processor, and so more, can be found on specification of ESP 12-E, Wi-Fi module [4] [5].

### D. RTD (PT-100) Sensor

RTD stands for resistance temperature detector. It means with the use of resistance as a parameter we can find the temperature data. Therefore, we are using the PT-100 sensor, means Platinum 100 as internal sensor used in the RTD to sense the resistance value. Moreover, it is called PT-100 because at the resistance of the 100ohm it gives the temperature as 0ºC [9].

PT-100 provides the temperature range between -200 ºC to 850 ºC. Moreover, PT-100 can be used at different platform with different wiring connection. PT-100 is used to monitor the temperature of the boilers as well as mixtures placed in the industry for the production purpose [9]. In real-time scenario to get the higher resolution of the temperature we are using 3-wire configuration of the RTD. There are mainly three ways to operate the RTD, first one is two wired configurations which is used for the general purpose to measure the temperature of the liquid in the chamber at bakery, second three wired configurations are used for the industrial purpose to measure the boiler temperature, mixture temperature, and many more places at the industrial plant it is required, third is the four wire configuration where exact and a perfect temperature resolution is required, generally laboratory where the testing and the quality assurance, requires a four wire configuration [9]. In our project we are using the three wire configuration.

### E. MQ-135 Sensor

MQ-135 is gas sensor. MQ-135 gas sensor applies $SnO_2$ which has a lower conductivity in the clear air as a gas-sensing material [10]. In an atmosphere where there may be polluting gas, the conductivity of the gas sensor raises along with the concentration of the polluting gas increases. MQ-135 performs a good detection to smoke and other harmful gas, especially sensitive to ammonia, sulfide and benzene steam. Its ability to detect various harmful gas and lower cost make MQ-135 an ideal choice of different applications of gas detection.

Moreover, MQ-135 is sensing gases like NH3, NOx, alcohol, Benzene, smoke, CO2 and many more [10]. For the safety and security purpose in the industry it needs to be placed and in real time scenarios it needs to be sensed so the worst can be prevented.

### F. MQ Telemetry Transport (MQTT) Protocol

MQTT is the proposed protocol that we are using for making the Industry 4.0 as a reality. MQTT is the lightweight protocol. It is unlike the request response type protocol. Moreover, MQTT uses the broker as server and broker will provide between the sensor to remote devices to monitor the data with the use of the subscribe and publishing topic [8].

MQTT works with four major components. The foremost component of MQTT protocol is broker. Broker is same like a server but here broker will automatically allow to connect the different devices with the use of three different Quality of Services (QoS) [7]. In order to connect with the broker and getting the data second component come to picture. Second component of MQTT protocol is topic. Topics generally works as an identity of the information provided by the sensors as well as the devices which may get the information based on the same topics. Now to send the data and to receive the data MQTT protocol has another two terms named as Publish and Subscribe. Publish means, it sends the data to the broker and Subscribe means, it will get the data from the broker. In addition to that, publishing information on the topic (Topic like "Temp") will be send the information to the broker and at the subscriber side if the subscribing topic (Topic like "Temp") is same as the publishing topic then at subscribing side the information will be reached [6].

Unlike a traditional request response type protocol. MQTT is work with publish subscribe type protocol. Once the device needs to be connected to the broker device will send the Connect message to the broker, after that broker will send the Connack message to the device. Once the device is registered with the broker then it will get all the information regarding the topics on which it subscribed, also it can send the information to the same topics by publishing the information [8] [7] [6]. Only one time it required to connect to the broker and then it will automatically have connected with the broker when the connection get lost. It is possible because broker will continuously send the keep alive message check the device is connected or not. In addition to that once, the device may have disconnected for some time and then reconnected then again it will start to receive the data provided by the broker with QoS. We are using this protocol to our system to monitor the sensors data in real time scenario.

## III. COMPARISION OF MQTT, CoAP, AMQP, HTPP

For many years HTTP has been used as the reference communications protocol in this context. HTTP is a very wide spread protocol, and APIs for its use are available basically for every programming language. However, in the new era of internet technology there are various protocols has been introduced in last 5 to 6 years. Advance Message Queuing protocol (AMQP), MQ Telemetry Transport protocol (MQTT), Constrained Application Protocol (CoAP) and many more. Every single protocol is used for the particular application [12] [13]. Every system does not require a same protocol as we all used in the traditional way HTTP were used. Now a days Internet of Things protocol are used in many different application likes industrial application, medical application, military application, commercial communication application etc. So, based on the comparison of different protocols we have choose the MQTT protocol in context with the industry 4.0.

