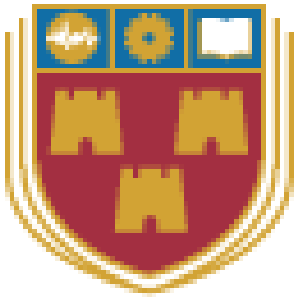

INSTITUTE of TECHNOLOGY CARLOW

Institiúid Teicneolaíochta Cheatharlach

Bachelor of Science (Honours)
Software Development

Home Automation System
Technical Manual

Course Code: CW238
Student ID: C00212235
Student Name: Xiaohui Ling
Supervisor: Paul Barry

Abstract

This document provides the technical details of the Home Automation System. Describe the programming languages, development environment, mid-wares, communication process and the details of the system requirements implementation. Finally, present the process of how to deploy the applications which were implemented in the project.

Contents

Table of Figures

Table of Code List

1. Introduction

The home automation system as a research project of my final year that is designed to be a prototype system which provides a basis features to remote management and control home appliances, sensor, or other devices. The home local network Wi-Fi is used for communication way between the mobile app and Raspberry Pi thus maintaining the home equipment.

2. Overview

2.1. Purpose

The purpose of this document aims to introduce the details of the technologies used to implement the system requirements. Conform strictly the requirement specifications document and functional design document.

2.2. Goal

According to the functional design document, to develop the Home Automation System and record the implementation details of the process.

3. Technologies & Components

3.1. Key Technologies

OpenWRT

- It is a Linux OS
- It can be logined vai SSH
- It can install software customize
- It provides Router functionalities

MQTT

- It is a lightweight MQ
- It consist of Publisher, Subscriber and Broker

AWS ActiveMQ

- It is a Message Queue midware
- It is also as a MQTT Broker
- It can be accessed with Internet via public IP address

Flutter

- It is a mobile app framework using Dart programming language
- It compatible with Android and iOS platform

Python3

- Scripts to control devices

Drivers & 3rd APIs

- Mjpg-streamer
- GoogleSpeechRecognition

Figure 1 key technologies

3.2. Key Components

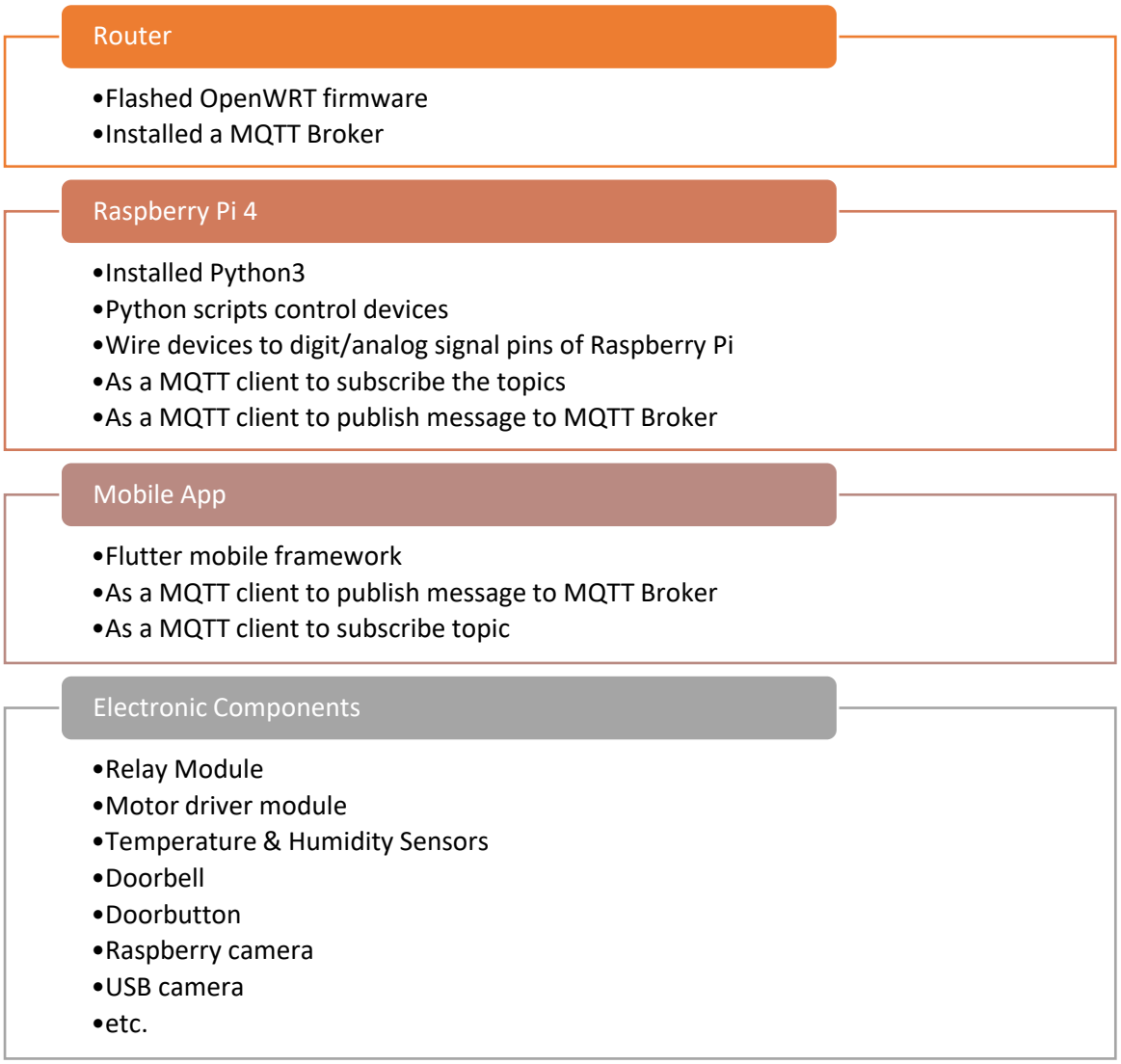

Figure 2 key components

4. System Environment

I would like to declare system environment versions which this project used that is listed below for each of the those I used.

Dart

Dart 3.9.1

Python

Python3.7.3

Raspberry Pi Linux

Raspberry Pi 4.19.75-v7l+ #1270

Flutter

Channel dev, v1.15.18

Android SDK

version is 28.0.3

OpenWRT

openwrt-mt300n-v2-3.102

MQTT Broker in Router

Version 3.0.1-1

Mjpeg-stremer

Version: 501f6362c5afddcfb41055f97ae484252c85c912

Project Source Code Git Hub

https://github.com/hellolingxh/HomeAutomation.git

5. Technical Manual

5.1. Structure

In this project, two systems were developed that respectively are Flutter App and Raspberry Pi Python Scripts.

5.1.1. Flutter App Structure

The Flutter App is a mobile app that will be installed on the mobile phone. The app code structure diagram below.

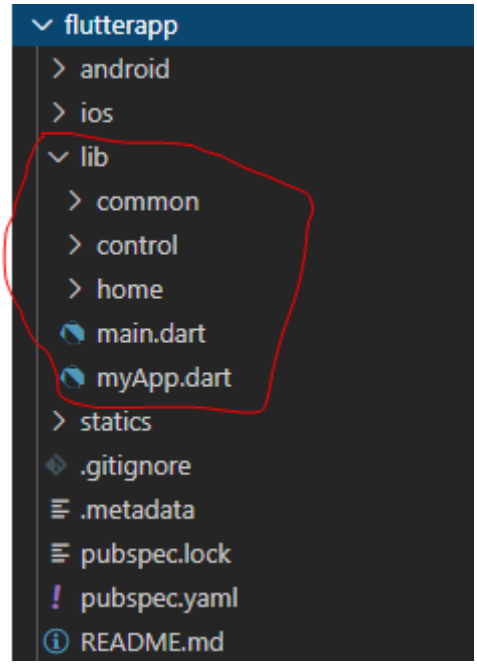

Figure 3 flutter app structure

The main.dart file under the lib folder that is an entry point in this app when the app is launched. The common folder included the files that are re-usable classes definition such as constant classes, utility classes etc. The control folder contains all devices' control panel in the app. The home folder just is home UI in the app.

Each of those classes in the directories will be explained next chapter in detail.

5.1.2. Raspberry Pi Python Scripts Structure

The Python scripts are used to control the devices that is connected on the Raspberry Pi. The scripts code structure diagram below.

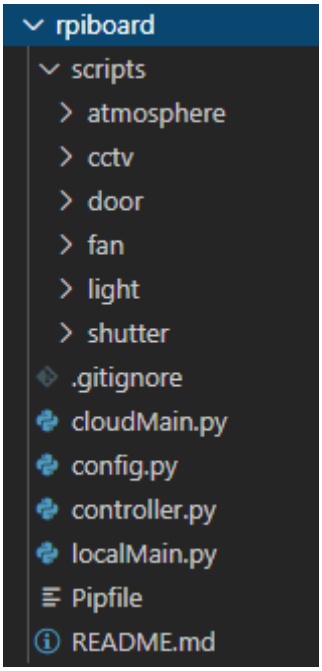

Figure 4 python scripts structure

The cloudMain.py file that is responsible for communicate with the Cloud MQTT Broker between the Raspberry Pi and mobile app. The localMain.py that is responsible for communicate with the Local MQTT Broker between the Raspberry Pi and mobile app. In the project, the Cloud MQTT Broker is a cloud service named AWS ActiveMQ.

The controller.py file schedules either cloudMain.py or localMain.py to execute if a message will be received. The cloudMain.py and localMain.py both execute as the respective thread in the controller.py.

The config.py is a global configuration file that defined the common constant string used in the whole python scripts project.

The subdirectories under the scripts are respectively responsible for a variety of devices control scripts that executed on Raspberry Pi.

Each of those scripts will be explained next chapter in detail.

5.2. Flutter App Classes

In this project, the Flutter mobile app development framework is used to build the mobile app.

5.2.1. Main Entry Point Classes

The main entry point class is an entry point in the app when the app is launched by a user. The home screen is loaded by the entry point class will be shown up on the mobile app. The location where the main entry point class is in the structure diagram below.

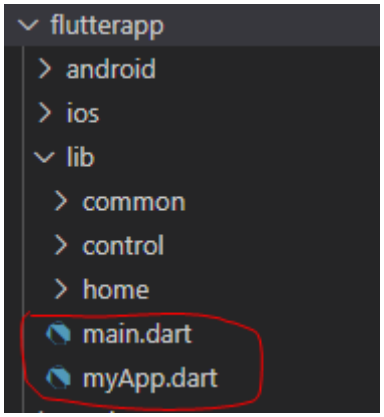

Figure 5 main entry point classes structure

- **main.dart**

Path: flutterapp/lib/main.dart
Description: That is main entry point of the mobile app when it is launched by a user.

```
import 'package:flutter/material.dart';
import 'package:flutterapp/home/myApp.dart';
```

```
void main() => runApp(MyApp());
```

Code 1 main.dart

- **myApp.dart**

Path: flutterapp/lib/myApp.dart
Description: That is a class is loaded by the main entry point class, meanwhile, this class loaded the home screen class to build the home screen UI to show up on the screen of the mobile app.

```
import 'package:flutter/material.dart';
import 'appHome.dart';

class MyApp extends StatefulWidget {

  @override
  _MyAppState createState() => new _MyAppState();

}

class _MyAppState extends State<MyApp> {

  @override
  Widget build(BuildContext context) {

    return new MaterialApp(
        title: 'My Home Automation',
        color: Colors.grey,
        home: AppHome(),
        routes: {'/home': (_)=> new AppHome()}
    );

  }

}
```

Code 2 myApp.dart

5.2.2. Common classes.

The classes defined are commonly used to the whole project is placed under the common directory of the Flutter app. The common directory structure diagram below.

