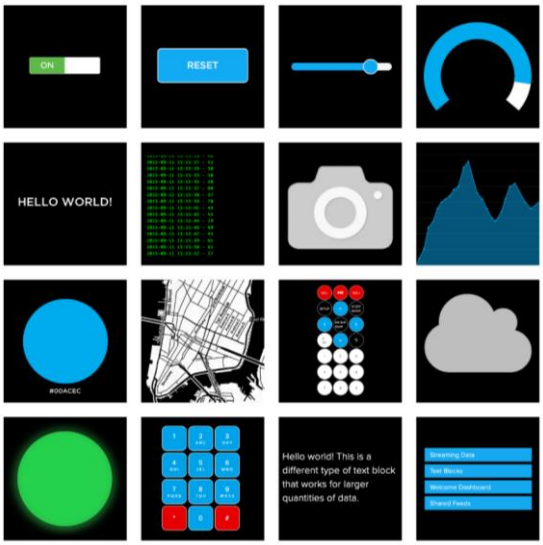
Phần 1: Xây dựng giao diện điều khiển thiết bị trên Adafruit IO

1. Giới thiệu

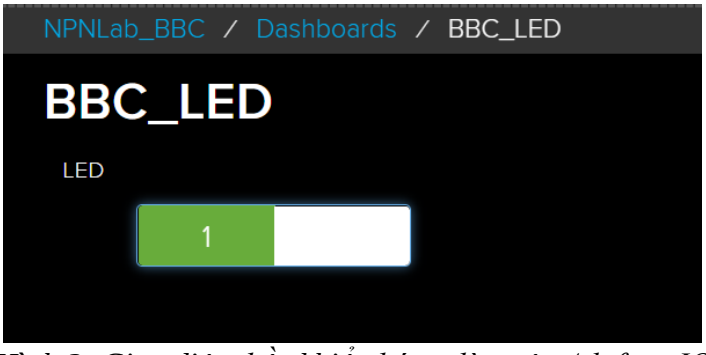
Trong bài này, chúng ta thậm chí còn có thể điều khiển mạch Microbit từ xa nhờ vào sự hỗ trợ của server Adafruit IO. Trang web này cùng cấp cho chúng ta rất nhiều phần tử giao diện đẹp để có thể thiết kết một màn hình thân thiện với người dùng. Chúng ta có thể dùng phần tử có 2 trạng thái ON và OFF để liên kết với một bóng đèn trên mạch Microbit và bật tắt nó từ xa một cách dễ dàng. Hình ảnh bên dưới là một số phần tử giao diện đang được hỗ trợ trên Adafruit IO và sẽ còn nhiều phần tử hơn nữa sẽ được mở rộng trong tương lai.



Hình 1: Các khối hỗ trợ tạo giao diện trên Adafruit IO

Khác với các server truyền thống của mạng Internet, các ứng dụng điều khiển thiết bị qua điện thoại thông minh sẽ bị một thời gian trễ nhất định. Server Adafruit IO được xây dựng theo một dạng dịch vụ trên mạng, nó tạo một liên kết ảo giữa 2 thiết bị đầu cuối, trường hợp này là điện thoại di động và mạch Microbit. Chính cơ chế này, còn gọi là MQTT, giúp cho việc gửi lệnh từ điện thoại tới mạch Microbit gần như là tức thời. Với các server truyền thống, muốn làm được việc này nó phải tốn rất nhiều tài nguyên của mạng, và trở nên bất hợp lý khi mở rộng với số lượng lớn. Nhờ cơ chế MQTT, mà các ứng dụng điều khiển thiết bị qua mạng Internet trở nên thực tế và khả thi hơn.

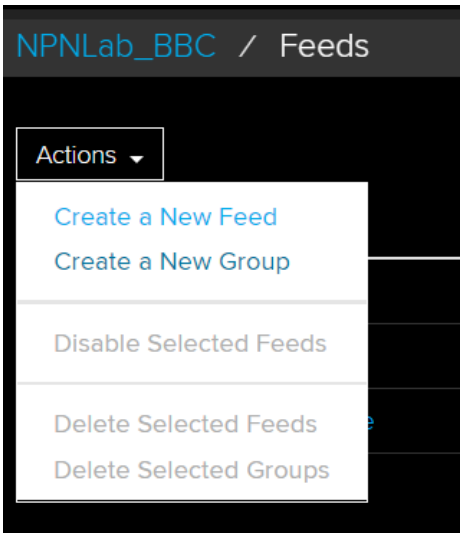
Trong bài này chúng ta sẽ xây dựng một giao diện để điều khiển một đèn LED trên mạch Microbit, giao diện mà chúng ta thiết kế sẽ như sau:



Hình 2: Giao diện điều khiển bóng đèn trên Adafruit IO

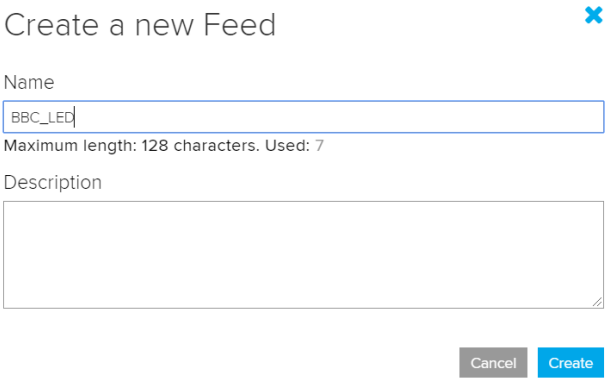
2. Tạo một feeds cho đèn LED

Trước tiên, chúng ta cần tạo một kênh dữ liệu riêng cho đèn LED này. Việc phân chia module tách biệt sẽ giúp chúng ta dễ quản lý hơn. Cách tạo một feed tương tự như ở Bài 7, bạn vào trang https://io.adafruit.com/, chọn mục Feeds > Actions và chọn Create a New Feed.



Hình 3: Tạo một kênh dữ liệu mới cho đèn LED

Giao diện sau đây sẽ hiện ra, bạn đặt tên cho nó là BBC_LED. Bạn phải ghi nhớ tên của kênh dữ liệu, vì sau đó bạn còn dùng nó để đăng kí trên điện thoại.

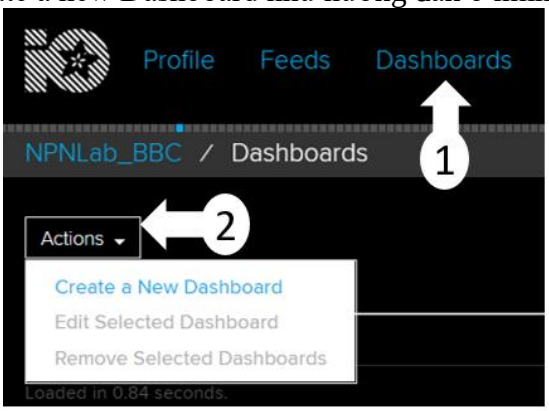


Hình 4: Đặt tên cho kênh dữ lệu, rồi nhấn Create

Chi tiết hơn, bạn có thể xem lại hướng dẫn ở Bài 7. Chúng ta có thể thấy rằng, Feeds là một dạng để lưu trữ dữ liệu lịch sử. Tất cả các lần gửi dữ liệu lên Feeds, nó đều lưu lại dưới dạng ngày tháng năm, cộng thêm 1 đồ thị để hiển thị đơn giản. Trong chuyên ngành máy tính, Feeds có thể xem là một dạng "back-end". Chính vì nó là dạng "back-end", một dạng xem dành cho người quản trị, nó không cần phải quá đẹp và thân thiện, miễn sao nó chính xác và không sót dữ liệu là được.

