

2025

nguyen.a.tu@gmail.com

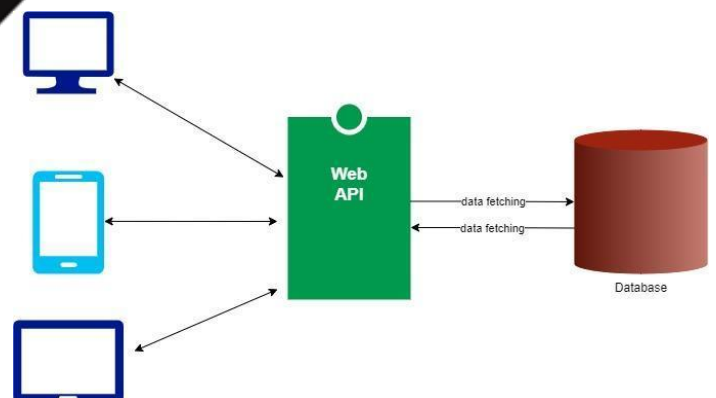
www.facebook.com/nguyenanhtucom

www.youtube.com/@nguyenanhtucom

EXERCISE

Internet of Things

PROGRAMMING



MỤC LỤC

Setup Development Environment	1
Example 0.01	1
IoT Broker	1
Example 1.01	1
Example 1.02	5
Example 1.03	13
IoT Device	22
Example 2.01	22
Example 2.02	25
Example 2.03	28
Example 2.04	31
Example 2.05	34
Example 2.06	38

Setup Development Environment

Example 0.01

Mục tiêu: Thiết lập môi trường lập trình Internet of Things và tạo project

Yêu cầu:

- ✓ Cài đặt JDK và Visual Studio Code
- ✓ Tạo project có tên project là **example01**

Hướng dẫn:

Bước 1: Cài đặt JDK và Visual Studio Code

JDK Development Kit 17.0.7 downloads

JDK 17 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions.

JDK 17 will receive updates under these terms, until September 2024, a year after the release of the next LTS.

Linux macOS Windows

Product/file description	File size	Download
x64 Compressed Archive	172.19 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip (sha256)
x64 Installer	153.28 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe (sha256)
x64 MSI Installer	152.07 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi (sha256)

Bước 2: Để cài đặt Extension Java cho Visual Studio Code bạn hãy thực hiện Mở Extensions (Ctrl+Shift+X), tìm kiếm **java**



Extension Pack for Java

v0.25.12

Preview

Microsoft [microsoft.com](https://www.microsoft.com) | 20,296,269 | ★★★★★ (62)

Popular extensions for Java development that provides Java IntelliSense, debugging, testing, Maven/Gradle support, project management and more

Disable

Uninstall

Switch to Pre-Release Version



This extension is enabled globally.

Bước 3: Để cài đặt Extension Spring Boot cho Visual Studio Code bạn hãy thực hiện Mở Extensions (Ctrl+Shift+X), tìm kiếm **Spring Boot Extension Pack**



Spring Boot Extension Pack

v0.2.1

VMware [vmware.com](https://www.vmware.com) | 1,643,692 | ★★★★★ (15)

A collection of extensions for developing Spring Boot applications

Installing



Bước 4: Sử dụng Visual Studio Code tạo project **Maven** với thông tin như sau:

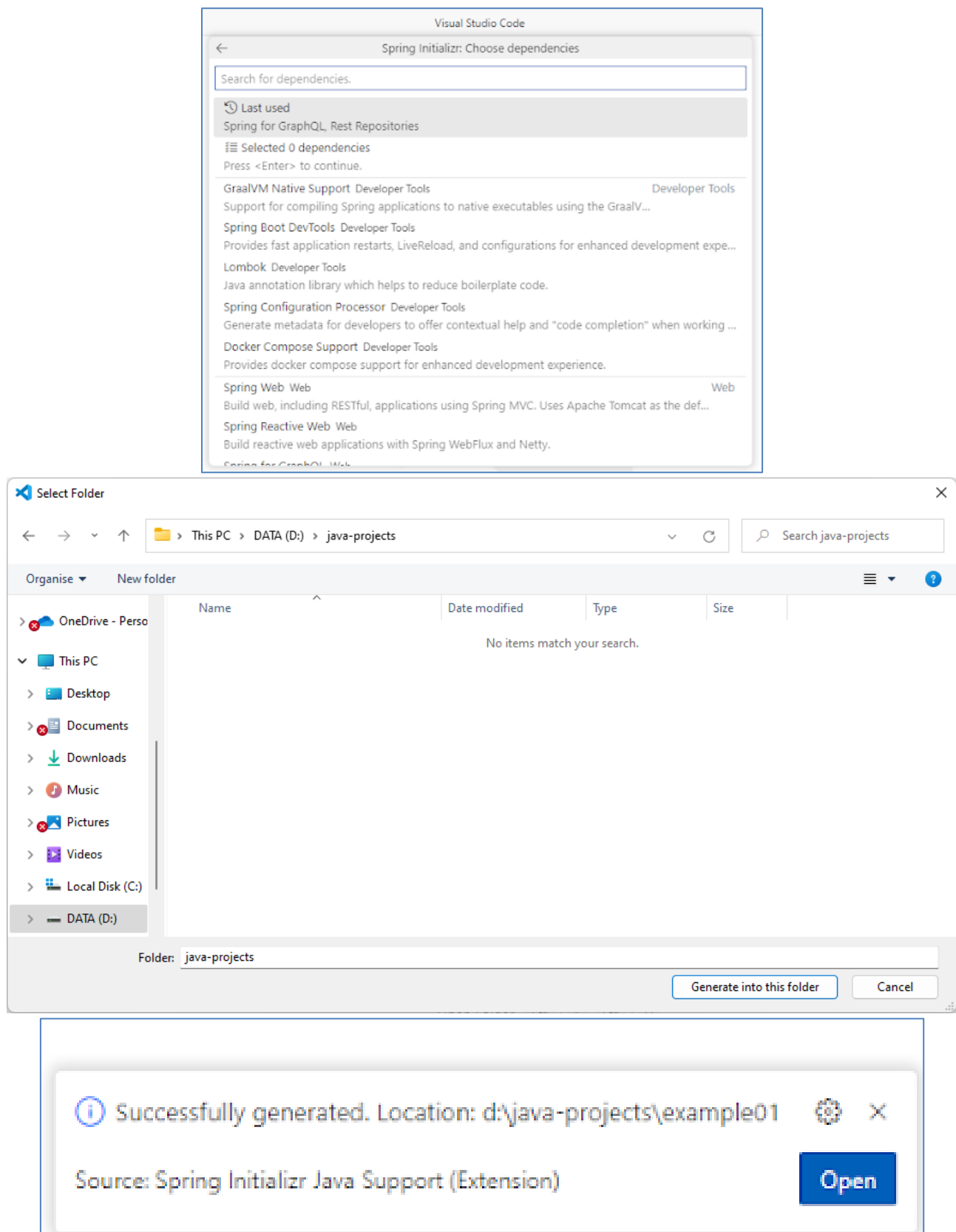
- ✓ Spring Boot version: **3.x.x**
- ✓ Language: **Java**
- ✓ Group Id: **com.nguyenanhtu**
- ✓ Artifact Id: **example01**
- ✓ Packaging type: **Jar**
- ✓ Java Version: **17**
- ✓ Dependencies: **Selected 0 dependencies**

✓ Vị trí project: **D:\java-projects**

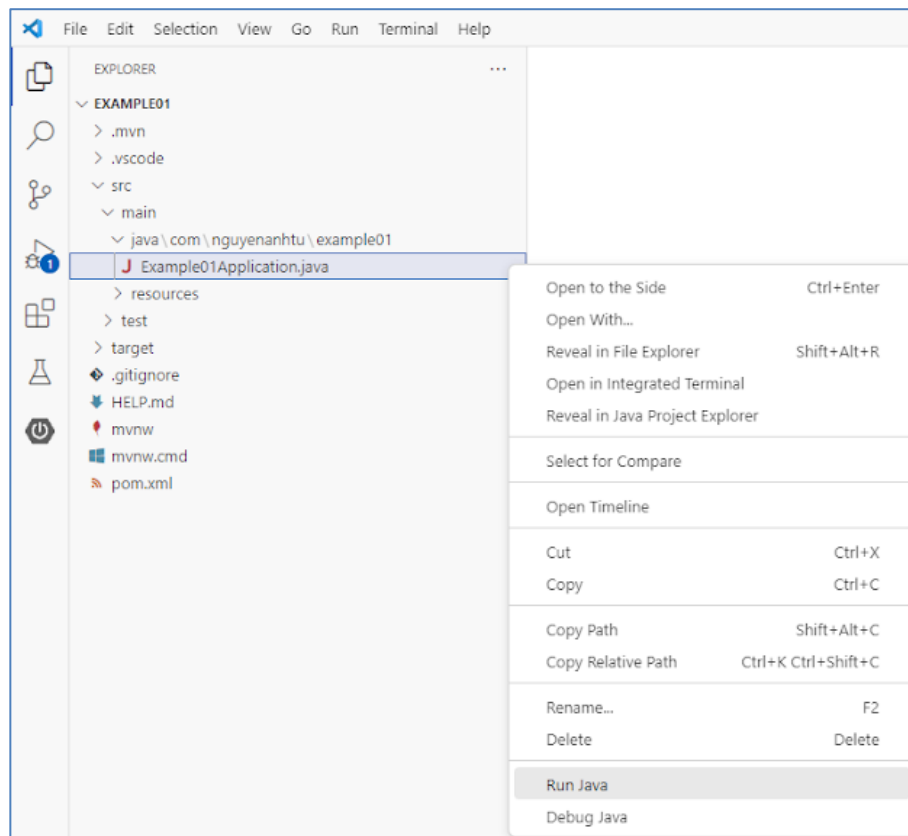
Hãy mở Bảng lệnh (Ctrl+Shift+P) và nhập **Spring Initializr** để bắt đầu tạo dự án **Maven**

The following steps illustrate the Spring Initializr wizard configuration:

- Step 1:** Command palette search results for `>Spring Initializr`.
 - Spring Initializr: Create a Maven Project... (recently used)
 - Spring Initializr: Add Starters...
 - Spring Initializr: Create a Gradle Project...
- Step 2:** Spring Initializr: Specify Spring Boot version.
 - Specify Spring Boot version.
 - 3.1.1 (selected)
 - 3.2.0 (SNAPSHOT)
 - 3.1.2 (SNAPSHOT)
 - 3.0.9 (SNAPSHOT)
 - 3.0.8
 - 2.7.14 (SNAPSHOT)
 - 2.7.13
- Step 3:** Spring Initializr: Specify project language.
 - Specify project language.
 - Java (selected)
 - Kotlin
 - Groovy
- Step 4:** Spring Initializr: Input Group Id.
 - com.nguyenanhtu
 - Input Group Id for your project. (Press 'Enter' to confirm or 'Escape' to cancel)
- Step 5:** Spring Initializr: Input Artifact Id.
 - example01
 - Input Artifact Id for your project. (Press 'Enter' to confirm or 'Escape' to cancel)
- Step 6:** Spring Initializr: Specify packaging type.
 - Specify packaging type.
 - Jar (selected)
 - War
- Step 7:** Spring Initializr: Specify Java version.
 - Specify Java version.
 - 17 (selected)
 - 20
 - 11
 - 8



Bước 5: Click chuột phải vào lớp **Example01Application** sau đó Click vào **Run Java**.



PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL
			<div> Run: Example01Application </div> <pre> 2023-07-09T11:21:45.752+07:00 INFO 7960 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2076 ms 2023-07-09T11:21:46.951+07:00 INFO 7960 --- [main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 1 endpoint(s) beneath base path '/actuator' 2023-07-09T11:21:47.030+07:00 INFO 7960 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path '' 2023-07-09T11:21:47.046+07:00 INFO 7960 --- [main] c.n.example01.Example01Application : Started Example01Application in 4.152 seconds (process running for 5.033) 2023-07-09T11:21:49.082+07:00 INFO 7960 --- [on(1)-127.0.0.1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet' 2023-07-09T11:21:49.083+07:00 INFO 7960 --- [on(1)-127.0.0.1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet' 2023-07-09T11:21:49.086+07:00 INFO 7960 --- [on(1)-127.0.0.1] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms </pre>

IoT Broker

Example 1.01

Mục tiêu: Quản lý ứng dụng **MQTT-Explorer** kết nối **MQTT Broker** có kiến trúc như sau:



Yêu cầu: Thực hiện các chức năng sau:

- ✓ Sử dụng **MQTT-Explorer** Subscribe với **Broker** với topic có tên */test/topic*
- ✓ Sử dụng **MQTT-Explorer** Publish nội dung "Hi from the IoT application" đến **Broker** với topic có tên */test/topic*

Hướng dẫn:

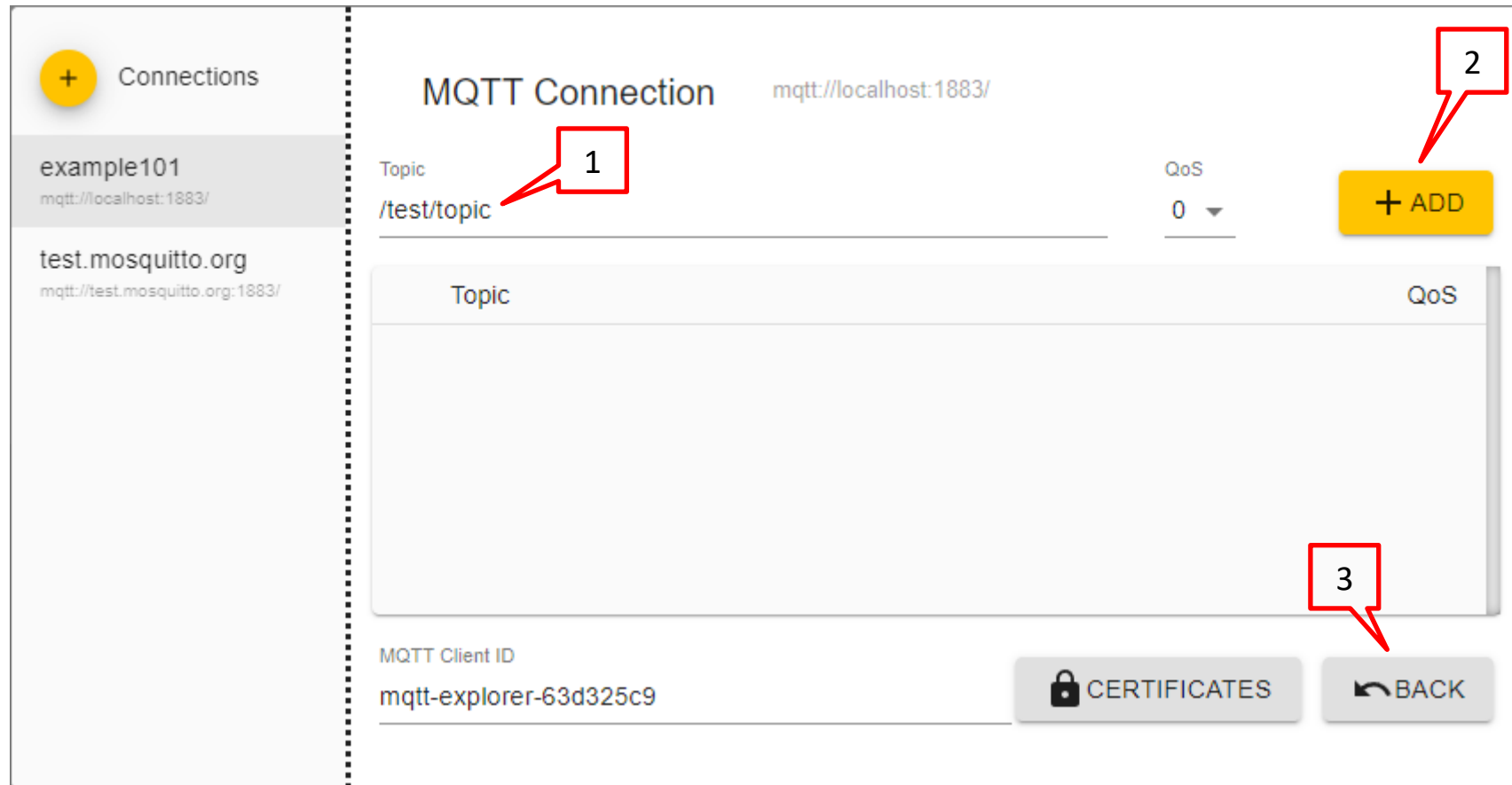
Bước 1: Sử dụng Visual Studio Code, mở Command Palette (Ctrl+Shift+P) và nhập **Java: Create Java Project...** để bắt đầu tạo project với các thông tin như sau:

Bước 1: Sử dụng **MQTT-Explorer** đăng ký topic có tên **/test/topic** với Broker

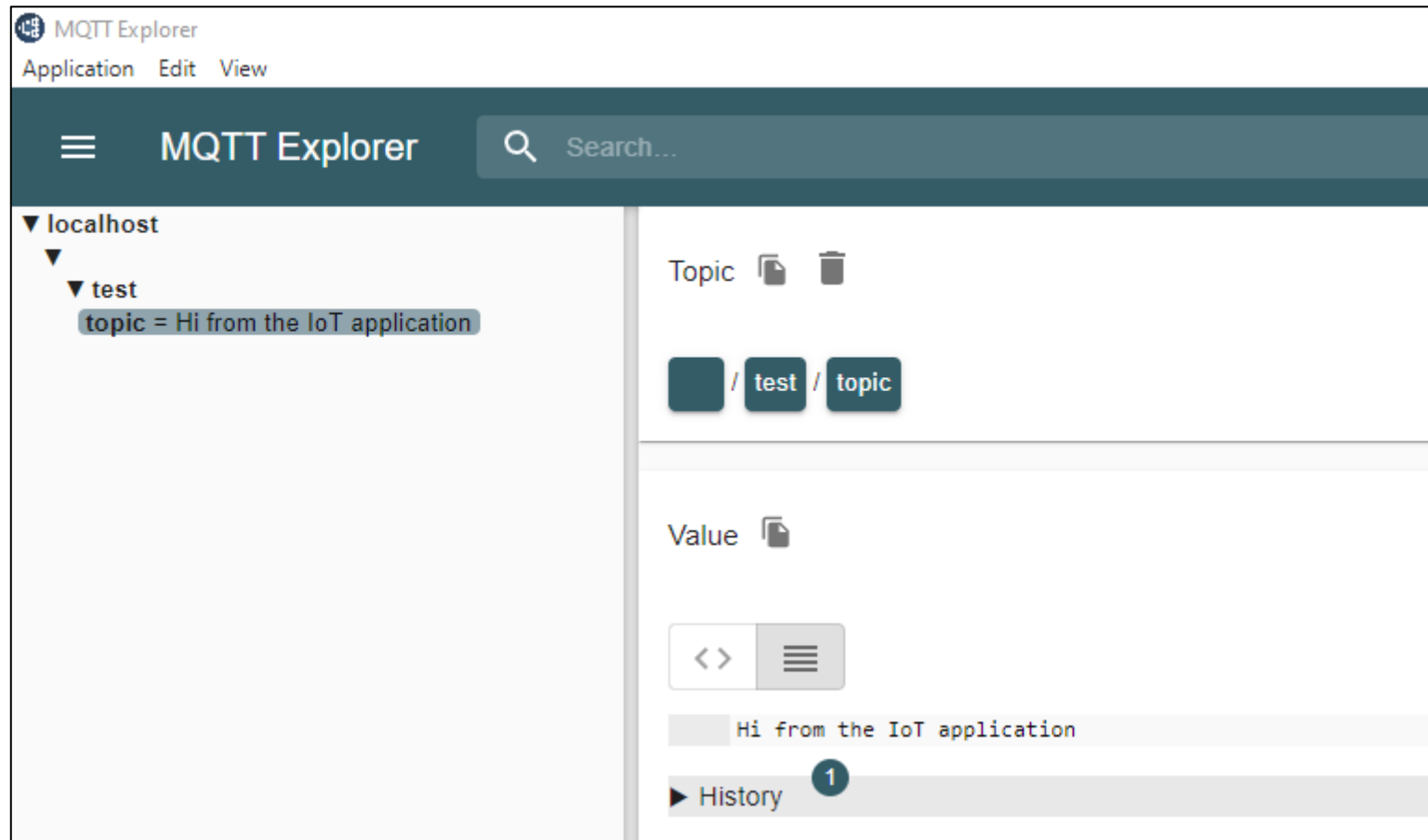
The screenshot displays the MQTT-Explorer application interface. On the left sidebar, under the 'Connections' section, two connections are listed: 'example101' (mqtt://localhost:1883/) and 'test.mosquitto.org' (mqtt://test.mosquitto.org:1883/). The main area is titled 'MQTT Connection' and shows the configuration for the selected connection 'mqtt://localhost:1883/'. The configuration fields are as follows:

- Name:** example101 (highlighted with a red box and arrow labeled 1)
- Validate certificate:** Toggle switch (off)
- Encryption (tls):** Toggle switch (off)
- Protocol:** mqtt:// (dropdown menu)
- Host:** localhost (highlighted with a red box and arrow labeled 2)
- Port:** 1883 (highlighted with a red box and arrow labeled 3)
- Username:** (empty field, highlighted with a red box and arrow labeled 4)
- Password:** (empty field with a toggle icon)

At the bottom of the configuration area, there are four buttons: 'DELETE' (with a trash icon), 'ADVANCED' (with a gear icon), 'SAVE' (yellow button with a floppy disk icon), and 'CONNECT' (blue button with a power icon).



