ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

——————— * ———————

PHÁT TRIỂN ỨNG DỤNG INTERNET OF THINGS THỰC HÀNH
(CO3038)

LAB05 Kết nối và điều khiển thiết bị trên Adafruit IO bằng chương trình di động

BỘ MÔN: KỸ THUẬT MÁY TÍNH
GVHD: VŨ TRỌNG THIÊN
NHÓM LỚP: L01
—o0o—
SVTH: MAI THỊNH PHÁT (1914590)

TP. HỒ CHÍ MINH, 4/2024

Mục lục

Danh sách hình vẽ

Chương 1

Tổng quan

1.1 Giới thiệu sơ lược về mục đích của bài lab này

Kết nối và điều khiển các thiết bị thông qua internet đã trở thành một xu hướng không thể phủ nhận trong thế giới công nghệ ngày nay. Adafruit IO, một trong những nền tảng IoT server phổ biến, cung cấp một cách tiếp cận linh hoạt và dễ dàng để quản lý các thiết bị từ xa. Thế nên việc tạo ra ứng dụng di động để kết nối và điều khiển các thiết bị từ xa trở nên vô cùng đơn giản và nhanh chóng. Chính vì thế trong bài lab này sẽ trình bày cách tạo ra một chương trình di động đơn giản kết nối đến các kênh thông tin của Adafruit IO để có thể gửi nhận và điều kiển các thiết bị từ xa.

1.2 Cấu trúc bài báo cáo

Cấu trúc của bài báo cáo này gồm có các chương sau:

- **Chương 1 Tổng quan**: Giới thiệu về bài thực hành.
- **Chương 2 Chuẩn bị các thành phần liên quan** : tiến hành cài đặt thiết bị ảo để lập trình
- **Chương 3 Hiện thực chương trình:** Trình bày các bước thực hiện chương trình

Chương 2

Chuẩn bị các thành phần liên quan

2.1 Các bước để tạo chương trình trên Androi Studio

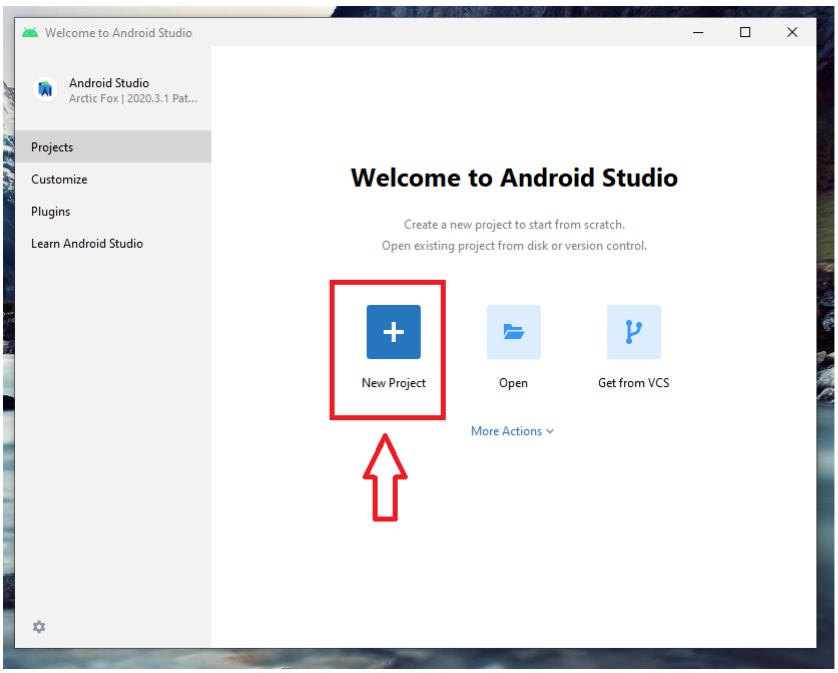
Để có thể hiện thực giao diện trên androi Studio chúng ta cần phải tải vào cài đặt ứng dụng Androi Studio vào máy tính. Đường dẫn để file cài đặt ở đây: `https://developer.android.com/studio`

Để tạo một project về giao diện trên Androi Studio ta là theo các bước sau:

