Connections

- `#define DHTPIN 2` -> D4
- `#define DHTPIN 4` -> D2
- `#define DHTPIN 5` -> D1
- `#define DHTPIN 14` -> D5
- `#define DHTPIN 12` -> D6
- `#define DHTPIN 13` -> D7
- `#define DHTPIN 16` -> D0

EXP-1

```
#include <DHT.h> // Including library for dht
#define DHTPIN 2 //pin where the dht11 is connected
DHT dht(DHTPIN, DHT11);
void setup()
{
  Serial.begin(9600); // Setting Baud rate to 9600
}
void loop()
{
  float h = dht.readHumidity(); // read humidity
  float t = dht.readTemperature(); // read temp
  Serial.println("Kshitij 47");
  Serial.print(" Current humidity = ");
  Serial.print(h);
  Serial.print("% ");
  Serial.print("temperature = ");
  Serial.print(t);
  Serial.println("C ");
  delay(2000);
}
```

EXP-2

```
#include <DHTesp.h> // Including library for DHT
#include <ESP8266WiFi.h>

String apiKey = "993LLLQYV403CGAH"; // Replace with your Write API Key from ThingSpeak
const char *ssid = "TP-LINK"; // Replace with your WiFi SSID
const char *pass = "Swapna1981"; // Replace with your WiFi Password
const char *server = "api.thingspeak.com";
#define DHTPIN 4 // Pin where the DHT11 is connected

DHTesp dht;
WiFiClient client;

void setup() {
    Serial.begin(9600); // Setting Baud rate to 9600
    delay(10);
    dht.setup(DHTPIN, DHTesp::DHT11); // Connect DHT sensor to GPIO 4
    Serial.println("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nWiFi connected");
}

void loop() {
    float h = dht.getHumidity(); // Read humidity
    float t = dht.getTemperature(); // Read temperature


    if (isnan(h) || isnan(t)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }

    if (client.connect(server, 80)) {
        String postStr = apiKey;
        postStr += "&field1=";
        postStr += String(t);
        postStr += "&field2=";
        postStr += String(h);
        postStr += "\r\n\r\n";

        client.print("POST /update HTTP/1.1\n");
        client.print("Host: api.thingspeak.com\n");
        client.print("Connection: close\n");
        client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
        client.print("Content-Type: application/x-www-form-urlencoded\n");
        client.print("Content-Length: ");
        client.print(postStr.length());
        client.print("\n\n");
        client.print(postStr);
        Serial.println("Data Sent to ThingSpeak.");
    }
    client.stop();
    delay(15000); // ThingSpeak requires a 15-second delay between updates
}
```

EXP-3

```
#define BLYNK_TEMPLATE_ID "TMPL34XZr63KO"
#define BLYNK_TEMPLATE_NAME "Led Blink"
#define BLYNK_AUTH_TOKEN "qVmqLOSUwG4AUa8x0KTTwq_MeYGaWma8"

#define DHTTYPE DHT11
#define BLYNK_PRINT Serial
#define DHT_PIN 4

#include <DHT.h>
#include <DHT_U.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

DHT dht(DHT_PIN, DHTTYPE);
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Mahek";
char pass[] = "12345678";
float t;
float h;
void setup() {
  Serial.begin(115200);
  delay(100);
  WiFi.begin(ssid, pass);
  Serial.print("Connecting to WiFi");
  int attempt = 0;
  while (WiFi.status() != WL_CONNECTED && attempt < 15) {
    delay(1000);
    Serial.print(".");
    attempt++;
  }
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nConnected to WiFi!");
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("\nFailed to connect to WiFi. Check credentials.");
  }
  Blynk.begin(auth, ssid, pass);
  dht.begin();
  Serial.println("DHT sensor initialized");
}
void sendUptime() {
  h = dht.readHumidity();
  t = dht.readTemperature();
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.println("\nHumidity and Temperature:");
  Serial.print("Kshitij 47 | Sarthak 40 | Mahek 31 Current humidity = ");
  Serial.print(h);
  Serial.print("%, Temperature = ");
  Serial.print(t);
  Serial.println("°C");
  Blynk.virtualWrite(V0, t);
  Blynk.virtualWrite(V1, h);
}
void loop() {
  Blynk.run();
  static unsigned long lastUpdate = 0;
  if (millis() - lastUpdate >= 2000) {
    sendUptime();
    lastUpdate = millis();
  }
}
```

