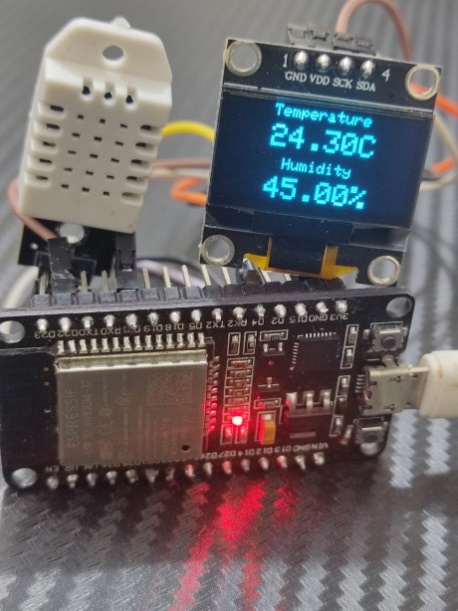
**Lab Report**

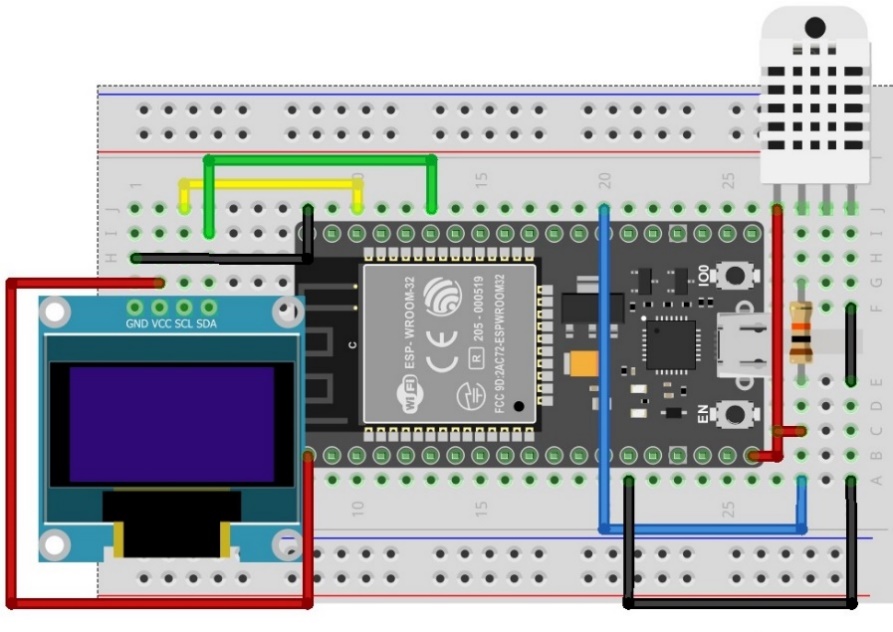
**Internet of Things (IOT)**

The term internet of things encompasses everything that is connected to internet. But it is mostly used to define devices that talk to each other. IOT comprises of most of the devices we use today including smart phones, smart watches, toasters, many kitchen equipment’s, sensors and many more. These sensors can be active sensors which means they required power source to run or passive sensors which do not require an active power source to drive them.

The layers of communication vary some time but the main purpose of them is to have some architectures upon which the whole system based on. The physical or perception layer shows the wiring or physical connections of devices and sensors which can be communicating on different protocols but are physically linked with each other. Then the overall network layer comes into play which is responsible for connecting to other smart things, network devices, and servers. Its features are also used for transmitting and processing sensor data. After that we have application layer which is used to deliver application specific for end user which some user-friendly interface.

**System Design**

Internet of things has helped connect our devices, systems and even sensors with cloud and with each other. A few components interfaced together can form an IOT. This project is also an IOT based system, which uses ESP32 as Microcontroller which has Wi-Fi and Bluetooth functionality.



In this project DHT22 sensor is used to monitor temperature and humidity of atmosphere. The 0.96-inch OLED display is used to display the sensor data in a very presentive way for end user to check temperature and humidity in real time. The heart of this system is ESP32 which is a very powerful microcontroller which multiple communication interfaces and protocols. The OLED is 12c based display and is communicating with ESP32, whereas DHT22 is a digital sensor. MQTT protocol is used in this project to publish the sensor data. Furthermore, the data is presented in MQTT dashboard application where end user can keep track of sensor data online at any time.