Bước 4: Sử dụng **MQTT-Explorer** publish tới Broker nội dung **"Hi from the IoT application"** với topic có tên **/test/topic**



Example 1.02

Mục tiêu: Tạo và quản lý ứng dụng **Java** sử dụng **MQTT Paho** kết nối **MQTT Broker** có kiến trúc như sau:



Yêu cầu: Ứng dụng **Java** Thực hiện các chức năng sau:

- ✓ Subscribe với **Broker** với topic có tên */test/topic*
- ✓ Publish nội dung "Hi from the IoT application" đến **Broker** với topic có tên */test/topic*

Hướng dẫn:

Bước 2: Sửa file pom.xml như sau:**pom.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.nguyenanhtu</groupId>
  <artifactId>example102</artifactId>
  <version>1.0-SNAPSHOT</version>

  <dependencies>
    <dependency>
      <groupId>org.eclipse.paho</groupId>
      <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
      <version>1.2.5</version>
    </dependency>
  </dependencies>

  <properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
  </properties>
</project>
```

Bước 4: Tạo Entity **Employee** như sau:**Main.java**

```
package com.nguyenanhtu.example101;

import java.io.IOException;

import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttClient;

public class Main {
    public static void main(String[] args) {
        String mqttBroker = "tcp://localhost:1883";
        String mqttTopic = "/test/topic";
        String username = "IoTClient";
        String password = "IoTPass";
        String testMsg = "Hi from the IoT application";
        int qos = 1;

        try {
            MqttClient mqttClient = new MqttClient(mqttBroker, "mqttClient");
            MqttConnectOptions mqttOptions = new MqttConnectOptions();
            mqttOptions.setUserName(username);
            mqttOptions.setPassword(password.toCharArray());
            mqttClient.connect(mqttOptions);

            if (mqttClient.isConnected()) {
                mqttClient.setCallback(new MqttCallback() {
                    @Override
```

```
        public void messageArrived(String topic, MqttMessage message) throws Exception {
            System.out.println("Received message: " + new String(message.getPayload()));
        }

        @Override
        public void connectionLost(Throwable cause) {
            System.out.println("Connection is lost: " + cause.getMessage());
        }

        @Override
        public void deliveryComplete(IMqttDeliveryToken token) {
            System.out.println("Message publish is complete: " + token.isComplete());
        }
    });

    /* The client subscribe to a topic */
    mqttClient.subscribe(mqttTopic, qos);
    /* Preparing a message to be published */
    MqttMessage mqttMsg = new MqttMessage(testMsg.getBytes());
    mqttMsg.setQos(qos);
    /* A message is published on the same subscribed topic */
    mqttClient.publish(mqttTopic, mqttMsg);
}

/* Keep the application open, so that the subscribe operation can tested */
System.out.println("Press Enter to disconnect");
System.in.read();
/* Proceed with disconnecting */
mqttClient.disconnect();
mqttClient.close();

} catch (MqttException e) {
    e.printStackTrace();
}
```

```
} catch (IOException e) {  
    throw new RuntimeException(e);  
}  
}  
}
```

Bước 3: Sử dụng **MQTT-Explorer** đăng ký topic có tên **/test/topic** với Broker

The screenshot displays the MQTT-Explorer application interface. On the left, a sidebar shows a list of connections: 'example101' (selected) and 'test.mosquitto.org'. The main panel is titled 'MQTT Connection' and shows the configuration for 'example101'. The connection URL is 'mqtt://localhost:1883/'. The interface includes fields for 'Name' (example101), 'Protocol' (mqtt://), 'Host' (localhost), 'Port' (1883), 'Username', and 'Password'. There are also toggle switches for 'Validate certificate' and 'Encryption (tls)'. At the bottom, there are buttons for 'DELETE', 'ADVANCED', 'SAVE', and 'CONNECT'. Four red callout boxes with numbers 1, 2, 3, and 4 point to the 'Name' field, 'Host' field, 'Port' field, and 'ADVANCED' button respectively.

Connections

example101
mqtt://localhost:1883/

test.mosquitto.org
mqtt://test.mosquitto.org:1883/

MQTT Connection mqtt://localhost:1883/

Name example101

Validate certificate

Encryption (tls)

Protocol mqtt://

Host localhost

Port 1883

Username

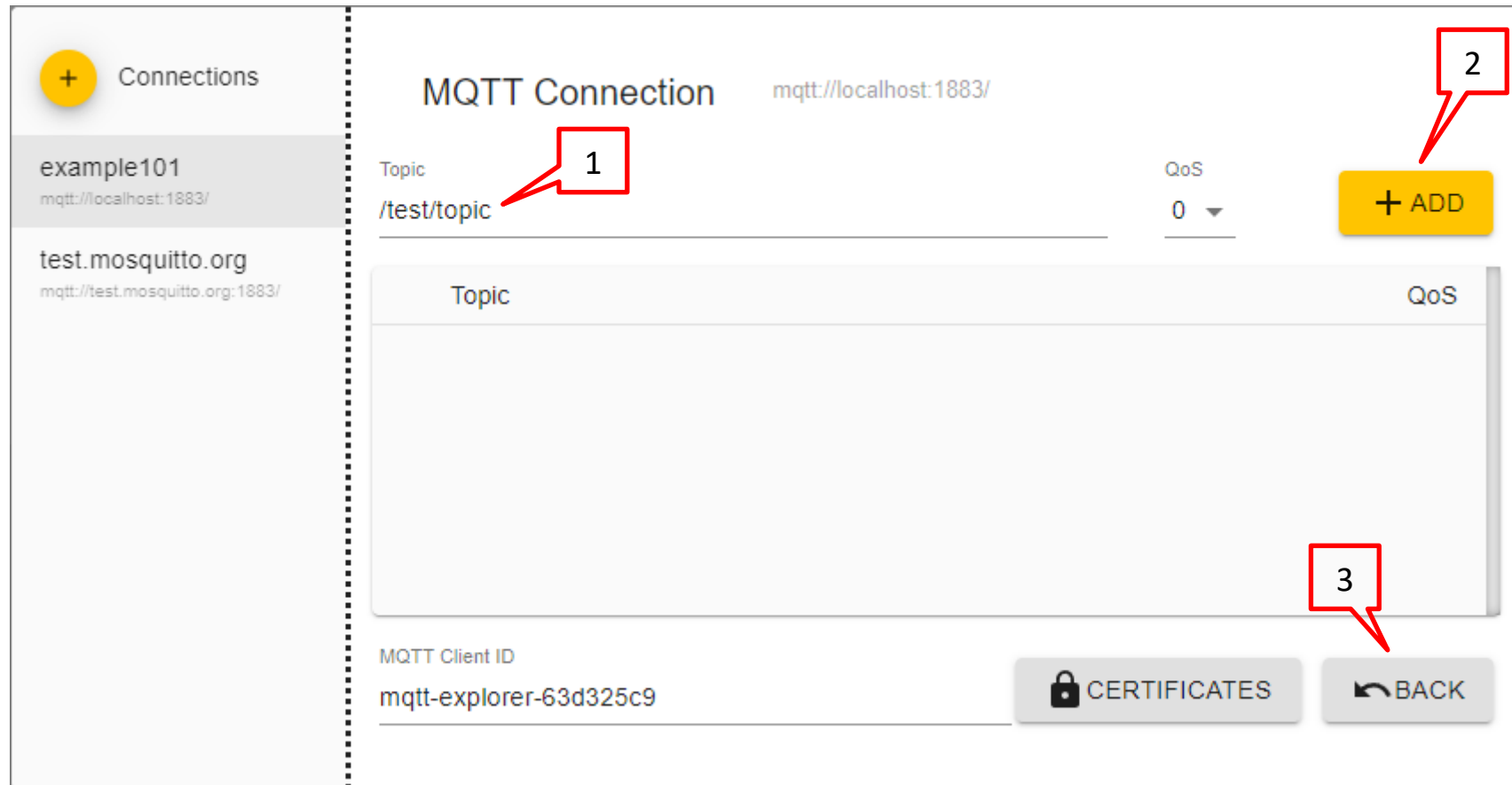
Password

DELETE

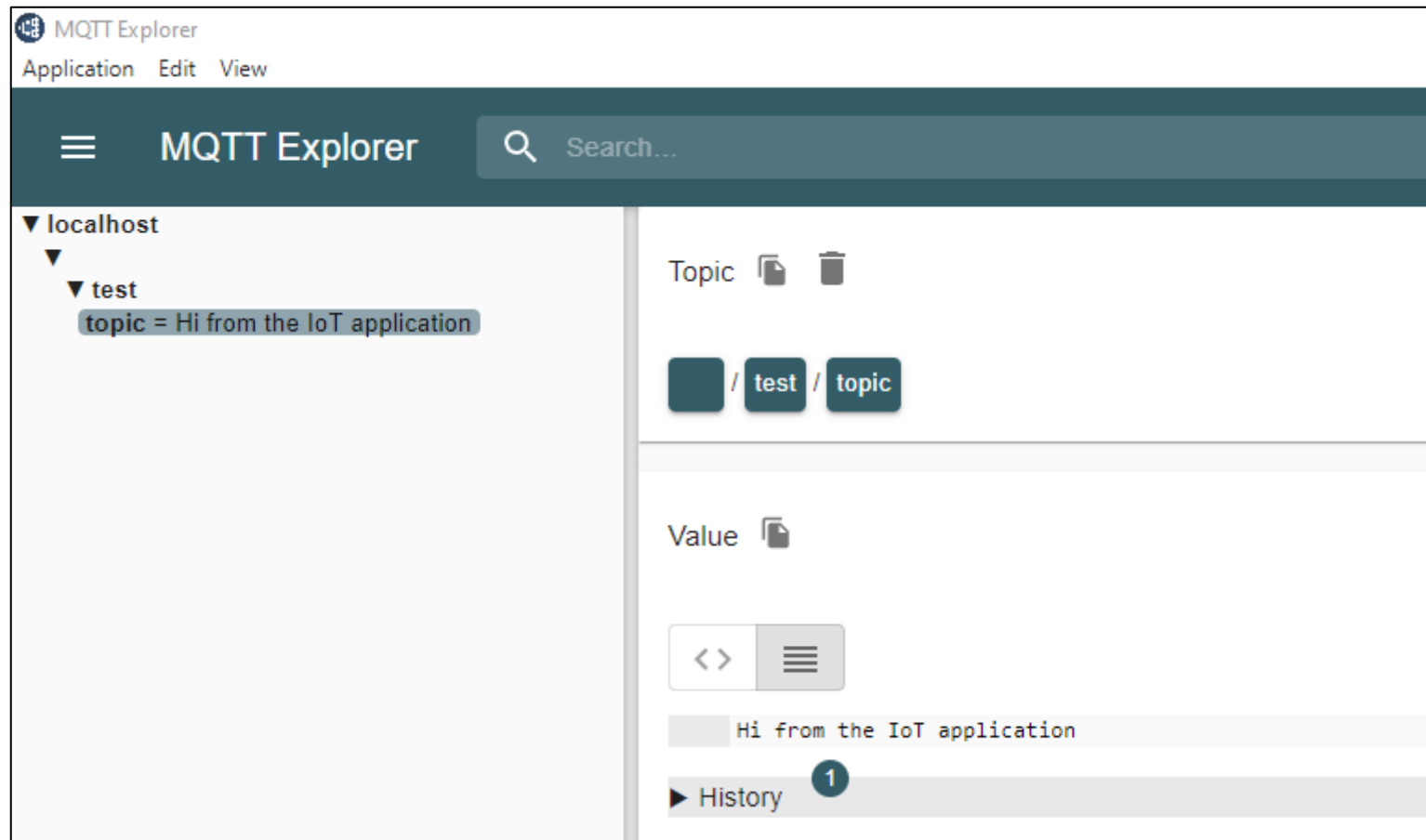
ADVANCED

SAVE

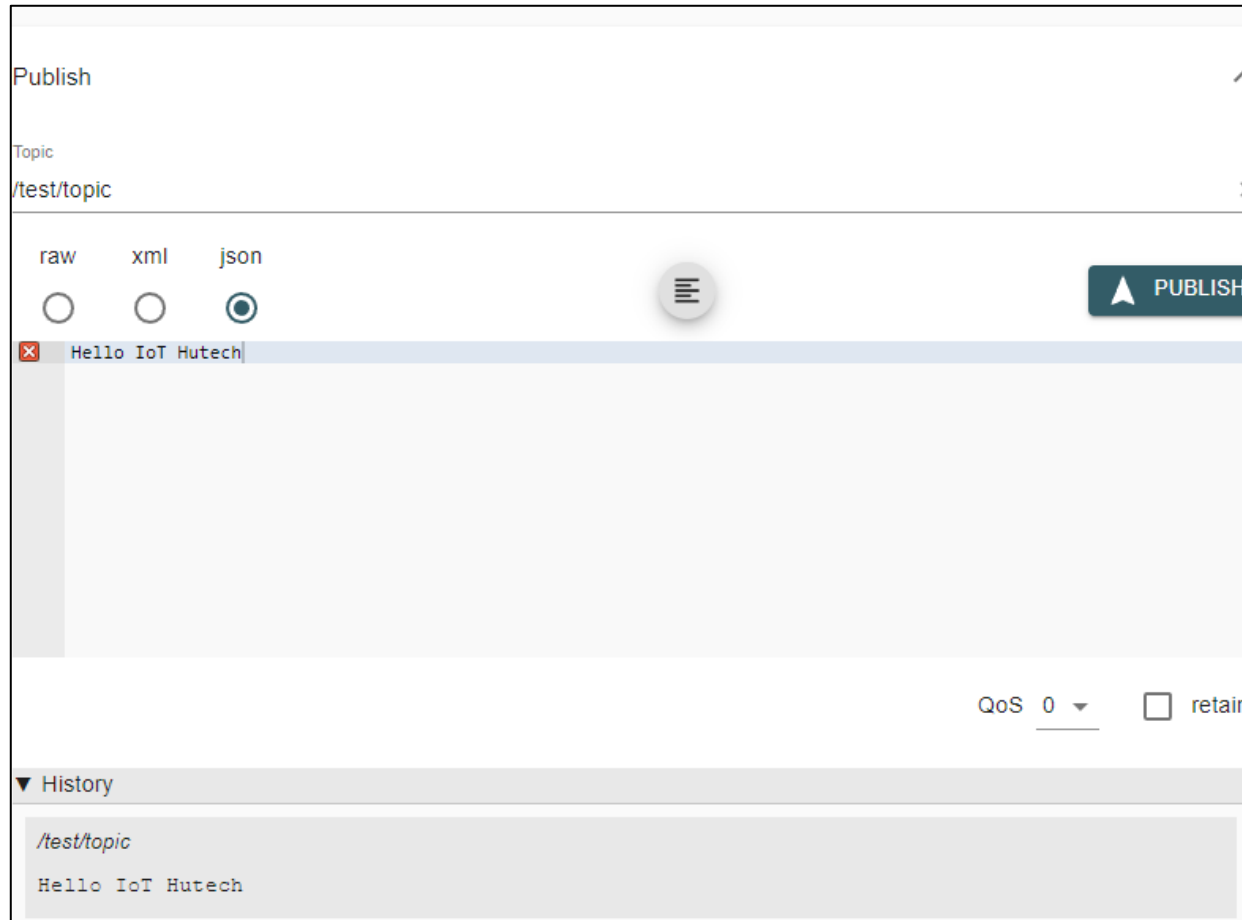
CONNECT



Bước 4: Click chuột phải vào lớp **Main** → **Run Java**, sau đó quan sát ứng dụng **MQTT-Explorer** như sau:



Bước 4: Sử dụng **MQTT-Explorer** publish tới Broker nội dung **Hello IoT Hutech** với topic có tên **/test/topic**

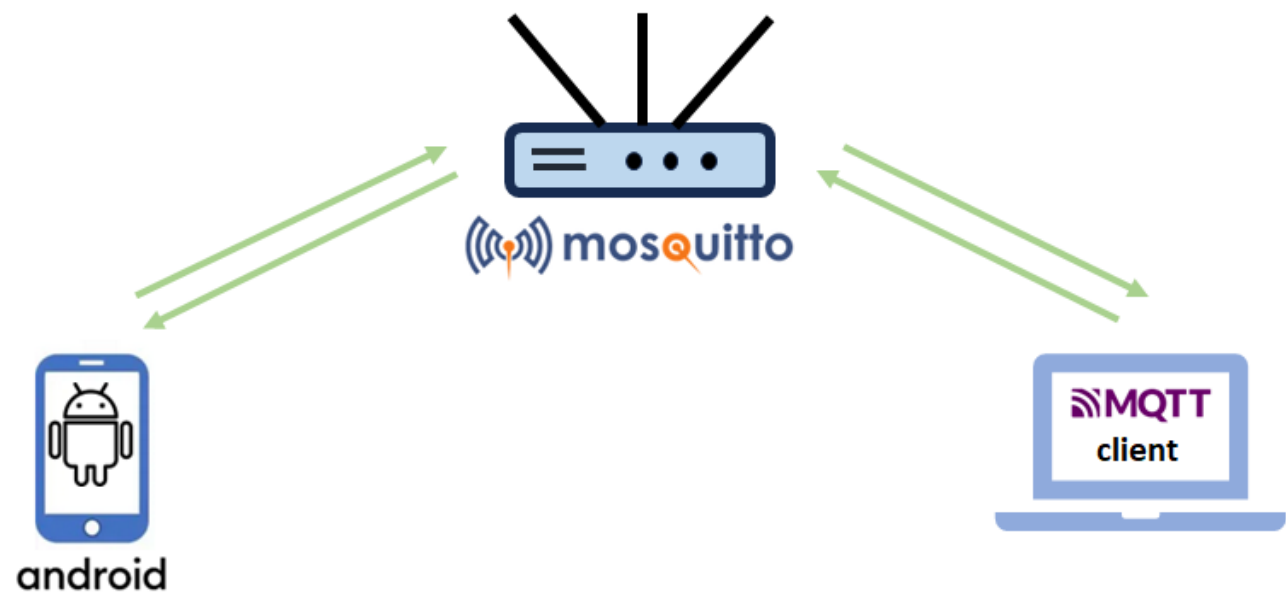


Bước 5: Kiểm tra dữ liệu nhận được trong cửa sổ Terminal của Visual Code

