PRACTICE 9: MODERN IoT

1. Connect a Streetlight Device to ThingsBoard

Register a streetlight device and send telemetry (e.g., brightness, status).

Steps:

a) Create a new Device on ThingsBoard (e.g., “Streetlight-001”)

b) Generate the access token

c) On your device (e.g., ESP32):

● Read LDR sensor data for ambient light

● If brightness < threshold, consider it “night”

d) Send telemetry to ThingsBoard via

i) MQTT

ii) HTTP

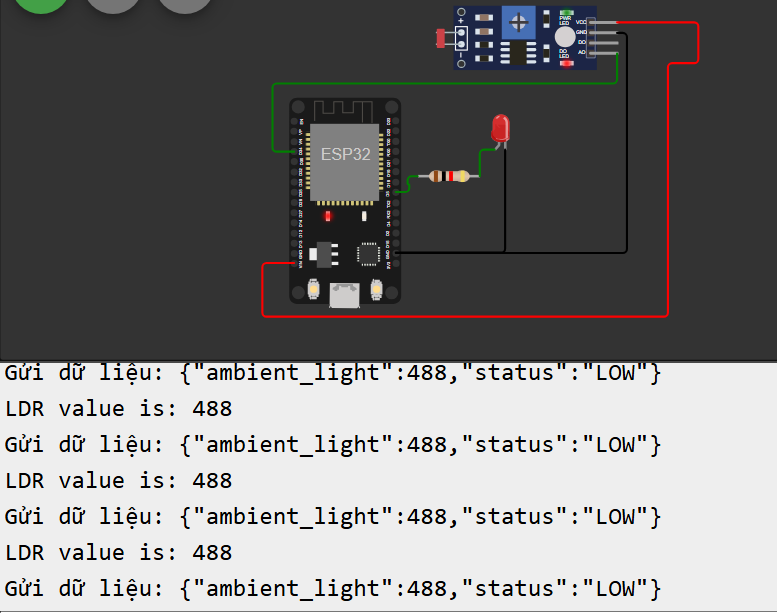
{

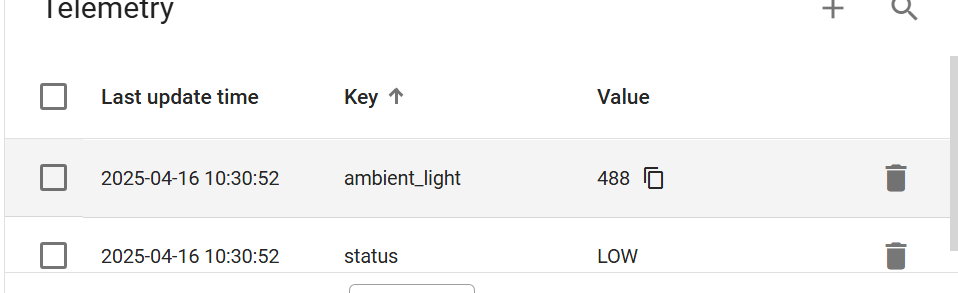
"ambient\_light": 150,

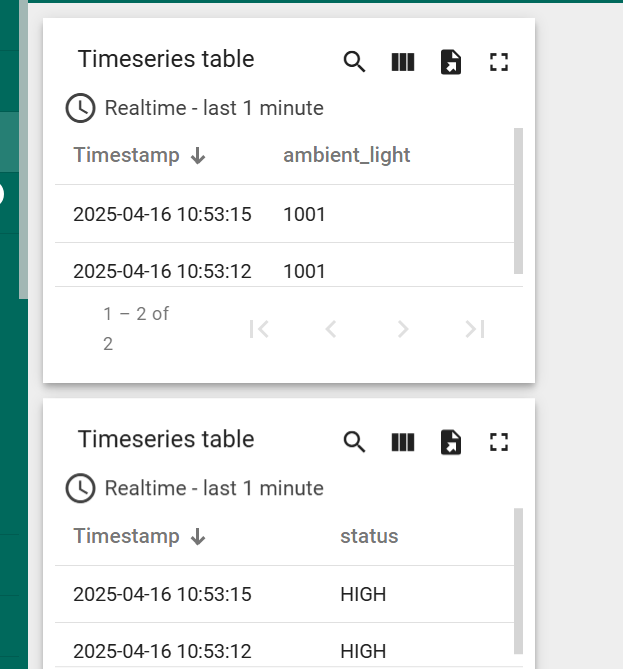
"status": "ON"