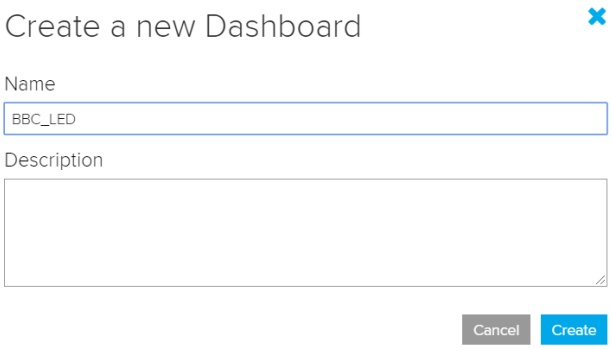
3. Tạo giao diện Dashboard

Ngược lại với Feeds, Dashboard có thể xem như một dạng "front-end", dùng để cho người sử dụng hệ thống sử dụng. Do đó, nó phải có giao diện đẹp mắt và dễ xài. Chúng ta bắt đầu tạo một Dashboard bằng cách vào tab Daskboard > Actions và chọn Create a new Dashboard như hướng dẫn ở hình bên dưới:



Hình 5: Tạo một Dashboard cho việc thiết kế giao diện

Một hộp hội thoại sẽ xuất hiện, cũng tương tự như khi tạo một Feeds, chúng ta cũng đặt tên cho Daskboard:

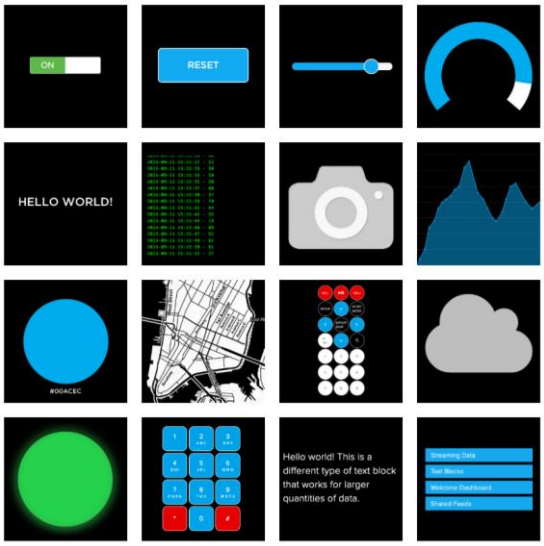


Hình 6: Đặt tên cho Dashboard

Bây giờ, chúng ta bắt đầu thiết kế các khối hiển thị trên Dashboard, bằng cách nhấn vào biểu tượng dấu +, như minh họa ở hình bên dưới:

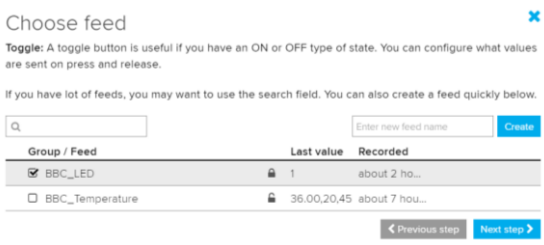
Hình 7: Tạo mới một khối trên Dashboard

Với rất nhiều các khối được hỗ trợ sẵn như minh họa ở hình bên dưới, chúng ta sẽ bắt đầu với khối đầu tiên, là Toggle. Khối này có thể dùng để mô phỏng cho bất kì thiết bị nào có 2 trạng thái, như công tác ON/OFF, bóng đèn sáng và tắt, và rất nhiều thiết bị khác nữa.



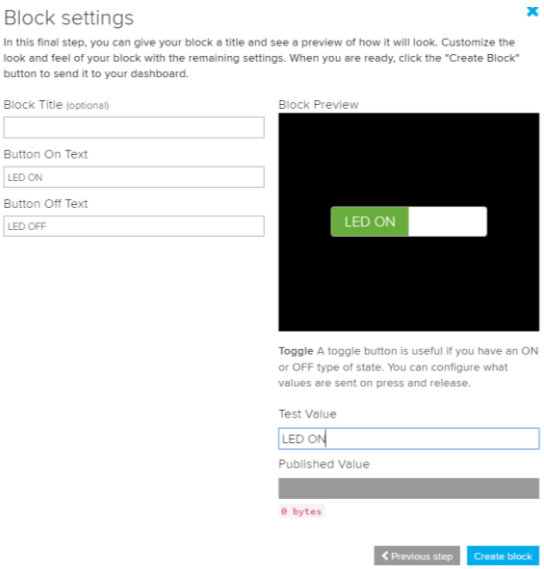
Hình 8: Các khối được hỗ trợ sẵn trên Adafruit IO

Chọn khối đầu tiên, giao diện sau đây sẽ hiển thị ra, chúng ta check vào BBC_LED, là kênh dữ liệu cho khối này:



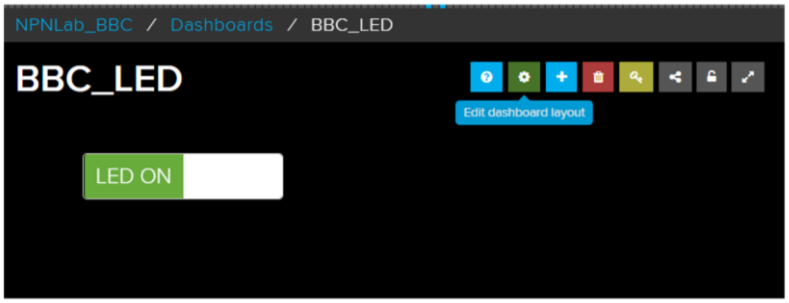
Hình 9: Chọn kênh dữ liệu cho khối

Sau đó, nhấn tiếp vào nút Next step để hiện ra giao diện sau đây:



Hình 10: Thiết lập dữ liệu cho khối

Việc thiết lập dữ liệu cho khối là khá quan trọng, ở 2 ô **Button On Text** và **Button Off Text**. Dữ liệu ở 2 trường này vừa phục vụ cho việc hiển thị, và là dữ liệu gửi xuống mạch Microbit. Trong trường hợp bạn muốn hiển thị được đồ thị bên kênh dữ liệu BBC_LED, thì dữ liệu ở đây phải là dạng số, 0 hoặc 1. Ở đây tôi xin chọn việc hiển thị là 1 và 0, tương ứng với 2 trạng thái của đèn LED. Cuối cùng, chúng ta nhấn vào Create Block. Giao diện hiển thị của chúng ta sẽ như sau:



Hình 11: Giao diện hiển thị trên Dashboard với khối Toggle

Nếu muốn thay đổi vị trí hiển thị, bạn có thể nhấn vào biểu tượng Edit dashboard layout, như minh họa ở hình trên, sau đó kéo thả nó tới vị trí mong muốn rồi nhấn nút SAVE hoặc nhấn CANCEL nếu muốn hủy bỏ.