1. Khởi tạo Project
2. Chọn thiết bị di động để chạy giao diện.
3. tiến hành viết mã giao diện

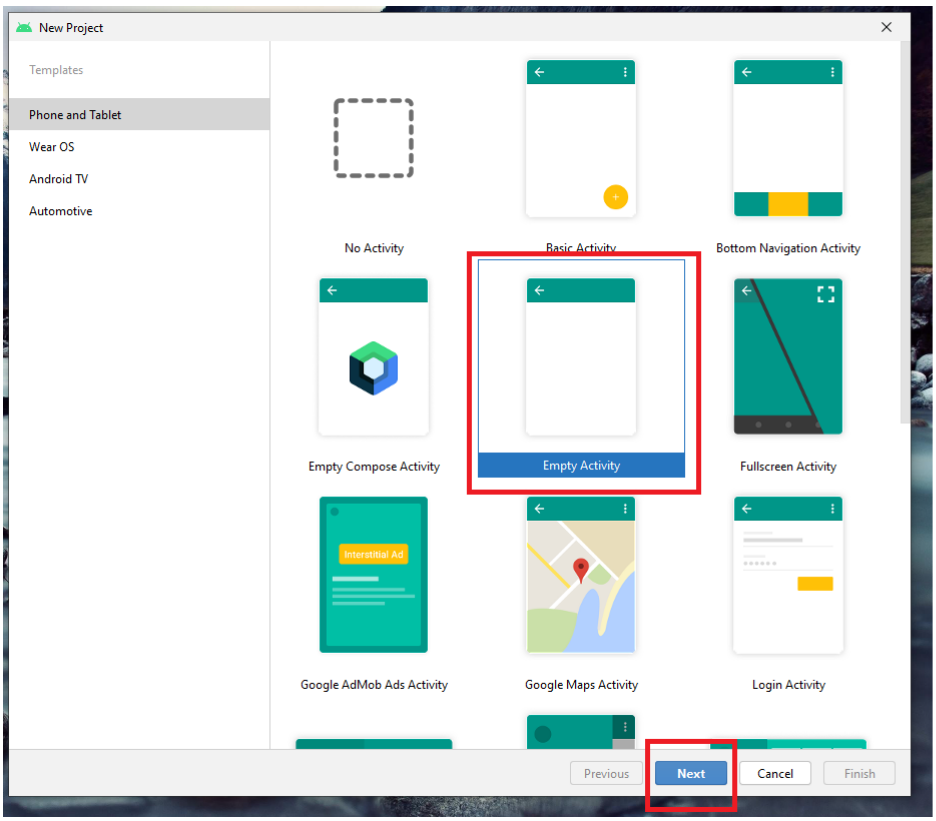
2.2 Khởi tạo Project

Để khởi tạo project ta sẽ ở ứng dụng Androi Studio lên và chọn **New Project**



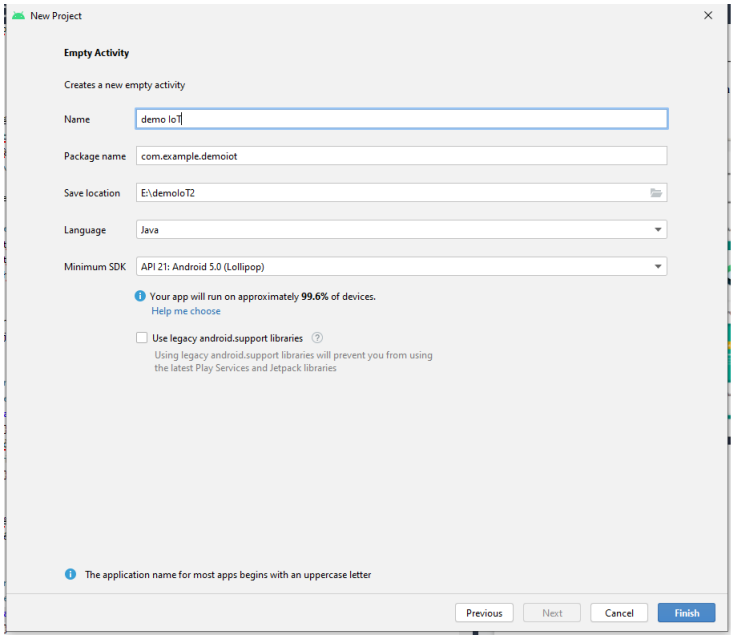
Hình 2.1: Khởi tạo project 1

Sau đó ta sẽ tiến hành chọn **Empty Activity** để tạo một project trống vào chọn **next** để tiếp tục:



Hình 2.2: Khởi tạo project 2

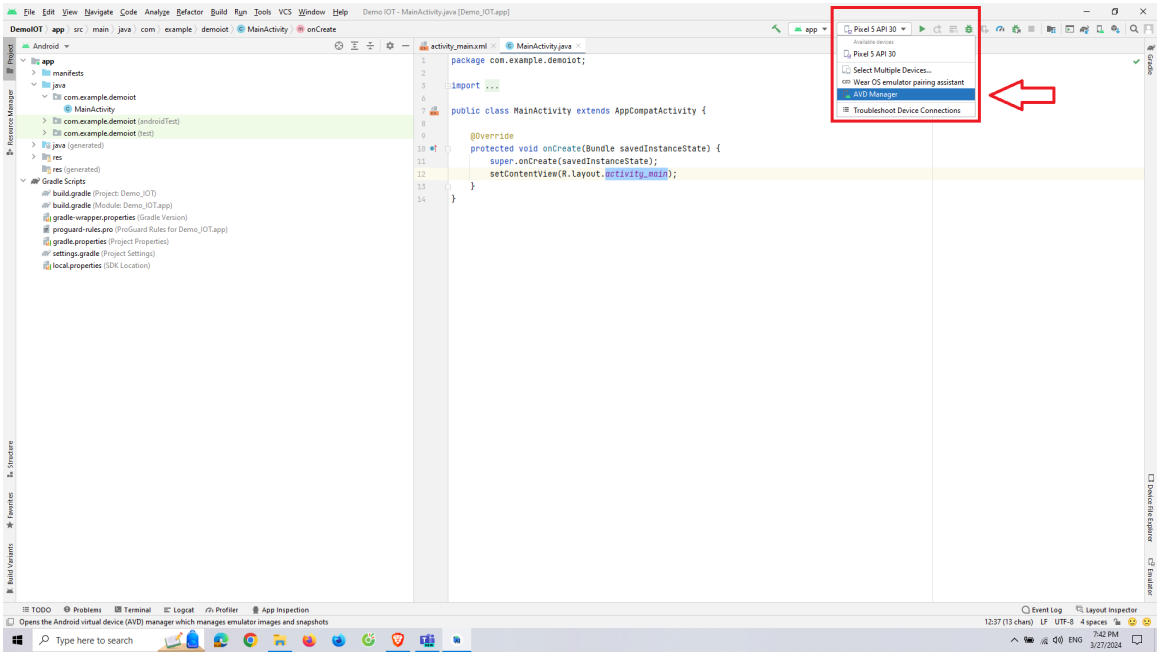
Tiếp theo đó ta sẽ tiến hành nhập các thông tin cần thiết về project:



Hình 2.3: Nhập các thông tin cho Project

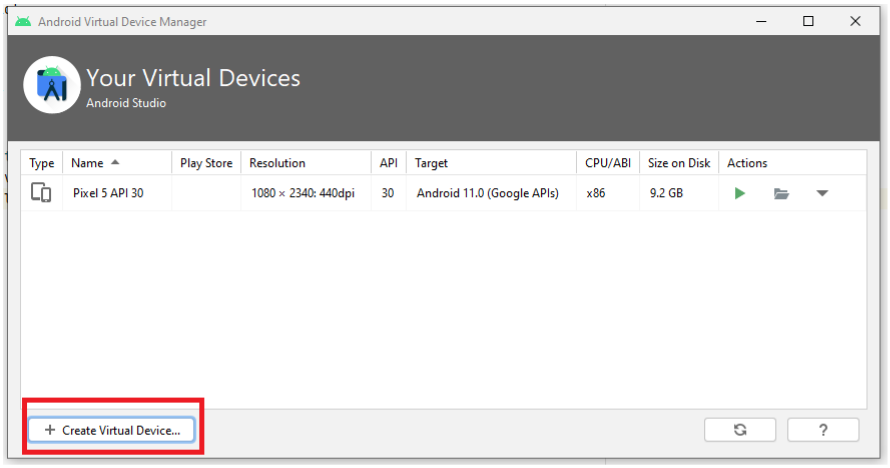
2.3 Tiến hành chọn thiết bị di động để chạy giao diện

Để tiến hành tạo thiết bị ảo trên ứng dụng Androi Studio ta cho **Available Device** sau đó chọn **AVD Manager** để hiển thị danh sách thiết bị.