**Code and Implementation**

The code is very simple and well explained with describing functionality on every function. The Arduino IDE is used for development of firmware.

#include <Wire.h> // 12c protocol library

#include <WiFi.h> // wifi library for esp32

#include <PubSubClient.h> // MQTT library

#include <Adafruit\_Sensor.h> // Sensor library

#include <Adafruit\_GFX.h> // Sensor library

#include <Adafruit\_SSD1306.h> // Oled library

#include "DHT.h" // DHT22 library

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

#define DHTTYPE DHT22 // DHT sensor type we are using DHT22

#define DHTPIN 4 // Digital Pin for DHT22 sensor

const char\* ssid = "Your SSID"; // Network SSID

const char\* password = "Your Password"; // Nerwork Password

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1);

DHT dht(DHTPIN, DHTTYPE); // Temperature Sensor Pin and Type

WiFiClient espClient; // WiFi Client

PubSubClient client(espClient); // MQTT Client

const char\* mqtt\_server = "137.135.83.217"; // MQTT Broker IP

int mqtt\_port = 1883; // MQTT Broker Port

float h, t;

String hum, temp;

This portion of code is including necessary libraries used in this project and defining the required paraments and variables that will be used in code later on. Wire.h is 12c library of arduino IDE which is used for 0.96 inch OLED SSD1306 display. WiFi.h is library of ESP32 which uses three modes Access Point(AP), Station(STA) and Both in this code we will use only Station mode in ESP32 will connect to our router. PubSubClient.h is library of MQTT Communcation for ESP8266 and ESP32. This library has two main functions to publish some data to topic and subscribe a topic to get data when its ready there is very less letancy. Then there are sensor libraries of adafruit for 12c Display and DHT sensor. DHT is temperature and humidity sensor which is most commonly used sensor for DIY projects and wide range of temperature almost -50 to 150 degree centigrates. SSD1306 is 0.96 inch OLED display with screen width of 128 pixels and heigth of 64 pixel. In last portion of this we have created WiFi Client and also MQTT Client to use MQTT protocol on ESP32 over WiFi Network. We are using free MQTT broker "137.135.83.217" with port 1883, there are number of avaiable brokers which gives free access for communcation but I found this broker most relaible so far.

void setup() {

Serial.begin(115200); // Serial Started

setup\_wifi(); // Call Wifi Setup Function

client.setServer(mqtt\_server, mqtt\_port); //Setup MQTT Server

dht.begin(); // begin dht temperature sensor

// Address 0x3D for 128x64

if (!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) {

Serial.println("Failed to begin 12C\_OLED Display");

delay(1000);

} else {

display.clearDisplay(); //clearing display

display.setTextSize(1); // text size can be selected from 1-5

display.setTextColor(WHITE); //text color

}

}

This is the setup function which is used to initialize all the required functions of program this is one time declaration and after that code will enter infinite loop. In this function we started Serial for UART communication, there are three hardware Serial available in ESP32 if we want to use other two, we can simply begin them by Serial1 and Serial2. We are staring Serial on 115200 baud rate which is speed of Serial and can be selected differently if want to. Setup\_wifi is function to begin Wi-Fi in Station mode which will be explained in next portion of code. Then we are setting MQTT server with provided MQTT broker IP and port, client.setServer(mqtt\_server, mqtt\_port); function is from pubsubclient library through which we are setting MQTT server. Similarly dht.begin(); is function from DHT library to initialize the dht22 sensor. Then there is initialization of OLED display display.begin(SSD1306\_SWITCHCAPVCC, 0x3C) in which are passing OLED display setting and 12c address. After that we are simply declaring one time parameters of display like text size and text color.

void setup\_wifi() {

// Defining The mode of WIFI in our we only need station

WiFi.mode(WIFI\_STA);

