

Code mẫu MQTT gửi tọa độ GPS tới sever dùng 4G

Dưới đây là code mẫu dùng cho **Mạch phát triển 4G GPS ESP32-C3 TDM2421**, một số sản phẩm khác bạn cần tìm hiểu tập lệnh AT để chỉnh sửa cho phù hợp.

Sever ở đây mình dùng là Thingsboard, nếu dùng sever khác thì các bạn chỉnh **TOPIC**, **ACESSTOKEN** cho phù hợp.

Để sử dụng được MQTT cho module này, các lệnh AT cần thiết phải dùng là:

1. AT+QMTOPEN
2. AT+QMTCONN
3. AT+QMTPUBEX
4. AT+QMTDISC

Lập trình

```
#include <HardwareSerial.h>

#define simSerial Serial0 // Sử dụng Serial0 cho module SIM
#define MCU_SIM_BAUDRATE 115200
#define MCU_SIM_TX_PIN 21
#define MCU_SIM_RX_PIN 20
#define MCU_SIM_EN_PIN 2
#define BUF_SIZE 256
#define PAYLOAD_SIZE 128
#define MSG_SIZE 128

char data[BUF_SIZE];
char payload[PAYLOAD_SIZE];
char msg[MSG_SIZE];
float latDecimalGlobal = 0.0;
float lonDecimalGlobal = 0.0;
```



```

bool is_publishing = false;

// Hàm chuyển đổi tọa độ sang dạng thập phân
float convertToDecimal(char *coordinate, char direction) {
    float degrees, minutes, decimal;
    char degStr[4], minStr[10];

    if (strlen(coordinate) > 9) { // Kinh độ (DDDMM.MMMM)
        strncpy(degStr, coordinate, 3);
        degStr[3] = '\0';
        strcpy(minStr, coordinate + 3);
    } else { // Vĩ độ (DDMM.MMMM)
        strncpy(degStr, coordinate, 2);
        degStr[2] = '\0';
        strcpy(minStr, coordinate + 2);
    }

    degrees = atof(degStr);
    minutes = atof(minStr);
    decimal = degrees + (minutes / 60.0);

    if (direction == 'S' || direction == 'W') {
        decimal = -decimal;
    }

    return decimal;
}

// Hàm phân tích chuỗi GPS
void parseGPSData(char *gpsData) {
    char *token;
    char latitude[12], longitude[12];
    char latDir, lonDir;
    float latDecimal, lonDecimal;

    // Kiểm tra lỗi GPS hoặc dữ liệu không hợp lệ
    if (strstr(gpsData, "+CME ERROR:") != NULL) {
        Serial.print("GPS error: ");
        Serial.println(gpsData);
    }
}

```



```

        // Tạo payload mặc định
        snprintf(payload, sizeof(payload), "{\"latitude\": %.6f, \"longitude\": %.6f}", latDecimalGlobal,
lonDecimalGlobal);

        Serial.print("Payload (default): ");
        Serial.println(payload);
        return;
    }

    if (strstr(gpsData, "+QGPSLOC:") == NULL) {
        Serial.println("No valid GPS data found");
        // Tạo payload mặc định
        snprintf(payload, sizeof(payload), "{\"latitude\": %.6f, \"longitude\": %.6f}", latDecimalGlobal,
lonDecimalGlobal);

        Serial.print("Payload (default): ");
        Serial.println(payload);
        return;
    }

    // Bỏ qua phần đầu "+QGPSLOC: "
    token = strtok(gpsData, " ");
    token = strtok(NULL, ",");
    // Bỏ qua thời gian
    token = strtok(NULL, ",");
    // Lấy vĩ độ
    strcpy(latitude, token);
    latDir = latitude[strlen(latitude) - 1];
    latitude[strlen(latitude) - 1] = '\0';
    // Lấy kinh độ
    token = strtok(NULL, ",");
    strcpy(longitude, token);
    lonDir = longitude[strlen(longitude) - 1];
    longitude[strlen(longitude) - 1] = '\0';
    // Chuyển đổi sang thập phân
    latDecimal = convertToDecimal(latitude, latDir);
    lonDecimal = convertToDecimal(longitude, lonDir);

    // Cập nhật biến toàn cục
    latDecimalGlobal = latDecimal;
    lonDecimalGlobal = lonDecimal;

