Source Code

NodeMCU.ino Code

```
#include <ESP8266WiFi.h>
#include "DHT.h"
#include <PubSubClient.h>
const char * ssid = "Tahmid";
const char * password = "passwordnai";
const char * mqtt_server = "91.121.93.94"; // test.mosquitto.org
const char * pubSubTopic = "CSE323/SEC07";
DHT dht(D5, DHT22);
const int PIN_NUM = 4;
const int PINS[] = { -1, D1, D2, D7, D8};
int pin_status[] = {-1, 0, 0, 0, 0};
String status_message = "m0000";
WiFiClient espClient;
PubSubClient client(espClient);
const int MSG_BUFFER_SIZE = 50;
char msg[MSG_BUFFER_SIZE];
void turn_on(const int pin) {
  digitalWrite(PINS[pin], HIGH);
  pin_status[pin] = 1;
}
void turn_off(const int pin) {
  digitalWrite(PINS[pin], LOW);
  pin_status[pin] = 0;
}

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  randomSeed(micros());
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
char buffer[10];
void reportStatus() {
  for (int i = 1; i <= PIN_NUM; ++i) {
```

```
    status_message[i] = pin_status[i] + '0';
  }
  String report = status_message;
  report += '~';
  report += dtostrf(dht.readHumidity(), 5, 2, buffer);
  report += '~';
  report += dtostrf(dht.readTemperature(), 5, 2, buffer);

  client.publish(pubSubTopic, report.c_str());
}

void callback(char * topic, byte * payload, unsigned int length) {
  if (payload[0] == 'm') {
    return;
  }
  if ((char) payload[2] == '1') {
    turn_on((char) payload[1] - '0');
  } else if ((char) payload[2] == '0') {
    turn_off((char) payload[1] - '0');
  }
  reportStatus();
}
void reconnect() {
  digitalWrite(BUILTIN_LED, HIGH);
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");

    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      digitalWrite(BUILTIN_LED, LOW);
      client.subscribe(pubSubTopic);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
void setup() {
  dht.begin();
  pinMode(BUILTIN_LED, OUTPUT);
  for (int i = 1; i <= PIN_NUM; ++i) {
    pinMode(PINS[i], OUTPUT);
  }
  for (int i = 1; i <= PIN_NUM; ++i) {
    digitalWrite(PINS[i], LOW);
  }
  digitalWrite(BUILTIN_LED, HIGH);
  Serial.begin(115200);
```

```
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

unsigned long last_refresh_time = 0;
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  unsigned long now = millis();
  if (now - last_refresh_time >= 2000) {
    last_refresh_time = now;
    reportStatus();
  }
}
```