EXP-4

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <DHT.h>
// DHT sensor setup
#define DHTPIN 4 // Pin where the DHT11 is connected
#define DHTTYPE DHT11 // DHT11 sensor
DHT dht(DHTPIN, DHTTYPE);
const char* ssid = "TP-LINK"; // Replace with your Wi-Fi network name
const char* password = "Swapna1981"; // Replace with your Wi-Fi password
String pushbulletAPIKey = "o.ta6TKZPdl0LvneGz7xyQSTAe3y7AgcuT"; // Replace
with your Pushbullet API key
String pushbulletURL = "https://api.pushbullet.com/v2/pushes";
WiFiClientSecure client;
void setup() {
  Serial.begin(9600);
  dht.begin();
  // Connect to Wi-Fi
  Serial.println("Connecting to Wi-Fi...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("Connected to Wi-Fi");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
  // Set up the secure client
  client.setInsecure();
}
void loop() {
  // Read temperature and humidity from DHT11
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  // Check if the reading failed and exit if so
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor! Retrying...");
    delay(2000); // Wait a bit before retrying
    return;
  }
  // Prepare the notification message with DHT11 data
  String message = "{\"type\": \"note\", \"title\": \"NodeMCU DHT11
Notification\", ";
  message += "\"body\": \" Roll no.47 , Temperature: " +
String(temperature) + "°C, Humidity: " + String(humidity) + "%\"}";
  // Send the notification to Pushbullet
  bool pushSent = sendPushbulletNotification(message); // Store the return
value
  // Print to serial monitor
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print("°C, Humidity: ");
  Serial.print(humidity);
  if (pushSent) {
    Serial.println("% | Notification sent on Pushbullet");
  } else {
    Serial.println("% | Notification sending failed"); //Added error
message
  }
  // Wait for 60 seconds before sending the next notification
  delay(10000); // 10 second delay. Consider increasing to 60000 for 1
minute.
}
bool sendPushbulletNotification(String message) {
  if (client.connect("api.pushbullet.com", 443)) {
    // Sending POST request to Pushbullet API
    client.println("POST /v2/pushes HTTP/1.1");
    client.println("Host: api.pushbullet.com");
    client.println("Authorization: Bearer " + pushbulletAPIKey);
    client.println("Content-Type: application/json");
    client.print("Content-Length: ");
    client.println(message.length());
    client.println(); // End of headers
    // Send the JSON body (message)
    client.println(message);
    // Wait for the server response
    unsigned long timeout = millis();
    while (client.available() == 0) {
      if (millis() - timeout > 5000) {
        Serial.println("Connection timed out");
        client.stop();
        return false; // Return false on timeout
      }
    }
    // Read the response from Pushbullet API
    String response = "";
    while (client.available()) {
        response += char(client.read());
    }
    // Check the response for success (simplified)
    if (response.indexOf("200 OK") > 0) { //Basic check.
      client.stop();
      return true;
    } else {
      Serial.println("Pushbullet API error:");
      Serial.println(response); // Print the response for debugging
      client.stop();
      return false;
    }
  }
  else {
    Serial.println("Connection failed");
    return false; // Return false on connection failure
  }
  return false; //added default return
}
```

EXP-5