We need to identify the requirements of the industry 4.0 and based on the requirements we have to find which protocol will be the best solution for our application. Moreover, we have made an analyses that industry 4.0 requires a continuous connection with the server in order to send the data in the real time scenario for that it does not required a large amount of bandwidth, power consumption of needs to be lesser means more energy efficient, at the loss of connection it needs to be connected automatic with the server, losing the connection and reconnects events needs to be FIFO (First Input First Out) because in the industrial environment if the real time data will be delayed or not shown at the connection reestablished then the worst may happen and it damage the company revenue. Also, the system needs to be protected, data or the information provided by the system will be securely send to server and no external traffic affects to that system.

AMQP will not be used for our system because connect and disconnect will change the information sequence from FIFO (First Input First Out) to LIFO (Last Input First Out). HTTP will not be used for the system because it is request response type protocol, also it consumes more bandwidth so more energy it requires so it is not used. CoAP will not be used for our system because packet loss rate under degraded network condition as well as implementation of CoAP is more complex due to its unavailability s open source [12] [13].

With the above mention requirements and comparison of the protocols, we found the MQTT protocol as the best solution for the industry 4.0 make a reality term.

## IV. PROPOSED SYSTEM

Aim of this research is to monitor the industrial sensors data in real time scenarios. Also it needs to be stable, very low cost, light weight, low power consumption.

This system is used to monitor and control the industrial sensors data with the use of Wi-Fi module as ESP8266 and implementing the MQTT protocol to that Wi-Fi device so, can get the data to the internet connected devices as well as local area connected devices. Here, given proposed monitoring system diagram.

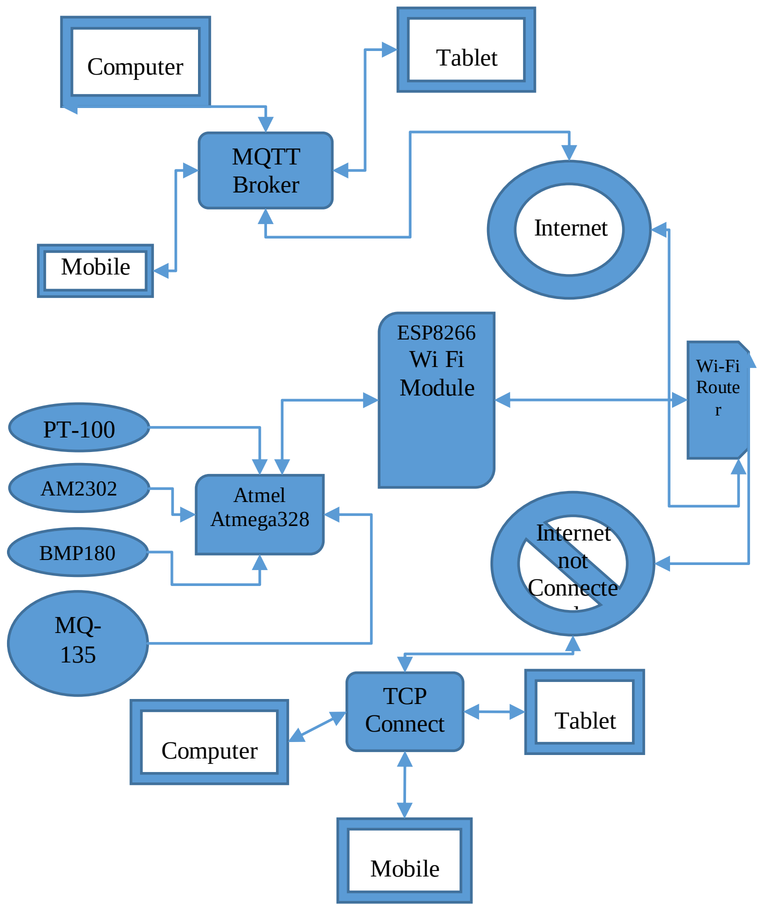
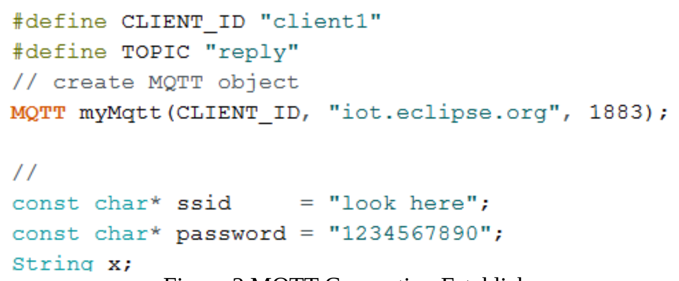


Figure 1 Proposed Monitoring System

As mention the component details in above section, we have used these sensors PT-100, AM2302, BMP180, MQ-135. All the sensors have produced the output, which has been processed by the microcontroller. Microcontroller sent the data to the ESP Wi-Fi module. Moreover, MQTT was implemented in the ESP module as shown in the below figure. Also, basic implementation method is shown in the below coding.