WiFi.begin(ssid, password); // Setting WiFi

// Waiting until WiFi connects

while (WiFi.status() != WL\_CONNECTED) {

delay(500); //delay 500 miliseconds

Serial.print(".");

}

Serial.println("WiFi connected"); //printing wifi connected

}

This is WIFI begin function which put ESP32 in WIFI Station mode and connects to available wireless network. There are three types of Wi-Fi modes in esp32, Wi-Fi AP mode in which ESP32 act as Access Point and provide network to other devices and they can connect with it. In AP mode ESP32 can support up to five connections, which means five devices can connect to ESP32 at a time. The second mode is Wi-Fi Station mode which we are using in this project. In Wi-Fi Station mode ESP32 connects with connect Access Point (Home Wi-Fi Router) and gets DHCP IP and communication over the Internet like other devices. Third is Wi-Fi AP and Station Mode in which ESP32 uses both functionalities AP and Station at a same time.

void readDHT() {

h = dht.readHumidity(); // Getting Humidity

t = dht.readTemperature(); // Getting Temperature

if (isnan(h) || isnan(t)) { // Checking if any value is NAN or not if yes then there is some issue with sensor

Serial.println("Failed to read from DHT sensor!");

return; // Return to if satement until sensor works again

}

hum = ""; temp = ""; // Clearing String

hum = String(h); // Converting Float to String

hum += '%'; // Converting Float to String

temp = String(t); // Converting Float to String

temp += 'C'; // Converting Float to String

}

This is the function in which DHT22 performs reading and then this value is converted into string to show on OLED display. dht.readHumidity() and dht.readTemperature() functions are used get Humidity and Temperature value in float from dht library. DHT22 is a digital sensor which means it is connected to digital pin of controller to perform its value. After getting the values from sensor it is stored in String to later on Display this value to Serial monitor, OLED display and also send to MQTT server.

void showDisplay() {

display.clearDisplay(); // Clear previous display

display.setTextSize(1); //text size is set to 1

display.setTextColor(WHITE); //text color

// x,y cursor point on display of 128,64

display.setCursor(30, 0); //set cursor where to print

display.println("Temperature"); // print temperature

display.setCursor(40, 35); //set cursor where to print

display.println("Humidity"); // print humidity

display.setTextSize(2); //text size is set to 2

display.setTextColor(WHITE); //text color

display.setCursor(30, 13); //set cursor where to print

display.println(temp); // print temperature value

display.setCursor(30, 48); //set cursor where to print

display.println(hum); // print humidity values

display.display(); // Display new values

}

This function shows the temperature and humidity values on OLED display. setCursor function is used to define the cursor potion where we want to print on display in below figure, we can see that where we are printing temperature and humidity and where we are displaying the values of temperature and humidity, respectively. For printing the temperature and humidity on display we are using setTextSize(1) and for values we are using setTextSize(2). After that we are going to call display.display to show printed values on display.



void reconnect() {

// Loop until we're reconnected

while (!client.connected()) {

Serial.print("Attempting MQTT connection...");

String clientId = "ESP32Client-";

// Create a random client ID

clientId += String(random(0xffff), HEX);

if (client.connect(clientId.c\_str())) { // Attempt to connect

Serial.println("connected");

} else {

Serial.print("failed, rc=");

Serial.print(client.state());

Serial.println(" try again in 3 seconds");

delay(3000); // Wait 3 seconds before retrying

}

}

}

This is MQTT reconnect function in case we lost connection with MQTT broker. In while function we are checking connection with MQTT broker if it is not connected it will enter in while and stays there until connection is established again. Client ID is required for connection to make that why we are assigning new ID every time it can also work on fixed ID as well. Attempt of connecting a client is done every three it can be reduced but its better to have some delay before attempting again.

void MQTTpublish() {

char sendtemp[4]; // Character of 4 bytes as our range is 0-100

sprintf(sendtemp, "%d", int(t)); // Storing value in Char array

// Character of 4 bytes as our range is 0-100

char sendhum[4];

sprintf(sendhum, "%d", int(h)); // Storing value in Char array

//printing temperature and humidity values on serial monitor

Serial.print("Temperature : "); Serial.println(sendtemp);