}

e) Visualize data on a dashboard







Code:

#include <WiFi.h>

#include <PubSubClient.h>

#define WIFI\_SSID "Wokwi-GUEST"

#define WIFI\_PASSWORD ""

#define TOKEN "muVXtktIfnk1WchCUXgD"

#define THINGSBOARD\_SERVER "thingsboard.cloud"

#define THRESHOLD 800

const int LDR = 34;

const int LED = 5;

WiFiClient espClient;

PubSubClient client(espClient);

void setup\_wifi() {

Serial.print("Kết nối WiFi...");

WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);

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

delay(500);

Serial.print(".");

}

Serial.println(" Đã kết nối WiFi!");

}

void reconnect() {

while (!client.connected()) {

Serial.print("Đang kết nối ThingsBoard...");

if (client.connect("ESP32Client", TOKEN, NULL)) {

Serial.println(" Thành công!");

} else {

Serial.print(" Thất bại, mã lỗi = ");

Serial.print(client.state());

Serial.println(". Thử lại sau 5 giây.");

delay(5000);

}

}

}

void setup() {

Serial.begin(115200);

pinMode(LED, OUTPUT);

setup\_wifi();

client.setServer(THINGSBOARD\_SERVER, 1883);

}

void loop() {

if (!client.connected()) {

reconnect();

}

client.loop();

int input\_val = analogRead(LDR);

Serial.print("LDR value is: ");

Serial.println(input\_val);

String status;

if (input\_val > THRESHOLD) {

status = "HIGH";

digitalWrite(LED, HIGH);

} else {

status = "LOW";

digitalWrite(LED, LOW);

}

// Tạo payload JSON

String payload = "{";

payload += "\"ambient\_light\":" + String(input\_val) + ",";

payload += "\"status\":\"" + status + "\"";

payload += "}";

Serial.println("Gửi dữ liệu: " + payload);

client.publish("v1/devices/me/telemetry", payload.c\_str());

delay(1000); // Gửi mỗi 5 giây

}

2. Remote Control of a Streetlight

Turn the light ON/OFF remotely using a ThingsBoard dashboard switch.

Steps:

a) Create a switch control widget on your dashboard

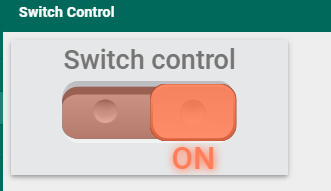
b) On your device, subscribe to the RPC topic

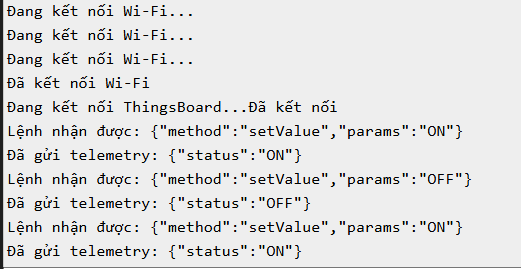
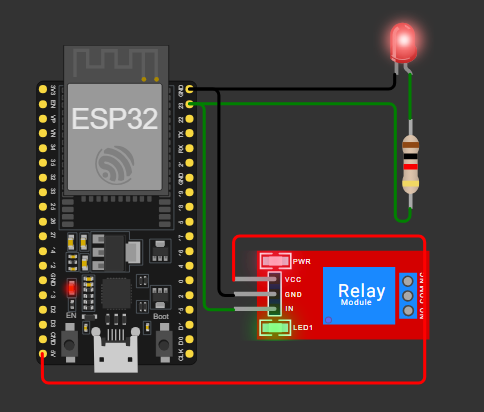
c) When a command is received (e.g.,

{"method":"setStatus","params":"ON"}), control the streetlight

relay/LED. If params is OFF then turn off the light

d) Report the new status back





Code:

#include <WiFi.h>

#include <PubSubClient.h>

// Cấu hình Wi-Fi

const char\* ssid = "Wokwi-GUEST";

const char\* password = "";