Phần 2: Hướng dẫn kết nối IoT_Gateway và Server MQTT trên thiết bị Android

1 Thông tin tổng quan

1.1 Gửi lệnh từ ứng dụng trên di động xuống phần cứng

Trong trường hợp ứng dụng trên di động muốn điều khiển thiết bị phần cứng, ứng dụng trên di động chỉ cần pusblish 1 message lên server MQTT Broker. Ứng dụng trên di động có thể kiểm tra việc message này đã được published lên server hay chưa mà không cần quan tâm message đã thực sự tới phần cứng hay chưa.

Trong trường hợp muốn tăng xác suất message này tới thiết bị phần cứng, sinh viên chỉ cần chỉnh thông số QoS của message là đủ. Tuy nhiên, với QoS mặc định, xác suất message được gửi đi thành công là trên 90%.

1.2 IoT gateway gửi dữ liệu lên server

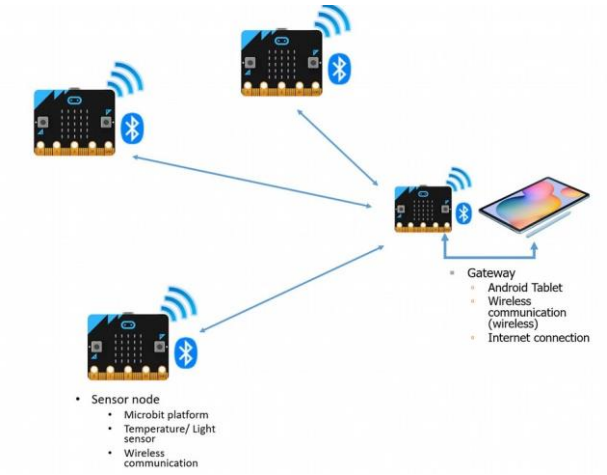
IoT gateway là một thiết bị nhúng xài hệ điều hành Android. Do đó, sinh viên hoàn toàn có thể sử dụng điện thoại Android để hiện thực chức năng của một Gateway mà không cần phải chờ đợi thiết bị phần cứng thật sự. Việc kết nối từ điện thoại ra các thiết bị cảm biến, giao tiếp không dây sẽ được trình bày ở phần hướng dẫn lập trình bên dưới .

2 Hướng dẫn lập trình

2.1 Mở rộng kết nối của điện thoại

Để có thể thực sự giao tiếp với các thiết bị IoT, điện thoại cần phải được mở rộng kết nối ra bên ngoài. Do đặc thù của 1 hệ thống có hệ điều hành, là không có khả năng xử lý thời gian thực. Do đó, chỉ cần kết nối thêm với một hệ thống vi điều khiển là được. Có rất nhiều hệ thống vi điều khiển thông dụng và rẻ tiền, Arduino là một ví dụ. Cuối cùng, giữa 1 hệ thống có hệ điều hành và hệ thống vi điều khiển, kết nối đơn giản nhất giữa chúng là USB UART.

Mô hình tổng quát của hệ thống cảm biến, giao tiếp không dây và kết nối với Gateway được mô tả như hình bên dưới:



Hình 12: Mô hình kết nối mạng cảm biến

Lý do mà điện thoại hoặc table phải kết nối thêm với 1 hệ thống vi điều khiển khác, là bản thân điện thoại không có các kênh giao tiếp không dây dựa trên sóng Radio, hoặc LoRa, vốn là các giao tiếp đặc thù trên mạng IoT. Điện thoại cần phải kết nối với phần cứng bên ngoài thông qua kết nối USB OTG như sau:

Hình 13: USB OTG mở rộng kết nối với điện thoại

Như vậy, mạch phần cứng nối với điện thoại, chỉ đóng vai trò là một thiết bị trung gian, chuyển hết dữ liệu mà nó nhận từ các nốt cảm biến khác lên điện thoại. Tất cả các tác vụ xử lý (upload server, publish message,...) sẽ được hiện thực trên điện thoại.

2.2 Lập trình trên mở rộng USB UART trên Android Studio

Để mở cổng kết nối USB UART trên di động hoặc tablet Android, sinh viên có thể tham khảo trên github sau đây

https://github.com/mik3y/usb-serial-for-android Phiên bản thu

gọn của project này, sinh viên có thể tải từ đường dẫn sau:

https://www.dropbox.com/s/4045vcf1a90mg3t/TerminalUart.zip?dl=1

Một số lỗi do không tương thích với Android Studio SDK tại máy của sinh viên, hoặc lỗi nhỏ do lỗi code bị dư, sinh viên chủ động sửa lỗi.

Các thông tin quan trọng cần chú ý như sau:

Thêm thư viện trong build.graddle (module app)

```
android {

   allprojects { repositories {
         maven { url 'https://jitpack.io' }
      }
   }
}

dependencies {
      implementation 'com.github.mik3y:usb-serial-for-android:2.+'
}
```

Thêm quyền truy cập USB trong AndroidManifest, lưu ý file device_filter (thư mục res/ xml/)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="uart.terminal.androidstudio.com.myapplication">
```

```
    <uses-feature android:name="android.hardware.usb.host" />
    <uses-permission android:name="android.permission.USB_PERMISSION" />

    <application

        <meta-data android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
          android:resource="@xml/device_filter" />
              </activity>
           </application>

</manifest>
```

Nội dung của device_filter, bao gồm các driver thông dụng của cổng USB UART:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- 0x0403 / 0x6001: FTDI FT232R UART -->
    <usb-device vendor-id="1027" product-id="24577" />

    <!-- 0x0403 / 0x6015: FTDI FT231X -->
    <usb-device vendor-id="1027" product-id="24597" />

    <!-- 0x2341 / Arduino -->
    <usb-device vendor-id="9025" />

    <!-- 0x16C0 / 0x0483: Teensyduino  -->
    <usb-device vendor-id="5824" product-id="1155" />

    <!-- 0x10C4 / 0xEA60: CP210x UART Bridge -->
    <usb-device vendor-id="4292" product-id="60000" />

    <!-- 0x067B / 0x2303: Prolific PL2303 -->
    <usb-device vendor-id="1659" product-id="8963" />

    <!-- 0x1a86 / 0x7523: Qinheng CH340 -->
    <usb-device vendor-id="6790" product-id="29987" /><usb-device vendor-id="1155" product-
    id="22352" />

    <usb-device vendor-id="8208" product-id="30264" />

    <usb-device vendor-id="3368" product-id="516" />

</resources>
```

Cấp quyền trong MainActivity.java

```
PendingIntent usbPermissionIntent = PendingIntent.getBroadcast(this, 0, new
Intent(INTENT_ACTION_GRANT_USB), 0);
manager.requestPermission(driver.getDevice(), usbPermissionIntent);

manager.requestPermission(driver.getDevice(), PendingIntent.getBroadcast(this, 0, new
Intent(ACTION_USB_PERMISSION), 0));
```

```
Nhận dữ liệu từ phần cứng @Override
public void onNewData(final byte[] data) {
   runOnUiThread(new Runnable() {
      @Override
      public void run() {
         txtOut.append(Arrays.toString(data));
                        }
                    });
}
```

2.3 Lập trình subscribe, publish dữ liệu lên server MQTT

Phần hướng dẫn này tập trung vào thiết bị Android, đóng vai trò là gateway trong hệ thống. Tuy nhiên, phần code này cũng có thể sử dụng để hiện thực ứng dụng giám sát, điều khiển từ xa từ điện thoại.