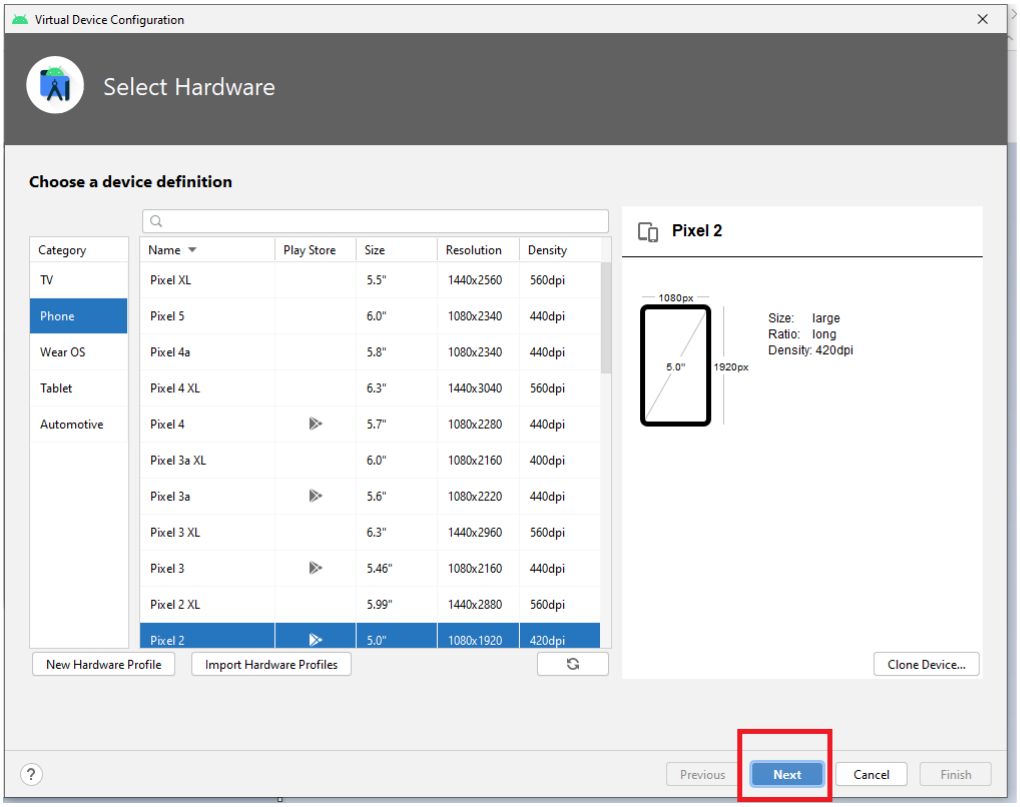


Hình 2.4: Mở cửa sổ quản lý thiết bị

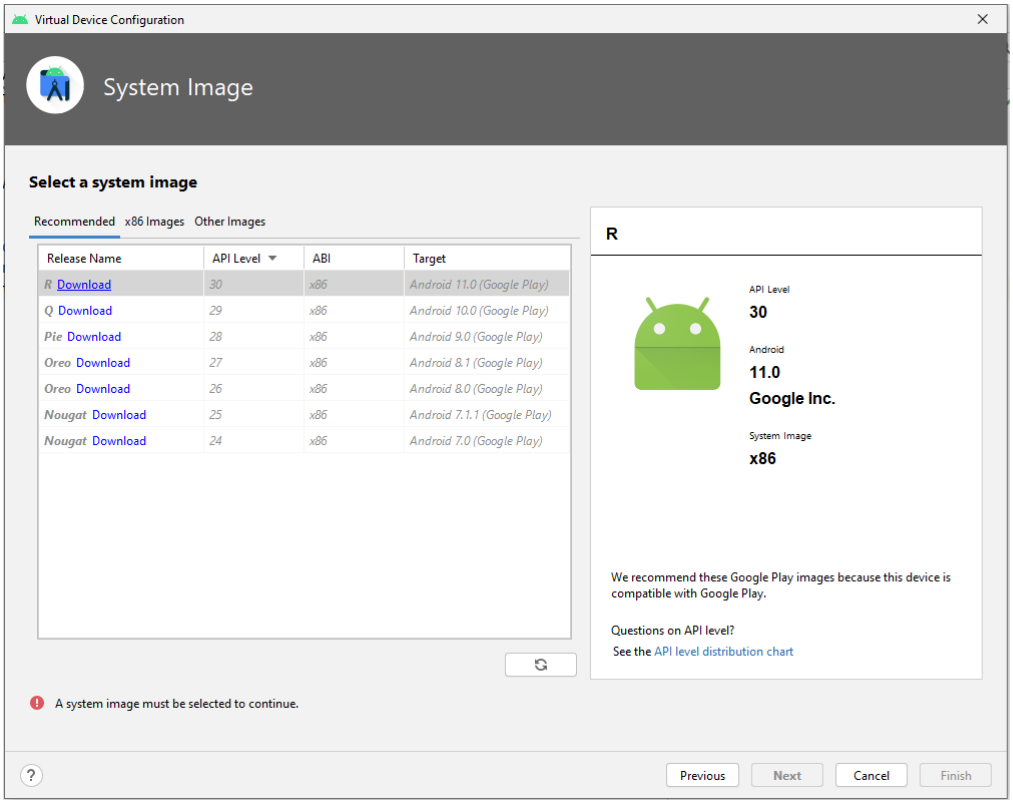
Sau đó chọn **create virtual device** và chọn thiết bị phù hợp muốn khởi tạo rồi bấm **next** để tiếp tục chọn hệ điều hành cho thiết bị



Hình 2.5: Chọn thiết bị phần cứng



Hình 2.6: Chọn thiết bị phần cứng



Hình 2.7: Chọn hệ điều hành cho máy

Chương 3

Hiện thực chương trình

3.1 Kết nối MQTT với Adafruit IO trên Androi Studio

3.1.1 Cài đặt

1. chỉnh sửa trong file *build.gradle* như sau:

```
android {
    ...
    defaultConfig {
        ...
        minSdk 21
        targetSdk 29
        ...
    }
    ...
}
dependencies {
    ...
    implementation 'org.eclipse.paho:org.eclipse.paho.client.
       mqttv3:1.1.0'
    implementation 'org.eclipse.paho:org.eclipse.paho.android
       .service:1.1.1'
    implementation 'com.github.angads25:toggle:1.1.0'
}
```