Serial.print("Humidity : "); Serial.println(sendhum);

Serial.println();

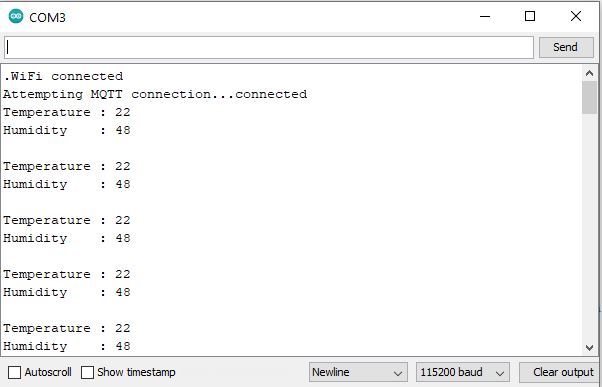
// Publishing Sensor value to Esp32-DHT\_Hum Topic

client.publish("Esp32-DHT\_Hum", sendhum);

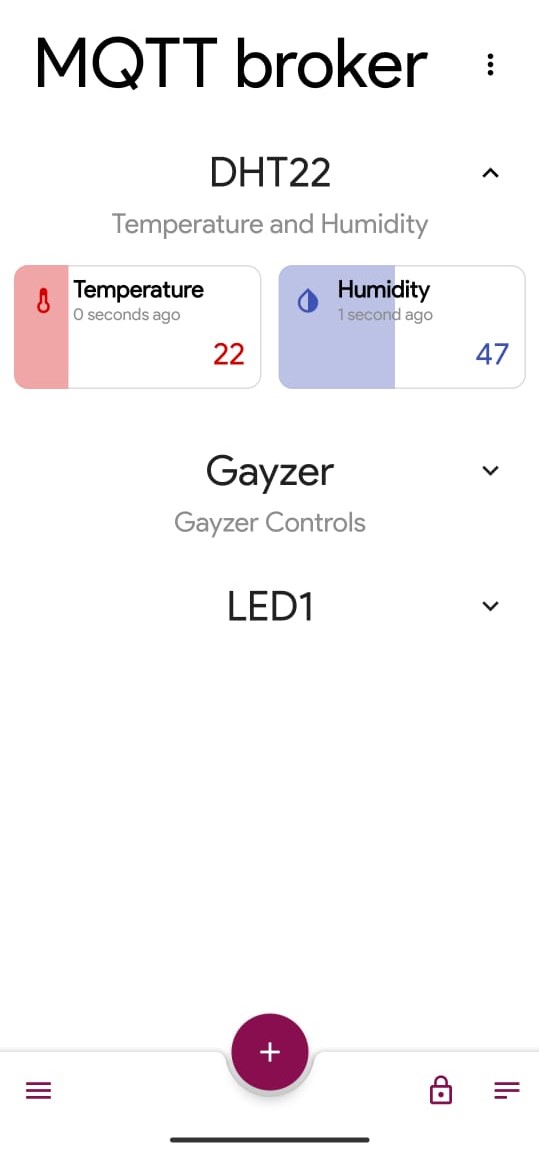
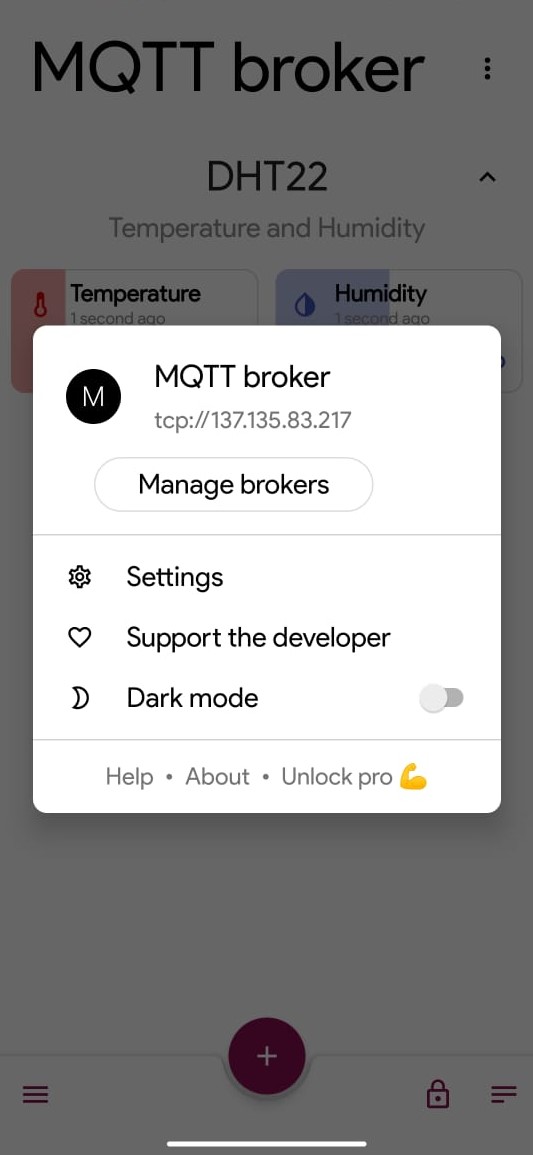
// Publishing Sensor value to Esp32-DHT\_Temp Topic