// Thông tin ThingsBoard

const char\* thingsboardServer = "thingsboard.cloud"; // Địa chỉ server ThingsBoard

const int thingsboardPort = 1883; // Port MQTT (mặc định là 1883)

const char\* accessToken = "kYUfEw6B3rNghi92RkJa"; // Access Token của thiết bị trên ThingsBoard

WiFiClient espClient;

PubSubClient client(espClient);

// Chân GPIO kết nối Relay/LED

const int relayPin = 23;

// Hàm callback nhận các lệnh RPC từ ThingsBoard

void callback(char\* topic, byte\* payload, unsigned int length) {

// Chuyển payload thành chuỗi

String command = "";

for (unsigned int i = 0; i < length; i++) {

command += (char)payload[i];

}

Serial.println("Lệnh nhận được: " + command);

if (command.indexOf("\"method\":\"setValue\"") >= 0) {

if (command.indexOf("\"params\":\"ON\"") >= 0) {

digitalWrite(relayPin, HIGH);

} else if (command.indexOf("\"params\":\"OFF\"") >= 0) {

digitalWrite(relayPin, LOW);

}

// Gửi lại trạng thái về ThingsBoard

String status = digitalRead(relayPin) == HIGH ? "ON" : "OFF";

String payload = "{\"status\":\"" + status + "\"}";

client.publish("v1/devices/me/telemetry", payload.c\_str());

Serial.println("Đã gửi telemetry: " + payload);

}

}

void setup() {

// Khởi tạo Serial

Serial.begin(115200);

// Kết nối Wi-Fi

WiFi.begin(ssid, password);

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

delay(1000);

Serial.println("Đang kết nối Wi-Fi...");

}

Serial.println("Đã kết nối Wi-Fi");

// Kết nối tới ThingsBoard

client.setServer(thingsboardServer, thingsboardPort);

client.setCallback(callback);

// Cài đặt chân GPIO cho Relay/LED

pinMode(relayPin, OUTPUT);

digitalWrite(relayPin, LOW); // Đảm bảo Relay tắt khi khởi động

}

void reconnect() {

// Kết nối lại nếu mất kết nối với ThingsBoard

while (!client.connected()) {

Serial.print("Đang kết nối ThingsBoard...");

if (client.connect("ESP32Client", accessToken, NULL)) {

Serial.println("Đã kết nối");

client.subscribe("v1/devices/me/rpc/request/+"); // Subscribe để nhận lệnh RPC từ ThingsBoard

} else {

Serial.print("Lỗi kết nối, mã lỗi: ");

Serial.println(client.state());

delay(5000);

}

}

}

void loop() {

// Nếu mất kết nối, reconnect lại

if (!client.connected()) {

reconnect();

}

client.loop();