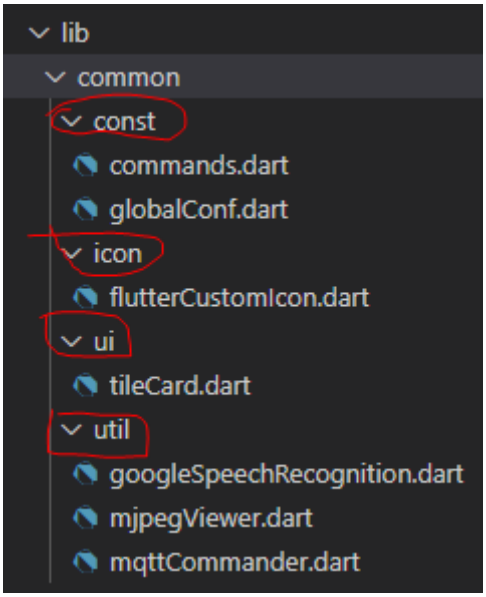

Figure 6 common classes structure

There are three subdirectories under the common directory that respectively are const, icon, ui, and util directories that will be described next section in detail.

5.2.2.1. Constant Classes

There are two constant classes are defined under const folder in this project that respectively are commands.dart and globalConf.dart.

- **commands.dart**

Path: flutterapp/lib/common/const/commands.dart

Description: That class defined all the topics of the MQTT message queue which named command in this project. There are 10 commands used totally in the whole project listed in this dart file. Each of those commands respectively corresponds to one device.

```
class Commands {

  static const String LIGHT_CONTROL = 'topic/light/control';

  static const String FAN_CONTROL = 'topic/fan/control';

  static const String SHUTTER_CONTROL = 'topic/shutter/control';

  static const String CCTV_CONTROL = 'topic/camera/cctv';

  static const String INDOOR_TEMPERATURE_HUMIDITY_DATA_READ = 'topic/indoor/measurement/read'
;
```

```
  static const String INDOOR_TEMPERATURE_HUMIDITY_DATA_RECEIVE = 'topic/indoor/measurement/da
ta';

  static const String OUTDOOR_TEMPERATURE_HUMIDITY_DATA_READ = 'topic/outdoor/measurement/r
ead';

  static const String OUTDOOR_TEMPERATURE_HUMIDITY_DATA_RECEIVE = 'topic/outdoor/measuremen
t/data';

  static const String DOOR_LOCKER_CONTROL = 'topic/door/lock/control';

  static const String DOOR_CAMERA_CONTROL = 'topic/door/camera/control';
}
```

Code 3 commands.dart

- **globalConfig.dart**

Path:flutterapp/lib/common/const/globalConfig.dart

Description: This is a global configuration file that defined all constants used in the whole project.

```
import 'package:flutter/material.dart';

class GlobalConfig {

   static final ThemeData myTheme = new ThemeData(
     brightness: Brightness.light,
     primarySwatch: Colors.teal,
     accentColor: Colors.redAccent,
   );

   static const String CEISIUS_SYMBOL = '\u2103';

   static const String HOME_SCREEN_BACKGROUND_IMAGE = 'statics/images/home_logo.jpg';

   static const String DEVICE_CONTROL_PANEL_SCREEN_BACKGROUND_IMAGE = 'statics/images/devices.
png';

   static const String CCTV_VIDEO_STREAM_URL = 'http://192.168.8.133:8080/?action=stream';

   static const String DOOR_CAMERA_VIDEO_STREAM_URL = 'http://192.168.8.133:8088/?action=stream
';

   static const String LOCAL_MQTT_BROKER_HOST = '192.168.8.1';

   static const int LOCAL_MQTT_BROKER_LISTEN_PORT = 1883;
```

```
  static const String AWS_ACTIVEMQ_HOST = 'b-ddb6ba7b-55f1-4ad2-b3c9-7754a11843ac-1.mq.eu-west-1.amazonaws.com';

  static const int AWS_ACTIVEMQ_LISTEN_PORT = 8883;

  static const String AWS_ACTIVEMQ_USERNAME = 'mqtt';

  static const String AWS_ACTIVEMQ_PASSWORD = '1qaz@WSX3edc';

  static const String INDOOR = 'indoor';

  static const String OUTDOOR = 'outdoor';

  static const String SPEECH_RECOGNITION_KEY_WORD_TURN_LIGHT_ON = 'light on';

  static const String SPEECH_RECOGNITION_KEY_WORD_TURN_LIGHT_OFF = 'light off';

  static const String DEFAULT_INITIAL_TEMPERATURE_VALUE = '17.8';

  static const String DEFAULT_INITIAL_HUMIDITY_VALUE = '43.7';

  static const String MQTT_CLIENT_IDENTIFIER_LIGHT = 'Light_MQTT_Client';

  static const String MQTT_CLIENT_IDENTIFIER_FAN = 'Fan_MQTT_Client';

  static const String MQTT_CLIENT_IDENTIFIER_SHUTTER = 'Shutter_MQTT_Client';

  static const String MQTT_CLIENT_IDENTIFIER_CCTV = 'CCTV_MQTT_Client';

  static const String MQTT_CLIENT_IDENTIFIER_ATMOSPHERE = 'Atmosphere_MQTT_Client';

  static const String MQTT_CLIENT_IDENTIFIER_DOORACCESS = 'DoorAccess_MQTT_Client';

  static const String NETWORK_OPTION_WIFI = 'WiFi';

  static const String NETWORK_OPTION_INTERNET = 'Internet';

}

enum DEVICE_NAME {LIGHT, FAN, SHUTTER, CCTV, DOOR_CAMERA, DOOR_LOCK, ATMOSPHERE_SENSOR }
```

Code 4 globalConfig.dart

5.2.2.2. Icon Classes

- **flutterCustomIcon.dart**

Path: flutterapp/lib/common/icon/flutterCustomIcon.dart

Description: That is an open-source extension class of Flutter Icon derived from Fontello. This class provided the icons used in this project.

```
/// Flutter icons Flutter_custom_icon
/// Copyright (C) 2019 by original authors @ fluttericon.com, fontello.com
/// This font was generated by FlutterIcon.com, which is derived from Fontello.
///
/// To use this font, place it in your fonts/ directory and include the
/// following in your pubspec.yaml
///
/// flutter:
///   fonts:
///    - family:  Flutter_custom_icon
///      fonts:
///       - asset: fonts/Flutter_custom_icon.ttf
///
///
/// * Font Awesome, Copyright (C) 2016 by Dave Gandy
///       Author:    Dave Gandy
///       License:   SIL ()
///       Homepage:  http://fortawesome.github.com/Font-Awesome/
/// * Entypo, Copyright (C) 2012 by Daniel Bruce
///       Author:    Daniel Bruce
///       License:   SIL (http://scripts.sil.org/OFL)
///       Homepage:  http://www.entypo.com
/// * Typicons, (c) Stephen Hutchings 2012
///       Author:    Stephen Hutchings
///       License:   SIL (http://scripts.sil.org/OFL)
///       Homepage:  http://typicons.com/
/// * Zocial, Copyright (C) 2012 by Sam Collins
///       Author:    Sam Collins
///       License:   MIT (http://opensource.org/licenses/mit-license.php)
///       Homepage:  http://zocial.smcllns.com/
/// * Material Design Icons, Copyright (C) Google, Inc
///       Author:    Google
///       License:   Apache 2.0 (https://www.apache.org/licenses/LICENSE-2.0)
///       Homepage:  https://design.google.com/icons/
/// * Linearicons Free, Copyright (C) Linearicons.com
///       Author:    Perxis
///       License:   CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0/)
///       Homepage:  https://linearicons.com
///
import 'package:flutter/widgets.dart';

class FlutterCustomIcon {
  FlutterCustomIcon._();
```

```
  static const _kFontFam = 'Flutter_custom_icon';

  static const IconData cog_alt = const IconData(0xe800, fontFamily: _kFontFam);
  static const IconData videocam = const IconData(0xe801, fontFamily: _kFontFam);
  static const IconData camera = const IconData(0xe802, fontFamily: _kFontFam);
  static const IconData thermometer = const IconData(0xe803, fontFamily: _kFontFam);
  static const IconData temperatire = const IconData(0xe804, fontFamily: _kFontFam);
  static const IconData lamp = const IconData(0xe805, fontFamily: _kFontFam);
  static const IconData light_down = const IconData(0xe806, fontFamily: _kFontFam);
  static const IconData light_up = const IconData(0xe807, fontFamily: _kFontFam);
  static const IconData windows = const IconData(0xe808, fontFamily: _kFontFam);
  static const IconData autorenew = const IconData(0xe809, fontFamily: _kFontFam);
  static const IconData enter = const IconData(0xe81f, fontFamily: _kFontFam);
  static const IconData gamepad = const IconData(0xf11b, fontFamily: _kFontFam);
  static const IconData chart_line = const IconData(0xf201, fontFamily: _kFontFam);
}
```

Code 5 flutterCustomIcon.dart

5.2.2.3. UI Classes

- **tileCart.dart**

Path: flutterapp/lib/common/ui/tileCard.dart

Description: This is a commonly used UI layout style that the tile is mainly used in this project.

```
import 'package:flutter/material.dart';
import 'package:flutterapp/common/const/globalConf.dart';

class TileCard extends StatelessWidget {
  const TileCard(this.text, this.textStyle, this.action, this.backgroundColor, this.icon);

  final String text;
  final TextStyle textStyle;
  final Function action;
  final Color backgroundColor;
  final Icon icon;

  @override
  Widget build(BuildContext context) {
    return Card(
      color: backgroundColor,
      child: new InkWell(
        onTap: () {navigate(context);},
        child: new Center(
          child: new Column(
            mainAxisAlignment: MainAxisAlignment.center,
```

```
            crossAxisAlignment: CrossAxisAlignment.center,
            children: <Widget>[
              Text(text, style: textStyle,),
              icon,
            ],
          ),
        ),
      )
    );
  }

  void navigate(BuildContext context){
    Navigator.push(context, MaterialPageRoute<void>(
        builder: (BuildContext context){
          return Theme(
            data: GlobalConfig.myTheme.copyWith(platform: Theme.of(context).platform),
            child: action(),
          );
        }
    ));
  }

}
```