```
Press Enter to disconnect
Received message: Hi from the IoT application
Message publish is complete: true
Received message: Hello IoT Hutech
```

Example 1.03

Mục tiêu: Tạo và quản lý ứng dụng **Android** sử dụng **MQTT** (Eclipse Paho Java MQTT client) kết nối đến **MQTT Broker** có topic tên **/sensor/temp**



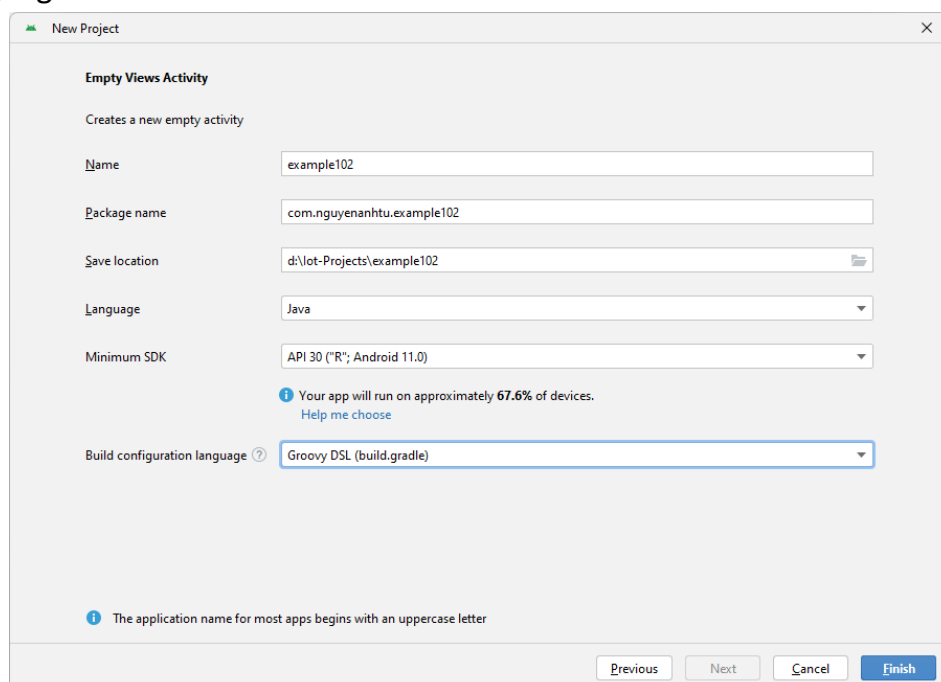
Yêu cầu: Ứng dụng **Android** Thực hiện các chức năng sau:

- ✓ Subscribe với **Broker** với topic có tên **/sensor/temp**

Hướng dẫn:

Bước 1: Sử dụng Android Studio tạo project với các thông tin như sau:

- ✓ Build configuration language: **Groovy DSL(build.gradle)**
- ✓ Vị trí project: **D:\IoT-projects**
- ✓ Artifact Id: **example102**
- ✓ Group Id: **com.nguyenanhtu**
- ✓ Language: **Java**



Bước 2: Thêm vào file **build.gradle** như sau:

build.gradle

```
dependencies {  
    . . .  
    implementation 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.2.5'  
    implementation 'org.eclipse.paho:org.eclipse.paho.android.service:1.1.1'  
}
```

Bước 3: Thêm vào file **AndroidManifest.xml** như sau:

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools">  
  
    <uses-permission android:name="android.permission.INTERNET" />  
    <uses-permission android:name="android.permission.WAKE_LOCK" />  
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />  
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />  
  
    <application  
        android:allowBackup="true"  
        android:dataExtractionRules="@xml/data_extraction_rules"  
        android:fullBackupContent="@xml/backup_rules"  
        android:icon="@mipmap/ic_launcher"  
        android:label="@string/app_name"  
        android:roundIcon="@mipmap/ic_launcher_round"  
        android:supportRtl="true"  
        android:theme="@style/Theme.Example102"  
        tools:targetApi="31">  
        <activity  
            android:name=".MainActivity"  
            android:exported="true">  
            <intent-filter>  
                <action android:name="android.intent.action.MAIN" />  
                <category android:name="android.intent.category.LAUNCHER" />  
            </intent-filter>  
        </activity>  
    </application>  
</manifest>
```

```
</activity>

    <service android:name="org.eclipse.paho.android.service.MqttService" />

</application>
</manifest>
```

Bước 4: Tạo file **MqttHelper.java** với nội dung như sau:

MqttHelper.java

```
package com.nguyenanhtu.example102;

import android.content.Context;
import android.util.Log;

import org.eclipse.paho.android.service.MqttAndroidClient;
import org.eclipse.paho.client.mqttv3.DisconnectedBufferOptions;
import org.eclipse.paho.client.mqttv3.IMqttActionListener;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.IMqttToken;
import org.eclipse.paho.client.mqttv3.MqttCallbackExtended;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;

public class MqttHelper {
    public MqttAndroidClient mqttAndroidClient;

    final String serverUri = "tcp://localhost:1883";

    final String clientId = "AndroidClient";
    final String subscriptionTopic = "/sensor/temp";

    final String username = "IoTClient";
    final String password = "IoTClient";

    public MqttHelper(Context context){
```

```
mqttAndroidClient = new MqttAndroidClient(context, serverUri, clientId);
mqttAndroidClient.setCallback(new MqttCallbackExtended() {
    @Override
    public void connectComplete(boolean b, String s) {
        Log.w("mqtt", s);
    }

    @Override
    public void connectionLost(Throwable throwable) {

    }

    @Override
    public void messageArrived(String topic, MqttMessage mqttMessage) throws Exception {
        Log.w("Mqtt", mqttMessage.toString());
    }

    @Override
    public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {

    }
});
connect();
}

public void setCallback(MqttCallbackExtended callback) {
    mqttAndroidClient.setCallback(callback);
}

private void connect() {
    MqttConnectOptions mqttConnectOptions = new MqttConnectOptions();
    mqttConnectOptions.setAutomaticReconnect(true);
    mqttConnectOptions.setCleanSession(false);
    mqttConnectOptions.setUsername(username);
    mqttConnectOptions.setPassword(password.toCharArray());
}
```

```
try {

    mqttAndroidClient.connect(mqttConnectOptions, null, new IMqttActionListener() {
        @Override
        public void onSuccess(IMqttToken asyncActionToken) {

            DisconnectedBufferOptions disconnectedBufferOptions = new DisconnectedBufferOptions();
            disconnectedBufferOptions.setBufferEnabled(true);
            disconnectedBufferOptions.setBufferSize(100);
            disconnectedBufferOptions.setPersistBuffer(false);
            disconnectedBufferOptions.setDeleteOldestMessages(false);
            mqttAndroidClient.setBufferOpts(disconnectedBufferOptions);
            subscribeToTopic();

        }

        @Override
        public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
            Log.w("Mqtt", "Failed to connect to: " + serverUri + exception.toString());
        }
    });

} catch (MqttException ex) {
    ex.printStackTrace();
}

}

private void subscribeToTopic() {
    try {
        mqttAndroidClient.subscribe(subscriptionTopic, 0, null, new IMqttActionListener() {
            @Override
            public void onSuccess(IMqttToken asyncActionToken) {
                Log.w("Mqtt", "Subscribed!");
            }
        })
    }
}
```

```

        @Override
        public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
            Log.w("Mqtt", "Subscribed fail!");
        }
    });

} catch (MqttException ex) {
    System.err.println("Exception whilst subscribing");
    ex.printStackTrace();
}
}
}