client.publish("Esp32-DHT\_Temp", sendtemp);

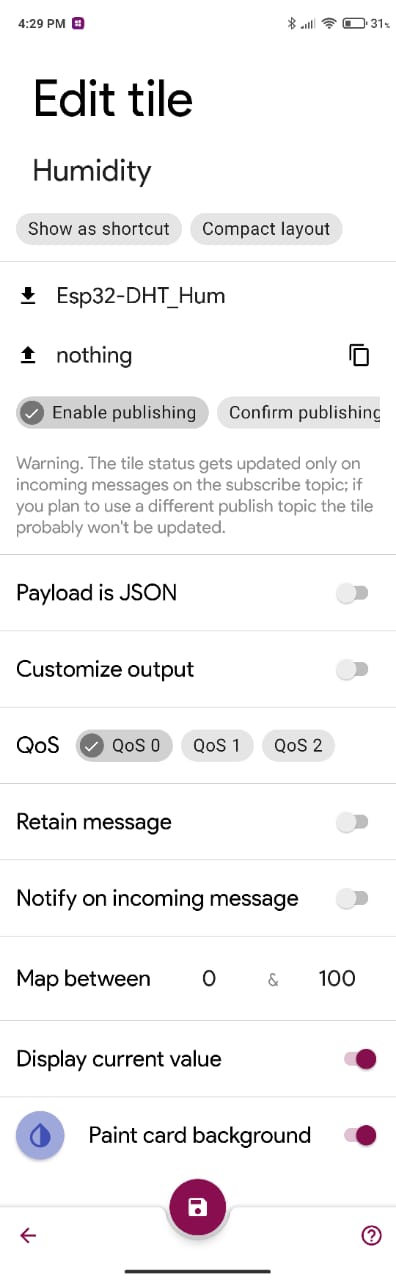
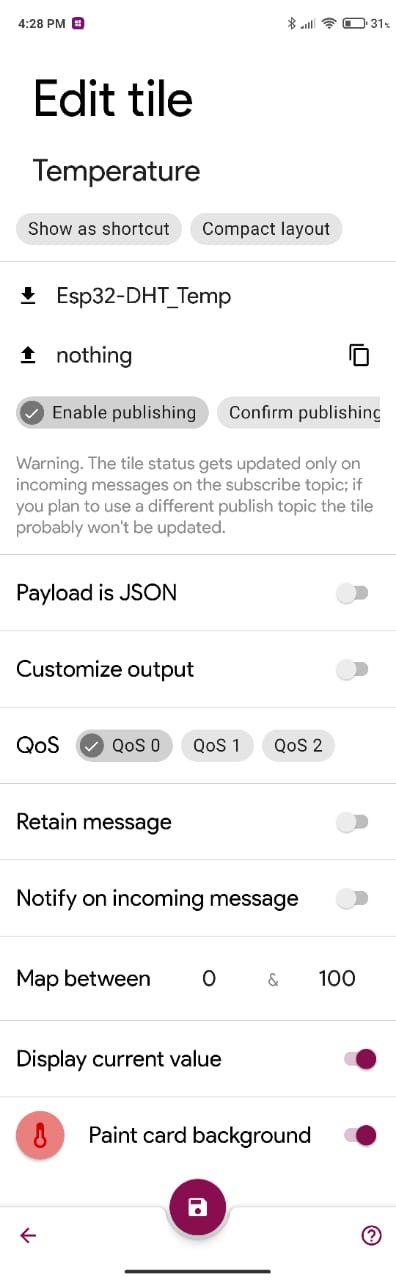
}