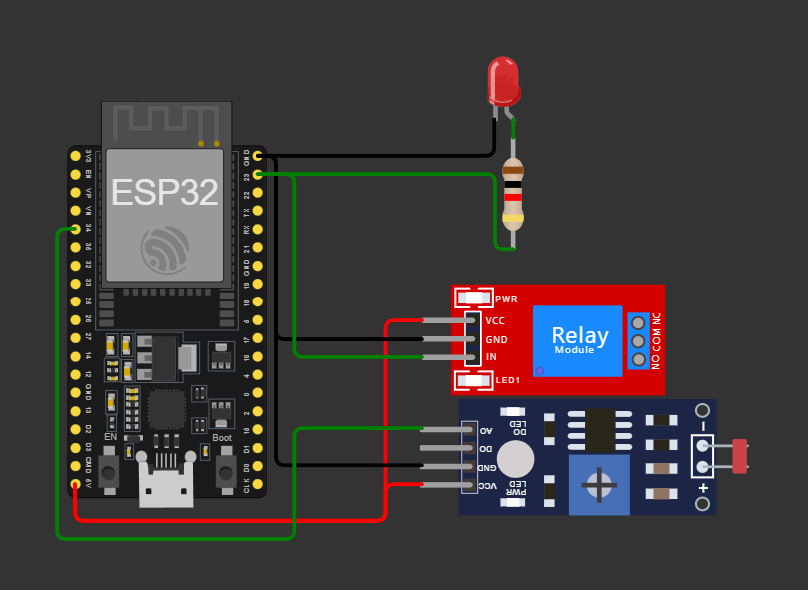
}

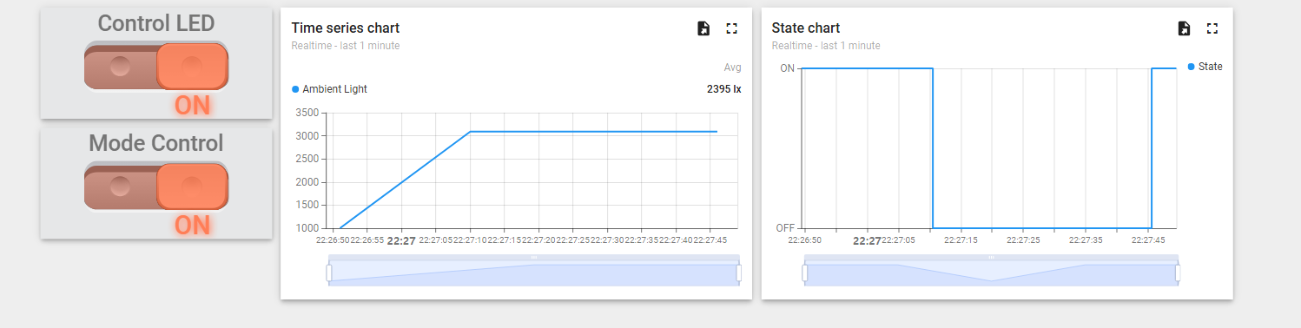
3. Smart Streetlight with Auto and Manual Modes

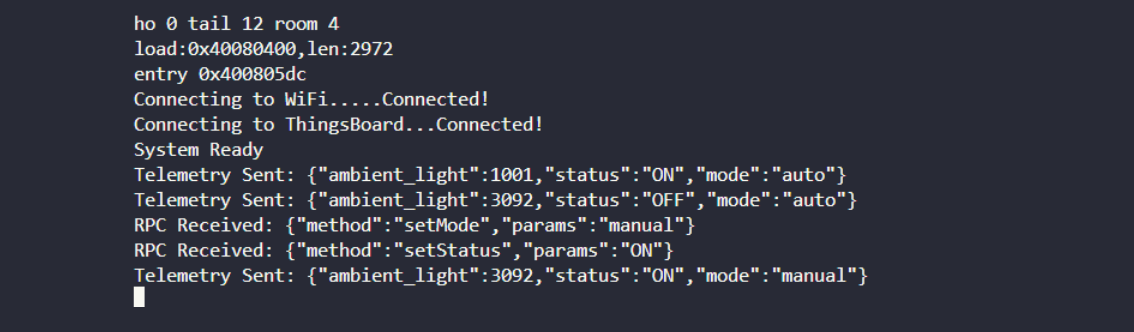
Combine sensor automation with manual override to control in cases:

● If in auto mode, control streetlight based on light sensor

● If in manual mode, control via dashboard switch







Code:  
#include <WiFi.h>

#include <PubSubClient.h>

#define LDR\_PIN 34

#define RELAY\_PIN 23

#define LED\_PIN 23

const char\* ssid = "Wokwi-GUEST";

const char\* password = "";

// ThingsBoard

const char\* mqttServer = "thingsboard.cloud";

const int mqttPort = 1883;

const char\* token = "kYUfEw6B3rNghi92RkJa";

WiFiClient espClient;

PubSubClient client(espClient);

unsigned long lastSend = 0;

const int interval = 5000; // gửi mỗi 5 giây

String currentMode = "auto"; // "auto" hoặc "manual"

String lightStatus = "OFF"; // trạng thái đèn hiện tại

void connectToWiFi() {

Serial.print("Connecting to WiFi...");

WiFi.begin(ssid, password);

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

delay(500);

Serial.print(".");

}

Serial.println("Connected!");

}

void connectToMQTT() {

while (!client.connected()) {

Serial.print("Connecting to ThingsBoard...");

if (client.connect("ESP32", token, NULL)) {

Serial.println("Connected!");

client.subscribe("v1/devices/me/rpc/request/+");

} else {

Serial.print("Failed. Error: ");

Serial.print(client.state());

delay(1000);

}

}

}

void callback(char\* topic, byte\* payload, unsigned int length) {

String request = "";

for (int i = 0; i < length; i++) {

request += (char)payload[i];