Code 6 tileCard.dart

5.2.2.4. Util Classes

- **googleSpeechRecognition.dart**

Path: flutterapp/lib/common/util/googleSpeechRecognition.dart

Description: This class encapsulated the speech recognition features provided by Google that used to voice control of devices in the project.

```
import 'package:speech_recognition/speech_recognition.dart';

///
/// This class encapsulate the google speech recognition features
/// into this project.
///
abstract class GoogleSpeechRecognition {

  final SpeechRecognition speechRecognition = new SpeechRecognition();

  bool isAvailable = false;
  bool isListening = false;
  bool isComplete = false;
```

```
 String resultText = '';

 void setAvailableState(bool result);

 void setRecognitionStartedState();

 void setRecognitionResultState(String speech);

 void setRecognitionCompleteState();

 void activeCallback(bool result);

 void initSpeechRecognizer(){

    speechRecognition.setAvailabilityHandler(
       (bool result) => setAvailableState(result)
    );
    speechRecognition.setRecognitionStartedHandler(
       () => setRecognitionStartedState(),
    );
    speechRecognition.setRecognitionResultHandler(
       (String speech) => setRecognitionResultState(speech)
    );
    speechRecognition.setRecognitionCompleteHandler(
       () => setRecognitionCompleteState()
    );
    speechRecognition.activate().then(
       (result) => activeCallback(result)
    );
 }


}
```

Code 7 googleSpeechRecognition.dart

- **mjpegViewer.dart**

Path: flutterapp/lib/common/util/mjpegViewer.dart

Description: This class is a utility that used to load the video stream in real-time then shows up on the mobile app screen.

```
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

import 'dart:typed_data';
import 'dart:async';
```

```dart
///
/// This class encapsulate the feature that receive the video stream through http protocol.
///
class MjpegView extends StatefulWidget {
  MjpegView({this.url, this.fps});

  final String url;
  final int fps;

  @override
  State<MjpegView> createState() => _MjpegViewState();
}

class _MjpegViewState extends State<MjpegView> {
  Image _mjpeg;
  var _imgBuf;
  Stopwatch _timer = Stopwatch();

  http.Client _client = http.Client();
  StreamSubscription videoStream;

  @override
  void initState() {
    super.initState();
    _buildImageStream();
  }

  void _buildImageStream() {

    var request = http.Request("GET", Uri.parse(widget.url));

    _client.send(request).then((response) {
      var startIndex = -1;
      var endIndex = -1;
      List<int> buf = List<int>();

      Duration ts;

      _timer.start();

      videoStream = response.stream.listen((List<int> data) {
        for (var i = 0; i < data.length - 1; i++) {
          if (data[i] == 0xff && data[i + 1] == 0xd8) {
            startIndex = buf.length + i;
          }

          if (data[i] == 0xff && data[i + 1] == 0xd9) {
            endIndex = buf.length + i;
```

```
      }
     }

     buf.addAll(data);

     if (startIndex != -1 && endIndex != -1) {
      // print('start $startIndex, end $endIndex');

      _timer.stop();
      ts = _timer.elapsed;

      if (ts.inMilliseconds > 1000 / widget.fps) {
       // print('duration ${ts.inMilliseconds / 1000}');

       _imgBuf = List<int>.from(buf.getRange(startIndex, endIndex + 2));
       _mjpeg = Image.memory(Uint8List.fromList(_imgBuf));

       precacheImage(_mjpeg.image, context);

       Future.delayed(const Duration(milliseconds: 100)).then((_) {
        if (mounted) setState(() {});
       });

       _timer.reset();
      }

      startIndex = endIndex = -1;
      buf = List<int>();
      _timer.start();
     }
    });
  });
 }

 @override
 void deactivate() {
  _timer?.stop();
  videoStream?.cancel();
  _client?.close();

  super.deactivate();
 }

 @override
 Widget build(BuildContext context) {
  return _mjpeg == null ? new Center(child: CircularProgressIndicator()) : _mjpeg;
}
```

```
}
```

Code 8 mjpegViewer.dart

- **mqttCommander.dart**

Path: flutterapp/lib/common/util/mqttCommander.dart

Description: This is a utility that used for MQTT Client to connect the MQTT Broker then publish or subscribe to the topics of the message. It is a core class in this project which is responsible for the communication in the whole project between the Raspberry Pi and mobile phone of which on the home local network even across the Internet.

```
import 'dart:async';
import 'package:mqtt_client/mqtt_client.dart';

///
/// This class encapsulate the MQTT Client that
/// is implemented by the examples
/// of official flutter mqtt client provided.
///
class MqttCommander {

  String host; //MQTT Server Host Address

  int port; //MQTT Server Port

  String clientIdentifier; //MQTT Client Identifier

  bool secure; //established the connection with a scurity way.

  String username; // if secure is true then should given the username

  String password; // if secure is true then should given the password

  /// An annotated simple subscribe/publish usage example for mqtt_client. Please read in with referenc
e
  /// to the MQTT specification. The example is runnable, also refer to test/mqtt_client_broker_test...dar
t
  /// files for separate subscribe/publish tests.

  /// First create a client, the client is constructed with a broker name, client identifier
  /// and port if needed. The client identifier (short ClientId) is an identifier of each MQTT
  /// client connecting to a MQTT broker. As the word identifier already suggests, it should be unique per
 broker.
  /// The broker uses it for identifying the client and the current state of the client. If you don't need a st
ate
  /// to be hold by the broker, in MQTT 3.1.1 you can set an empty ClientId, which results in a connection
 without any state.
```

```
  /// A condition is that clean session connect flag is true, otherwise the connection will be rejected.
  /// The client identifier can be a maximum length of 23 characters. If a port is not specified the standar
d port
  /// of 1883 is used.
  /// If you want to use websockets rather than TCP see below.
  MqttClient client;

  MqttCommander(String remoteHost, int remotePort, String clientName, {bool isSecure:false, String use
rname, String password}){
    host = remoteHost;
    port = remotePort;
    clientIdentifier = clientName;
    secure = isSecure;
    this.username = username;
    this.password = password;
    _connect();
  }

  _initial(){
    /// to create a mqtt client
    client = new MqttClient.withPort(host, clientIdentifier, port);

    /// Set logging on if needed, defaults to off
    client.logging(on: false);

    /// If you intend to use a keep alive value in your connect message that is not the default(60s)
    client.keepAlivePeriod = 20;

    /// Add the unsolicited disconnection callback
    client.onDisconnected = onDisconnected;

    /// Add the successful connection callback
    client.onConnected = onConnected;

    /// Add a subscribed callback, there is also an unsubscribed callback if you need it.
    client.onSubscribed = onSubscribed;

    /// Set a ping received callback if needed, called whenever a ping response(pong) is received
    /// from the broker.
    client.pongCallback = pong;

    // Specific the definition protocol version.
    client.setProtocolV311();

    // The data transimission through the secure way.
    client.secure = secure;

    /// Create a connection message to use or use the default one.
```

```
    /// the default keepalive interval(60s)
    final MqttConnectMessage connMess = MqttConnectMessage()
        .withClientIdentifier(clientIdentifier)
        .keepAliveFor(20) // Must agree with the keep alive set above or not set
        .withWillTopic('willtopic') // If you set this you must set a will message
        .withWillMessage('My Will message')
        .startClean() // Non persistent session for testing
        .withWillQos(MqttQos.atLeastOnce);
    print('Mosquitto client connecting....');
    client.connectionMessage = connMess;

  }

  Future _connect() async{

    /// to inital the context of the mqtt client
    _initial();

    /// Connect the client, any errors here are communicated by raising of the appropriate exception. Not
e
    /// in some circumstances the broker will just disconnect us, see the spec about this, we however eill
    /// never send malformed messages.
    try {
      if(secure)
        await client.connect(this.username, this.password);
      else
        await client.connect();
    } on Exception catch (e) {
      print('client exception - $e');
      client.disconnect();
    }

    /// Check we are connected
    if (client.connectionStatus.state == MqttConnectionState.connected) {
      print('Mosquitto client connected');
    } else {
      /// Use status here rather than state if you also want the broker return code.
      print(
          'ERROR Mosquitto client connection failed - disconnecting, status is ${client.connectionStatus}');
      client.disconnect();
    }

  }

  Future<int> _publish(final String topic, String message) async {

    final MqttClientPayloadBuilder builder = MqttClientPayloadBuilder();
    builder.addString(message);
```

```
  /// Publish it
  print('Publishing our topic');
  client.publishMessage(topic, MqttQos.exactlyOnce, builder.payload);

  return 0;

}

disconnect() async{
  /// Wait for the unsubscribe message from the broker if you wish.
  await MqttUtilities.asyncSleep(2);
  print('Disconnecting');
  client.disconnect();
  return 0;
}

Future<int> _subscribe(String topic) async{
  await MqttUtilities.asyncSleep(2);
  client.subscribe(topic, MqttQos.atMostOnce);

  return 0;

}

Future<int> unsubscribe(String topic) async{
  /// Finally, unsubscribe and exit gracefully
  print('Unsubscribing');
  client.unsubscribe(topic);
  return 0;
}

/// The subscribed callback
void onSubscribed(String topic) {
  print('Subscription confirmed for topic $topic');
}

/// The unsolicited disconnect callback
void onDisconnected() {
  print('OnDisconnected client callback - Client disconnection');
  if (client.connectionStatus.returnCode == MqttConnectReturnCode.solicited) {
    print('OnDisconnected callback is solicited, this is correct');
  }
}

/// The successful connect callback
void onConnected() {
  print('OnConnected client callback - Client connection was sucessful');
```

```
  }

  /// Pong callback
  void pong() {
    print('Ping response client callback invoked');
  }

  ///To send the message according to the command which is a topic concept on MQTT.
  Future send(String command, String param) async{

    _publish(command, param);

  }

  ///To receive the message from broker according to the specific command which
  ///is a topic concept on MQTT, then called the callback method passed by caller.
  Future receive(String command, Function(String) callback) async{

    _subscribe(command);

    client.updates.listen((List<MqttReceivedMessage<MqttMessage>> list) {

      final MqttPublishMessage receivedPayload = list[0].payload;
      final String message = MqttPublishPayload.bytesToStringAsString(receivedPayload.payload.message)
;


      print('notification: topic is <${list[0].topic}>, payload message is <-- $message -->');

      if(callback != null)
        callback(message);
    });
  }

}
```

Code 9 mqttCommander.dart

5.2.3. Home Screen Classes

The classes defined that used to build the home screen of the mobile app under the home folder in this Flutter app project. The home directory structure diagram below.

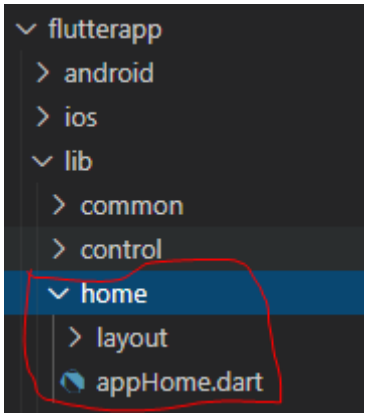

Figure 7 home screen classes structure

There is a subdirectory named layout and the appHome.dart file under the home folder. That will be described next section in detail.