Phần hướng dẫn tương tác với server adafruit thông qua giao thức HTTP, các bạn có thể tham khảo tại link sau: https://io.adafruit.com/api/docs/#adafruit-io-http-api

Hai thuật ngữ mà bạn sẽ thường gặp khi làm việc với MQTT, được tóm tắt ngắn gọn dưới đây:

- **Publish, subscribe**

Trong một hệ thống sử dụng giao thức MQTT, nhiều node cảm biến (gọi là mqtt client - gọi tắt là client) kết nối tới một MQTT server (gọi là broker). Mỗi client sẽ đăng ký một hoặc nhiều kênh (topic), ví dụ như "/client1/channel1", "/client1/channel2". Quá trình đăng ký này gọi là "subscribe", giống như chúng ta đăng ký nhận tin trên một kênh Youtube vậy. Mỗi client sẽ nhận được dữ liệu khi bất kỳ trạm nào khác gởi dữ liệu và kênh đã đăng ký. Khi một client gởi dữ liệu tới kênh đó, gọi là "publish".

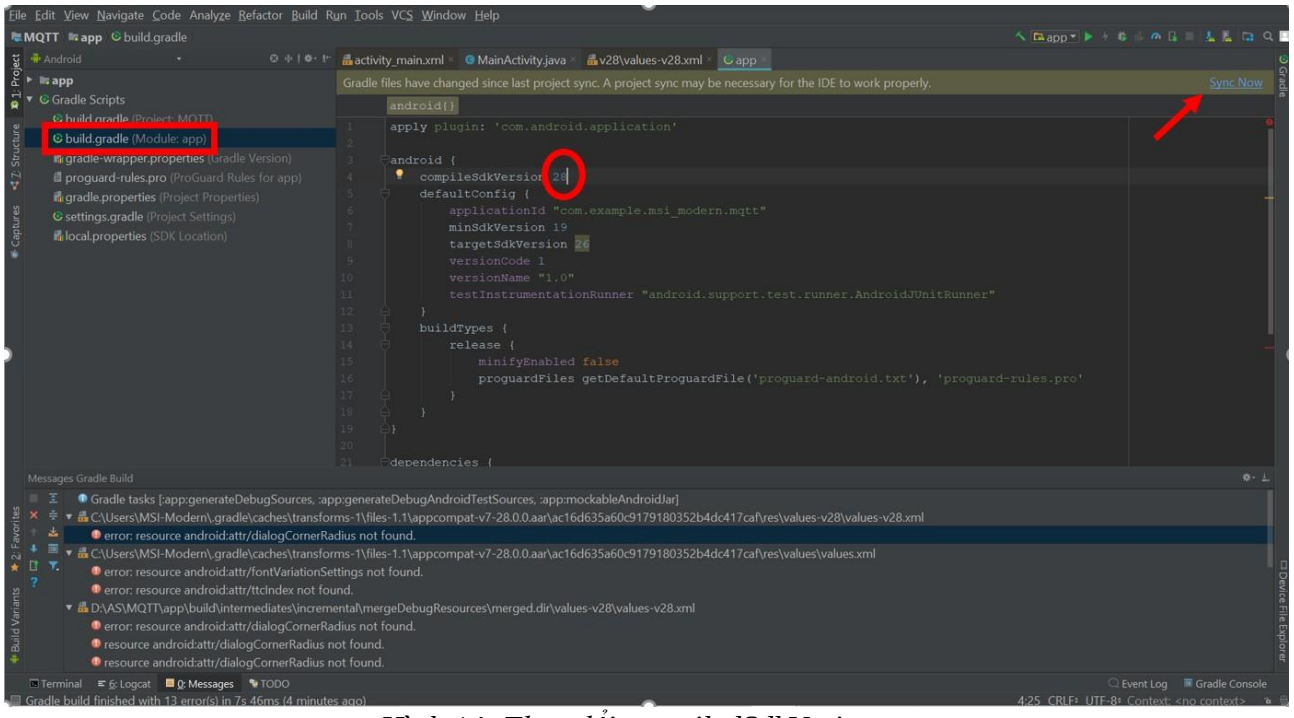
Phần tiếp theo, chúng tôi sẽ hướng dẫn bạn thông qua một dự án đơn giản, dùng giao thức MQTT kết nối với máy chủ Adafruit để gửi và thu thập dữ liệu cho ứng dụng Android của bạn.

- **build.gradle**

Lúc đầu sau khi bạn tạo dự án, bạn sẽ thấy dự án có một số lỗi. Chúng tôi sẽ sửa nó bằng cách vào tệp build.gradle trong thư mục Gradle Scripts. Lưu ý rằng bạn sẽ phát hiện ra có 2 trong số các tệp như vậy trong đó. Chúng tôi chỉ làm việc với một build.gradle (Module: app). Bây giờ chúng ta sẽ thay đổi compileSdkVersion từ 26 thành 28:

```
1       compileSdkVersion 28
```

Màn hình sẽ hiển thị như sau:



Hình 14: Thay đổi compiledSdkVerion

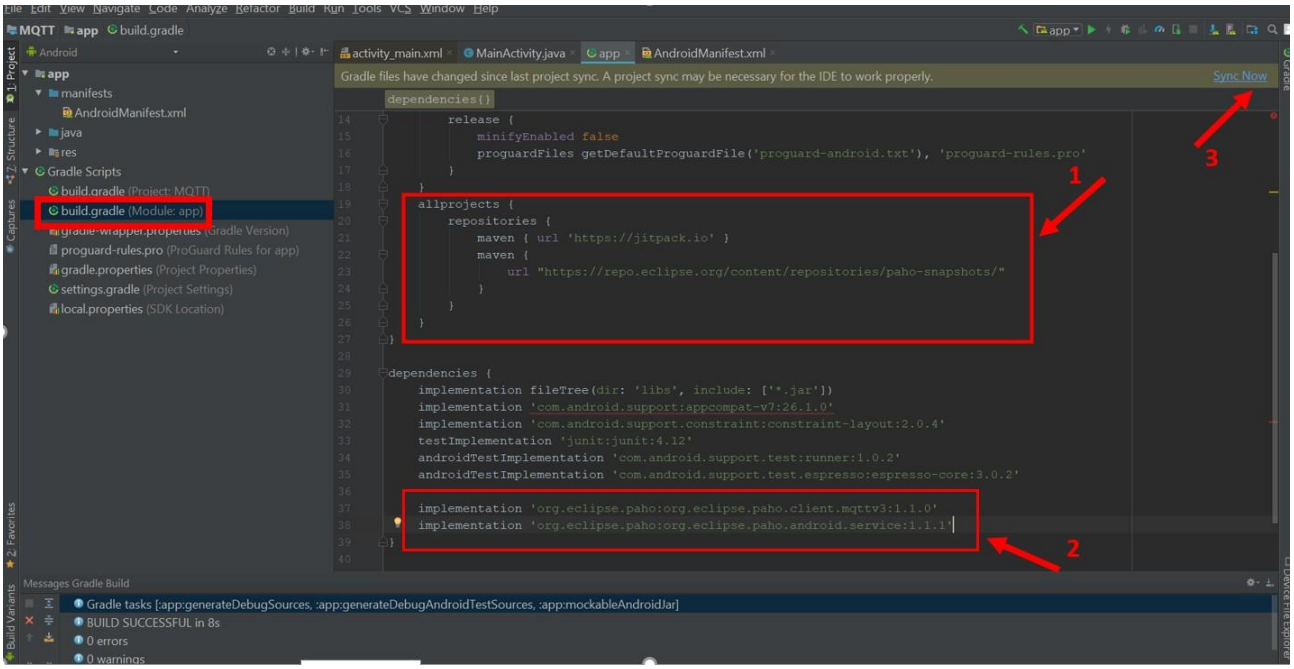
Sau đó bạn nhấn Sync Now, kết quả là Gradle Sync thành công và không xảy ra lỗi. Tuy nhiên, chúng tôi vẫn chưa hoàn thành, chúng tôi chỉ sửa phiên bản và chưa làm bất kỳ điều gì với MQTT. Do đó, trong cùng một tệp, hãy thêm các dòng này vào các phần phụ thuộc:

```
1    implementation'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.1.0'
2    implementation'org.eclipse.paho:org.eclipse.paho.android. service:1.1.1'
```

Các bạn tiếp tục thêm các đoạn lệnh sau vào project:

```
1    allprojects {
2        repositories {
3            maven { url'https://jitpack.io' }
4            maven {
5                url"https://repo.eclipse.org/content/repositories/
                     paho-snapshots/"
6            }
7        }
8    }
```

Màn hình kết quả sẽ được hiển thị như sau:



Hình 15: Thêm implementation và github link

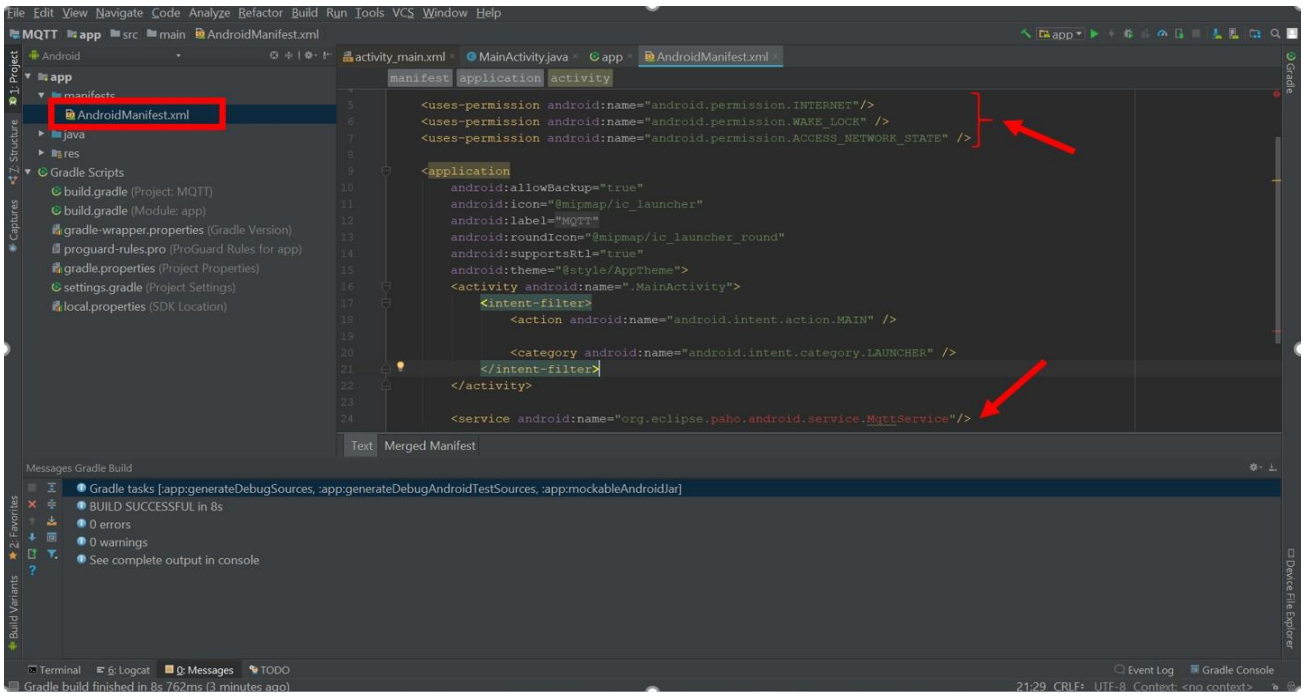
- **AndroidManifest.xml**

Bây giờ chúng ta sẽ chuyển sang tệp AndroidManifest.xml và thêm quyền cũng như dịch vụ. Các quyền được thêm vào phía trên thẻ ứng dụng và ở dạng các dòng mã sau:

```
1    <uses-permission android:name="android.permission.INTERNET"/>
2    <uses-permission android:name="android.permission.WAKE_LOCK"/>
3    <uses-permission android:name="android.permission.
         ACCESS_NETWORK_STATE"/>
```

Sau đó, dịch vụ MQTT sẽ được bao gồm trong thẻ ứng dụng. Đây là mã cho các dịch vụ MQTT:

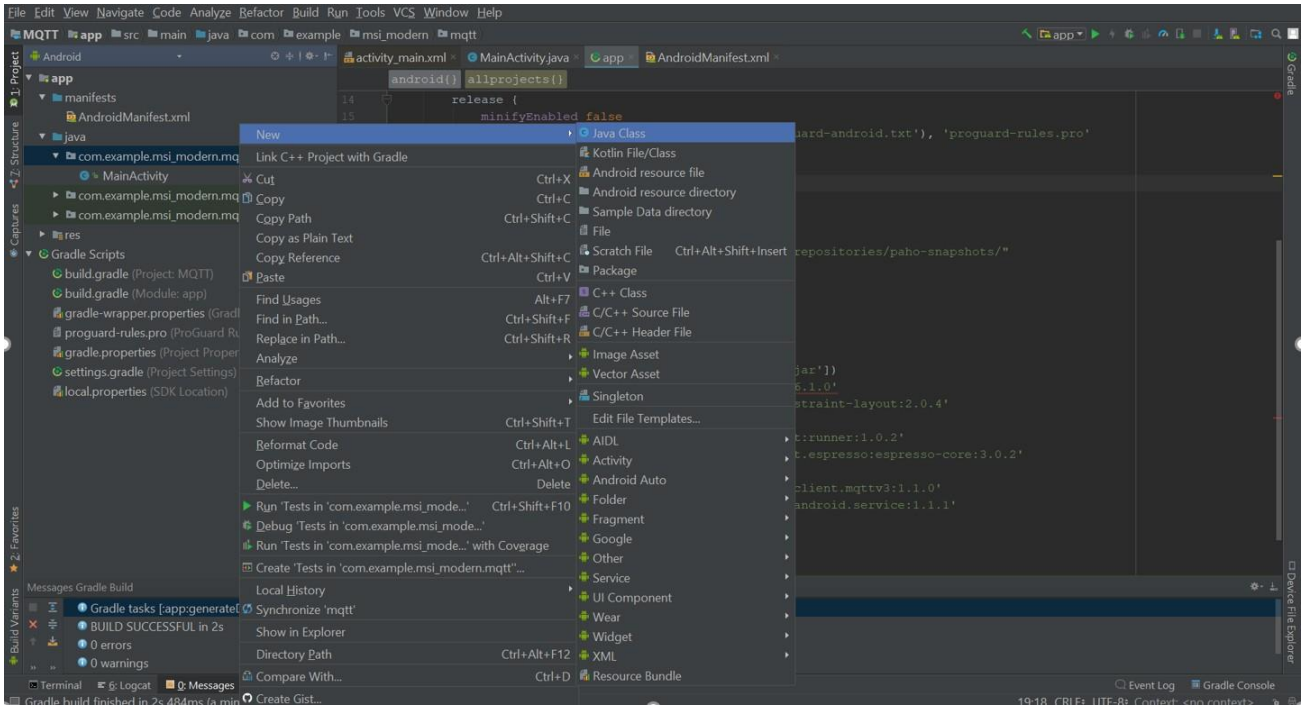
```
1        <serviceandroid:name="org.eclipse.paho.android.service.
             MqttService"/>
```



Hình 16: Thêm users-permission và service

MQTTService.java

Bây giờ bạn phải tạo một lớp mới trong cùng thư mục với MainActivity.java và đặt tên là MQTTService.java.



Hình 17: Các bước để tạo một lớp mới

Thay thế mọi thứ ngoại trừ dòng đầu tiên bằng các dòng sau:

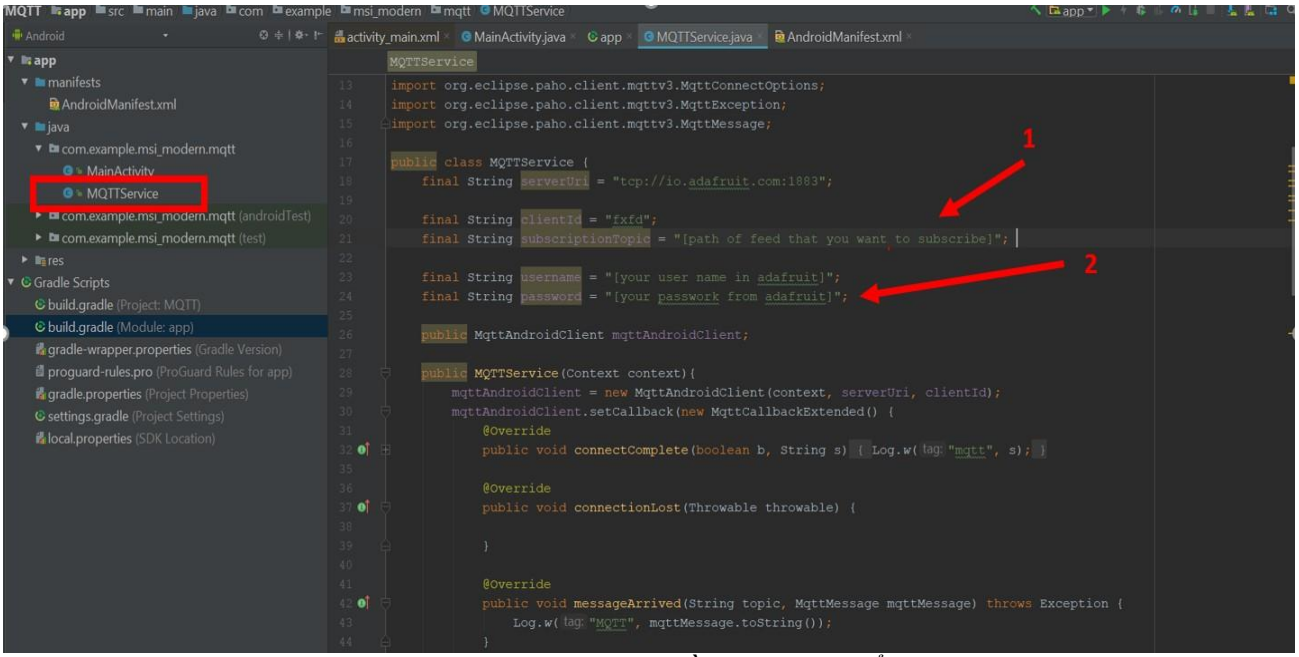
```
1  import android.content.Context;
2  import android.util.Log;
3
4  import org.eclipse.paho.android.service.MqttAndroidClient;
5  import org.eclipse.paho.client.mqttv3.DisconnectedBufferOptions;
6  import org.eclipse.paho.client.mqttv3.IMqttActionListener;
7  import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
8  import org.eclipse.paho.client.mqttv3.IMqttToken;
9  import org.eclipse.paho.client.mqttv3.MqttCallbackExtended;
10 import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
11 import org.eclipse.paho.client.mqttv3.MqttException;
12 import org.eclipse.paho.client.mqttv3.MqttMessage;
13
14 publicclass        MQTTService {

15          final    String    serverUri ="tcp://io.adafruit.com:1883";
16
17
18          final    String    clientId ="[YourclientID]";
19          final    String    subscriptionTopic ="[Yoursubscriptiontopic]";
20
```

```
21
22          final    String    username ="[Yourusername]";
23          final    String    password ="[Yourpassword]";
24
25      public MqttAndroidClient  mqttAndroidClient;
26
27      public MQTTService(Context context){
28          mqttAndroidClient = new MqttAndroidClient(context,
                serverUri, clientId);
29          mqttAndroidClient.setCallback(new MqttCallbackExtended()                {
30              @Override
31              publicvoid       connectComplete( boolean b, String s) {
32                  Log.w("mqtt", s);
33              }
34
35              @Override
36              publicvoid       connectionLost( Throwable throwable){
37
38              }
39
40              @Override
41              publicvoid       messageArrived(String topic, MqttMessage
                    mqttMessage) throws Exception  {
42                  Log.w("Mqtt", mqttMessage.toString());
43              }
44
45              @Override
46              publicvoid       deliveryComplete(IMqttDeliveryToken
                    iMqttDeliveryToken) {
47
48              }
49          });
50          connect();
51      }
52      publicvoid       setCallback( MqttCallbackExtended callback){
53          mqttAndroidClient.setCallback(callback);
54      }
55
56      privatevoid       connect(){
57          MqttConnectOptions mqttConnectOptions =  new
                MqttConnectOptions();
58          mqttConnectOptions.setAutomaticReconnect(true);
```

```
59          mqttConnectOptions.setCleanSession(false);
60          mqttConnectOptions.setUserName(username);
61          mqttConnectOptions.setPassword(password.toCharArray());
62
63          try {
64
65              mqttAndroidClient.connect(mqttConnectOptions, null, new
                    IMqttActionListener() {
66                  @Override
67                  publicvoid      onSuccess( IMqttTokenasyncActionToken)
                        {
68
69                      DisconnectedBufferOptions
                            disconnectedBufferOptions = new
                              DisconnectedBufferOptions();
70                          disconnectedBufferOptions.setBufferEnabled(true
                        );
71                      disconnectedBufferOptions.setBufferSize(100);
72                      disconnectedBufferOptions.setPersistBuffer(
                            false);
73                      disconnectedBufferOptions.
                              setDeleteOldestMessages(false);
74                      mqttAndroidClient.setBufferOpts(
                              disconnectedBufferOptions);
75                      subscribeToTopic();
76                  }
77
78                  @Override
79                  publicvoid      onFailure(IMqttToken asyncActionToken,
                        Throwable exception) {
80                      Log.w("Mqtt","Failedtoconnectto:"+
                        serverUri + exception. toString());
81                  }
82              });
83
84
85          } catch (MqttException ex) {
86              ex.printStackTrace();
87          }
88      }
89
```