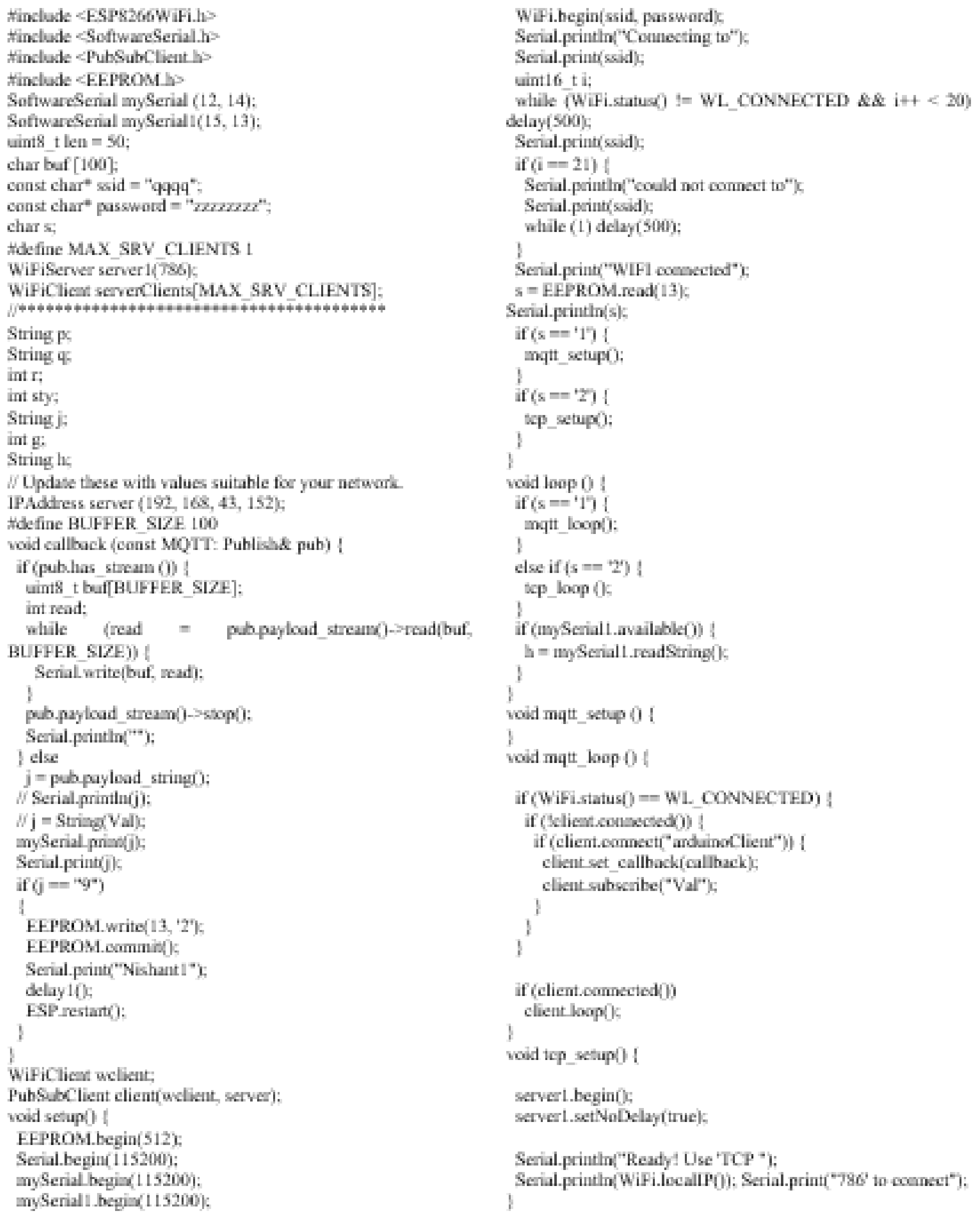


```
#define CLIENT_ID "client1"
#define TOPIC "reply"
// create MQTT object
MQTT myMqtt(CLIENT_ID, "iot.eclipse.org", 1883);

//
const char* ssid     = "look here";
const char* password = "1234567890";
String x;
```

Figure 2 MQTT Connection Establish

Here, we are using the open source broker provided by the eclipse corporation. Port number for the MQTT broker as per the security level is fixed 1883. Moreover, we have given a ssid and password for the router to connect with.