5.2.3.1 Home Screen UI

The classes under the home folder that will build the UI on the screen below.

Figure 8 home screen

5.2.3.2 Classes Description

- **appHome.dart**

Path: flutterapp/lib/home/appHome.dart

Description: This class build the home screen UI on the mobile app when the app is launched by a user.

```
import 'package:flutter/material.dart';
import 'layout/tileLayout.dart';

class AppHome extends StatefulWidget {
 @override
 _AppHomeState createState() => new _AppHomeState();

 }

class _AppHomeState extends State<AppHome> {

 static final GlobalKey<ScaffoldState> _scaffoldKey = new GlobalKey<ScaffoldState>();

 @override
 Widget build(BuildContext context) {
  return new Scaffold(
     key: _scaffoldKey,
     appBar: AppBar(
           automaticallyImplyLeading: true,
           title: Text("My Home Automation"),
           elevation: 10.0,
           centerTitle: true,
           backgroundColor: Colors.teal,
     ),
     body: new Padding(
        padding: const EdgeInsets.all(0.0),
        child: staggeredGridView,
     ),
     bottomNavigationBar: BottomNavigationBar(
        currentIndex: 0, // This will be set when a new tab is tapped
        backgroundColor: Colors.teal,
        items: [
           BottomNavigationBarItem(
              icon: new Icon(Icons.home, color: Colors.white,),
              title: new Text('Home', style: TextStyle(color: Colors.white),),
           ),
           BottomNavigationBarItem(
              icon: new Icon(Icons.account_circle, color: Colors.white),
              title: new Text('Me', style: TextStyle(color: Colors.white),),
           )
        ],
     ),
  );
 }
```

```
}
```

Code 10 appHome.dart

- **tileLayout.dart**

Path: flutterapp/lib/home/tileLayout.dart

Description: This class is to create the elements on the UI using the tile layout such as a background image, remote control tile card, data statistics tile card and configuration tile card. The class will be called by the appHome.dart to build the home screen UI.

```
import 'package:flutter/material.dart';
import 'package:flutter_staggered_grid_view/flutter_staggered_grid_view.dart';
import 'package:flutterapp/common/const/globalConf.dart';
import 'package:flutterapp/common/ui/tileCard.dart';
import 'package:flutterapp/common/icon/flutterCustomIcon.dart';
import 'package:flutterapp/control/deviceControlPanel.dart';

List<StaggeredTile> staggeredTiles = const <StaggeredTile>[
      const StaggeredTile.count(4, 3),
      const StaggeredTile.count(2, 3),
      const StaggeredTile.count(2, 2),
      const StaggeredTile.count(2, 1),
];

DeviceControlPanel deviceControlPanel() => DeviceControlPanel();

const textStyle = const TextStyle(color: Colors.blueGrey, fontWeight: FontWeight.w500, fontSize: 20.0);

List<Widget> tiles = const <Widget> [
   const Card(
      elevation: 10,
      shape: BeveledRectangleBorder(borderRadius: BorderRadius.all(Radius.zero), side: BorderSide(width
: 0, color: Colors.white)),
      child: const Image(image: AssetImage(GlobalConfig.HOME_SCREEN_BACKGROUND_IMAGE), fit: Box
Fit.fill,),
   ),
   const TileCard(
         'Remote Control',
         textStyle,
         deviceControlPanel,
         Colors.green,
         Icon(
            Icons.wifi,
            size: 50,
            color: Colors.white,
         )
```

```
    ),
  const TileCard(
      'Data Statistics',
      textStyle,
      deviceControlPanel,
      Colors.orange,
      Icon(
        FlutterCustomIcon.chart_line,
        size: 50,
        color: Colors.pink,
      )
    ),
  const TileCard(
      'Configuration',
      textStyle,
      deviceControlPanel,
      Colors.cyan,
      Icon(
       FlutterCustomIcon.cog_alt,
        size: 50,
        color: Colors.white,
      )
    ),
];

StaggeredGridView staggeredGridView = new StaggeredGridView.count(
          crossAxisCount: 4,
          staggeredTiles: staggeredTiles,
          children: tiles,
          mainAxisSpacing: 0.0,
          crossAxisSpacing: 0.0,
          padding: const EdgeInsets.all(0.0),
);
```

Code 11 tileLayout.dart

5.2.4. Device Control Panel Classes

The classes defined are the device control panels used to the whole project is placed under the control folder of the Flutter app. The control folder structure diagram below.

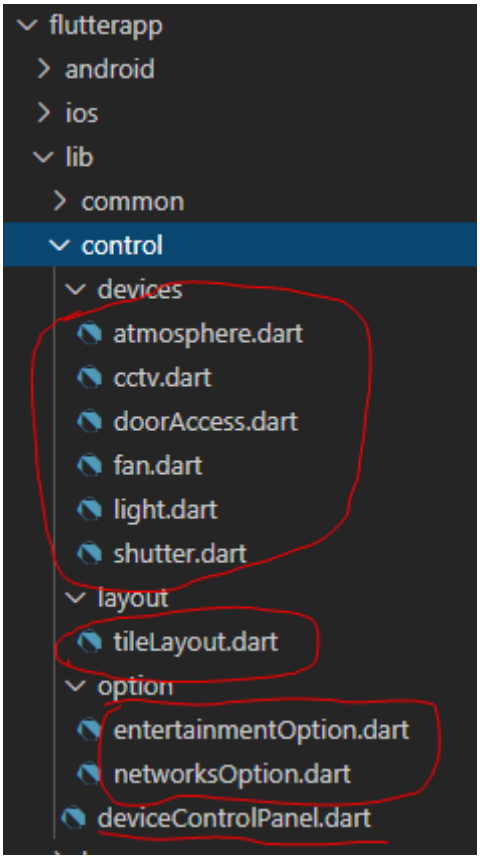

Figure 9 device control panel classes structure

There are three subdirectories under the control directory that respectively are devices, layout, option directories that will be described next section in detail.

5.2.4.1 Device Control Panel Classes

The classes deviceControlPanel.dart that will build the UI on the screen below.

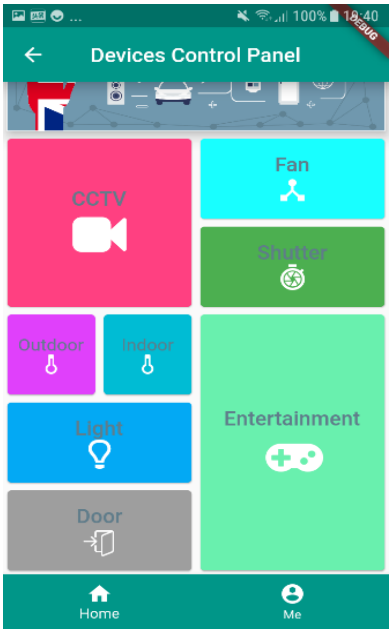

Figure 10 device control panel screen

- **deviceControlPanel.dart**

Path: flutterapp/lib/control/deviceControlPanel.dart

Description: In this class will list what the devices is controlled then built the UI components on the screen with mobile app.

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutterapp/common/const/globalConf.dart';
import 'package:flutterapp/home/appHome.dart';
import '../control/layout/tileLayout.dart';

class DeviceControlPanel extends StatefulWidget {

  @override
  State<StatefulWidget> createState() => new _DeviceControlPanel();

}

class _DeviceControlPanel extends State<DeviceControlPanel> {

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
        key: new GlobalKey<ScaffoldState>(),
        appBar: AppBar(
                automaticallyImplyLeading: true,
                title: Text("Devices Control Panel"),
```

```
                elevation: 10.0,
                centerTitle: true,
                backgroundColor: Colors.teal,
        ),
        body: new Padding(
            padding: const EdgeInsets.all(0.0),
            child: staggeredGridView,
        ),
        bottomNavigationBar: BottomNavigationBar(
            currentIndex: 0, // This will be set when a new tab is tapped
            backgroundColor: Colors.teal,
            items: [
                BottomNavigationBarItem(
                    icon: new Icon(Icons.home, color: Colors.white,),
                    title: new Text('Home', style: TextStyle(color: Colors.whi
te),),
                ),
                BottomNavigationBarItem(
                    icon: new Icon(Icons.account_circle, color: Colors.white),
                    title: new Text('Me', style: TextStyle(color: Colors.white
),),
                )
            ],
            onTap: (index) => Navigator.push(context, MaterialPageRoute<void>(
              builder: (BuildContext context){
                  return Theme(
                      data: GlobalConfig.myTheme.copyWith(platform: Theme.of(c
ontext).platform),
                      child: AppHome(),
                  );
              }
            )),
        ),
    );
  }

}
```

Code 12 deviceControlPanel.dart

- **tileLayout.dart**

Path: flutterapp/lib/control/layout/tileLayout.dart

Description: This class is a layout within the deviceControlPanel.dart to layout the UI components in this screen.

```
import 'package:flutter/material.dart';
import 'package:flutter_staggered_grid_view/flutter_staggered_grid_view.dart';
```

```dart
import 'package:flutterapp/common/const/globalConf.dart';
import 'package:flutterapp/common/ui/tileCard.dart';
import 'package:flutterapp/common/icon/flutterCustomIcon.dart';
import 'package:flutterapp/control/devices/atmosphere.dart';
import 'package:flutterapp/control/devices/cctv.dart';
import 'package:flutterapp/control/devices/doorAccess.dart';
import 'package:flutterapp/control/option/entertainmentOption.dart';
import 'package:flutterapp/control/option/networksOption.dart';

List<StaggeredTile> staggeredTiles = const <StaggeredTile>[
        const StaggeredTile.count(4, 2), // control panel picture
        const StaggeredTile.count(2, 2), // Camera control
        const StaggeredTile.count(2, 1), // Fan control
        const StaggeredTile.count(2, 1), // shutter control
        const StaggeredTile.count(1, 1), // indoor thermometer control
        const StaggeredTile.count(1, 1), // outdoor thermometer control
        const StaggeredTile.count(2, 3), // entertainment control
        const StaggeredTile.count(2, 1), // lighting control
        const StaggeredTile.count(2, 1), // door control
];

const textStyle = const TextStyle(color: Colors.blueGrey, fontWeight: FontWeig
ht.w500, fontSize: 20.0);
const smallTextStyle = const TextStyle(color: Colors.blueGrey, fontWeight: Fon
tWeight.w400, fontSize: 17.0);

CCTVWidget cctvWidget() => CCTVWidget();
NetworksOptionWidget lightWidget() => NetworksOptionWidget(deviceName: DEVICE_
NAME.LIGHT);
NetworksOptionWidget fanWidget() => NetworksOptionWidget(deviceName:DEVICE_NAM
E.FAN);
NetworksOptionWidget shutterWidget() => NetworksOptionWidget(deviceName:DEVICE
_NAME.SHUTTER);
AtmosphereWidget indoorAtmosphereWidget() => AtmosphereWidget(GlobalConfig.IND
OOR);
AtmosphereWidget outdoorAtmosphereWidget() => AtmosphereWidget(GlobalConfig.OU
TDOOR);
DoorAccessWidget doorAccessWidget() => DoorAccessWidget();
EntertainmentOptionWidget entertainmentOptionWidget() => EntertainmentOptionWi
dget();
List<Widget> tiles = const <Widget> [
    const Card(
        elevation: 10,
        shape: BeveledRectangleBorder(borderRadius: BorderRadius.all(Radius.ze
ro), side: BorderSide(width: 0, color: Colors.white)),
        child: const Image(image: AssetImage(GlobalConfig.DEVICE_CONTROL_PANEL
_SCREEN_BACKGROUND_IMAGE), fit: BoxFit.fill,),
    ),
```