```



```

// In kết quả
Serial.print("Vi do: ");
Serial.println(latDecimal, 6);
Serial.print("Kinh do: ");
Serial.println(lonDecimal, 6);

// Tạo payload JSON
snprintf(payload, sizeof(payload), "{\"latitude\": %.6f, \"longitude\": %.6f}", latDecimalGlobal,
lonDecimalGlobal);
Serial.print("Payload: ");
Serial.println(payload);
}

void sim_at_wait() {
    delay(1000); // Tăng delay để đảm bảo nhận đủ dữ liệu
    int len = 0;
    while (simSerial.available()) {
        char c = simSerial.read();
        if (len < BUF_SIZE - 1) {
            data[len++] = c;
        }
    }
    if (len > 0) {
        data[len] = '\0'; // Null-terminate chuỗi
        Serial.print("==== SIM receive: ");
        Serial.println(data);
        // Kiểm tra lỗi MQTT
        if (strstr(data, "+QMTSTAT:") != NULL) {
            Serial.print("MQTT connection error: ");
            Serial.println(data);
        }
        // Phân tích GPS nếu nhận được và không đang publish
        if (strstr(data, "+QGPSLOC:") != NULL && !is_publishing) {
            parseGPSData(data);
        } else if (strstr(data, "+CME ERROR:") != NULL) {
            parseGPSData(data); // Xử lý lỗi GPS
        }
    }
} else {

```



```
    Serial.println("No data received from SIM module");
}
}
```

```
bool sim_at_cmd(String cmd) {
    Serial.print("Sending command: ");
    Serial.println(cmd);
    simSerial.println(cmd);
    sim_at_wait();
    // Kiểm tra phản hồi OK hoặc ERROR
    if (strstr(data, "OK") != NULL) {
        return true;
    } else if (strstr(data, ">") != NULL) {
        return true; // Dấu > cho biết module chờ payload
    } else if (strstr(data, "ERROR") != NULL) {
        Serial.print("Command failed: ");
        Serial.println(cmd);
        return false;
    }
    return false;
}
```

```
bool sim_at_send(char c) {
    simSerial.write(c);
    return true;
}
```

```
void get_gps_data() {
    Serial.println("Requesting GPS data...");
    sim_at_cmd("AT+QGPS?");
    for (int i = 0; i < 5; i++) { // Thử 5 lần
        Serial.print("GPS attempt ");
        Serial.println(i + 1);
        sim_at_cmd("AT+QGPSLOC=0");
        delay(5000); // Chờ 5 giây để GPS thử định vị
        if (strstr(data, "+QGPSLOC:") != NULL) {
            Serial.println("GPS data received!");
            break;
        }
    }
}
```



```

    }
}

bool check_network() {
    sim_at_cmd("AT+CSQ");
    if (strstr(data, "+CSQ: 0,0") != NULL || strstr(data, "+CSQ: 99,99") != NULL) {
        Serial.println("No network signal!");
        return false;
    }
    Serial.println("Network signal OK");
    // Kiểm tra GPRS
    sim_at_cmd("AT+CGATT?");
    if (strstr(data, "+CGATT: 1") == NULL) {
        Serial.println("GPRS not attached, attempting to attach...");
        sim_at_cmd("AT+CGATT=1");
        delay(2000);
        return strstr(data, "OK") != NULL;
    }
    return true;
}

bool send_mqtt_data() {
    // Khởi tạo payload mặc định nếu rỗng
    if (strlen(payload) == 0) {
        snprintf(payload, sizeof(payload), "{\"latitude\": %.6f, \"longitude\": %.6f}", latDecimalGlobal,
lonDecimalGlobal);
        Serial.print("Payload initialized: ");
        Serial.println(payload);
    }
    Serial.print("Sending MQTT data with payload: ");
    Serial.println(payload);
    is_publishing = true;

    if (!check_network()) {
        Serial.println("Network signal weak, skipping MQTT send");
        is_publishing = false;
        return false;
    }
}