Moreover, as we have mention Online mode and Offline mode, below figure 3(a) and 3(b) shows the implementation of the online and offline mode concept. Online mode will work when the internet connection will available and sensors information needs to be monitor from anywhere on the earth with the use of mobile devices. Offline mode will work when the internet connection will not available and locally sensors information needs to be monitor in the remote places in the industry within a local area network then the offline mode will work. In addition to that, we have put one status read function which will detect the current mode of the ESP module and manually we can switch from one mode to another. It will be useful, in case of online as well as offline scenarios.

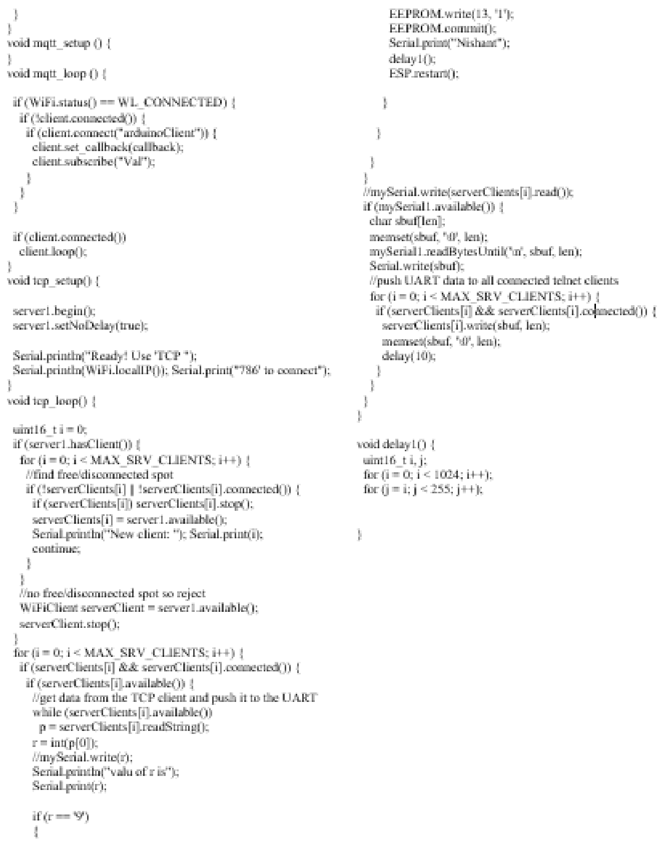


```
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>
#include <PubSubClient.h>
#include <EEPROM.h>
SoftwareSerial mySerial (12, 14);
SoftwareSerial mySerial1(15, 13);
uint8_t len = 50;
char buf [100];
const char* ssid = "qqqq";
const char* password = "zzzzzzzz";
char s;
#define MAX_SRV_CLIENTS 1
WiFiServer server1(786);
WiFiClient serverClients[MAX_SRV_CLIENTS];
//*****************************************
String p;
String q;
int r;
int sty;
String j;
int g;
String h;
// Update these with values suitable for your network.
IPAddress server (192, 168, 43, 152);
#define BUFFER_SIZE 100
void callback (const MQTT: Publish& pub) {
 if (pub.has_stream ()) {
  uint8_t buf[BUFFER_SIZE];
  int read;
  while     (read     =     pub.payload_stream()->read(buf,
BUFFER_SIZE)) {
    Serial.write(buf, read);
  }
  pub.payload_stream()->stop();
  Serial.println("");
 } else
  j = pub.payload_string();
 // Serial.println(j);
 // j = String(Val);
 mySerial.print(j);
 Serial.print(j);
 if (j == "9")
 {
  EEPROM.write(13, '2');
  EEPROM.commit();
  Serial.print("Nishant1");
  delay1();
  ESP.restart();
 }
}
WiFiClient wclient;
PubSubClient client(wclient, server);
void setup() {
 EEPROM.begin(512);
 Serial.begin(115200);
 mySerial.begin(115200);
 mySerial1.begin(115200);
 WiFi.begin(ssid, password);
 Serial.println("Connecting to");
 Serial.print(ssid);
 uint16_t i;
 while (WiFi.status() != WL_CONNECTED && i++ < 20)
delay(500);
 Serial.print(ssid);
 if (i == 21) {
  Serial.println("could not connect to");
  Serial.print(ssid);
  while (1) delay(500);
 }
 Serial.print("WIFI connected");
 s = EEPROM.read(13);
Serial.println(s);
 if (s == '1') {
  mqtt_setup();
 }
 if (s == '2') {
  tcp_setup();
 }
}
void loop () {
 if (s == '1') {
  mqtt_loop();
 }
 else if (s == '2') {
  tcp_loop ();
 }
 if (mySerial1.available()) {
  h = mySerial1.readString();
 }
}
void mqtt_setup () {
}
void mqtt_loop () {

 if (WiFi.status() == WL_CONNECTED) {
  if (!client.connected()) {
   if (client.connect("arduinoClient")) {
    client.set_callback(callback);
    client.subscribe("Val");
   }
  }
 }

 if (client.connected())
  client.loop();
}
void tcp_setup() {

 server1.begin();
 server1.setNoDelay(true);

 Serial.println("Ready! Use 'TCP ");
 Serial.println(WiFi.localIP()); Serial.print("786' to connect");
}
```