```
    const TileCard(
            'CCTV',
            textStyle,
            cctvWidget,
            Colors.pinkAccent,
            Icon(
                FlutterCustomIcon.videocam,
                size: 50,
                color: Colors.white,
            )
         ),
    const TileCard(
            'Fan',
            textStyle,
            fanWidget,
            Colors.cyanAccent,
            Icon(
                Icons.device_hub,
                size: 30,
                color: Colors.white,
            )
        ),
    const TileCard(
            'Shutter',
            textStyle,
            shutterWidget,
            Colors.green,
            Icon(
                Icons.shutter_speed,
                size: 30,
                color: Colors.white,
            )
        ),
    const TileCard(
            'Outdoor',
            smallTextStyle,
            outdoorAtmosphereWidget,
            Colors.purpleAccent,
            Icon(
                FlutterCustomIcon.thermometer,
                size: 20,
                color: Colors.white,
            )
        ),
    const TileCard(
            'Indoor',
            smallTextStyle,
            indoorAtmosphereWidget,
```

```
            Colors.cyan,
            Icon(
                FlutterCustomIcon.thermometer,
                size: 20,
                color: Colors.white,
            )
        ),
    const TileCard(
            'Entertainment',
            textStyle,
            entertainmentOptionWidget,
            Colors.greenAccent,
            Icon(
                FlutterCustomIcon.gamepad,
                size: 50,
                color: Colors.white,
            )
        ),
    const TileCard(
            'Light',
            textStyle,
            lightWidget,
            Colors.lightBlue,
            Icon(
                FlutterCustomIcon.lamp,
                size: 30,
                color: Colors.white,
            )
        ),
    const TileCard(
            'Door',
            textStyle,
            doorAccessWidget,
            Colors.grey,
            Icon(
                FlutterCustomIcon.enter,
                size: 30,
                color: Colors.white,
            )
        ),

];

StaggeredGridView staggeredGridView = new StaggeredGridView.count(
                crossAxisCount: 4,
                staggeredTiles: staggeredTiles,
                children: tiles,
                mainAxisSpacing: 0.0,
```

```
                crossAxisSpacing: 0.0,
                padding: const EdgeInsets.all(0.0),
);
```

Code 13 tileLayout.dart

5.2.4.2 Devices Classes

The each of those device classes defined under the devices folder is controlled through the mobile app. Currently, there is a set of devices is controlled such as table light, fan, shutter, cctv camera, door access system devices and temperature & humidity sensors. Those device classes will be described in detail.

5.2.4.2.1 Light Device

The device is an abstraction with table light home appliance in the real world.

- **light.dart**

UI:

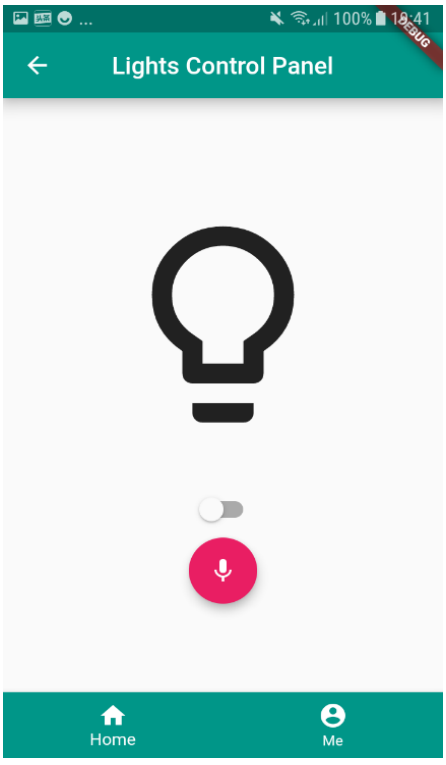

Figure 11 light control panel screen

Path: flutterapp/lib/control/devices/light.dart

Description: The class is the UI of the light control panel on the mobile app to turn on or off the light through the switch button on this screen. It also can be controlled via the user voice through the voice button on this screen.

```
import 'package:flutter/material.dart';
import 'package:flutterapp/common/const/commands.dart';
import 'package:flutterapp/common/const/globalConf.dart';
import 'package:flutterapp/common/util/googleSpeechRecognition.dart';
import 'package:flutterapp/common/util/mqttCommander.dart';
import 'package:flutterapp/home/appHome.dart';

class LightWidget extends StatefulWidget {
  final int networkType; // to decide the control through either WiFi or Inter
net

  LightWidget(this.networkType);

  @override
  State<StatefulWidget> createState() {
    _LightState _state ;

    if(networkType == 0)
      _state = new _LightState( new MqttCommander(
                    GlobalConfig.LOCAL_MQTT_BROKER_HOST,
                    GlobalConfig.LOCAL_MQTT_BROKER_LISTEN_PORT,
                    GlobalConfig.MQTT_CLIENT_IDENTIFIER_LIGHT
                    )
            );
    else
      _state = new _LightState( new MqttCommander(
                    GlobalConfig.AWS_ACTIVEMQ_HOST,
                    GlobalConfig.AWS_ACTIVEMQ_LISTEN_PORT,
                    GlobalConfig.MQTT_CLIENT_IDENTIFIER_LIGHT,
                    isSecure: true,
                    username: GlobalConfig.AWS_ACTIVEMQ_USERNAME,
                    password: GlobalConfig.AWS_ACTIVEMQ_PASSWORD
                  )
      );

      return _state;
  }

}

class _LightState extends State<LightWidget> with GoogleSpeechRecognition{

  final MqttCommander _commander;

  bool _isLightOn = false;
```

```
  _LightState(this._commander);

  @override
  void initState() {
    super.initState();
    initSpeechRecognizer();
  }

  @override
  void dispose() {
    super.dispose();
    _commander.disconnect();
  }

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
        key: new GlobalKey<ScaffoldState>(),
        appBar: AppBar(
                automaticallyImplyLeading: true,
                title: Text("Lights Control Panel"),
                elevation: 10.0,
                centerTitle: true,
                backgroundColor: Colors.teal,
        ),
        body: new Padding(
            padding: const EdgeInsets.all(0.0),
            child: _buildBody(),
        ),
        bottomNavigationBar: BottomNavigationBar(
            currentIndex: 0, // This will be set when a new tab is tapped
            backgroundColor: Colors.teal,
            items: [
                BottomNavigationBarItem(
                    icon: new Icon(Icons.home, color: Colors.white,),
                    title: new Text('Home', style: TextStyle(color: Colors.whi
te),),),
                ),
                BottomNavigationBarItem(
                    icon: new Icon(Icons.account_circle, color: Colors.white),
                    title: new Text('Me', style: TextStyle(color: Colors.white
),),),
                )
            ],
            onTap: (index) => Navigator.push(context, MaterialPageRoute<void>(
                builder: (BuildContext context){
                  return Theme(
```

```
                    data: GlobalConfig.myTheme.copyWith(platform: Theme.of(c
ontext).platform),
                    child: AppHome(),
                  );
                }
            )),
        ),
    );
  }


  Widget _buildBody(){
      return new Container(
          child: new Center(
                 child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    crossAxisAlignment: CrossAxisAlignment.center,
                    children: <Widget>[
                        Container(
                            height: 260,
                            margin: EdgeInsets.only(top: 3.0, bottom: 3.0),
                            child: _isLightOn==false ? Icon(Icons.lightbulb_ou
tline, size: 200,) :  Icon(Icons.lightbulb_outline, size: 200, color: Colors.d
eepOrangeAccent,),
                        ),
                        Switch(
                            value: _isLightOn,
                            onChanged: (bool value){
                                _commander.send(Commands.LIGHT_CONTROL, _isLig
htOn ? 'off' : 'on');
                                setState(() {
                                  _isLightOn = _isLightOn ? false : true; //no
tificate the flutter to refresh to component.
                                });
                            },
                        ),
                        FloatingActionButton(
                            heroTag: 'microphone',
                            child: Icon(Icons.mic),
                            onPressed: () {
                                if(isAvailable && !isListening)
                                    speechRecognition.listen(locale: 'en_US').
then((result) => print('$result'));
                            },
                            backgroundColor: Colors.pink,
                        ),
                        isComplete ? Padding(child:Text(resultText,), padding:
 EdgeInsets.only(top: 5.0),) : Text(' '),
                    ],
```

```
            ),
          ),
      );
  }

  @override
  void setAvailableState(bool result) {
    setState(() {
      isAvailable = result;
    });
  }

  @override
  void setRecognitionStartedState() {
    setState(() {
      isComplete = false;
      isListening = true;
    });
  }

  @override
  void setRecognitionResultState(String speech) {
    setState(() {
        resultText = speech;
    });
  }

  @override
  void activeCallback(bool result) {
    setState(() {
        isAvailable = result;
    });
  }

  @override
  void setRecognitionCompleteState() {
    setState(() {
      if(resultText==GlobalConfig.SPEECH_RECOGNITION_KEY_WORD_TURN_LIGHT_ON)
        _isLightOn = true;
      else if(resultText==GlobalConfig.SPEECH_RECOGNITION_KEY_WORD_TURN_LIGHT_
OFF)
        _isLightOn = false;

      isListening = false;
      isComplete = true;
    });
    if(resultText==GlobalConfig.SPEECH_RECOGNITION_KEY_WORD_TURN_LIGHT_ON || r
esultText==GlobalConfig.SPEECH_RECOGNITION_KEY_WORD_TURN_LIGHT_OFF)
```

```
      _commander.send(Commands.LIGHT_CONTROL, _isLightOn ? 'on' : 'off');
  }

}
```

Code 14 light.dart

5.2.4.2.2 Fan Device

The device is an abstraction with the Fan home appliance in the real world.

- fan.dart

UI:

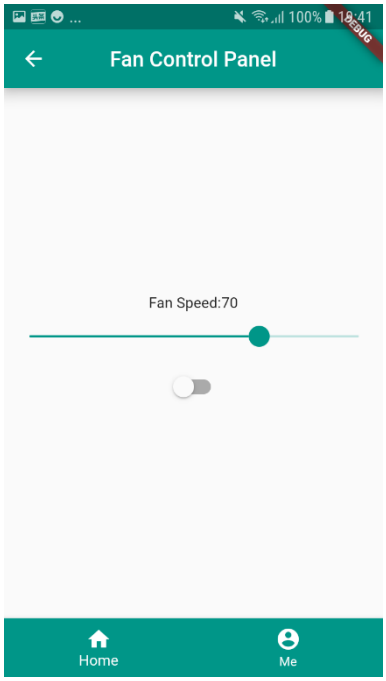

Figure 12 fan control panel screen

Path: flutterapp/lib/control/devices/fan.dart

Description: The class is the UI of the fan control panel on the mobile app to turn on or off the fan through the switch button on this screen. It also can be adjusted the speed of the fan via the user slides the progress bar on this screen.