```
#define BLYNK_TEMPLATE_ID "TMPL34XZr63KO"
#define BLYNK_TEMPLATE_NAME "Led Blink"
#define BLYNK_AUTH_TOKEN "98y7rCepl0Khfxfh3bNQ552qic7H7X0e" // Replace
with your token
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
char auth[] = "98y7rCepl0Khfxfh3bNQ552qic7H7X0e"; // Your Blynk Auth Token
char ssid[] = "Kshitij's";
char pass[] = "kshitij20";  // Your WiFi Password
#define LED_PIN D4 // Built-in LED (GPIO2)
// Blynk Virtual Pin Handler
BLYNK_WRITE(V0) {
  int value = param.asInt(); // Get value from Blynk button (0 or 1)
  if (value == 1) {
    Serial.println("Kshitij Nangare 47 | LED OFF");
    digitalWrite(LED_PIN, LOW); // LED ON (Inverse logic)
  } else {
    Serial.println("Kshitij Nangare 47 | LED ON");
    digitalWrite(LED_PIN, HIGH); // LED OFF
  }
}
void setup() {
  Serial.begin(9600); // Initialize Serial communication
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, HIGH); // Initially OFF
  Blynk.begin(auth, ssid, pass); // Connect to Blynk
}
void loop() {
  Blynk.run(); // Keep Blynk connected
}
```

EXP-6

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <Hash.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
// Replace with your network credentials
const char* ssid = "Mahek";
const char* password = "12345678";

#define DHTPIN 4     // Digital pin connected to the DHT sensor
// Uncomment the type of sensor in use:
#define DHTTYPE    DHT11     // DHT 11
//#define DHTTYPE    DHT22     // DHT 22 (AM2302)
//#define DHTTYPE    DHT21     // DHT 21 (AM2301)
DHT dht(DHTPIN, DHTTYPE);

// current temperature & humidity, updated in loop()
float t = 0.0;
float h = 0.0;

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0;    // will store last time DHT was
updated
// Updates DHT readings every 10 seconds
const long interval = 10000;

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
<h2>ESP8266 DHT Server Roll no. 31 | 40 | 47</h2>
```

```
<p>
Temperature: <span id="temperature">%TEMPERATURE%</span> &deg;C
</p>
<p>
Humidity: <span id="humidity">%HUMIDITY%</span> %
</p>
<script>
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("temperature").innerHTML =
this.responseText;
    }
  };
  xhttp.open("GET", "/temperature", true);
  xhttp.send();
}, 10000 ) ;
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("humidity").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/humidity", true);
  xhttp.send();
}, 10000 ) ;
</script>
</body>
</html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHp
vLbHG9Sr" crossorigin="anonymous">
    <style>
      html {
      font-family: Arial;
      display: inline-block;
      margin: 0px auto;
      text-align: center;
      }
      h2 { font-size: 3.0rem; }
      p { font-size: 3.0rem; }
      .units { font-size: 1.2rem; }
      .dht-labels{
        font-size: 1.5rem;
        vertical-align:middle;
        padding-bottom: 15px;
      }
    </style>
  </head>
  <body>
    <h2>ESP8266 DHT Server Roll no. 31 | 40 | 47</h2>
    <p>
      <i class="fas fa-thermometer-half" style="color:#059e8a;"></i>
      <span class="dht-labels">Temperature</span>
      <span id="temperature">%TEMPERATURE%</span>
      <sup class="units">&deg;C</sup>
    </p>
    <p>
      <i class="fas fa-tint" style="color:#00add6;"></i>
      <span class="dht-labels">Humidity</span>
      <span id="humidity">%HUMIDITY%</span>
      <sup class="units">%</sup>
    </p>
  </body>
  <script>
  setInterval(function ( ) {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("temperature").innerHTML =
this.responseText;
      }
    };
    xhttp.open("GET", "/temperature", true);
    xhttp.send();
  }, 10000 ) ;
  setInterval(function ( ) {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("humidity").innerHTML = this.responseText;
      }
    };
    xhttp.open("GET", "/humidity", true);
    xhttp.send();
  }, 10000 ) ;
  </script>
  </html>
)rawliteral";

// Replaces placeholder with DHT values
String processor(const String& var){
  //Serial.println(var);
  if(var == "TEMPERATURE"){
    return String(t);
  }
  else if(var == "HUMIDITY"){
    return String(h);
  }
  return String();
}

void setup(){
  // Serial port for debugging purposes
  Serial.begin(9600);
  dht.begin();
  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  Serial.println("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println(".");
  }
  // Print ESP8266 Local IP Address
  Serial.println(WiFi.localIP());
  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
  });
  server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(t).c_str());
  });
  server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(h).c_str());
  });
  // Start server
  server.begin();
}

void loop(){
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    // save the last time you updated the DHT values
    previousMillis = currentMillis;
    // Read temperature as Celsius (the default)
    float newT = dht.readTemperature();
    // Read temperature as Fahrenheit (isFahrenheit = true)
    //float newT = dht.readTemperature(true);
    // if temperature read failed, don't change t value
    if (isnan(newT)) {
      Serial.println("Failed to read from DHT sensor!");
    }
    else {
      t = newT;
      Serial.println(t);
    }
    // Read Humidity
    float newH = dht.readHumidity();
    // if humidity read failed, don't change h value
    if (isnan(newH)) {
      Serial.println("Failed to read from DHT sensor!");
    }
    else {
      h = newH;
      Serial.println(h);
      Serial.println("31 | 40 | 47");
    }
  }
}
```