Figure 3(a) Online and Offline Mode Switching Concept

Online mode will directly connect the ESP with the MQTT broker and the industrial sensors information will be monitor at anywhere on the earth with the use of IoT. Offline mode will connect with the local area network as TCP/IP connection then monitor from local connected mobile devices.

MQTT broker is setup on the widows. Moreover, broker will work as offline as well as online. Also, broker will work as a server. Mobile devices can connect with the broker with the use of the android application my mqtt. Our proposed system is working on the publisher and subscriber. Sensors will publish the data to broker and mobile devices will subscribe to the broker to get the data of the sensors. Moreover, every sensor has their topic and based on their topic the send the data to the broker and mobile devices subscribe with that topic so that topic based sensor data will be monitor with the mobile devices.



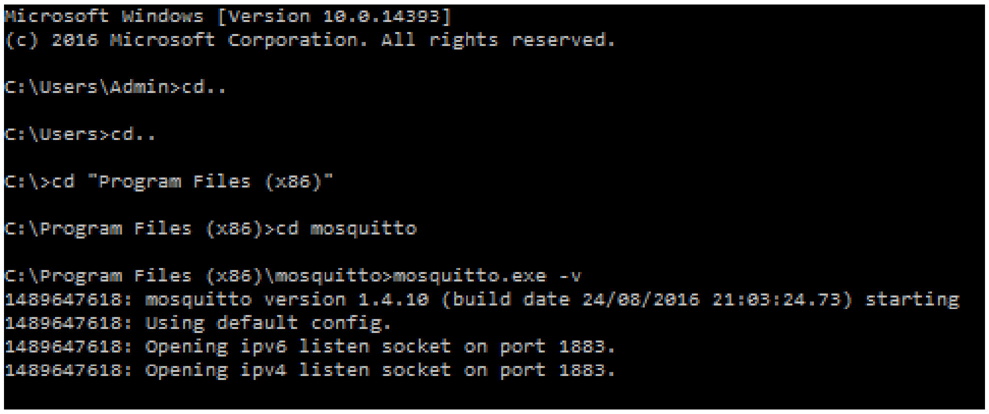
```
  }
}
void mqtt_setup () {
}
void mqtt_loop () {

 if (WiFi.status() == WL_CONNECTED) {
  if (!client.connected()) {
   if (client.connect("arduinoClient")) {
    client.set_callback(callback);
    client.subscribe("Val");
   }
  }
 }

 if (client.connected())
  client.loop();
}
void tcp_setup() {

 server1.begin();
 server1.setNoDelay(true);

 Serial.println("Ready! Use 'TCP ");
 Serial.println(WiFi.localIP()); Serial.print("786' to connect");
}
void tcp_loop() {

 uint16_t i = 0;
 if (server1.hasClient()) {
  for (i = 0; i < MAX_SRV_CLIENTS; i++) {
   //find free/disconnected spot
   if (!serverClients[i] || !serverClients[i].connected()) {
    if (serverClients[i]) serverClients[i].stop();
    serverClients[i] = server1.available();
    Serial.println("New client: "); Serial.print(i);
    continue;
   }
  }
  //no free/disconnected spot so reject
  WiFiClient serverClient = server1.available();
  serverClient.stop();
 }
 for (i = 0; i < MAX_SRV_CLIENTS; i++) {
  if (serverClients[i] && serverClients[i].connected()) {
   if (serverClients[i].available()) {
    //get data from the TCP client and push it to the UART
    while (serverClients[i].available())
     p = serverClients[i].readString();
    r = int(p[0]);
    //mySerial.write(r);
    Serial.println("valu of r is");
    Serial.print(r);

    if (r == '9')
    {
     EEPROM.write(13, '1');
     EEPROM.commit();
     Serial.print("Nishant");
     delay1();
     ESP.restart();

    }

   }

  }
 }
 //mySerial.write(serverClients[i].read());
 if (mySerial1.available()) {
  char sbuf[len];
  memset(sbuf, '\0', len);
  mySerial1.readBytesUntil('\n', sbuf, len);
  Serial.write(sbuf);
  //push UART data to all connected telnet clients
  for (i = 0; i < MAX_SRV_CLIENTS; i++) {
   if (serverClients[i] && serverClients[i].connected()) {
    serverClients[i].write(sbuf, len);
    memset(sbuf, '\0', len);
    delay(10);
   }
  }
 }
}

void delay1() {
 uint16_t i, j;
 for (i = 0; i < 1024; i++);
 for (j = i; j < 255; j++);


}
```