```
import 'package:flutter/material.dart';
import 'package:flutterapp/common/const/commands.dart';
import 'package:flutterapp/common/const/globalConf.dart';
import 'package:flutterapp/common/util/mqttCommander.dart';
import 'package:flutterapp/home/appHome.dart';
```

```dart
class FanWidget extends StatefulWidget {

  final int networkType;  // to decide the control through either WiFi or Inte
rnet

  const FanWidget(this.networkType, {Key key}) : super(key: key);

  @override
  State<StatefulWidget> createState() {
    _FanState _state ;

    if(networkType == 0)
      _state = new _FanState( new MqttCommander(
                    GlobalConfig.LOCAL_MQTT_BROKER_HOST,
                    GlobalConfig.LOCAL_MQTT_BROKER_LISTEN_PORT,
                    GlobalConfig.MQTT_CLIENT_IDENTIFIER_FAN
                    )
            );
    else
      _state = new _FanState( new MqttCommander(
                    GlobalConfig.AWS_ACTIVEMQ_HOST,
                    GlobalConfig.AWS_ACTIVEMQ_LISTEN_PORT,
                    GlobalConfig.MQTT_CLIENT_IDENTIFIER_FAN,
                    isSecure: true,
                    username: GlobalConfig.AWS_ACTIVEMQ_USERNAME,
                    password: GlobalConfig.AWS_ACTIVEMQ_PASSWORD
                  )
      );

      return _state;
  }

}

class _FanState extends State<FanWidget> {

  final MqttCommander _commander;

  bool _isRunning = false;
  int _speed = 70;

  _FanState(this._commander);

  @override
  void dispose() {
    super.dispose();
    _commander.disconnect();
  }
```

```
  @override
  Widget build(BuildContext context) {
    return new Scaffold(
        key: new GlobalKey<ScaffoldState>(),
        appBar: AppBar(
                automaticallyImplyLeading: true,
                title: Text("Fan Control Panel"),
                elevation: 10.0,
                centerTitle: true,
                backgroundColor: Colors.teal,
        ),
        body: new Padding(
            padding: const EdgeInsets.all(0.0),
            child: _buildBody(),
        ),
        bottomNavigationBar: BottomNavigationBar(
            currentIndex: 0, // This will be set when a new tab is tapped
            backgroundColor: Colors.teal,
            items: [
                BottomNavigationBarItem(
                    icon: new Icon(Icons.home, color: Colors.white,),
                    title: new Text('Home', style: TextStyle(color: Colors.whi
te),),),
                ),
                BottomNavigationBarItem(
                    icon: new Icon(Icons.account_circle, color: Colors.white),
                    title: new Text('Me', style: TextStyle(color: Colors.white
),),),
                )
            ],
            onTap: (index) => Navigator.push(context, MaterialPageRoute<void>(
                builder: (BuildContext context){
                  return Theme(
                      data: GlobalConfig.myTheme.copyWith(platform: Theme.of(c
ontext).platform),
                      child: AppHome(),
                  );
                }
            )),
        ),
    );
  }

  Widget _buildBody(){
      return new Container(
          child: new Column(
              mainAxisAlignment: MainAxisAlignment.center,
```

```
            crossAxisAlignment: CrossAxisAlignment.center,
            children: <Widget>[
                Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    crossAxisAlignment: CrossAxisAlignment.center,
                    children: <Widget>[
                        Text('Fan Speed:'),
                        Text(_speed.toString()),
                    ],
                ),
                Slider(
                      value: _speed.toDouble(),
                      min: 0.0,
                      max: 100.0,
                      onChanged: (double value) {
                          print('on changed method, the value is '+value.toS
tring());
                          setState(() {
                              _speed = value.round();
                          });
                          if(_isRunning)
                            _commander.send(Commands.FAN_CONTROL, _speed.toS
tring());
                      },
                      onChangeStart: (double value){
                          print('on change start method, the value is '+valu
e.toString());
                      },
                      onChangeEnd: (double value){
                          print('on change end method, the value is '+value.
toString());
                      },
                  ),
                  Switch(
                      value: _isRunning,
                      onChanged: (bool value){
                          setState(() {
                              _isRunning = _isRunning ? false : true;
                          });
                          _commander.send(Commands.FAN_CONTROL, _isRunning ?
 _speed.toString() : 'off');
                      },
                  ),
            ],
          ),
      );
  }
```

```
}
```

Code 15 fan.dart

5.2.4.2.3 Shutter Device

The device is an abstraction with the shutter home appliance in the real world.

- shutter.dart

UI:

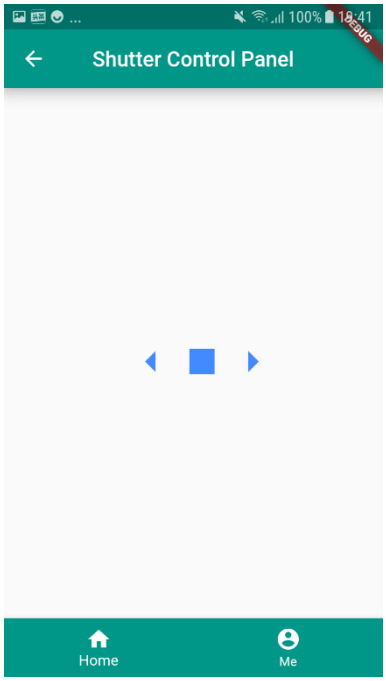

Figure 13 shutter control panel screen

Path: flutterapp/lib/control/devices/shutter.dart

Description: The class is the UI of the shutter control panel on the mobile app to move the curtain to left or right through the left or right arrow buttons on this screen. It also can be stopped via the stop button on this screen.

```
import 'package:flutter/material.dart';
import 'package:flutterapp/common/const/commands.dart';
import 'package:flutterapp/common/const/globalConf.dart';
import 'package:flutterapp/common/util/mqttCommander.dart';
import 'package:flutterapp/home/appHome.dart';

class ShutterWidget extends StatefulWidget {

  final int networkType;  // to decide the control through either WiFi or Inte
rnet
```

```
  const ShutterWidget(this.networkType, {Key key}) : super(key: key);

  @override
  State<StatefulWidget> createState(){
    _ShutterState _state ;

    if(networkType == 0)
      _state = new _ShutterState( new MqttCommander(
                    GlobalConfig.LOCAL_MQTT_BROKER_HOST,
                    GlobalConfig.LOCAL_MQTT_BROKER_LISTEN_PORT,
                    GlobalConfig.MQTT_CLIENT_IDENTIFIER_SHUTTER
                    )
            );
    else
      _state = new _ShutterState( new MqttCommander(
                    GlobalConfig.AWS_ACTIVEMQ_HOST,
                    GlobalConfig.AWS_ACTIVEMQ_LISTEN_PORT,
                    GlobalConfig.MQTT_CLIENT_IDENTIFIER_SHUTTER,
                    isSecure: true,
                    username: GlobalConfig.AWS_ACTIVEMQ_USERNAME,
                    password: GlobalConfig.AWS_ACTIVEMQ_PASSWORD
                  )
      );

      return _state;
  }
}

class _ShutterState extends State {

  final MqttCommander _commander;

  String _shutterAction = "stop";

  _ShutterState(this._commander);

  @override
  void dispose() {
    super.dispose();
    _commander.disconnect();
  }

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
        key: new GlobalKey<ScaffoldState>(),
        appBar: AppBar(
```

```
                automaticallyImplyLeading: true,
                title: Text("Shutter Control Panel"),
                elevation: 10.0,
                centerTitle: true,
                backgroundColor: Colors.teal,
        ),
        body: new Padding(
            padding: const EdgeInsets.all(0.0),
            child: _buildBody(),
        ),
        bottomNavigationBar: BottomNavigationBar(
            currentIndex: 0, // This will be set when a new tab is tapped
            backgroundColor: Colors.teal,
            items: [
                BottomNavigationBarItem(
                    icon: new Icon(Icons.home, color: Colors.white,),
                    title: new Text('Home', style: TextStyle(color: Colors.whi
te),),
                ),
                BottomNavigationBarItem(
                    icon: new Icon(Icons.account_circle, color: Colors.white),
                    title: new Text('Me', style: TextStyle(color: Colors.white
),),
                )
            ],
            onTap: (index) => Navigator.push(context, MaterialPageRoute<void>(
                builder: (BuildContext context){
                  return Theme(
                      data: GlobalConfig.myTheme.copyWith(platform: Theme.of(c
ontext).platform),
                      child: AppHome(),
                  );
                }
            )),
        ),
    );

  }

  Widget _buildBody(){
      return new Center(
        child: new Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: <Widget>[
                Row(
                    mainAxisSize: MainAxisSize.min,
                    mainAxisAlignment: MainAxisAlignment.center,
```

```
                    children: <Widget>[
                        IconButton(icon: new Icon(Icons.arrow_left, size: 48,
                            color: Colors.blueAccent,),
                            onPressed: (){
                              _shutterAction = 'left';
                              _commander.send(Commands.SHUTTER_CONTROL, _shutt
erAction);
                        }),
                        IconButton(icon: new Icon(Icons.stop, size: 48,
                            color: Colors.blueAccent,),
                            onPressed: (){
                              _shutterAction = 'stop';
                              _commander.send(Commands.SHUTTER_CONTROL, _shutt
erAction);
                        },),
                        IconButton(icon: new Icon(Icons.arrow_right, size: 48,

                            color: Colors.blueAccent,),
                            onPressed: (){
                              _shutterAction = 'right';
                              _commander.send(Commands.SHUTTER_CONTROL, _shutt
erAction);
                        },)
                    ],
                )

            ],
        ),
    );
  }

}
```

Code 16 shutter.dart

5.2.4.2.4 CCTV Device

The device is an abstraction with the CCTV camera in the real world.

- cctv.dart

UI:

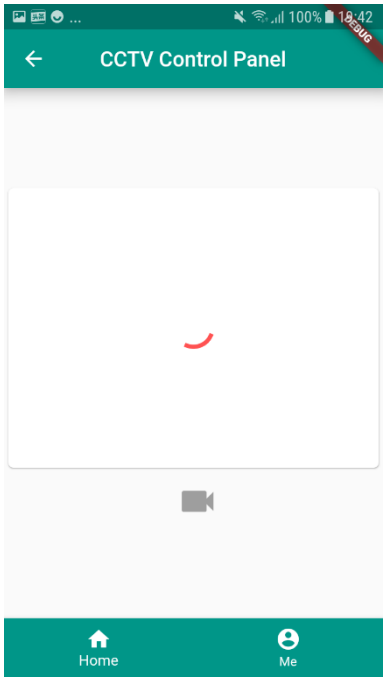

Figure 14 cctv control panel screen

Path: flutterapp/lib/control/devices/cctv.dart

Description: The class is the UI of the CCTV control panel on the mobile app to turn on or off the CCTV camera through the video button on this screen. The live view the camera captured will be shown up on the screen in real-time.