EXP-7

```
#include <ESP8266WiFi.h>
#include <ESPAsyncWebServer.h>
#include <ESPAsyncTCP.h>

// Set to true to define Relay as Normally Open (NO)
#define RELAY_NO true

// Set number of relays (4 for a 4-relay module)
#define NUM_RELAYS 4

// Assign each GPIO to a relay (D1, D2, D5, D6 on ESP8266)
int relayGPIOs[NUM_RELAYS] = {5, 4, 14, 12}; // GPIO5 (D1), GPIO4 (D2),
GPIO14 (D5), GPIO12 (D6)

// Replace with your network credentials
const char* ssid = "Kshitij's";
const char* password = "kshitij20";

const char* PARAM_INPUT_1 = "relay";
const char* PARAM_INPUT_2 = "state";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
    html {font-family: Arial; display: inline-block; text-align: center;}
    h2 {font-size: 3.0rem;}
    p {font-size: 3.0rem;}
    body {max-width: 600px; margin:0px auto; padding-bottom: 25px;}
    .switch {position: relative; display: inline-block; width: 120px;
height: 68px}
    .switch input {display: none}
    .slider {position: absolute; top: 0; left: 0; right: 0; bottom: 0;
background-color: #ccc; border-radius: 34px}
```

```
    .slider:before {position: absolute; content: ""; height: 52px; width:
52px; left: 8px; bottom: 8px; background-color: #fff; -webkit-transition:
.4s; transition: .4s; border-radius: 68px}
    input:checked+.slider {background-color: #2196F3}
    input:checked+.slider:before {-webkit-transform: translateX(52px);
-ms-transform: translateX(52px); transform: translateX(52px)}
  </style>
</head>
<body>
  <h2>Sohan 68 | Shifa 61 | Om 59 | Sameeksha 57 | Amruta 56 | Arjun 54 |
Mohish 49 | Roshan 72 | Anish 66</h2>
  <h3>ESP Web Server</h3>
  %BUTTONPLACEHOLDER%
<script>function toggleCheckbox(element) {
  var xhr = new XMLHttpRequest();
  if(element.checked){ xhr.open("GET",
"/update?relay="+element.id+"&state=1", true); }
  else { xhr.open("GET", "/update?relay="+element.id+"&state=0", true); }
  xhr.send();
}</script>
</body>
</html>
)rawliteral";

// Replaces placeholder with button section in your web page
String processor(const String& var) {
  if (var == "BUTTONPLACEHOLDER") {
    String buttons = "";
    for (int i = 1; i <= NUM_RELAYS; i++) {
      String relayStateValue = relayState(i);
      buttons += "<h4>Relay #" + String(i) + " - GPIO " + relayGPIOs[i -
1] + "</h4><label class=\"switch\"><input type=\"checkbox\"
onchange=\"toggleCheckbox(this)\" id=\"" + String(i) + "\" " +
relayStateValue + "><span class=\"slider\"></span></label>";
    }
    return buttons;
  }
  return String();
}
```

```
String relayState(int numRelay) {
  if (RELAY_NO) {
    if (digitalRead(relayGPIOs[numRelay - 1])) {
      return "";
    } else {
      return "checked";
    }
  } else {
    if (digitalRead(relayGPIOs[numRelay - 1])) {
      return "checked";
    } else {
      return "";
    }
  }
}

void setup() {
  Serial.begin(9600);

  // Initialize relay pins
  for (int i = 0; i < NUM_RELAYS; i++) {
    pinMode(relayGPIOs[i], OUTPUT);
    if (RELAY_NO) {
      digitalWrite(relayGPIOs[i], HIGH); // Relay OFF for NO
    } else {
      digitalWrite(relayGPIOs[i], LOW);  // Relay OFF for NC
    }
  }

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
    Serial.println("Soham 68 | Shifa 61 | Om 59 | Sameeksha 57 | Amruta 56
| Arjun 54 | Mohish 49 | Roshan 72 | Anish 66");
  }

  Serial.println("Connected to WiFi");
  Serial.println(WiFi.localIP());
  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
    String output = index_html;
    output.replace("%BUTTONPLACEHOLDER%", processor("BUTTONPLACEHOLDER"));
    request->send(200, "text/html", output);
  });

  // Route to update relay state
  server.on("/update", HTTP_GET, [](AsyncWebServerRequest *request) {
    String relayNum;
    String state;

    if (request->hasParam(PARAM_INPUT_1) &&
request->hasParam(PARAM_INPUT_2)) {
      relayNum = request->getParam(PARAM_INPUT_1)->value();
      state = request->getParam(PARAM_INPUT_2)->value();
      int relayIndex = relayNum.toInt() - 1;

      if (relayIndex >= 0 && relayIndex < NUM_RELAYS) {
        if (RELAY_NO) {
          digitalWrite(relayGPIOs[relayIndex], !state.toInt()); // Invert
for NO
        } else {
          digitalWrite(relayGPIOs[relayIndex], state.toInt());  // Direct
for NC
        }
        Serial.println("Relay " + relayNum + " set to state " + state);
      }
    } else {
      Serial.println("Invalid request parameters");
    }
    request->send(200, "text/plain", "OK");
  });

  // Start server
  server.begin();
}

void loop() {
  // Nothing needed in loop for Async server
}
```