}

Serial.print("RPC Received: ");

Serial.println(request);

if (request.indexOf("setStatus") != -1) {

if (request.indexOf("ON") != -1) {

lightStatus = "ON";

} else if (request.indexOf("OFF") != -1) {

lightStatus = "OFF";

}

}

if (request.indexOf("setMode") != -1) {

if (request.indexOf("auto") != -1) {

currentMode = "auto";

} else if (request.indexOf("manual") != -1) {

currentMode = "manual";

}

}

// Gửi phản hồi trạng thái hiện tại

String response = "{\"mode\":\"" + currentMode + "\",\"status\":\"" + lightStatus + "\"}";

client.publish("v1/devices/me/attributes", response.c\_str());

}

void sendTelemetry(int ldr, String status) {

String payload = "{\"ambient\_light\":" + String(ldr) + ",\"status\":\"" + status + "\",\"mode\":\"" + currentMode + "\"}";

client.publish("v1/devices/me/telemetry", payload.c\_str());

Serial.println("Telemetry Sent: " + payload);

}

void setup() {

Serial.begin(115200);

pinMode(LDR\_PIN, INPUT);

pinMode(RELAY\_PIN, OUTPUT);

connectToWiFi();

client.setServer(mqttServer, mqttPort);

client.setCallback(callback);

connectToMQTT();

Serial.println("System Ready");

}

void loop() {

if (!client.connected()) {

connectToMQTT();

}

client.loop();

if (millis() - lastSend > interval) {

lastSend = millis();

int ldrValue = analogRead(LDR\_PIN);

String status;

if (currentMode == "auto") {

if (ldrValue < 2000) {

digitalWrite(RELAY\_PIN, HIGH);

status = "ON";

} else {

digitalWrite(RELAY\_PIN, LOW);

status = "OFF";

}

} else {

status = lightStatus;

digitalWrite(RELAY\_PIN, (status == "ON") ? HIGH : LOW);

}

sendTelemetry(ldrValue, status);

}

}

4. Streetlight Monitoring Dashboard

Building a real-time visualization for smart streetlights using ThingsBoard

dashboards. Implement on two devices:

Mordern Iot

Nguyễn Ngọc Lễ

1

a) ESP32

b) Raspberry Pi

Create an interactive dashboard on ThingsBoard that displays:

● Ambient brightness (from sensor)

● Light ON/OFF status

● Device uptime

● Mode (Auto/Manual)

● Event counters (e.g., how many times light turned ON)

● Device type

Telemetry (data sent periodically):

● ambient\_light (Integer): Light level from LDR (0-1023)

● status (String): "ON" or "OFF" based on whether the light is currently lit

● uptime (Integer): Seconds or milliseconds since the device started

● mode (String): "AUTO" or "MANUAL"

● on\_count (Integer): Number of times the light has been turned on

Attributes (optional static data):

● location: Device physical location

● model: Device model or ID

Ensure your device is sending telemetry like this:

{

"ambient\_light": 120,

"status": "ON",

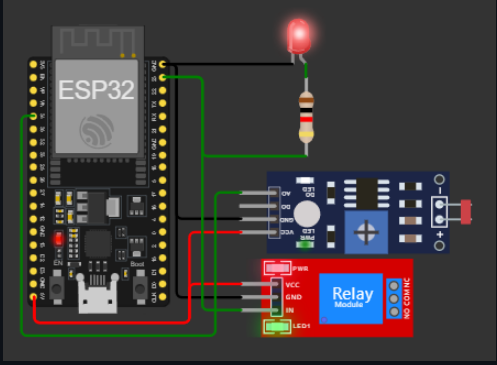
"uptime": 45600,

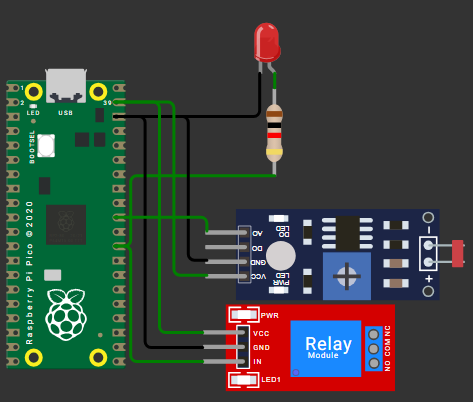
"mode": "AUTO",

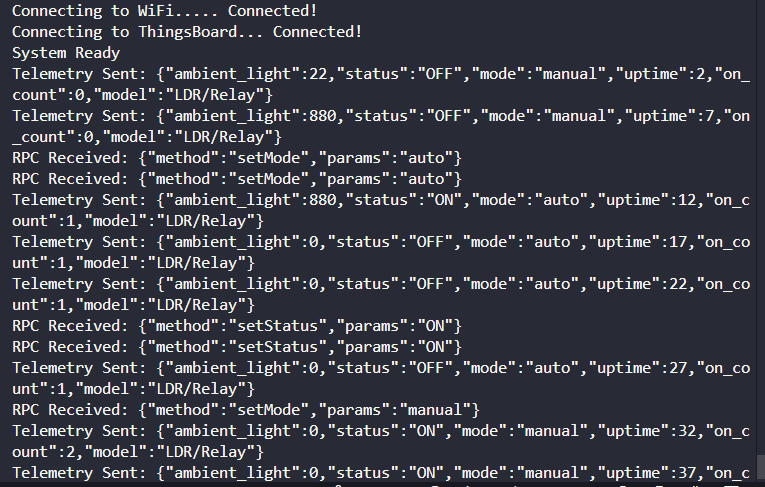
"on\_count": 5,

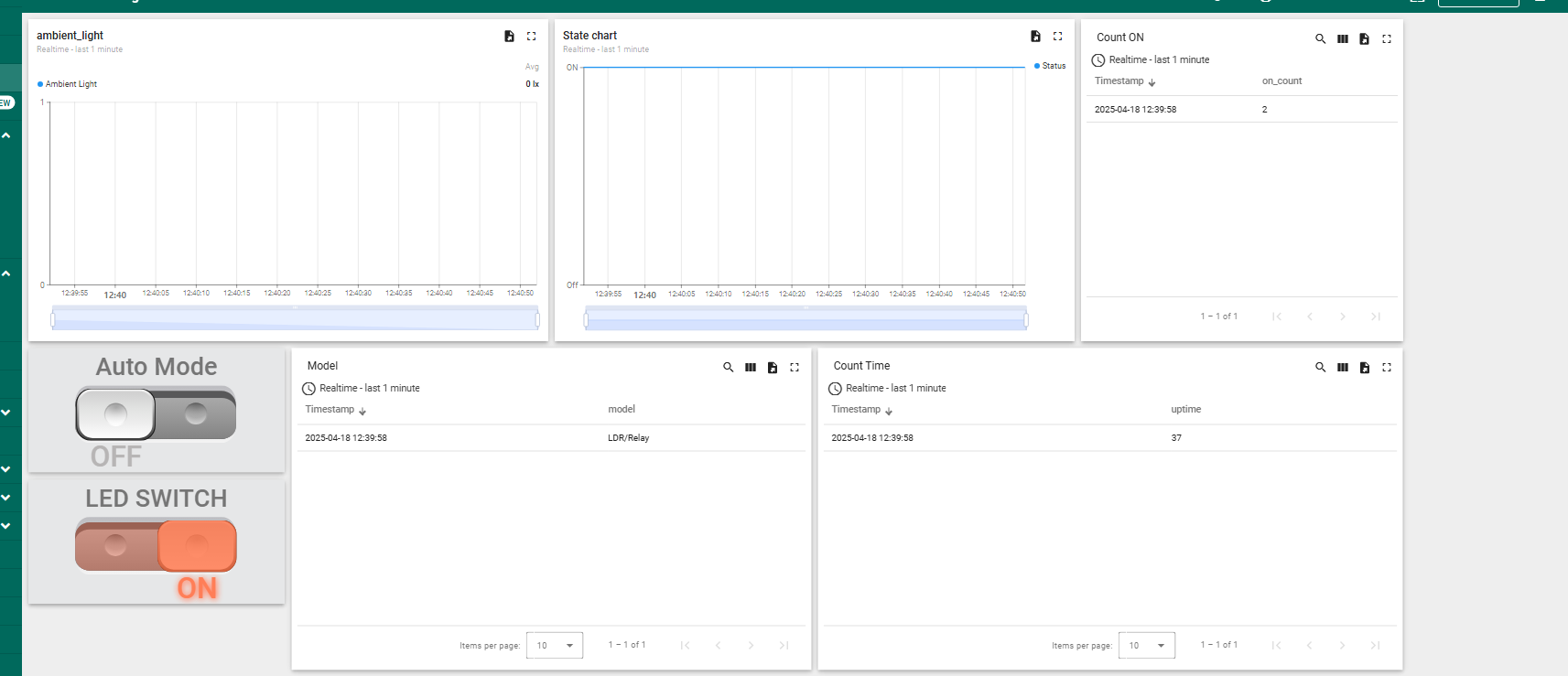
“model”: “DHT20”