Figure 3(b) Online and Offline Continue Coding

## V. EXPERIMENT RESULT

Figure 1 shows the proposed system block diagram. Moreover, in a practical scenario every sensor's output is the input of the microcontroller (we are using Atmel Atmega328 controller) and controller will serially send the data to the ESP Wi Fi module. MQTT broker needs to be run to start communication between sensors and mobile devices. ESP will publish the data to the broker with the use of the publishing topics. Mobile devices have to subscribe the topics in order to get the industrial sensors data.

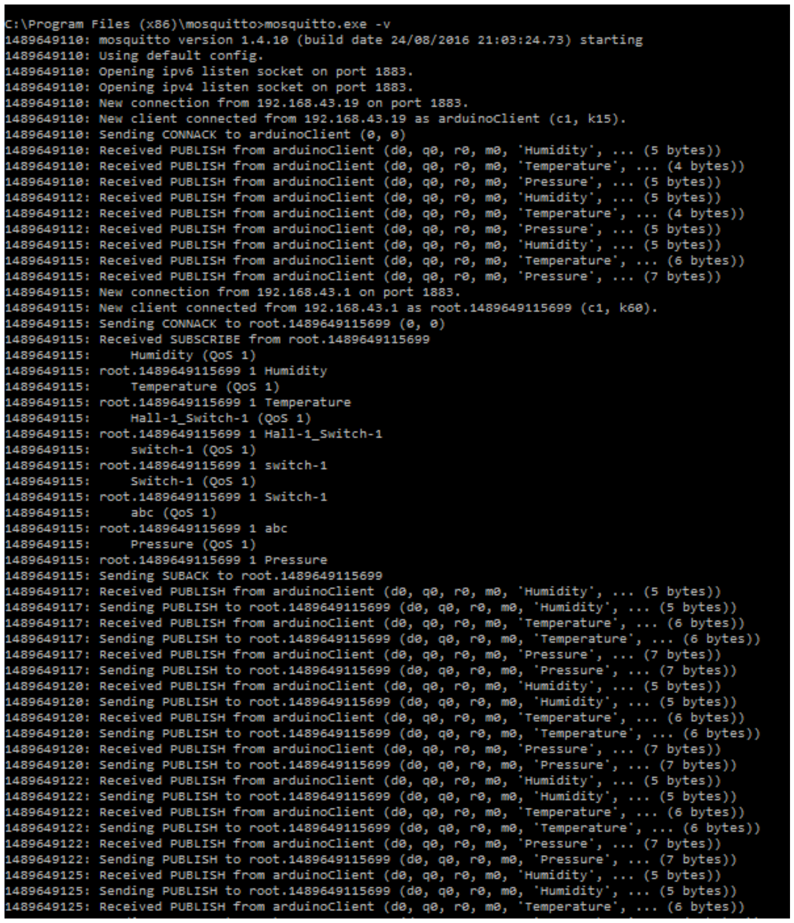
Below is the method to implement the broker on the windows. We are using the Mosquitto broker for the testing purpose as shown on the figure 4.



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd..

C:\Users>cd..

C:\>cd "Program Files (x86)"

C:\Program Files (x86)>cd mosquitto

C:\Program Files (x86)\mosquitto>mosquitto.exe -v
1489647618: mosquitto version 1.4.10 (build date 24/08/2016 21:03:24.73) starting
1489647618: Using default config.
1489647618: Opening ipv6 listen socket on port 1883.
1489647618: Opening ipv4 listen socket on port 1883.
```

Figure 4 Broker Initialization

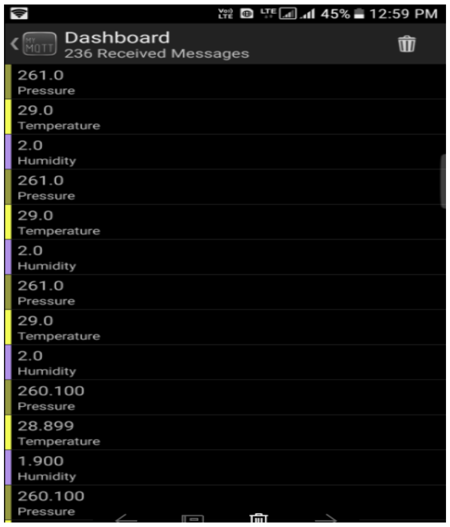
Once the broker will be establishing, ESP will be automatically connected to the broker and at the same time mobile device, with the use of android application will connect to the broker. Broker will show the ESP module as connected and Mobile devices as connected. Also it gives the details of the topics which are published and subscribed as shown in figure 5. However, in the figure 6 it shows the mobile application in which the sensors data will be publish in real time scenarios.