EXP-8

Arduino Code

```
#include <PubSubClient.h>
#include <ESP8266WiFi.h>
// WiFi Credentials
const char* ssid = "iPhone";
const char* password = "Ambernath";
// MQTT Broker Details
const char* mqtt_server = "mqtt.eclipseprojects.io";
const int mqtt_port = 1883;
const char* mqtt_topic_sub = "S/topic"; // Subscribe to this topic
const char* mqtt_topic_pub = "S/topic"; // Publish to this topic
WiFiClient espClient;
PubSubClient client(espClient);
void setup_wifi() {
  delay(10);
  Serial.println("\nConnecting to WiFi...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi Connected!");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
}
void reconnect() {
  while (!client.connected()) {
    Serial.println("Attempting MQTT connection...");
    if (client.connect("NodeMCUClient")) {
      Serial.println("Connected to MQTT Broker!");
      client.subscribe(mqtt_topic_sub);
      client.publish(mqtt_topic_pub, "Hello from NodeMCU");
    }
    else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" Retrying in 5 seconds...");
      delay(5000);
    }
  }
}
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message received on topic: ");
  Serial.print("Om 59 Arjun 54 Soham 68 Shifa 61 Amruta 56 Sameeksha 57 
");
  Serial.println(topic);
  Serial.print("Payload: ");
  for (int i = 0; i < length; i++) {
   Serial.print((char)payload[i]);
  }
  Serial.println();
}
void setup() {
  Serial.begin(9600);
  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
}
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
}
```

Python Code

```python
import paho.mqtt.client as mqtt
# Create a client instance
c = mqtt.Client()
# Define the on_connect callback
def onc(c, userdata, flag, rc):
    print("Connected with result code:", rc)
    c.subscribe('pi/data')  # Subscribe to topic upon connection

# Define the on_message callback
def onm(c, userdata, msg):
    print("Received message:", msg.payload.decode())

# Assign callbacks
c.on_connect = onc
c.on_message = onm

# Connect to the public MQTT broker
c.connect("mqtt.eclipseprojects.io", 1883)

# Start the loop
c.loop_forever()
```