```

Bước 5: Sửa file activity_main.xml như sau:

res/layout/activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView

        android:id="@+id/dataReceived"

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

Bước 6: Sửa file MainActivity.java như sau:

MainActivity.java


```
package com.nguyenanhtu.example102;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallbackExtended;
import org.eclipse.paho.client.mqttv3.MqttMessage;

public class MainActivity extends AppCompatActivity {
    MqttHelper mqttHelper;
    TextView dataReceived;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        dataReceived = (TextView) findViewById(R.id.dataReceived);

        startMqtt();
    }

    private void startMqtt() {
        mqttHelper = new MqttHelper(getApplicationContext());
        mqttHelper.setCallback(new MqttCallbackExtended() {
            @Override
            public void connectComplete(boolean b, String s) {

            }

            @Override
            public void connectionLost(Throwable throwable) {

            }

            @Override
            public void messageArrived(String topic, MqttMessage mqttMessage) throws Exception {
```

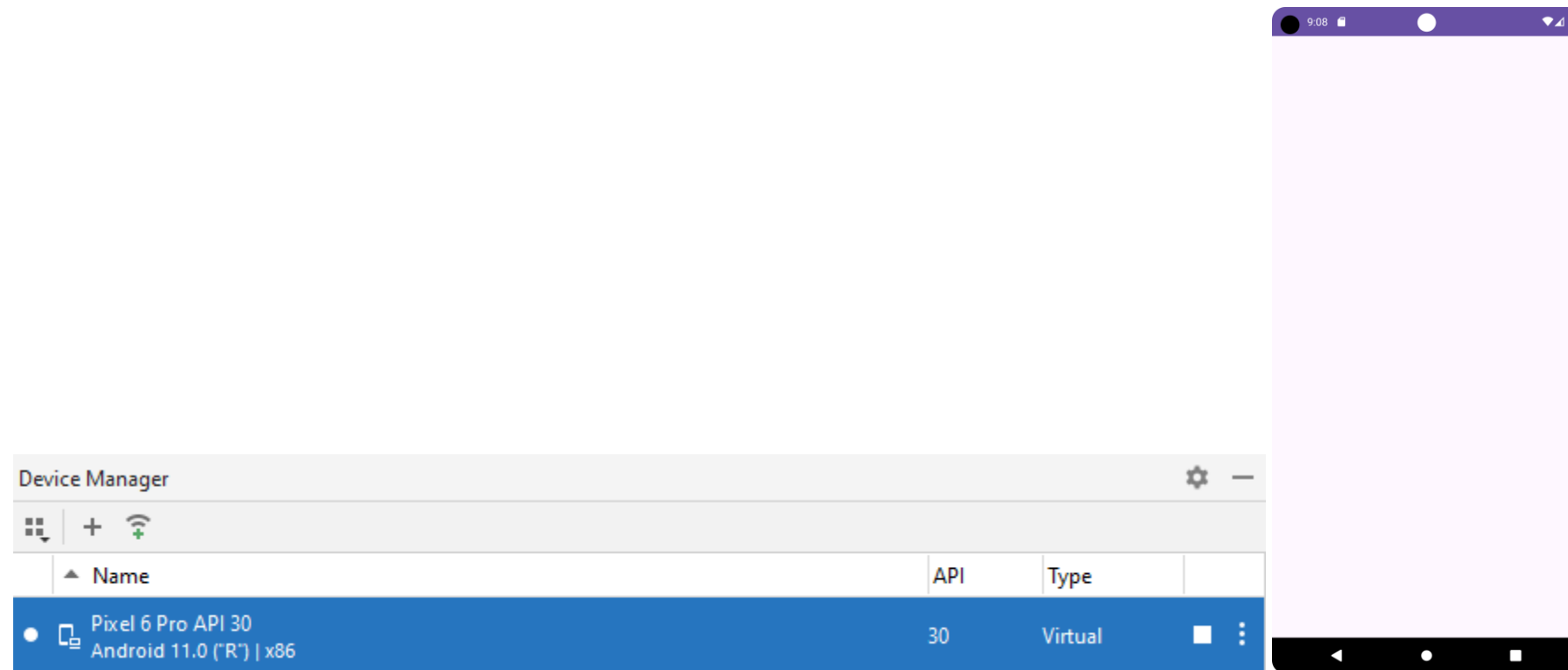
```
        Log.w("Debug", mqttMessage.toString());
        dataReceived.setText(mqttMessage.toString());
    }

    @Override
    public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {
    }

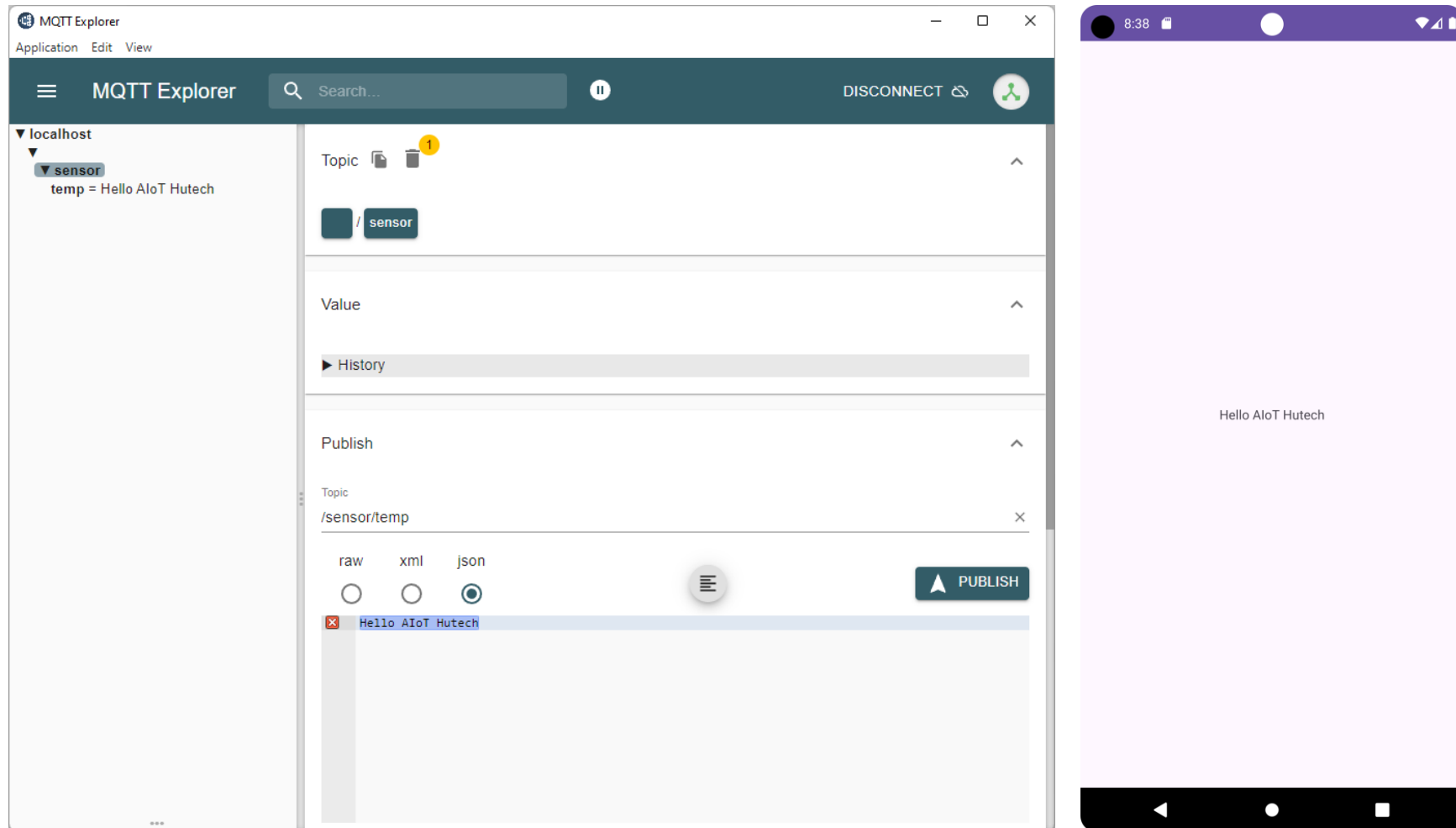
    });
}
}
```

Bước 7: Sử dụng **MQTT-Explorer** subscribe với Broker một topic có tên **/sensor/temp**

Bước 8: Run App Android



Bước 9: Sử dụng **MQTT-Explorer** publish tới Broker nội dung **Hello AIoT Hutech** với topic có tên **/sensor/temp**



IoT Device

Example 2.01

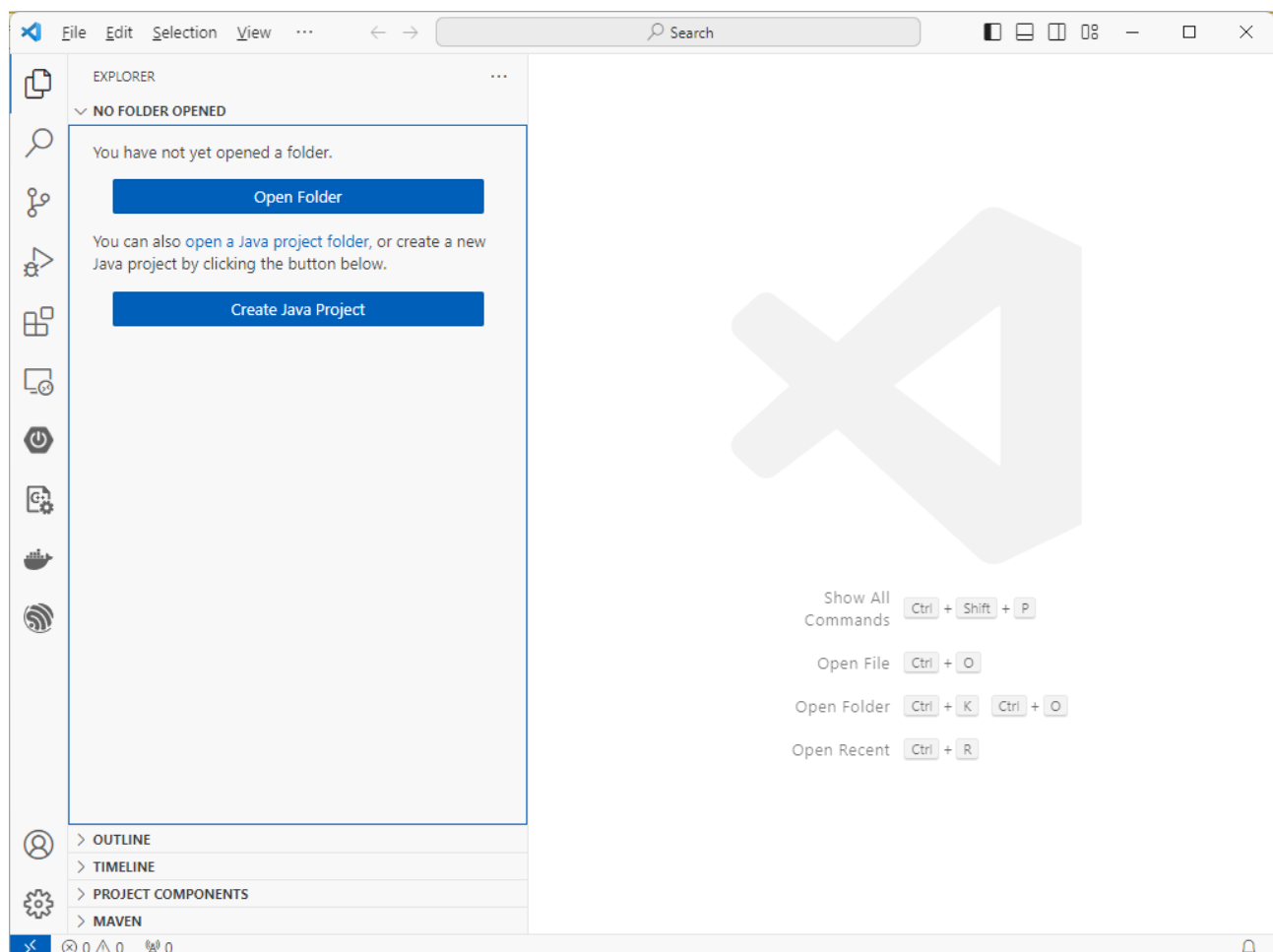
Mục tiêu: Thiết lập môi trường lập trình ESP-IDF (Espressif IoT Development Framework)