```
import 'package:flutter/material.dart';
import 'package:flutterapp/common/const/commands.dart';
import 'package:flutterapp/common/const/globalConf.dart';
import 'package:flutterapp/common/util/mjpegViewer.dart';
import 'package:flutterapp/common/util/mqttCommander.dart';
import 'package:flutterapp/home/appHome.dart';

class CCTVWidget extends StatefulWidget {

  @override
  State<StatefulWidget> createState() => new _CCTVState();

}

class _CCTVState extends State<CCTVWidget> {

  final MqttCommander _commander = new MqttCommander(
    GlobalConfig.LOCAL_MQTT_BROKER_HOST,
    GlobalConfig.LOCAL_MQTT_BROKER_LISTEN_PORT,
    GlobalConfig.MQTT_CLIENT_IDENTIFIER_CCTV
  );
```

```
  bool _isVideoLoading = false;

  @override
  void dispose() {
    super.dispose();
    _commander.disconnect();
  }

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
        key: new GlobalKey<ScaffoldState>(),
        appBar: AppBar(
                automaticallyImplyLeading: true,
                title: Text("CCTV Control Panel"),
                elevation: 10.0,
                centerTitle: true,
                backgroundColor: Colors.teal,
        ),
        body: new Padding(
            padding: const EdgeInsets.all(0.0),
            child: _buildBody(),
        ),
        bottomNavigationBar: BottomNavigationBar(
            currentIndex: 0, // This will be set when a new tab is tapped
            backgroundColor: Colors.teal,
            items: [
                BottomNavigationBarItem(
                    icon: new Icon(Icons.home, color: Colors.white,),
                    title: new Text('Home', style: TextStyle(color: Colors.whi
te),),
                ),
                BottomNavigationBarItem(
                    icon: new Icon(Icons.account_circle, color: Colors.white),
                    title: new Text('Me', style: TextStyle(color: Colors.white
),),
                )
            ],
            onTap: (index) => Navigator.push(context, MaterialPageRoute<void>(
                builder: (BuildContext context){
                  return Theme(
                      data: GlobalConfig.myTheme.copyWith(platform: Theme.of(c
ontext).platform),
                      child: AppHome(),
                  );
                }
            )),
        ),
```

```
    );
  }

  Widget _buildBody(){

      return new Container(
          child: new Center(
                 child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    crossAxisAlignment: CrossAxisAlignment.center,
                    children: <Widget>[
                        new Card(
                            shape: RoundedRectangleBorder(borderRadius: Border
Radius.circular(5.0)),
                            child: Container(
                                    height: 260,
                                    margin: EdgeInsets.only(top: 3.0, bottom:
3.0),
                                    alignment: Alignment.topRight,
                                    child: _isVideoLoading==false ?
                                          Center(child: CircularProgressIndica
tor()) :
                                          MjpegView(url: GlobalConfig.CCTV_VID
EO_STREAM_URL, fps: 100),
                                    ),
                        ),
                        IconButton(
                            icon: Icon(Icons.videocam, size:40,),
                            onPressed: () {
                                _commander.send(Commands.CCTV_CONTROL, _isVide
oLoading ? 'off' : 'on');
                                _updateVideoState();
                            },
                            color: _isVideoLoading == false ? Colors.grey : Co
lors.blue,
                        ),
                ],
            ),
          ),
      );
  }

  void _updateVideoState() async {
    //when the command has already sent to CCTV, then will get the live video
after 3 seconds.
    await Future.delayed(const Duration(milliseconds: 3000), () {
      setState(() {
```

```
        _isVideoLoading = _isVideoLoading ? false : true; //notificate the flu
tter to refresh to component.
      });
    });

  }

}
```

Code 17 cctv.dart

5.2.4.2.5. Door Access Devices

The devices in the door access subsystem is an abstraction with the door access system in the real world.

- doorAccess.dart

UI:

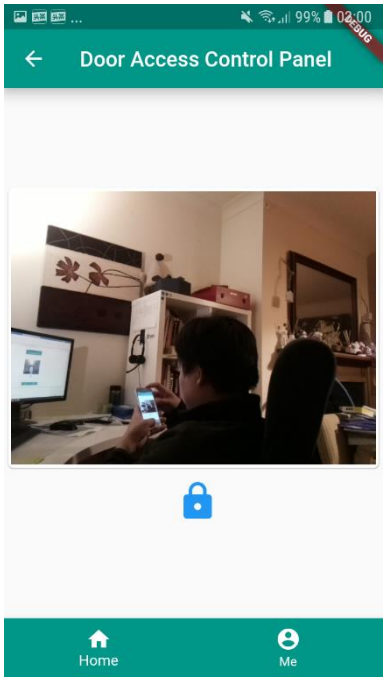

Figure 15 door access control panel screen

Path: flutterapp/lib/control/devices/dooraccess.dart

Description: The class is the UI of the door access subsystem control panel on the mobile app to unlock or unlock the door through the lock button on this screen. The live view the door camera captured will be shown up on the screen in real-time.

```
import 'package:flutter/material.dart';
```

```dart
import 'package:flutterapp/common/const/commands.dart';
import 'package:flutterapp/common/const/globalConf.dart';
import 'package:flutterapp/common/util/mjpegViewer.dart';
import 'package:flutterapp/common/util/mqttCommander.dart';
import 'package:flutterapp/home/appHome.dart';

class DoorAccessWidget extends StatefulWidget {
  @override
  State<StatefulWidget> createState() => _DoorAccessState();

}

class _DoorAccessState extends State<StatefulWidget> {

  final MqttCommander _commander = new MqttCommander(
    GlobalConfig.LOCAL_MQTT_BROKER_HOST,
    GlobalConfig.LOCAL_MQTT_BROKER_LISTEN_PORT,
    GlobalConfig.MQTT_CLIENT_IDENTIFIER_DOORACCESS
  );

  bool _isVideoLoading = false;
  bool _isLocked = true;

  @override
  void initState() {
    super.initState();
    Future.delayed(const Duration(seconds: 1), (){
      ///to notify the door camera to turn on.
      _commander.send(Commands.DOOR_CAMERA_CONTROL, "on");
    });

  }

  @override
  void dispose(){
    super.dispose();
    _isVideoLoading = false;
    _commander.send(Commands.DOOR_CAMERA_CONTROL, "off");
    _commander.disconnect();
  }

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
        key: new GlobalKey<ScaffoldState>(),
        appBar: AppBar(
                automaticallyImplyLeading: true,
                title: Text("Door Access Control Panel"),
```

```
                elevation: 10.0,
                centerTitle: true,
                backgroundColor: Colors.teal,
        ),
        body: new Padding(
            padding: const EdgeInsets.all(0.0),
            child: _buildBody(),
        ),
        bottomNavigationBar: BottomNavigationBar(
            currentIndex: 0, // This will be set when a new tab is tapped
            backgroundColor: Colors.teal,
            items: [
                BottomNavigationBarItem(
                    icon: new Icon(Icons.home, color: Colors.white,),
                    title: new Text('Home', style: TextStyle(color: Colors.whi
te),),
                ),
                BottomNavigationBarItem(
                    icon: new Icon(Icons.account_circle, color: Colors.white),
                    title: new Text('Me', style: TextStyle(color: Colors.white
),),
                )
            ],
            onTap: (index) => Navigator.push(context, MaterialPageRoute<void>(
                builder: (BuildContext context){
                  return Theme(
                      data: GlobalConfig.myTheme.copyWith(platform: Theme.of(c
ontext).platform),
                      child: AppHome(),
                  );
                }
            )),
        ),
    );
  }

  Widget _buildBody(){
      /// To notifiy the UI engine to re-render.
      _updateVideoState();

      return new Container(
          child: new Center(
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    crossAxisAlignment: CrossAxisAlignment.center,
                    children: <Widget>[
                        new Card(
```

```
                            shape: RoundedRectangleBorder(borderRadius: Border
Radius.circular(5.0)),
                            child: Container(
                                    height: 260,
                                    margin: EdgeInsets.only(top: 3.0, bottom:
3.0),
                                    alignment: Alignment.center,
                                    child: !_isVideoLoading ?
                                          Center(child: CircularProgressIndica
tor()) :
                                          MjpegView(url: GlobalConfig.DOOR_CAM
ERA_VIDEO_STREAM_URL, fps: 100),
                                ),
                        ),
                        Row(
                            mainAxisAlignment: MainAxisAlignment.center,
                            children: <Widget>[
                                _lockButtonWidget(),

                            ],
                        ),

                ],
            ),
          ),
      );
  }


  Widget _lockButtonWidget (){
      return IconButton(
                icon: Icon(((_isLocked) ? Icons.lock: Icons.lock_open), size:4
0,),
                onPressed: () {
                    setState(() {
                        _isLocked = _isLocked ? false : true;
                    });

                    _commander.send(Commands.DOOR_LOCKER_CONTROL, _isLocked ?
"lock" : "unlock");

                },
                color: Colors.blue,
            );
  }

  void _updateVideoState() async {
    //when the command has already sent to CCTV, then will get the live video
after 3 seconds.
```

```
    await Future.delayed(const Duration(milliseconds: 3000), () {
      if(!_isVideoLoading){
        setState(() {
          _isVideoLoading = true; //notificate the flutter to refresh to compo
nent.
        });
      }

    });

  }


}
```

Code 18 doorAccess.dart

5.2.4.2.6 Sensor Device

The device is an abstraction with the temperature and humidity sensor for indoor and outdoor in the real world.

- atmosphere.dart

UI:

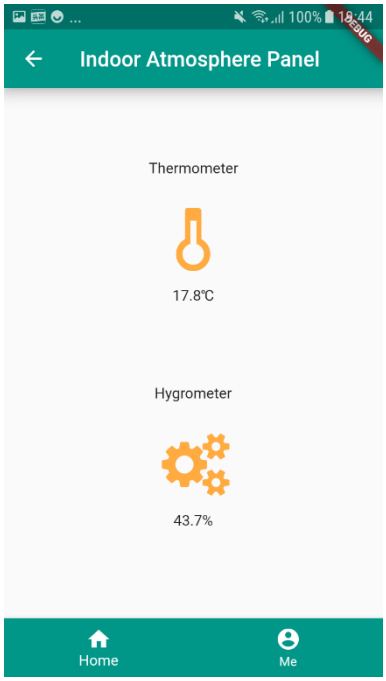

Figure 16 indoor sensor panel screen

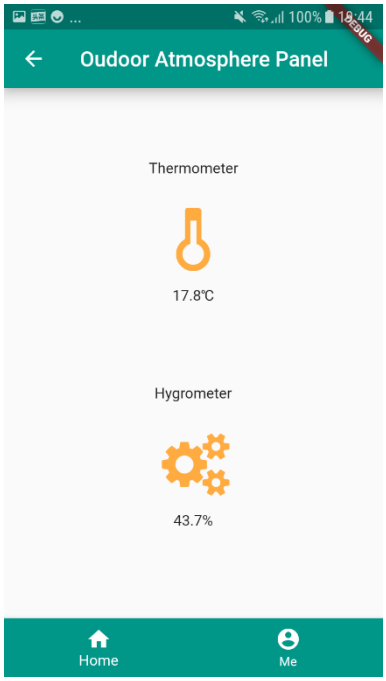

Figure 17 outdoor sensor panel screen

Path: flutterapp/lib/control/devices/atmosphere.dart

Description: The class is the UI of the temperature & humidity sensor control panel for indoor and outdoor on the mobile app to obtain the measurements from the sensors in real-time and then shown up the measurement on the screen.