```
C:\Program Files (x86)\mosquitto>mosquitto.exe -v
1489649110: mosquitto version 1.4.10 (build date 24/08/2016 21:03:24.73) starting
1489649110: Using default config.
1489649110: Opening ipv6 listen socket on port 1883.
1489649110: Opening ipv4 listen socket on port 1883.
1489649110: New connection from 192.168.43.19 on port 1883.
1489649110: New client connected from 192.168.43.19 as arduinoClient (c1, k15).
1489649110: Sending CONNACK to arduinoClient (0, 0)
1489649110: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Humidity', ... (5 bytes))
1489649110: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Temperature', ... (4 bytes))
1489649110: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Pressure', ... (5 bytes))
1489649112: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Humidity', ... (5 bytes))
1489649112: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Temperature', ... (4 bytes))
1489649112: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Pressure', ... (5 bytes))
1489649115: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Humidity', ... (5 bytes))
1489649115: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Temperature', ... (6 bytes))
1489649115: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Pressure', ... (7 bytes))
1489649115: New connection from 192.168.43.1 on port 1883.
1489649115: New client connected from 192.168.43.1 as root.1489649115699 (c1, k60).
1489649115: Sending CONNACK to root.1489649115699 (0, 0)
1489649115: Received SUBSCRIBE from root.1489649115699
1489649115:     Humidity (QoS 1)
1489649115: root.1489649115699 1 Humidity
1489649115:     Temperature (QoS 1)
1489649115: root.1489649115699 1 Temperature
1489649115:     Hall-1_Switch-1 (QoS 1)
1489649115: root.1489649115699 1 Hall-1_Switch-1
1489649115:     switch-1 (QoS 1)
1489649115: root.1489649115699 1 switch-1
1489649115:     Switch-1 (QoS 1)
1489649115: root.1489649115699 1 Switch-1
1489649115:     abc (QoS 1)
1489649115: root.1489649115699 1 abc
1489649115:     Pressure (QoS 1)
1489649115: root.1489649115699 1 Pressure
1489649115: Sending SUBACK to root.1489649115699
1489649117: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Humidity', ... (5 bytes))
1489649117: Sending PUBLISH to root.1489649115699 (d0, q0, r0, m0, 'Humidity', ... (5 bytes))
1489649117: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Temperature', ... (6 bytes))
1489649117: Sending PUBLISH to root.1489649115699 (d0, q0, r0, m0, 'Temperature', ... (6 bytes))
1489649117: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Pressure', ... (7 bytes))
1489649117: Sending PUBLISH to root.1489649115699 (d0, q0, r0, m0, 'Pressure', ... (7 bytes))
1489649120: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Humidity', ... (5 bytes))
1489649120: Sending PUBLISH to root.1489649115699 (d0, q0, r0, m0, 'Humidity', ... (5 bytes))
1489649120: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Temperature', ... (6 bytes))
1489649120: Sending PUBLISH to root.1489649115699 (d0, q0, r0, m0, 'Temperature', ... (6 bytes))
1489649120: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Pressure', ... (7 bytes))
1489649120: Sending PUBLISH to root.1489649115699 (d0, q0, r0, m0, 'Pressure', ... (7 bytes))
1489649122: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Humidity', ... (5 bytes))
1489649122: Sending PUBLISH to root.1489649115699 (d0, q0, r0, m0, 'Humidity', ... (5 bytes))
1489649122: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Temperature', ... (6 bytes))
1489649122: Sending PUBLISH to root.1489649115699 (d0, q0, r0, m0, 'Temperature', ... (6 bytes))
1489649122: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Pressure', ... (7 bytes))
1489649122: Sending PUBLISH to root.1489649115699 (d0, q0, r0, m0, 'Pressure', ... (7 bytes))
1489649125: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Humidity', ... (5 bytes))
1489649125: Sending PUBLISH to root.1489649115699 (d0, q0, r0, m0, 'Humidity', ... (5 bytes))
1489649125: Received PUBLISH from arduinoClient (d0, q0, r0, m0, 'Temperature', ... (6 bytes))
```

Figure 5 ESP and Mobile devices connected as well as publish subscribe topics