2. Chỉnh sửa trong file *settings.gradle*

```
dependencyResolutionManagement {
    ...
    repositories {
        ...
        maven{
            url "https://repo.eclipse.org/content/
               repositories/paho-snapshots/"
        }
    }
}
```

3. Cấp quyền xin cấp quyền cho ứng dụng android trong file *AndroidManifest.xml*

```
<uses-permission android:name="android.permission.
   WAKE_LOCK" />
<uses-permission android:name="android.permission.
   INTERNET" />

<uses-permission android:name="android.permission.
   CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.
   ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.
   ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.
   ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.
   ACCESS_FINE_LOCATION" />
```

4. Khai báo **mqtt** service trong file *AndroidManifest.xml*

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

3.1.2 Kết nối ứng dụng với MQTT

Để kết nối với Adafruit IO server thông qua giao thức mqtt đầu tiên ta sẽ tạo một lớp có tên là **MQTTHelper** để quản lý việc kết nối và giao tiếp với server Adafruit IO

```
public class MQTTHelper {
    public MqttAndroidClient mqttAndroidClient;

    public final String[] arrayTopics = {"feed_id_1", "feed_id_2",
        "feed_id_3", "nutnhan1","nutnhan2"};

    final String clientId = "123456789";
    final String username = "tai khoan";
    final String password = "mat khau";

    final String serverUri = "tcp://io.adafruit.com:1883";

    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() {
        for(int i = 0; i < arrayTopics.length; i++) {
            try {
                mqttAndroidClient.subscribe(arrayTopics[i], 0,
                   null, new IMqttActionListener() {
                    @Override
                    public void onSuccess(IMqttToken
                        asyncActionToken) {
                        Log.d("TEST", "Subscribed!");
                    }

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

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

}
```

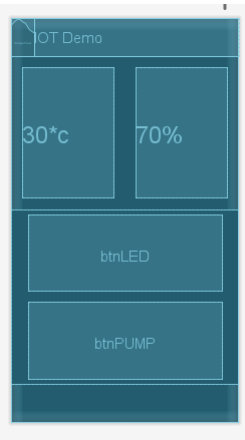
Ở trong đoạn mã trên có biến:

- **username** dùng để lưu tài khoản Adafruit IO,
- **password** chứa thông tin về mật khẩu của tài khoản
- **arrayTopics** chưa danh sách các topic mà ta cần subcribe vào

3.2 Giao diện trên Android Studio

3.2.1 Chia bố cục giao diện

Để hiện thực giao diện trên thiết bị di động ta sẽ tiến hành soạn thảo mã giao diện trong file **activity_main.xml**. Giao diện ứng dụng sẽ được chia bố cục như sau:



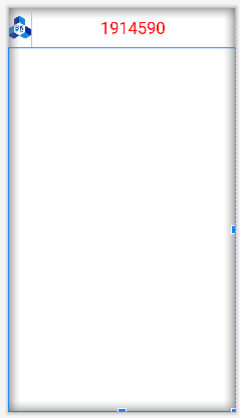
Hình 3.1: Bố cục giao diện ứng dụng

Đây là đoạn mã chia bố cục giao diện ứng dụng:

```
<LinearLayout 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"
     android:orientation="vertical" >

     <LinearLayout
         android:layout_width="match_parent"
         android:layout_height="0dp"
         android:layout_weight="10">
     </LinearLayout>

     <LinearLayout
         android:layout_width="match_parent"
         android:layout_height="0dp"
         android:layout_weight="40"
         android:orientation="horizontal">

         <LinearLayout
             android:layout_width="0dp"
             android:layout_height="match_parent"
             android:layout_weight="1" />

         <LinearLayout
```

```
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="20"></LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="20"></LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="10"></LinearLayout>

</LinearLayout>
```

Tiếp theo để hiện thực phần text và Logo ở vị trí trên cùng như hình sau:



Hình 3.2: header của giao diện

ta sẽ thêm thẻ **<ImageView>** như sau để hiển thị logo:

```
        <ImageView
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:src="@drawable/logo"
            />
```

Thẻ **<TextView>** để hiện thị đoạn text:

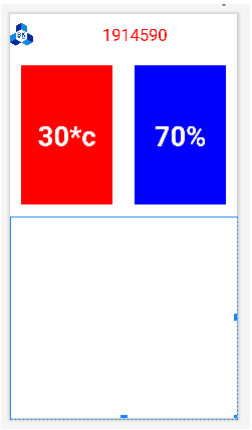
```
        <TextView
            android:layout_width="0dp"
            android:layout_height="match_parent"
```

```
            android:layout_weight="9"
            android:text="1914590"
            android:gravity="center"
            android:textColor="#ff0000"
            android:textSize="30dp"
            />
```