}









code:

**ESP32**

#include <WiFi.h>

#include <PubSubClient.h>

#define LDR\_PIN 34

#define RELAY\_PIN 23

const char\* ssid = "Wokwi-GUEST";

const char\* password = "";

const char\* mqttServer = "thingsboard.cloud";

const int mqttPort = 1883;

const char\* token = "jSNXYT0BdFgS69jXXfJr";

WiFiClient espClient;

PubSubClient client(espClient);

unsigned long lastSend = 0;

unsigned long startTime = 0;

const int interval = 5000;

String currentMode = "manual";

String lightStatus = "OFF";

int on\_count = 0;

bool prevRelay = false;

void connectToWiFi() {

Serial.print("Connecting to WiFi...");

WiFi.begin(ssid, password);

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

delay(500);

Serial.print(".");

}

Serial.println(" Connected!");

}

void connectToMQTT() {

while (!client.connected()) {

Serial.print("Connecting to ThingsBoard...");

if (client.connect("ESP32", token, NULL)) {

Serial.println(" Connected!");

client.subscribe("v1/devices/me/rpc/request/+");

// Gửi thông tin ban đầu

String initAttr = "{\"model\":\"ESP32-Relay-LDR\",\"firmware\_version\":\"1.0\"}";

client.publish("v1/devices/me/attributes", initAttr.c\_str());

} else {

Serial.print(" Failed. Error: ");

Serial.print(client.state());

delay(1000);

}

}

}

void callback(char\* topic, byte\* payload, unsigned int length) {

String request = "";

for (int i = 0; i < length; i++) {

request += (char)payload[i];

}

Serial.print("RPC Received: ");

Serial.println(request);

if (request.indexOf("setStatus") != -1) {

if (request.indexOf("ON") != -1) {

lightStatus = "ON";

} else if (request.indexOf("OFF") != -1) {

lightStatus = "OFF";

}

}

if (request.indexOf("setMode") != -1) {

if (request.indexOf("auto") != -1) {

currentMode = "auto";

} else if (request.indexOf("manual") != -1) {

currentMode = "manual";

}

}

// Phản hồi trạng thái

// Gửi riêng từng attribute

String modePayload = "{\"mode\":\"" + currentMode + "\"}";

client.publish("v1/devices/me/attributes", modePayload.c\_str());

String statusPayload = "{\"status\":\"" + lightStatus + "\"}";

client.publish("v1/devices/me/attributes", statusPayload.c\_str());

}

void sendTelemetry(int ldr, String status) {

unsigned long uptime = (millis() - startTime) / 1000;

String payload = "{";

payload += "\"ambient\_light\":" + String(ldr) + ",";

payload += "\"status\":\"" + status + "\",";

payload += "\"mode\":\"" + currentMode + "\",";

payload += "\"uptime\":" + String(uptime) + ",";

payload += "\"on\_count\":" + String(on\_count) + ",";

payload += "\"model\":\"LDR/Relay\"";

payload += "}";

client.publish("v1/devices/me/telemetry", payload.c\_str());

Serial.println("Telemetry Sent: " + payload);

}

void setup() {

Serial.begin(115200);

pinMode(LDR\_PIN, INPUT);

pinMode(RELAY\_PIN, OUTPUT);

digitalWrite(RELAY\_PIN, LOW);

connectToWiFi();

client.setServer(mqttServer, mqttPort);

client.setCallback(callback);

connectToMQTT();

startTime = millis();

Serial.println("System Ready");