As same way mobile device provide the monitoring data based on the topics subscribed to the broker of the industry. Sensors data is published on the “Temperature”, “Humidity” and “Pressure” topics. Based on the subscribed topics as given above mobile device will get the real time sensors information as shown in the figure 6 below.



45% 12:59 PM

Dashboard
236 Received Messages

261.0
Pressure

29.0
Temperature

2.0
Humidity

261.0
Pressure

29.0
Temperature

2.0
Humidity

261.0
Pressure

29.0
Temperature

2.0
Humidity

260.100
Pressure

28.899
Temperature

1.900
Humidity

260.100
Pressure

Figure 6 Mobile Device with Subscribe Sensors Topics

We can monitor the data with the use of command prompt, MQTT lens, MQTT android application (which we have used in figure 6) and many more software’s available in the market to monitor the subscribed data.

## VI. CONCLUSION & FUTURE WORK

With this paper, we have shown, the industrial sensors information monitoring system, which is low cost, low power and real time based. Industry 4.0 is the term associated with the Internet of Things and with the help of MQTT protocol we have provided the real time data monitoring for the smart production as well as industrial safety purpose.

As a part of future work, we have found that the proposed system which we had shown in this paper is limited to particular area so in future mesh node network implementation will help to cover every corner of the industry. Moreover, from the security point of view we will allot the SSL certificate based machine to machine communication so that no one outside of the industry will subscribe to the sensors topics.

## REFERENCES

[1] Temperature and humidity module DHT22 Product Manual [Online]Available:https://cdnshop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf (accessed on June 13, 2016)
[2] BMP180 Digital, barometric pressure sensor, [Online] Available:https://aebst.resource.bosch.com/media/_tech/media/product_flyer/Flyer_BMP180_08_2013_web.pdf (accessed on June 14, 2016)
[3] Data sheet BMP180 Digital pressure sensor, [Online] Available: https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS00009.-pdf (accessed on June 14, 2016)
[4] ESP8266 Wi-Fi Module, [Online]. Available: http://www.esp-ruino.com/ESP8266 (accessed on June 15, 2016)
[5] ESP12EBriefSpec[Online]Available:http://www.seeedstudio.-com/wiki/-images/7/7d/ESP-12E_brief_spec.pdf (accessed on June 15, 2016)
[6] Dave Locke, ―MQ Telemetry Transport (MQTT) V3.1 Protocol Specification‖ [Online]. Available: http://www.ibm.com/-developerworks/library/ws-mqtt/ (accessed on June 18, 2016)
[7] MQTT Essentials Part 1 to 9: Detail study on MQTT[Online]Available:http://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels (accessed on June 18, 2016)
[8] Home of RF and Wireless Vendors and Resources, ―What is MQTT in IoT‖ [Online]. Available: http://www.rfwireless-world.com/ Terminology/MQTT-protocol.html (accessed on September 10, 2016)
[9] RTD PT-100 Detail study on its temperature range and with the help of resistance calculate the temperature. Also its 2wire, 3wire, 4wire configuration details http://www.omega.com/temperature/z/pdf/z033-035.pdf (accessed on September 16, 2016)
[10] MQ-135 Sensors application, detecting gases with the study of https://www.olimex.com/Products/Components/Sensors/SNS-MQ135/resources/SNS-MQ135.pdf
[11] Michal Lom; Ondrej Pribyl; Miroslav Svitek "Industry 4.0 as a part of smart cities" 2016 Smart Cities Symposium Prague (SCSP) Year: 2016
[12] Jorge E. Luzuriaga; Miguel Perez; Pablo Boronat; Juan Carlos Cano; Carlos Calafate; Pietro Manzoni, "A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks," 2015 12th Annual IEEE Consumer Communications and Networking Conference(CCNC) Year: 2015.
[13] Surapon Kraijak; Panwit Tuwanut, "A survey on internet of things architecture, protocols, possible applications, security, privacy, real-world implementation and future trends," 2015 IEEE 16th International Conference on Communication Technology (ICCT) Year: 2015
[14] Ala Al-Fuqaha; Mohsen Guizani; Mehdi Mohammadi; Mohammed Aledhari; Moussa Ayyash "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications" IEEE Communications Surveys & Tutorials Pages: 2347 - 2376, DOI: 10.1109/COMST.2015.2444095 Year: 2015, Volume: 17, Issue: 4