Yêu cầu:

- ✓ Cài đặt Visual Studio Code
- ✓ Cài đặt và cấu hình Espressif IDF Extensions


Hướng dẫn:

Bước 1: Download tại <https://code.visualstudio.com>, sau đó tiến hành cài đặt Visual Studio Code



Bước 2: Cài đặt và cấu hình Espressif IDF

- ✓ Để cài đặt Extension Espressif IDF cho Visual Studio Code bạn hãy thực hiện Mở Extensions (Ctrl+Shift+X), tìm kiếm **Espressif IDF**




Espressif IDF

v1.6.5


Espressif Systems espressif.com | 515,985 | ★★★★★ (97)


Develop and debug applications for Espressif ESP32, ESP32-S2 chips with ESP-IDF

[Disable](#) [Uninstall](#) 

This extension is enabled globally.

- ✓ Để cấu hình Extension Espressif IDF cho Visual Studio Code bạn hãy thực hiện Mở Command Palette (Ctrl+Shift+P), chọn **ESP-IDF: Configure ESP-IDF extension**


ESP-IDF Explorer: Focus on ESP-IDF Docs search results View similar commands 

ESP-IDF: Configure ESP-IDF extension 

ESP-IDF Explorer: Focus on IDF App Tracer View

ESP-IDF: Add Arduino ESP32 as ESP-IDF Component

ESP-IDF: Clear ESP-IDF Search results

 **ESPRESSIF**


ESP-IDF Extension for Visual Studio Code

Welcome.

are installed before choosing the setup mode.

Choose a setup mode.

Select where to save these settings:

Global 

EXPRESS


Fastest option. Choose ESP-IDF, ESP-IDF Tools directory and python executable to create ESP-IDF.

ADVANCED

Configurable option. Choose ESP-IDF, ESP-IDF Tools directory and python executable to create ESP-IDF.
Can choose ESP-IDF Tools download or manually input each existing ESP-IDF tool path.

USE EXISTING SETUP

Select existing ESP-IDF setup saved in the extension or find ESP-IDF in your system.

 **ESPRESSIF**

ESP-IDF Extension for Visual Studio Code

Select download server:

Github ▾

☐ Show all ESP-IDF tags


Select ESP-IDF version:

v5.1.2 (release version) ▾

Enter ESP-IDF container directory


C:\Users\NAT\esp

\esp-idf



Enter ESP-IDF Tools directory (IDF_TOOLS_PATH)

C:\Users\NAT\.espressif



Install

Example 2.02

Mục tiêu: Tạo và quản lý ứng dụng chạy trên vi điều khiển **ESP32**:

Yêu cầu: Ứng dụng thực hiện các chức năng sau:

- ✓ In ra Terminal nội dung **Hello World**

Hướng dẫn:

Bước 1: Sử dụng Visual Studio Code, mở Command Palette (Ctrl+Shift+P) và nhập **IDF: New Project**) để bắt đầu tạo project với các thông tin như sau:

- ✓ Project Name: **example202**
- ✓ Enter Project directory: **D:\java-projects**
- ✓ Choose ESP-IDF Board: **ESP32-C3 chip(via ESP USB Bridge)**
- ✓ Choose serial port: **COMx**
- ✓ Choose Template: **template-app**

New Project

Project Name
example201

Enter Project directory
D:\IoT-Projects \example201

Choose ESP-IDF Board
ESP32-C3 chip (via ESP USB Bridge)

Choose serial port
COM1

Add your ESP-IDF Component directory

[Choose Template](#)

New Project

ESP-IDF

Search Template By Name

get-started

- blink
- hello_world**
- sample_project
- bluetooth

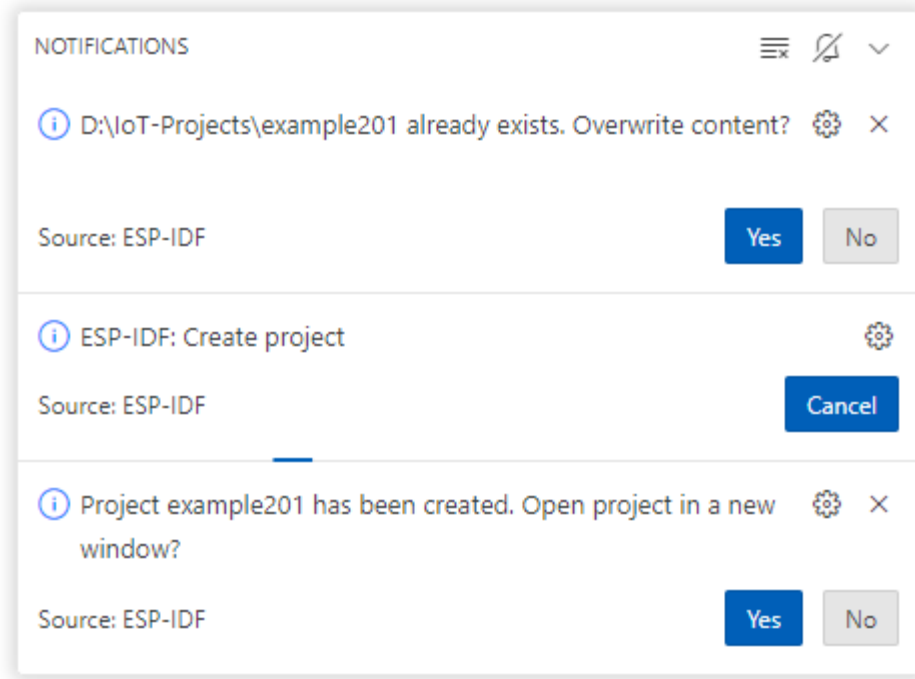
[Create project using template hello_world](#)

Supported Targets ESP32 ESP32-C2 **ESP32-C3** ESP32-C6 ESP32-H2 ESP32-S2 ESP32-S3

Hello World Example

Starts a FreeRTOS task to print "Hello World".

(See the README.md file in the upper level 'examples' directory for more information about examples.)



Bước 2: Sửa file **hello_world_main.c** như sau:

hello_world_main.c

```
#include <stdio.h>
#include <inttypes.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_chip_info.h"
#include "esp_flash.h"

void app_main(void)
```



```
{  
    printf("Hello world!\n");  
    printf("Restarting now.\n");  
    fflush(stdout);  
    esp_restart();  
}
```

Bước 3: Build project và nạp chương trình vào vi điều khiển



Example 2.03

Mục tiêu: Tạo và quản lý ứng dụng sử dụng FreeRTOS chạy trên vi điều khiển **ESP32**:

Yêu cầu: Ứng dụng thực hiện các chức năng sau:

- ✓ In ra Terminal nội dung Hello World

Hướng dẫn:

Bước 1: Sử dụng Visual Studio Code, mở Command Palette (Ctrl+Shift+P) và nhập **IDF: New Project**) để bắt đầu tạo project với các thông tin như sau:

- ✓ Project Name: **example203**
- ✓ Enter Project directory: **D:\java-projects**
- ✓ Choose ESP-IDF Board: **ESP32-C3 chip(via ESP USB Bridge)**
- ✓ Choose serial port: COMx
- ✓ Choose Template: **template-app**

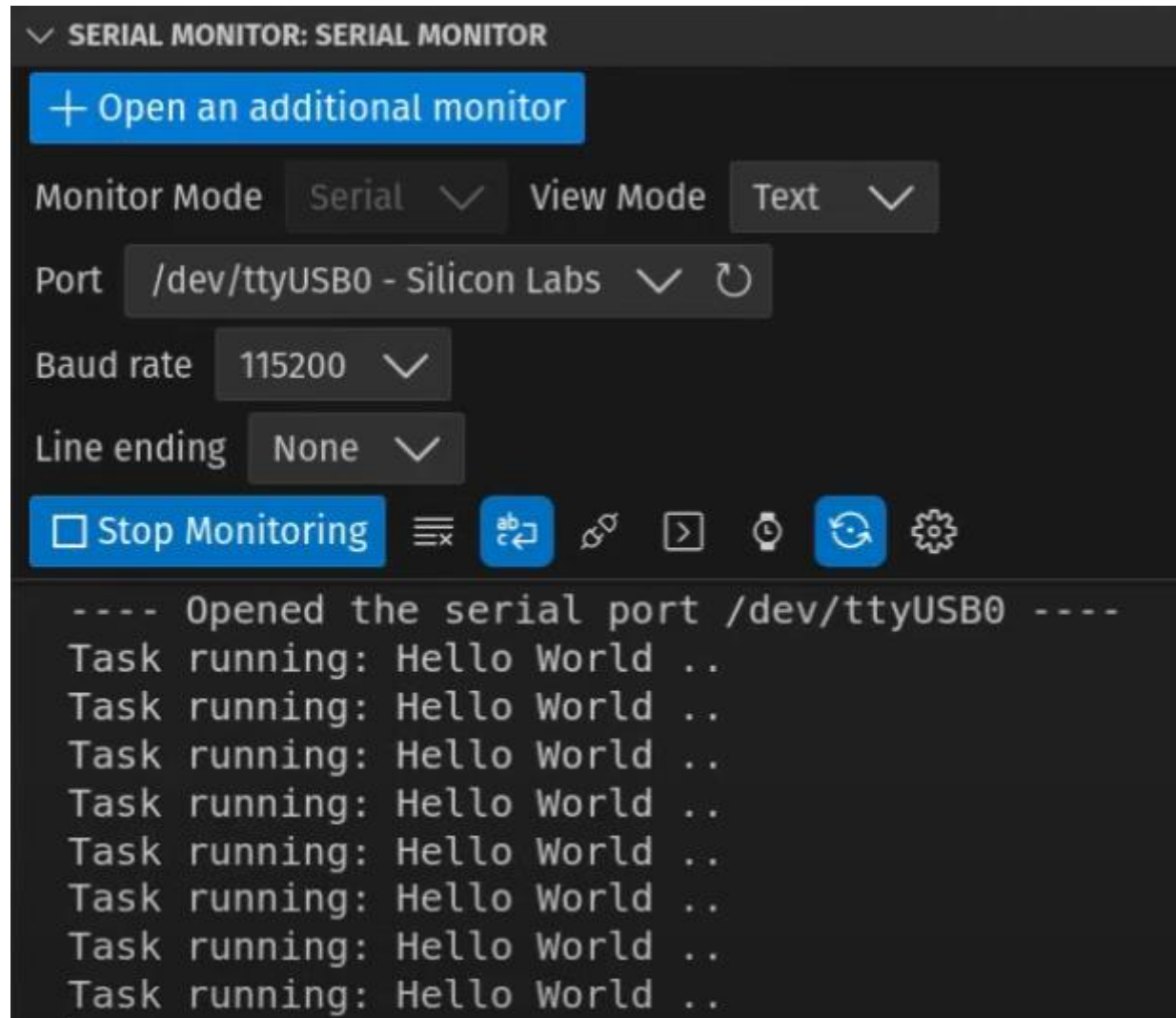
Bước 2: Sửa file `hello_world_main.c` như sau:

hello_world_main.c

```
#include <stdio.h>
#include <inttypes.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_chip_info.h"
#include "esp_flash.h"
TaskHandle_t HelloWorldTaskHandle = NULL;
void HelloWorld_Task(void *arg)
{
    while (1)
    {
        printf("Task running: Hello World ..\n");
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
void app_main(void)
{
    xTaskCreate(HelloWorld_Task, "HelloWorld", 4096, NULL, 10, &HelloWorldTaskHandle);
    //xTaskCreatePinnedToCore(HelloWorld_Task, "HelloWorld", 4096, NULL, 10, &HelloWorldTaskHandle, 1); // Run on Core 1
}
```

Bước 3: Build project và nạp chương trình vào vi điều khiển





Example 2.04

Mục tiêu: Tạo và quản lý ứng dụng sử dụng FreeRTOS chạy trên vi điều khiển **ESP32**:

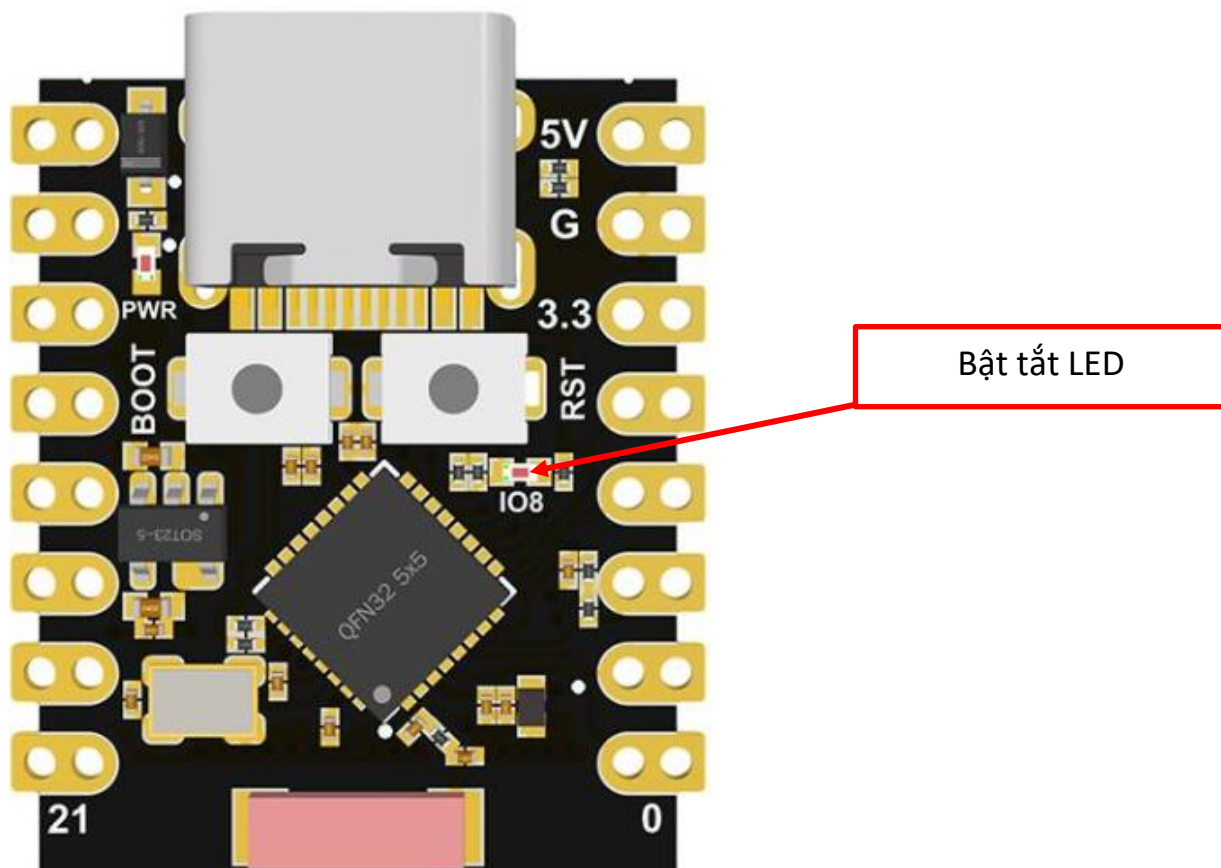
Yêu cầu: Ứng dụng thực hiện các chức năng sau:

- ✓ Thực hiện bật/tắt LED với chu kỳ 1000ms

Hướng dẫn:

Bước 1: Sử dụng Visual Studio Code, mở Command Palette (Ctrl+Shift+P) và nhập **IDF: New Project**) để bắt đầu tạo project với các thông tin như sau:

- ✓ Project Name: **example204**
- ✓ Enter Project directory: **D:\java-projects**
- ✓ Choose ESP-IDF Board: **ESP32-C3 chip(via ESP USB Bridge)**
- ✓ Choose serial port: COMx
- ✓ Choose Template: **template-app**



Bước 2: Sửa file **blink_main.c** như sau:

blink_main.c

```
#include <stdio.h>
#include <inttypes.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_chip_info.h"
#include "esp_flash.h"

#include "driver/gpio.h"

#define BLINK_GPIO GPIO_NUM_32

TaskHandle_t BlinkyTaskHandle = NULL;

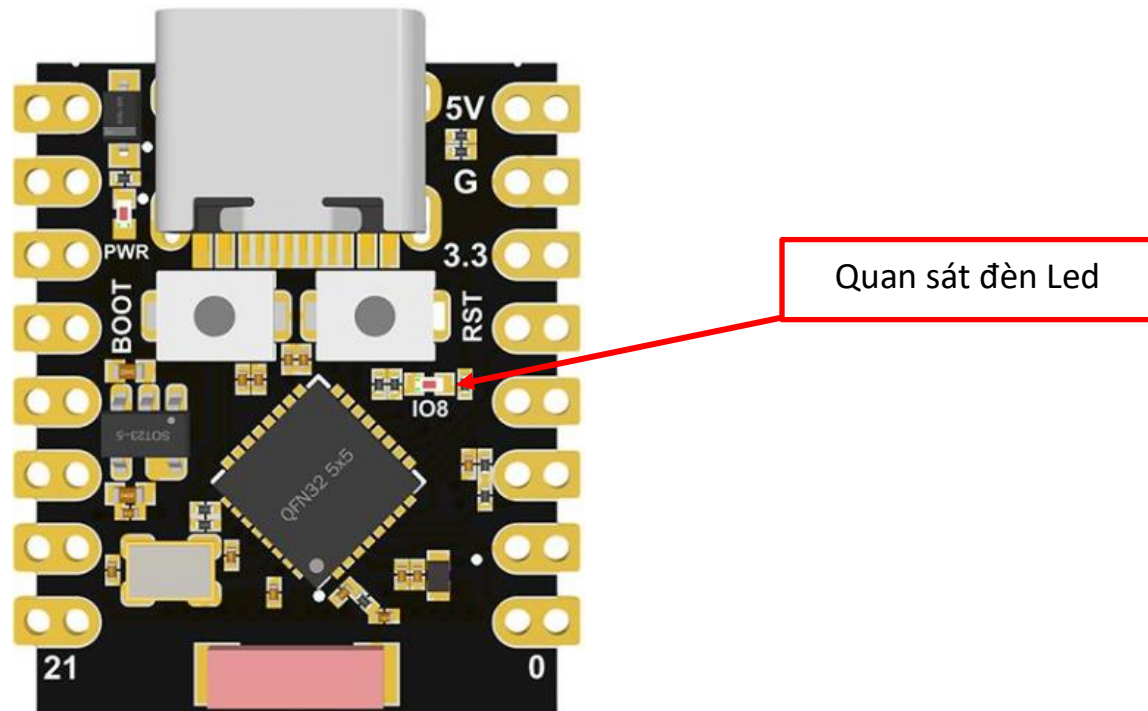
void Blinky_Task(void *arg)
{
    esp_rom_gpio_pad_select_gpio(BLINK_GPIO);
    gpio_set_direction(BLINK_GPIO, GPIO_MODE_OUTPUT);

    while (1)
    {
        gpio_set_level(BLINK_GPIO, 1);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        gpio_set_level(BLINK_GPIO, 0);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}

void app_main(void)
{
}
```

```
xTaskCreatePinnedToCore(Blinky_Task, "Blinky", 4096, NULL, 10, &BlinkyTaskHandle, 0); // Core 0  
}
```

Bước 3: Build project và nạp chương trình vào vi điều khiển, sau đó quan sát trạng thái của bóng đèn LED trên board



Example 2.05

Mục tiêu: Tạo và quản lý GIPO. Thực hiện bật tắt đèn Led trên board ESP32-C3

Mục tiêu: Tạo và quản lý ứng dụng sử dụng FreeRTOS chạy trên vi điều khiển **ESP32**:

Yêu cầu: Ứng dụng thực hiện các chức năng sau:

- ✓ Thực hiện bật/tắt LED với chu kỳ 1000ms

Hướng dẫn:

Bước 1: Sử dụng Visual Studio Code, mở Command Palette (Ctrl+Shift+P) và nhập **IDF: New Project**) để bắt đầu tạo project với các thông tin như sau:

- ✓ Project Name: **example204**
- ✓ Enter Project directory: **D:\java-projects**
- ✓ Choose ESP-IDF Board: **ESP32-C3 chip(via ESP USB Bridge)**
- ✓ Choose serial port: COMx
- ✓ Choose Template: **template-app**

Hướng dẫn:

Bước 1: Sử dụng Visual Studio Code tạo Project với thông tin như sau:

- ✓ Project Name: **Lab02**
- ✓ Choose ESP-IDF Board: **ESP32-C3 chip (via ESP USB Bridge)**
- ✓ Enter Project directory: **D:\IoT-projects**