JavaScript

Index.js

```
let start = new Date("2023-01-01");
let prev = "";
const num_of_switch = 4;
const topic = "CSE323/SEC07";
const mqttClient = mqtt.connect('wss://test.mosquitto.org:8081');
mqttClient.subscribe(topic);
mqttClient.on('message', function(topic, message) {
    payload = message.toString();
    processMessage(payload);
});
mqttClient.on('connect', () => {
    console.log('Connected to MQTT broker');

});
mqttClient.on('error', (error) => {
    console.error('Error:', error);
});
mqttClient.on('close', () => {
    console.log('Disconnected from MQTT broker');
    setTimeout(() => {
        mqttClient.connect();
    }, 5000);
});
function publishMessage(message) {
    mqttClient.publish(topic, message);
    setTimeout(function() {}, 1000);
}

async function updateHT(message){
  const info = message.split('~');
  const temparature = Number(info[2]);
```

```
  const humidity = Number(info[1]);
  changeChart(temparature, humidity);
  document.getElementById('tinfo').textContent = `Temparature:
${temparature} *C`;
  document.getElementById('hinfo').textContent = `Humidity: ${humidity} %`;
}

function processMessage(message) {
    if (message[0] == "u" || message[0] != 'm') {
        return
    }
    start = new Date();
    updateHT(message);
    if (message == prev) {
        return;
    }
    prev = message;
    for (i = 1; i <= num_of_switch; ++i) {
        if (message[i] == '1') {
            turn_on_switch(i);
        } else if (message[i] == '0') {
            turn_off_switch(i);
        }
    }
}
function turn_on_switch(switch_no) {
    const button = document.getElementById("switch-" + i.toString());
    button.classList.remove("off");
    button.classList.remove("disconnect");
    button.classList.add("on");
}
function turn_off_switch(switch_no) {
    const button = document.getElementById("switch-" + i.toString());
    button.classList.remove("on");
    button.classList.remove("disconnect");
    button.classList.add("off");
}
function disconnect_switch(i) {
    const button = document.getElementById("switch-" + i.toString());
    button.classList.remove("on");
    button.classList.remove("off");
    button.classList.add("disconnect");
}
function switch_click(i) {
    let message;
    const button = document.getElementById("switch-" + i.toString());
    if (button.classList.contains('off')) {
        message = "u" + i.toString() + "1";
    } else {
        message = "u" + i.toString() + "0";
    }
    publishMessage(message);
```

```
}
function refresh_connection() {
    const now = new Date();
    const elapsedSeconds = Math.floor((now - start) / 1000);
    if (elapsedSeconds >= 10) {
        for (let i = 1; i <= num_of_switch; ++i){
            disconnect_switch(i);
        }
    }
    prev = "";
}
refresh_connection();
setInterval(refresh_connection, 1000);
```

chart.js

```
var temChart;
var humiChart;
var labels = ['48', '46', '44', '42', '40', '38', '36', '34', '32', '30',
'28', '26', '24', '22', '20', '18', '16', '14', '12', '10', '8', '6', '4',
'2', '0'];
let temparatures = []
let humidities = [];
document.addEventListener("DOMContentLoaded", function () {
  temChart = new
Chart(document.getElementById('temparature').getContext('2d'), {
      type: 'line',
      data: {
          labels: labels,
          datasets: [{
              label: '',
              data: temparatures,
              borderColor: 'rgba(255, 99, 132, 0.8)',
              borderWidth: 2,
              fill: false
          }]
      },
      options: {
        scales: {
              y: {
                suggestedMin: null,
                suggestedMax: null,
                grid: {
                        color: 'rgba(255, 99, 132, 0.2)'
                    }
            }
        },
        animations: {
          tension: {
            duration: 1000,
            easing: 'linear',
            from: 1,
            to: 0,
```

```
            loop: true
          }
        },
      }
  });
  humiChart = new
Chart(document.getElementById('humidity').getContext('2d'), {
      type: 'line',
      data: {
          labels: labels,
          datasets: [{
              label: '',
              data: humidities,
              borderColor: 'rgba(54, 162, 235, 0.8)',
              borderWidth: 2,
              fill: true
          }]
      },
      options: {
          scales: {
                y: {
                  suggestedMin: null,
                  suggestedMax: null,
                  grid: {
                          color: 'rgba(54, 162, 235, 0.2)'
                      }
              }
          },
          animations: {
            tension: {
              duration: 1000,
              easing: 'linear',
              from: 1,
              to: 0,
              loop: true
            }
          },
        }
  });
});

function changeChart(t, h){
  temparatures.push(t);
  humidities.push(h);
  if(temparatures.length > 25){
    temparatures.shift();
    humidities.shift();
  }
  temChart.data.datasets[0].data = temparatures;
  humiChart.data.datasets[0].data = humidities;
  temChart.update();
  humiChart.update();
```

```
}
```