}

void loop() {

if (!client.connected()) connectToMQTT();

client.loop();

if (millis() - lastSend > interval) {

lastSend = millis();

int rawLdrValue = analogRead(LDR\_PIN);

rawLdrValue = constrain(rawLdrValue, 0, 1023);

int ldrValue = map(rawLdrValue, 0, 1023, 1023, 0);

String status;

if (currentMode == "auto") {

if (ldrValue > 500) {

digitalWrite(RELAY\_PIN, HIGH);

status = "ON";

} else {

digitalWrite(RELAY\_PIN, LOW);

status = "OFF";

}

} else {

status = lightStatus;

digitalWrite(RELAY\_PIN, (status == "ON") ? HIGH : LOW);

}

// Đếm số lần bật đèn

bool currentRelay = (status == "ON");

if (currentRelay && !prevRelay) on\_count++;

prevRelay = currentRelay;

sendTelemetry(ldrValue, status);

}

}

**Raspberry Pi**

import network

import time

import machine

import ubinascii

import ujson

from umqtt.simple import MQTTClient

# GPIO cấu hình

LDR\_PIN = 26 # GP26 (ADC0)

RELAY\_PIN = 22 # GP22

ldr = machine.ADC(LDR\_PIN)

relay = machine.Pin(RELAY\_PIN, machine.Pin.OUT)

# WiFi & MQTT config

SSID = "Wokwi-GUEST"

PASSWORD = ""

THINGSBOARD\_HOST = "thingsboard.cloud"

ACCESS\_TOKEN = "jSNXYT0BdFgS69jXXfJr"

client\_id = ubinascii.hexlify(machine.unique\_id())

mqtt\_client = MQTTClient(client\_id, THINGSBOARD\_HOST, 1883, ACCESS\_TOKEN, "")

# Biến trạng thái

current\_mode = "manual"

light\_status = "OFF"

on\_count = 0

prev\_relay = False

start\_time = time.ticks\_ms()

def connect\_wifi():

wlan = network.WLAN(network.STA\_IF)

wlan.active(True)

wlan.connect(SSID, PASSWORD)

print("Connecting to WiFi...", end="")

while not wlan.isconnected():

print(".", end="")

time.sleep(0.5)

print(" Connected!")

def mqtt\_callback(topic, msg):

global current\_mode, light\_status

print("RPC Received:", msg)

try:

request = ujson.loads(msg)

method = request.get("method")

params = request.get("params")

if method == "setMode":

current\_mode = params

elif method == "setStatus":

light\_status = params

# Gửi phản hồi attribute

mqtt\_client.publish("v1/devices/me/attributes", ujson.dumps({

"mode": current\_mode,

"status": light\_status

}))

except Exception as e:

print("Error in RPC:", e)

def send\_telemetry(ldr\_value, status):

uptime = (time.ticks\_ms() - start\_time) // 1000

payload = {

"ambient\_light": ldr\_value,

"status": status,

"mode": current\_mode,

"uptime": uptime,

"on\_count": on\_count,

"model": "PicoW-Relay-LDR"

}

mqtt\_client.publish("v1/devices/me/telemetry", ujson.dumps(payload))

print("Telemetry Sent:", payload)

# Setup

connect\_wifi()

mqtt\_client.set\_callback(mqtt\_callback)

mqtt\_client.connect()

mqtt\_client.subscribe(b"v1/devices/me/rpc/request/+")

mqtt\_client.publish("v1/devices/me/attributes", ujson.dumps({

"model": "PicoW-Relay-LDR",

"firmware\_version": "1.0"

}))

print("Connected to ThingsBoard")

# Loop

last\_send = time.ticks\_ms()

interval = 5000

while True:

mqtt\_client.check\_msg()

if time.ticks\_diff(time.ticks\_ms(), last\_send) > interval:

last\_send = time.ticks\_ms()

raw = ldr.read\_u16() >> 6 # scale to 10-bit (0-1023)

ldr\_value = 1023 - raw

if current\_mode == "auto":

if ldr\_value > 500:

relay.value(1)

status = "ON"

else:

relay.value(0)

status = "OFF"

else:

status = light\_status

relay.value(1 if status == "ON" else 0)

current\_relay = (status == "ON")

if current\_relay and not prev\_relay:

on\_count += 1

prev\_relay = current\_relay

send\_telemetry(ldr\_value, status)

time.sleep(0.1)