Để hiện thị 2 ô xuất kết quả nhiệt độ và độ ẩm ta sẽ sử dụng 2 thẻ **<TextView>** được lồng vào bên trong thẻ **<LinearLayout>** với thuộc tính *orientation="horizontal"* để hiện thị trên cùng một hàng ngang:

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="40"
    android:orientation="horizontal">
    <TextView
        android:id="@+id/txtTemperature"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="#ff0000"
        android:text="30*c"
        android:textSize="50dp"
        android:textStyle="bold"
        android:textColor="#FFFFFF"
        android:gravity="center"
        android:layout_margin="20dp"/>

    <TextView
        android:id="@+id/txtHumidity"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="#0000ff"
        android:text="70%"
        android:textSize="50dp"
        android:textStyle="bold"
        android:textColor="#FFFFFF"
        android:gravity="center"
        android:layout_margin="20dp"/>
</LinearLayout>
```

Ở đoạn mã trên có 2 thẻ **<TextView>** mỗi thẻ có 1 thuộc tính **id** để phân biệt đồng thời làm khóa để tham chiếu đế ở bước tới ở bước sau. Dưới đây là kết quả sau khi đã hiện thực giao diện cho thông tin về nhiệt độ và độ ẩm:



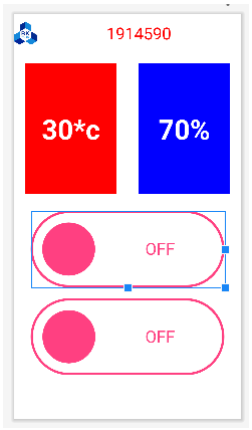
Hình 3.3: Giao diện khi có cửa sổ hiện thị nhiệt độ và đổ ẩm.

Cuối cùng để hiện thị giao diện hoàn chỉnh ta chỉ cần thêm 2 nút bấm để điều khiển máy đèn và máy bơm. Dưới đây là đoạn mã cho để hiện thị nút nhấn:

```
<com.github.angads25.toggle.widget.LabeledSwitch
    android:id="@+id/............................"

    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="20"
    android:layout_marginLeft="30dp"
    android:layout_marginTop="10dp"
    android:layout_marginRight="30dp"
    android:layout_marginBottom="10dp"
    android:textSize="30dp"
    app:textOn="ON"
    app:textOff="OFF"
    app:colorBorder="@color/colorAccent"
    app:on="false" />
```

Ta nhớ điền các giá trị cho thuộc tính **id** trong giao diện nút nhấn để phân biệt nút nhấn cho máy bơm và đèn.

Cuối cùng ta sẽ thu được giao diện sau:



Hình 3.4: Giao diện hoàn chỉnh

3.3 Hiện thực giao diện động khi có sự kiện

Để giao diện ứng dụng có thể thay đổi theo sự kiện chúng ta sẽ thực hiện theo các bước sau:

1. Tạo các biến tham chiếu đến các thành phần giao diện cần thay đổi khi có sự kiện xảy ra.
2. Viết các hàm thay đổi giao diện khi có sự kiện xảy ra

3.3.1 Tham chiếu đến giao diện

Để có thể thay đổi giá trị của các thành phần giao diện khi sự kiện xảy ra ta sẽ phải tạo ra các con trỏ tham chiếu trực tiếp đến các thành phần giao diện đó. Trong bài thực hành này chúng ta sẽ điều khiển nhiệt độ, độ ẩm, máy bơm và bóng đèn chính vì thế ta sẽ tạo ra các con trỏ tương ứng cho các thành phần này như sau:

```
TextView txtTemp, txtHumi;
LabeledSwitch btnLED,btnPUMP;