```



```

// Thử mở kết nối MQTT
bool mqtt_opened = false;
for (int i = 0; i < 3; i++) {
    if (sim_at_cmd("AT+QMTOPEN=0,\"demo.thingsboard.io\",1883")) { // CHỈNH SỬA CHO PHÙ HỢP VỚI
SEVER ĐANG DÙNG
        if (strstr(data, "+QMTOPEN: 0,0") != NULL) {
            Serial.println("MQTT connection opened");
            mqtt_opened = true;
            break;
        }
    }
    Serial.println("MQTT open failed, retrying...");
    delay(3000);
}
if (!mqtt_opened) {
    Serial.println("Failed to open MQTT connection");
    is_publishing = false;
    return false;
}

// Thử kết nối client
bool mqtt_connected = false;
for (int i = 0; i < 3; i++) {
    if (sim_at_cmd("AT+QMTCONN=0,\"clientExample\", \"StQb4lQec7VJK7BZsUI5\\'\") { // CHỈNH ACCESTOKEN
CHO PHÙ HỢP
        if (strstr(data, "+QMTCONN: 0,0,0") != NULL) {
            Serial.println("MQTT client connected");
            mqtt_connected = true;
            break;
        }
    }
    Serial.println("MQTT connect failed, retrying...");
    delay(3000);
}
if (!mqtt_connected) {
    Serial.println("Failed to connect MQTT client");
    is_publishing = false;
    return false;
}

```



```

// Gửi payload
bool mqtt_published = false;
for (int i = 0; i < 3; i++) {
    uint32_t dodai = strlen(payload);
    sprintf(msg, sizeof(msg), "AT+QMQTTPUBEX=0,0,0,0,\"v1/devices/me/telemetry\",%lu", dodai); // CHỈNH
TOPIC CHO PHÙ HỢP
    if (sim_at_cmd(msg)) {
        delay(1000); // Tăng delay để chờ dấu >
        if (strstr(data, ">") != NULL) {
            if (sim_at_cmd(payload)) {
                delay(1000); // Tăng delay để gửi payload
                sim_at_cmd(""); // Gửi \r\n
                delay(1000); // Chờ phản hồi
                if (strstr(data, "+QMQTTPUBEX: 0,0,0,0") != NULL) {
                    Serial.println("MQTT publish successful");
                    mqtt_published = true;
                    break;
                }
            }
        }
    }
    Serial.println("MQTT publish failed, retrying...");
    delay(3000);
}

if (!mqtt_published) {
    Serial.println("Failed to publish MQTT data");
}

// Ngắt kết nối
sim_at_cmd("AT+QMQTTPUBEX=0");
delay(2000);
is_publishing = false;
return mqtt_published;
}

void setup() {
    pinMode(MCU_SIM_EN_PIN, OUTPUT);
    digitalWrite(MCU_SIM_EN_PIN, HIGH); // Thả PWRKEY lên cao

```



```

delay(500);
Serial.begin(115200);
Serial.println("\n\n\n\n-----\nSystem started!!!!");
delay(5000);

simSerial.begin(MCU_SIM_BAUDRATE, SERIAL_8N1, MCU_SIM_RX_PIN, MCU_SIM_TX_PIN);
Serial.println("SIM Serial initialized!");
Serial.println("Checking AT command...");
sim_at_cmd("AT");
Serial.println("Getting product info...");
sim_at_cmd("ATI");
Serial.println("Checking signal quality...");
sim_at_cmd("AT+CSQ");
Serial.println("Getting IMSI...");
sim_at_cmd("AT+CIMI");
Serial.println("Activating GPS...");
sim_at_cmd("AT+QGPSCFG=\"antenna\",1"); // Bật nguồn cho anten chủ động
sim_at_cmd("AT+QGPS=1");
delay(15000); // Chờ 15 giây để GPS khởi động
// Khởi tạo payload mặc định
snprintf(payload, sizeof(payload), "{\"latitude\": %.6f, \"longitude\": %.6f}", latDecimalGlobal,
lonDecimalGlobal);
Serial.print("Initial payload: ");
Serial.println(payload);
get_gps_data();
}

void loop() {
  if (Serial.available()) {
    char c = Serial.read();
    sim_at_send(c);
  }
  if (!is_publishing) {
    get_gps_data(); // Đọc GPS trước khi gửi MQTT
  }
  send_mqtt_data();
  delay(5000);
}

```


Kết quả:

Giao diện ThingsBoard sau khi nạp code. State “Active” đã hiển thị ở mục Device có Name là “MiHi” cho thấy module đã kết nối thành công tới ThingsBoard.