This function is publishing sensor data to MQTT server and printing sensor values on serial monitor. We are using sprintf function to convert decimal value to character array because client.publish takes values in character array, the size of array is set to 4 bytes to accommodate the range of our sensor values which is 0-100 almost. client.publish uses topic and the value to publish, later on we will see how this value is received in Application.

This is output of Serial Monitor in which we can see that ESP32 got Wi-Fi Connection, then it shows that MQTT connection is being established. After that temperature and humidity values are printed every three seconds on serial monitor.



These above screen shots show the application interface how sensor data is displayed in the form of blocks where sensor data is updated every three seconds. The broker setting is required at application end it must be with same port other wise we will not be able to communicate.



In above screen shots we can see how MQTT setting is done in Application. First, we have assigned block with a tile like Temperature and then subscribe topic we enter the name of our topic which is "Esp32-DHT\_Temp" and "Esp32-DHT\_Hum". And then fill publish topic which nothing so we do not have to publish from our application we just want to receive value from device. At the end of this setting page, we have graphical features for sensor block through this we can select icon, icon color and background color etc. Map between the range at application it has nothing to do with data received from device this can be set according to users’ requirement.

void loop() {

if (!client.connected()) { // Check client is connected or not

reconnect(); // if not connected call reconnect function

}

client.loop(); // run Loop function while client is connected

readDHT(); // Read Sensor values

MQTTpublish(); // Publish Sensor data to MQTT Server

showDisplay(); // Show Sensor data on OLED Display

delay(3000); // Delay function to wait for 3 seconds before getting new data

}

This is loop which call required functions and runs infinitely. client.connected return 1 or 0 depending to weather it is connected or not in this case if we receive 0 it will go into if statement and calls reconnect() function. client.loop() runs while client is connected. After that sensor data is gathered every three seconds and publish the data on MQTT broker and print the data on Serial and OLED display similarly.

**Conclusion**

Message Queuing Telemetry Transport (MQTT) is widely used commination protocol. MQTT is very reliable and efficient communication with extremely light weight overhead, which is best suited for remote sensing and control. Publish/subscribe protocol collects more data with less bandwidth compared to polling protocols.

In our case we want to get sensor data as early as possible with less complexity and maximum reliability, thus MQTT is best suited for this kind of system. Where we want to gather sensor data and post it to server without any major overhead. On other hand mobile application can access the data through same topic at any time. MQTT is best approach for this type of implementation and can be deployed without much effort.

**REFERENCES**

Random Nerd Tutorials, (2017) “Installing the ESP32 Board in Arduino IDE” [online]. Available at: <https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>. (Accessed: 7 March 2021)

Dherrada B., (2020) “Adafruit\_sensor”[online]. Available at: <https://github.com/adafruit/Adafruit_Sensor> (Accessed: 6 March 2021)

Dherrada B., (2021) “Adafruit\_SSD1306”[online]. Available at: <https://github.com/adafruit/Adafruit_SSD1306> (Accessed: 4 March 2021)

Dherrada B., (2020) “DHT-sensor-library”[online]. Available at: <https://github.com/adafruit/DHT-sensor-library> (Accessed: 4 March 2021)

Dherrada B., (2021) “Adafruit-GFX-library”[online]. Available at: <https://github.com/adafruit/Adafruit-GFX-Library> (Accessed: 8 March 2021)

Knollearry T., (2020) “Pubsubclient”[online]. Available at: <https://github.com/knolleary/pubsubclient> (Accessed: 8 March 2021)