txtTemp = findViewById(R.id.txtTemperature);
txtHumi = findViewById(R.id.txtHumidity);
btnLED = findViewById(R.id.btnLED);
btnPUMP = findViewById(R.id.btnPUMP);
```

Trong đó các biến:

- **txtTemp**: con trỏ tham chiếu đến thành phần giao diện hiển thị nhiệt độ
- **txtHumi**: con trỏ tham chiếu đến thành phần giao diện hiển thị độ ẩm
- **btnLED**: con trỏ tham chiếu đến nút nhấn điều khiển đèn
- **btnPUMP**: con trỏ tham chiếu đến nút nhấn điều khiển máy bơm

3.3.2 Hàm callback cho sự kiện

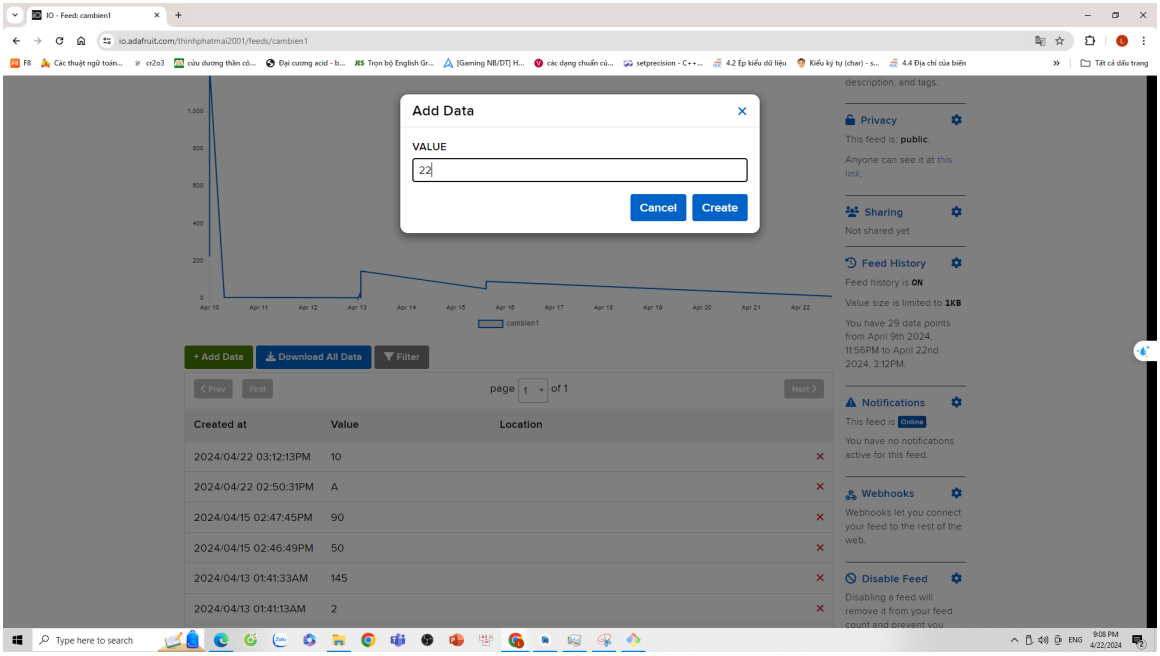
Khi có một sự khiện xảy ra hàm **messageArrived** của đối tượng thuộc lớp **MQTTHelper** sẽ được gọi. Chính vì thế ta sẽ hiện thực đoạn mã thay đổi trạng thái giao diện khi có sự kiện xảy ra ở trong hàm này. Cụ thể là cập nhật giao diện hiển thị nhiệt độ khi có nhiệt độ mới được gửi thông qua topic có tên là **cambien1**, cập nhật giao diện hiển thị độ ẩm khi có thông tin về độ ẩm mới được gửi thông qua topic có tên là **cambien2**, thay đổi trạng thái của nút nhấn khi nút nhấn trên server thay đổi và cuối cũng là gửi thông tin mới nhất về trạng thái của nút nhấn khi nó được thay đổi trên ứng dụng. dưới đây là đoạn mã làm các công việc trên:

```
btnLED.setOnToggledListener(new OnToggledListener() {
    @Override
    public void onSwitched(ToggleableView toggleableView, boolean
       isOn) {
        if(isOn == true){
            sendDataMQTT("thinhphatmai2001/feeds/nutnhan1","1");
        }else{
            sendDataMQTT("thinhphatmai2001/feeds/nutnhan1","0");
        }
    }
```

```
});
btnPUMP.setOnToggledListener(new OnToggledListener() {
    @Override
    public void onSwitched(ToggleableView toggleableView, boolean
       isOn) {
        if(isOn == true){
            sendDataMQTT("thinhphatmai2001/feeds/nutnhan2","1");
        }else{
            sendDataMQTT("thinhphatmai2001/feeds/nutnhan2","0");
        }
    }
});