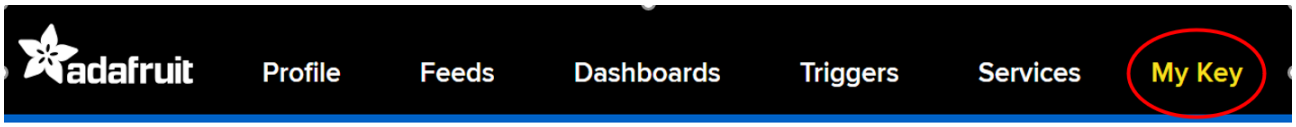
```
90        privatevoid        subscribeToTopic() {
91              try {
92                    mqttAndroidClient.subscribe(subscriptionTopic, 0, null, new
                          IMqttActionListener() {
93                          @Override
94                          publicvoid        onSuccess( IMqttTokenasyncActionToken)
                                {
95                              Log.w("Mqtt","Subscribed!");
96                          }
97
98                          @Override
99                          publicvoid        onFailure(IMqttToken asyncActionToken,
                              Throwable exception) {
100                             Log.w("Mqtt","Subscribedfail!");
101                         }
102                   });
103
104             } catch (MqttException ex) {
105                   System.err.println("Exceptionstsubscribing");
106                   ex.printStackTrace();
107             }
108       }
109
110 }
```

Hãy nhớ rằng thông tin máy chủ Adafruit phải được thay đổi thành các chủ đề và tài khoản mong muốn của bạn.



Hình 18: Các thông tin cần phải thay đổi

Thông tin Adafruit có thể được tìm thấy tại tab My Key trong chủ đề của bạn.



Hình 19: My Key tab

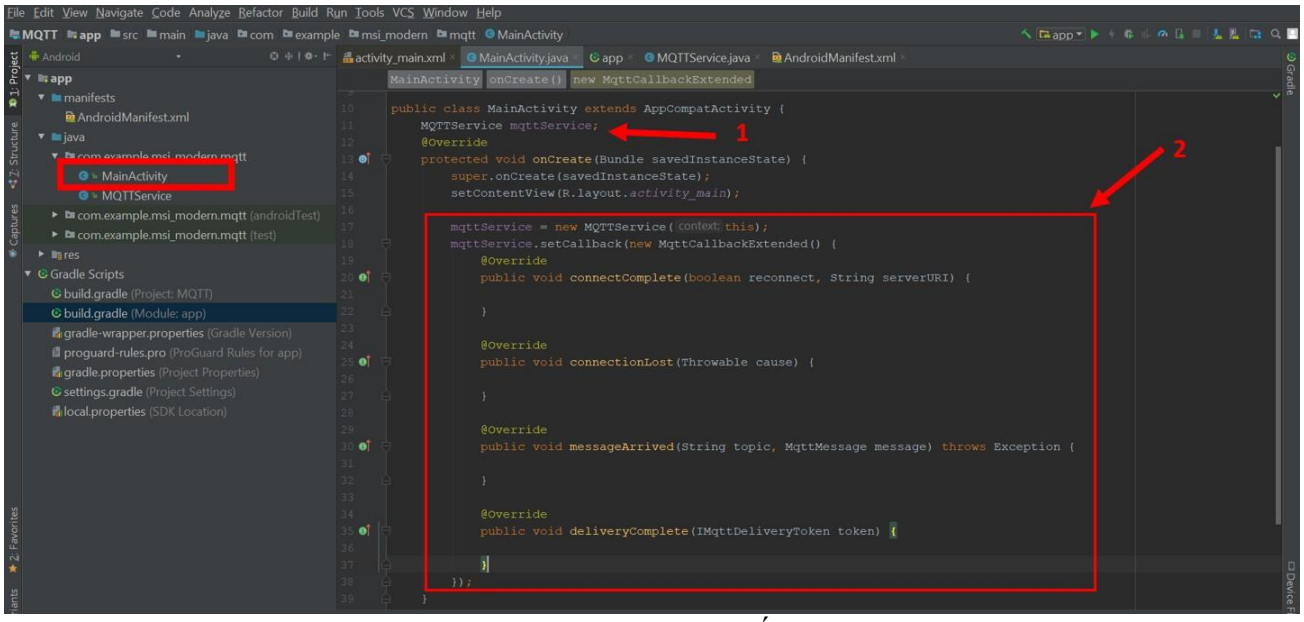
- **MainActivity.java**

Chúng ta đã gần hoàn tất, bước tiếp theo của chúng ta sẽ là trên tệp MainActivity.java. Thêm những thứ này vào phương thức onCreate:

```
1    MQTTService mqttService;
```

```
1        mqttService = new MQTTService( this);
2        mqttService.setCallback(new MqttCallbackExtended() {
3            @Override
4            publicvoid      connectComplete(boolean reconnect, String
                 serverURI) {
5
7
6            }

8            @Override
9            publicvoid      connectionLost( Throwable cause){
10
11           }
12
13           @Override
14           publicvoid      messageArrived(String topic, MqttMessage
                 message) throws Exception  {
15             String data_to_microbit =message. toString();
16             port.write(data_to_microbit.getBytes(),1000);
17           }

19           @Override
20           publicvoid      deliveryComplete( IMqttDeliveryTokentoken)
                 {
21
22           }
23       });
```