Bước 2: Sửa main.c file như sau:

main.c

```
#include <stdio.h>

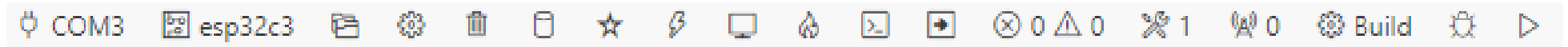
#include "driver\gpio.h"
#include "freeRTOS\freeRTOS.h"
#include "freeRTOS\task.h"

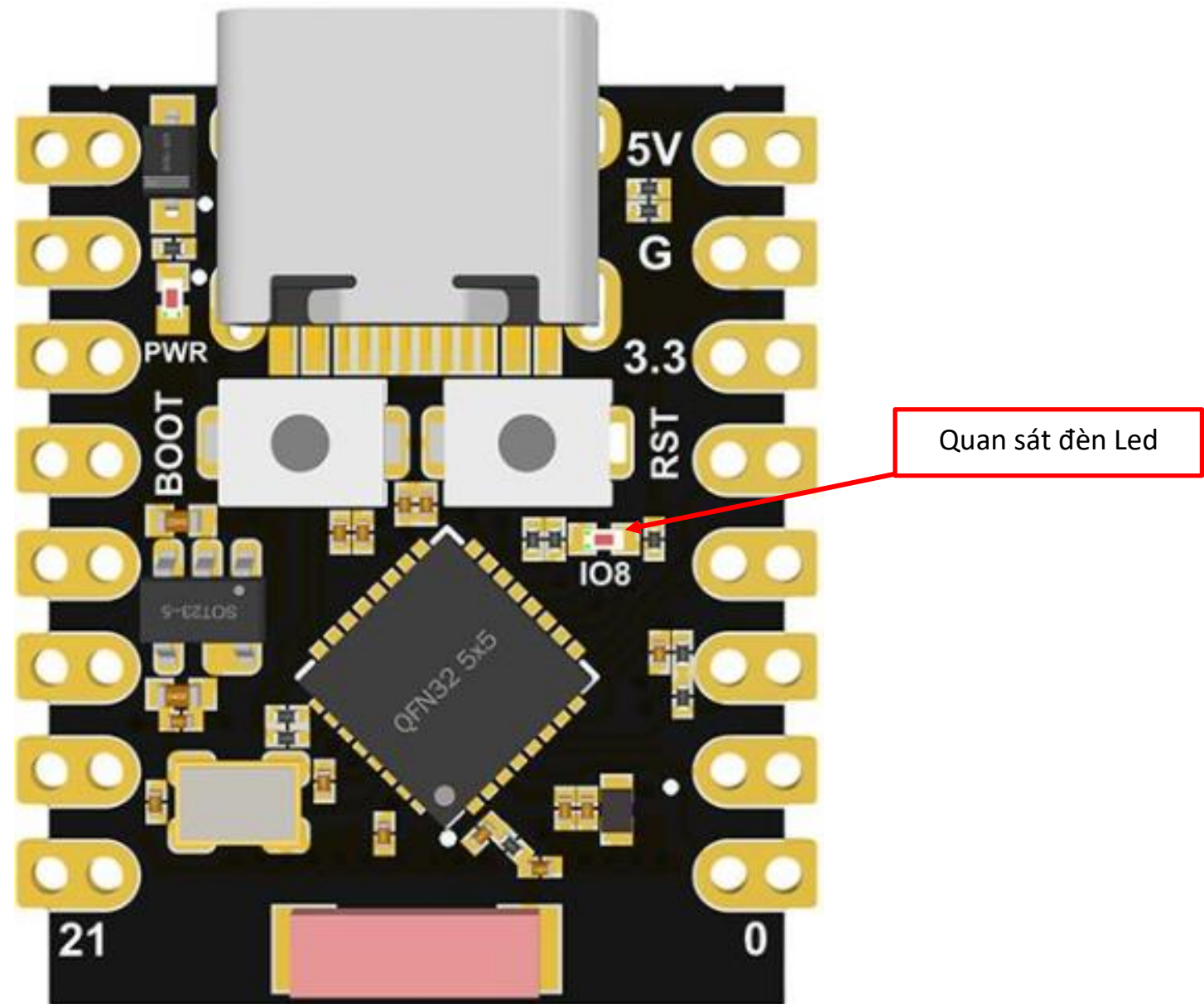
#define LED GPIO_NUM_8

void app_main(void)
{
    gpio_set_direction(LED, GPIO_MODE_DEF_OUTPUT);
    while(1)
    {
        gpio_set_level(LED, 1);
        vTaskDelay(100);
        gpio_set_level(LED, 0);
        vTaskDelay(100);
    }
}
```

Bước 3: Thực hiện build và flash chương trình lên board ESP32-C3

- ✓ Thực hiện mở Command Palette (Ctrl+Shift+P) sau đó chọn **ESP-IDF: Build your project** và **ESP-IDF: Flash (UART) your project**





Example 2.06

Mục tiêu: Tạo và quản lý ứng dụng sử dụng FreeRTOS chạy trên vi điều khiển **ESP32**:

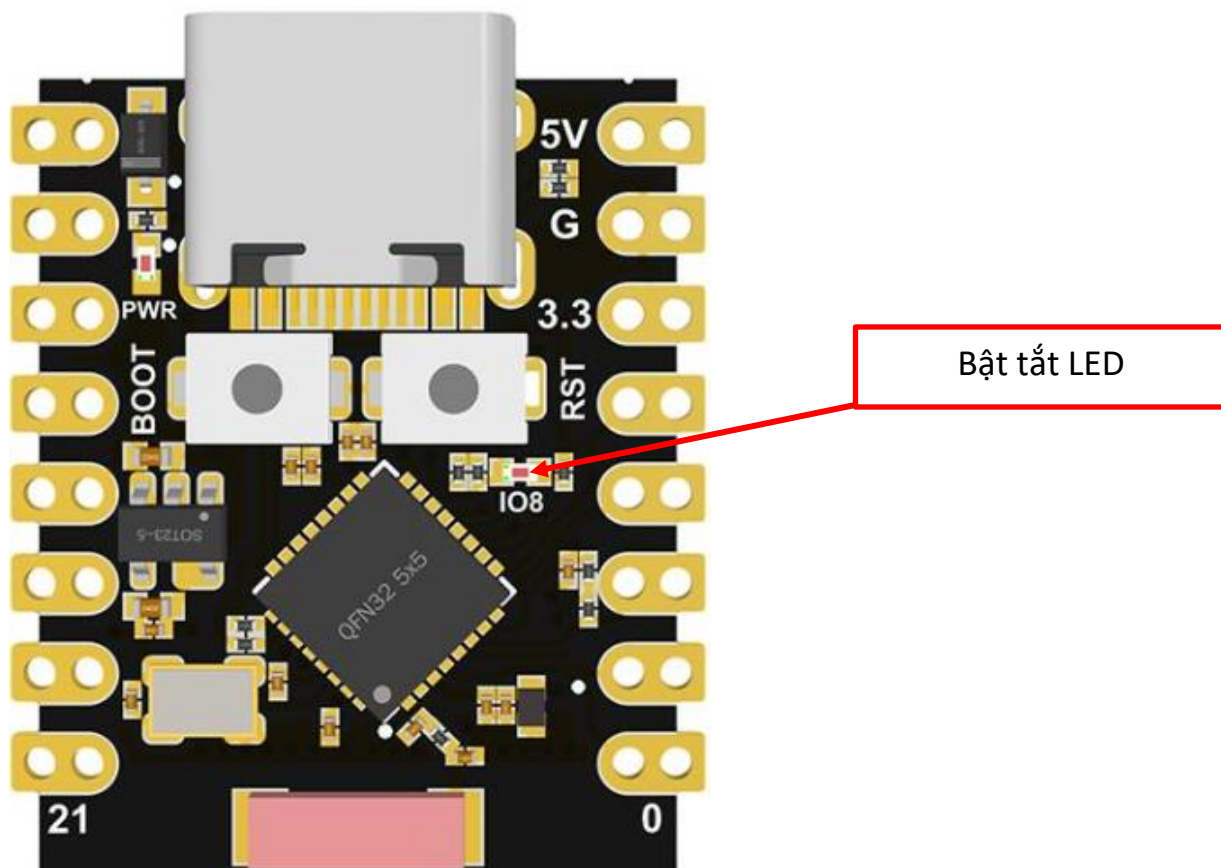
Yêu cầu: Ứng dụng thực hiện các chức năng sau:

- ✓ Thực hiện bật/tắt LED với chu kỳ 1000ms
- ✓ In ra Terminal nội dung Hello World ..

Hướng dẫn:

Bước 1: Sử dụng Visual Studio Code, mở Command Palette (Ctrl+Shift+P) và nhập **IDF: New Project**) để bắt đầu tạo project với các thông tin như sau:

- ✓ Project Name: **example205**
- ✓ Enter Project directory: **D:\java-projects**
- ✓ Choose ESP-IDF Board: **ESP32-C3 chip(via ESP USB Bridge)**
- ✓ Choose serial port: COMx



Bước 2: Sửa file **blink_main.c** như sau:

blink_main.c

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_log.h"
#include "sdkconfig.h"

#define BLINK_GPIO GPIO_NUM_32

TaskHandle_t HelloWorldTaskHandle = NULL;
TaskHandle_t BlinkyTaskHandle = NULL;

void HelloWorld_Task(void *arg)
{
    while (1)
    {
        printf("Task running: Hello World ..\n");
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}

void Blinky_Task(void *arg)
{
    esp_rom_gpio_pad_select_gpio(BLINK_GPIO);
    gpio_set_direction(BLINK_GPIO, GPIO_MODE_OUTPUT);

    int count_second = 0;

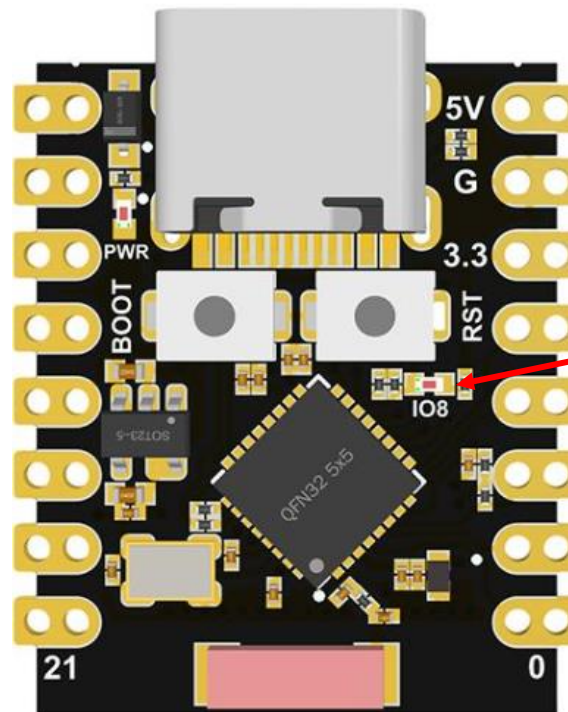
    while (1)
    {
```

```
    count_second += 1;

    switch (count_second)
    {
    case 10:
        vTaskSuspend(HelloWorldTaskHandle);
        printf("HelloWorld task suspended .. \n");
        break;
    case 14:
        vTaskResume(HelloWorldTaskHandle);
        printf("HelloWorld task resumed .. \n");
        break;
    case 20:
        vTaskDelete(HelloWorldTaskHandle);
        printf("HelloWorld task deleted .. \n");
        break;
    default:
        break;
    }
    gpio_set_level(BLINK_GPIO, 1);
    vTaskDelay(1000 / portTICK_PERIOD_MS);
    gpio_set_level(BLINK_GPIO, 0);
    vTaskDelay(1000 / portTICK_PERIOD_MS);
}

void app_main(void)
{
    xTaskCreatePinnedToCore(Blinky_Task, "Blinky", 4096, NULL, 10, &BlinkyTaskHandle, 0); // Core 0
    xTaskCreatePinnedToCore(HelloWorld_Task, "HelloWorld", 4096, NULL, 10, &HelloWorldTaskHandle, 1); // Core 1
}
```

Bước 3: Build project và nạp chương trình vào vi điều khiển, sau đó quan sát trạng thái của bóng đèn LED trên board



Quan sát đèn Led