@Override
public void messageArrived(String topic, MqttMessage message)
   throws Exception {
    Log.d("TEST", topic + " --- " + message.toString());
    if(topic.contains("cambien1")){
        txtTemp.setText(message.toString()+"*C");
    } else if(topic.contains("cambien2")){
        txtHumi.setText(message.toString()+"%");
    }else if(topic.contains("nutnhan1")){
        if(message.toString().equals("1")){
            btnLED.setOn(true);
        }else{
            btnLED.setOn(false);
        }
    }else if(topic.contains("nutnhan2")){
        if(message.toString().equals("1")){
            btnPUMP.setOn(true);
        }else{
            btnPUMP.setOn(false);
        }
    }
}
```

3.4 Demo kết quả

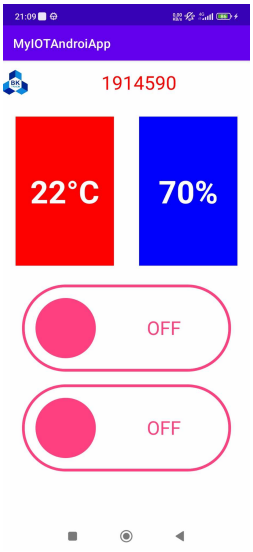
Khi thay đổi giá trị nhiệt độ là 22 trên server Adafruit thì ngay ngay sau đó hàm lắng nghe trạng thái sẽ được gọi và thông báo ra màn hình log đồng thời giao diện phía ứng dụng cũng thay đổi theo:



Hình 3.5: Thay đổi thông tin nhiệt độ

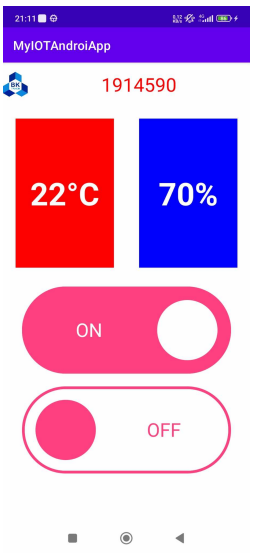
```
2024-04-22 21:09:14.478 21360-21360/com.example.myiotandroiapp D/TEST: thinhphatmai2001/feeds/cambien1 --- 22
```

Hình 3.6: Hàm lắng nghe sự kiện nhiệt độ

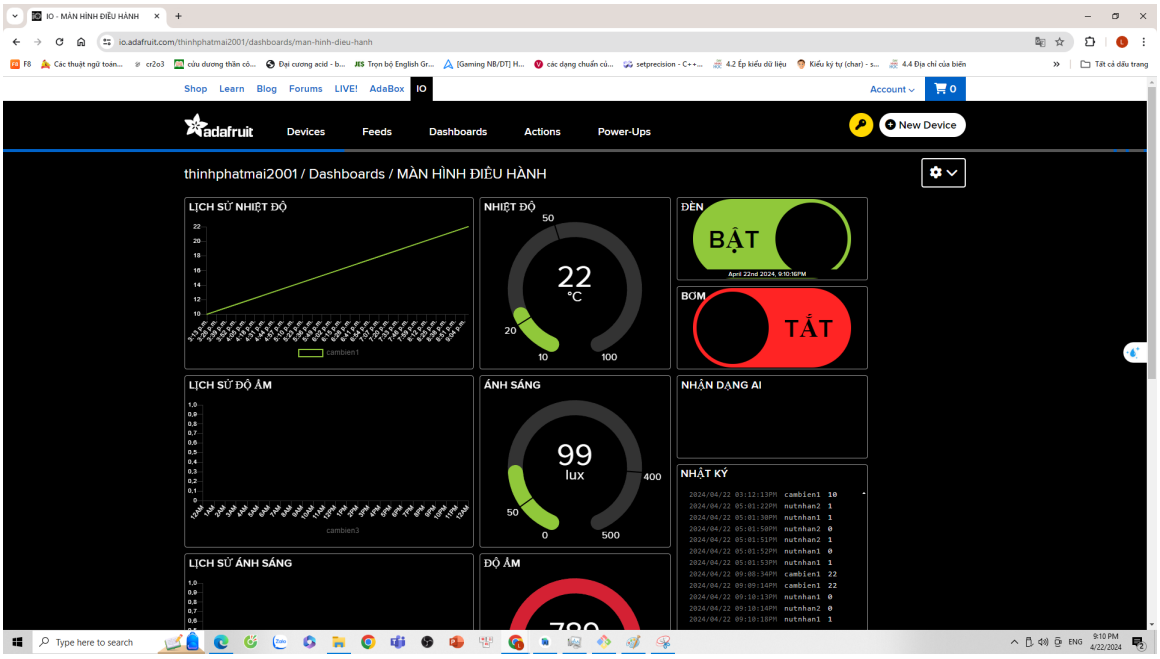


Hình 3.7: giao diện được cập nhật khi nhiệt độ thay đổi

Khi giao diện nút nhất phía ứng dụng thay đổi thì đồng thời giao diện trên server Adafruit IO cũng được đồng bộ theo



Hình 3.8: Thay đổi giao diện nút nhất



Hình 3.9: giao diện nút nhất trên server cũng thay đổi theo

3.5 Repo mã nguồn

Mã nguồn và báo cáo đều được lưu trữ tại: `https://github.com/LeoPkm2-1/Lab_IOT_-CO3038-/tree/master/IOT_LAB_FOR_WINDOWNS/MyIOTAndroiApp`

Tài liệu tham khảo

[1] Develop for Android
https://developer.android.com/develop [Truy cập: 20-02-2024]

[2] android-toggle
https://github.com/singhangadin/android-toggle

Adafruit IO API Cookbook

[3] Adafruit IO API Cookbook
https://io.adafruit.com/api/docs/cookbook.html#adafruit-io-api-cookbook