Web HTML

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="style.css" rel="stylesheet">
    <title>Smart Home</title>
  </head>
  <body>
    <div class="grid-container">
      <div class="grid-item disconnect" id="switch-1">
        <h1>Light 1</h1>
        <div class="switch">
          <button class="bulb-btn" type="button"
onclick="switch_click(1)"></button>
        </div>
      </div>
      <div class="grid-item disconnect" id="switch-2">
        <h1>Light 2</h1>
        <div class="switch">
          <button class="bulb-btn" type="button"
onclick="switch_click(2)"></button>
        </div>
      </div>
      <div class="grid-item disconnect" id="switch-3">
        <h1>Light 3</h1>
        <div class="switch">
          <button class="bulb-btn" type="button"
onclick="switch_click(3)"></button>
        </div>
      </div>
      <div class="grid-item disconnect" id="switch-4">
        <h1>Light 4</h1>
        <div class="switch">
          <button class="bulb-btn" type="button"
onclick="switch_click(4)"></button>
        </div>
      </div>
    </div>
    <div class="info">
        <h1 id="tinfo" style="color: rgba(255, 99, 132, 0.8)">Temparature:
null *C</h1>
      <canvas id="temparature" width="600" height="210"></canvas>
        <h1 id="hinfo" style="color: rgba(54, 162, 235, 0.8)" >Humidity:
null %</h1>
      <canvas id="humidity" width="600" height="210"></canvas>
```

```
    </div>
  </body>
  <script src="https://unpkg.com/mqtt/dist/mqtt.min.js"
type="text/javascript"></script>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <script src="index.js" type="text/javascript"></script>
  <script src="chart.js" type="text/javascript"></script>
</html>
```

Stylesheet

```
* {
  box-sizing: border-box;
}
body {
  display: flex;
  justify-content: center;
  align-items: center;
  text-align: center;
  height: 100vh;
  margin: 0;
  background-color: #111;
  font-family: "system-ui";
}
h1 {
  margin: 0 auto;
  margin-bottom: 10px;
}
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 1fr 1fr;
  gap: 10px;
  overflow: hidden;
}

.grid-item {
  margin: 30px;
  margin-top: 20px;
  margin-bottom: 20px;
  padding: 20px;
  padding-left: 30px;
  padding-right: 30px;
  text-align: center;
  background-color: #111;
  border-radius: 15px;
}
.switch {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-between;
```

```
  background-size: contain;
  background-position: top center;
  background-repeat: no-repeat;
  width: 150px;
  height: 210px;
  margin: 0 auto;
}
.on {
  box-shadow: 0 10px 20px rgba(255, 255, 255, 0.8);
  color: #ccc;
  transition: box-shadow 1s ease;
}
.off {
  box-shadow: 0 10px 20px rgba(255, 255, 255, 0.3);
  color: #aaa;
  transition: box-shadow 1s;
}
.disconnect {
  box-shadow: 0 10px 20px rgba(255, 120, 120, 0.3);
  color: rgba(255, 120, 120, 0.7);
  transition: box-shadow 1s;
}
.on .switch {
  background-image: url("src/bulb-on.png");
  transition: background-image 1s ease;
}

.off .switch {
  background-image: url("src/bulb-off-2.png");
  transition: background-image 1s ease;
}

.disconnect .switch {
  background-image: url("src/bulb-not.png");
  transition: background-image 1s ease;
}

.off .bulb-btn {
  background-image: url("src/switch-on.png");
  transition: background-image 1s ease;
}
.on .bulb-btn {
  background-image: url("src/switch-off.png");
  box-shadow: 0 0 15px rgba(255, 255, 255, 1);
  transition: background-image 1s ease;
}
.off .bulb-btn:hover {
  box-shadow: 0 0 15px rgba(255, 255, 255, 1);
}
.on .bulb-btn:hover {
  box-shadow: none;
}
```

```css
.disconnect .bulb-btn {
  display: none;
}
.bulb-btn {
  background-size: contain;
  background-repeat: no-repeat;
  display: block;
  margin-top: auto;
  width: auto;
  height: 50px;
  width: 50px;
  border-radius: 50%;
  outline: none;
  border: none;
  cursor: pointer;
}
.grid-item:hover .bulb-btn {
  height: 55px;
  width: 55px;
}
.connection-info {
  position: fixed;
  top: 50px;
  color: #aaa;
}

.info{
  margin-left: 80px;
  border: 1px solid white;
  width: 40%;
  color: #aaa;
  padding: 10px;
}
```