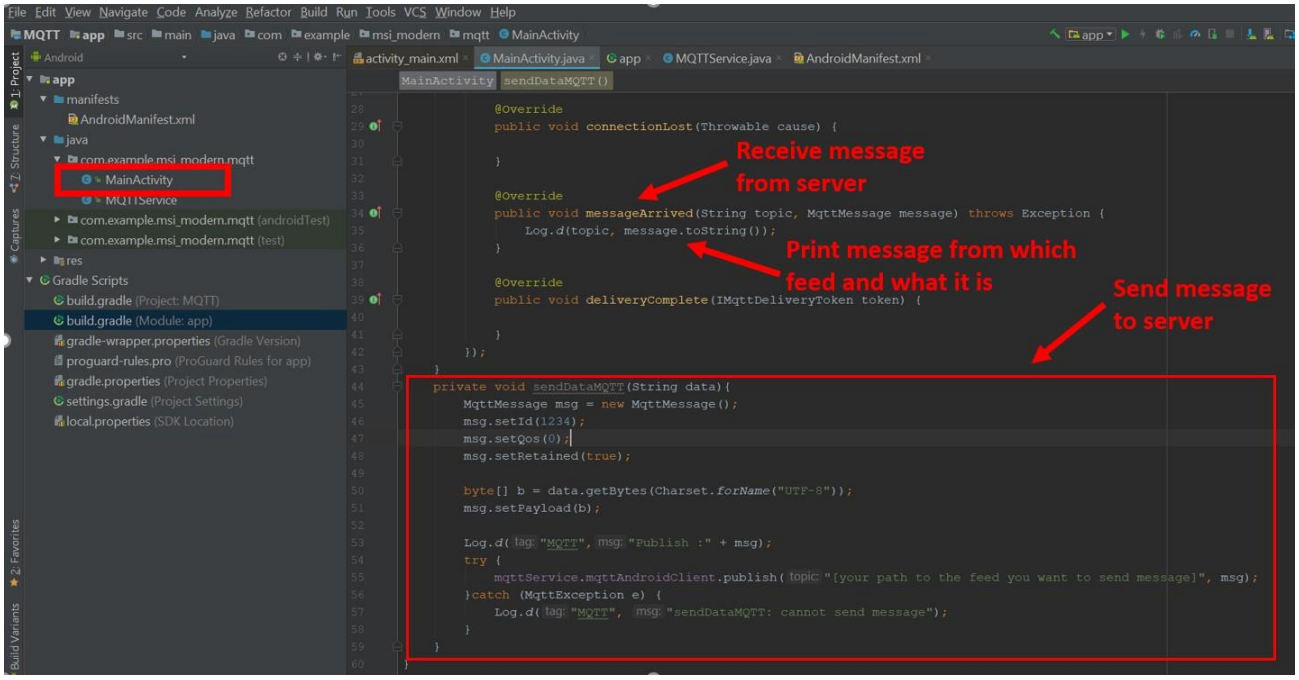
Dưới đây ảnh mô tả các bước cần hiện thực:



*Hình 20: Thêm các hàm MQTT và các biến vào hàm **onCreate***

Cuối cùng, thêm chức năng gửi và nhận dữ liệu MQTT đến máy chủ Adafruit:

```
1   privatevoid      sendDataMQTT( Stringdata){
2       MqttMessage msg = new MqttMessage();
3       msg.setId(1234);
4       msg.setQos(0);
5       msg.setRetained(true);
6
7       byte[] b = data.getBytes(Charset.forName("UTF-8"));
8       msg.setPayload(b);
9
10      Log.d("ABC","Publish:"+ msg);
11      try {
12          mqttService.mqttAndroidClient.publish("[Your
                subscriptiontopic]", msg);
13      } catch (MqttException e){
14
15      }
16  }
```



Hình 8: Thêm hàm sendDataMQTT

Như vậy là đã xong phần hướng dẫn gửi nhận dữ liệu, dùng giao thức MQTT trên cho server adafruit cho android.

Phần 3: Giả lập Publisher và Subscriber bằng Mosquitto trên server adafruit

Mô tả: Ứng dụng có thể giả lập đóng vai trò như 1 broker tại máy tính cá nhân. (Tên host: "localhost", port: 1883). Mặc định giao thức mosquito sử dụng là MQTT. Ngoài ra, ứng dụng có thể tạo ra Publisher hay Subscriber.

Link cài đặt: htttps://mosquito.org/

Sau khi cài đặt hoàn tất, vào thư mục cài đặt, sử dụng cmd chạy lệnh mosquito.exe để chạy ứng dụng.

Các bạn có thể kiểm tra việc tạo Publisher và Subscriber bằng cách mở 2 tab cmd, 1 tab dành cho Publisher và tab còn lại cho Subscriber.

- Để tạo subscriber, gõ lệnh:

mosquito_sub -h [tên_host] -p [port] -u [IO_USERNAME] -P [IO_KEY] -t [tên_topic]

- Để publish một payload, dùng lệnh:

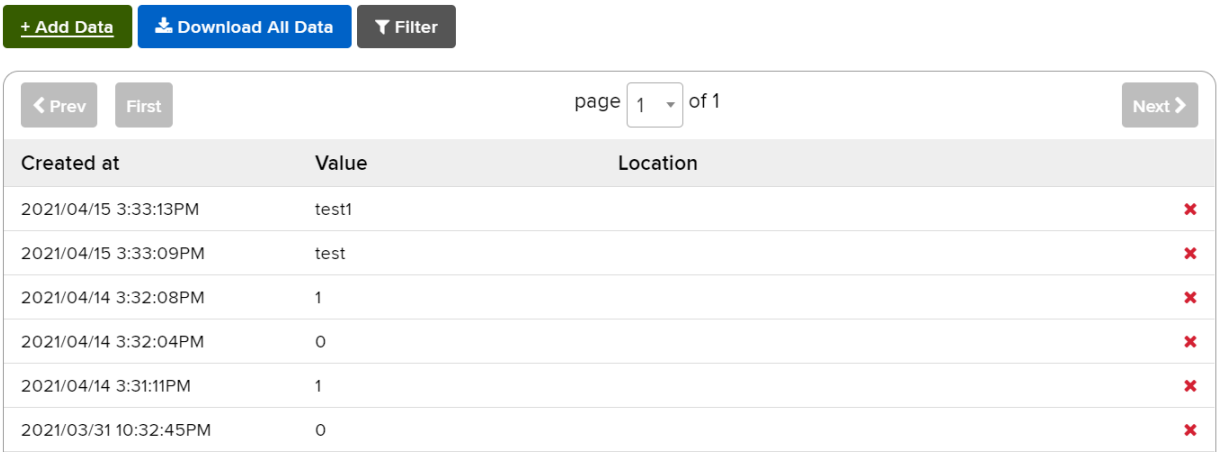
mosquito_pub -h [tên_host] -p [port] -u [IO_USERNAME] -P [IO_KEY] -t [tên_topic] -m [message]

Trong đó:

- tên_host: io.adafruit.com
- port: 1883
- IO_USERNAME và IO_KEY: Kiểm tra trong tab **My Key** ứng với tài khoản tạo server trên adafruit.
- tên_topic: Các bạn vào **Feeds**, chọn vào feed dữ liệu của mình, chọn **Feed Info** và điền giá trị ở dòng **MQTT by Key**
- message: Các bạn tham khảo format JSON ứng với các thiết bị.

Hình 1: Kết quả khi chạy các đoạn lệnh pub_sub với Mosquitto

Sau khi chạy xong, các bạn có thể vào feed mà các bạn vừa gửi dữ liệu lên server để kiểm tra:



Hình 2: Kết quả khi publish message “test” và “test1” lên feed “iot-test-led”