```
import 'package:flutter/material.dart';
import 'package:flutterapp/common/const/commands.dart';
import 'package:flutterapp/common/const/globalConf.dart';
import 'package:flutterapp/common/icon/flutterCustomIcon.dart';
import 'package:flutterapp/common/util/mqttCommander.dart';
import 'package:flutterapp/home/appHome.dart';
import 'package:mqtt_client/mqtt_client.dart';

class AtmosphereWidget extends StatefulWidget {

  /// to decide the temperature and humidity measurement in which environment.
  final String indoorOrOutdoor;

  const AtmosphereWidget(this.indoorOrOutdoor, {Key key}) : super(key: key);

  @override
  State<StatefulWidget> createState() => _AtmosphereState();

}

class _AtmosphereState extends State<AtmosphereWidget> {
```

```
final MqttCommander _commander = new MqttCommander(
  GlobalConfig.LOCAL_MQTT_BROKER_HOST,
  GlobalConfig.LOCAL_MQTT_BROKER_LISTEN_PORT,
  GlobalConfig.MQTT_CLIENT_IDENTIFIER_ATMOSPHERE
);

String _temperatureValue = GlobalConfig.DEFAULT_INITIAL_TEMPERATURE_VALUE;
String _humidityValue = GlobalConfig.DEFAULT_INITIAL_HUMIDITY_VALUE;

/// To decide if it continuously read the data from the sensor.
bool _isPlanToRead;

///To specific the interval seconds of the reading data from the sensor.
///Here just given the default value is 1 second interval between each read.
final int readIntervalSeconds = 1;

@override
void initState() {
  super.initState();
  _isPlanToRead = true;
  // notify the device to send the measurement data to mobile app interval.
  toReadValueFromSensor(readIntervalSeconds);
}

@override
void dispose(){
  super.dispose();
  _isPlanToRead = false;
  _commander.unsubscribe(
    widget.indoorOrOutdoor == GlobalConfig.INDOOR ?
      Commands.INDOOR_TEMPERATURE_HUMIDITY_DATA_RECEIVE :
      Commands.OUTDOOR_TEMPERATURE_HUMIDITY_DATA_RECEIVE
    );
  _commander.disconnect();
}

@override
Widget build(BuildContext context) {

  _commander.receive(
    widget.indoorOrOutdoor == GlobalConfig.INDOOR ?
      Commands.INDOOR_TEMPERATURE_HUMIDITY_DATA_RECEIVE :
      Commands.OUTDOOR_TEMPERATURE_HUMIDITY_DATA_RECEIVE,
    refreshMeasurement);

  return new Scaffold(
          key: new GlobalKey<ScaffoldState>(),
```

```
            appBar: AppBar(
                    automaticallyImplyLeading: true,
                    title: Text(
                      widget.indoorOrOutdoor == GlobalConfig.INDOOR ?
                        "Indoor Atmosphere Panel" :
                        "Oudoor Atmosphere Panel"
                      ),
                    elevation: 10.0,
                    centerTitle: true,
                    backgroundColor: Colors.teal,
            ),
            body: new Padding(
                padding: const EdgeInsets.all(0.0),
                child: _buildBody(),
            ),
            bottomNavigationBar: BottomNavigationBar(
                currentIndex: 0, // This will be set when a new tab is tapped
                backgroundColor: Colors.teal,
                items: [
                    BottomNavigationBarItem(
                        icon: new Icon(Icons.home, color: Colors.white,),
                        title: new Text('Home', style: TextStyle(color: Colors
.white),),
                    ),
                    BottomNavigationBarItem(
                        icon: new Icon(Icons.account_circle, color: Colors.whi
te),
                        title: new Text('Me', style: TextStyle(color: Colors.w
hite),),
                    )
                ],
                onTap: (index) => Navigator.push(context, MaterialPageRoute<vo
id>(
                    builder: (BuildContext context){
                      return Theme(
                          data: GlobalConfig.myTheme.copyWith(platform: Theme.
of(context).platform),
                          child: AppHome(),
                      );
                    }
                )),
            ),
        );
  }

  Widget _buildBody(){
        return new Container(
          child: new Center(
```

```
                child: Column(
                   mainAxisAlignment: MainAxisAlignment.center,
                   crossAxisAlignment: CrossAxisAlignment.center,
                   children: <Widget>[
                       Container(
                           margin: EdgeInsets.all(30.0),
                           child: Text("Thermometer"),
                       ),
                       Icon(FlutterCustomIcon.thermometer, size: 60, color: C
olors.orangeAccent,),
                       Container(
                           height: 60,
                           margin: EdgeInsets.only(top: 15.0, bottom: 3.0),
                           child: Text(_temperatureValue+GlobalConfig.CElSIUS
_SYMBOL),
                       ),
                       Container(
                           margin: EdgeInsets.all(30.0),
                           child: Text("Hygrometer"),
                       ),
                       Icon(FlutterCustomIcon.cog_alt, size: 60, color: Color
s.orangeAccent),
                       Container(
                           height: 60,
                           margin: EdgeInsets.only(top: 15.0, bottom: 3.0),
                           child: Text(_humidityValue+"%"),
                       ),

                   ],
               ),
           ),
       );
   }

   Future toReadValueFromSensor(int second) async{

     while (_isPlanToRead){
       /// Publish the message to the topic
       _commander.send(
         widget.indoorOrOutdoor == GlobalConfig.INDOOR ?
           Commands.INDOOR_TEMPERATURE_HUMIDITY_DATA_READ:
           Commands.OUTDOOR_TEMPERATURE_HUMIDITY_DATA_READ,
           'on'
       );
       await MqttUtilities.asyncSleep(second);
     }
   }
```

```
    // This is a callback method that is called when the message has already r
eceived.
    void refreshMeasurement(String message){
      if(message.contains("|")){
        List<String> result = message.split("|");
        setState(() {
         _temperatureValue = result[0];
         _humidityValue = result[1];
        });

      }

    }

}
```

Code 19 atmosphere.dart

5.2.5 Options Classes

The classes defined respectively are the networking options and the entertainment game options used to this project is placed under the option folder of the Flutter app. The option folder structure diagram below.

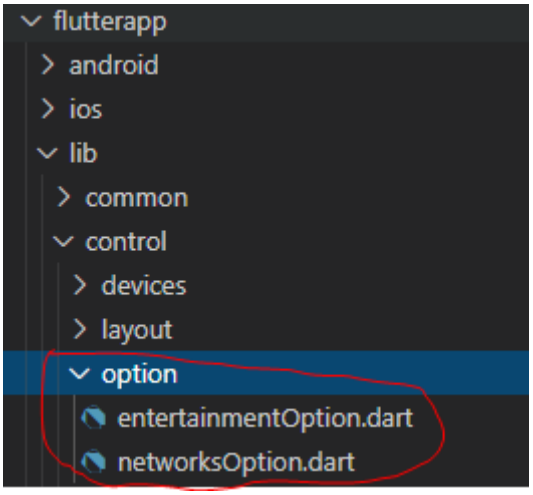

Figure 18 option classes structure

5.2.5.1. Networking Options Class

There are two networking options respectively are Wi-Fi and Internet in this class to be chosen by user when the user would like to control the light, fan, and shutter.

- networksOption.dart

UI:

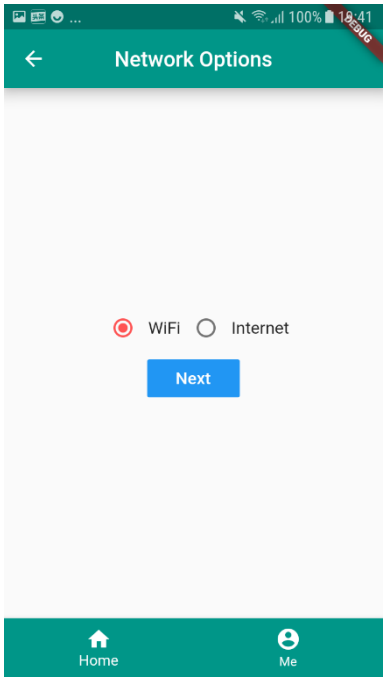

Figure 19 networking options screen

Path: flutterapp/lib/control/option/networksOption.dart

Description: when the user would like to turn on or off the light can be chosen through either Wi-Fi or Internet.

```
import 'package:flutter/material.dart';
import 'package:flutterapp/common/const/globalConf.dart';
import 'package:flutterapp/control/devices/fan.dart';
import 'package:flutterapp/control/devices/light.dart';
import 'package:flutterapp/control/devices/shutter.dart';

class NetworksOptionWidget extends StatefulWidget {

  final DEVICE_NAME deviceName;

  const NetworksOptionWidget({Key key, @required this.deviceName}) : super(key
: key);

  @override
  State<StatefulWidget> createState() => new _NetworksOptionState();

}

class _NetworksOptionState extends State<NetworksOptionWidget> {

  int radioValue = 0;

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
```

```
        key: new GlobalKey<ScaffoldState>(),
        appBar: AppBar(
                automaticallyImplyLeading: true,
                title: Text("Network Options"),
                elevation: 10.0,
                centerTitle: true,
                backgroundColor: Colors.teal,
        ),
        body: new Padding(
            padding: const EdgeInsets.all(0.0),
            child: _buildBody(),
        ),
        bottomNavigationBar: BottomNavigationBar(
            currentIndex: 0, // This will be set when a new tab is tapped
            backgroundColor: Colors.teal,
            items: [
                BottomNavigationBarItem(
                    icon: new Icon(Icons.home, color: Colors.white,),
                    title: new Text('Home', style: TextStyle(color: Colors.whi
te),),
                ),
                BottomNavigationBarItem(
                    icon: new Icon(Icons.account_circle, color: Colors.white),
                    title: new Text('Me', style: TextStyle(color: Colors.white
),),
                )
            ],
        ),
    );
  }


  void handleRadioValueChanged(int value) {
    setState(() {
      radioValue = value;
    });
  }

  Widget _buildBody(){
      return new Center(
      child: new Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
            Row(
                mainAxisSize: MainAxisSize.min,
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                    Radio<int>(
```

```
                    value: 0,
                    groupValue: radioValue,
                    onChanged: handleRadioValueChanged,
                  ),
                  new Text(GlobalConfig.NETWORK_OPTION_WIFI,style: new TextS
tyle(fontSize: 16.0)),
                  Radio<int>(
                    value: 1,
                    groupValue: radioValue,
                    onChanged: handleRadioValueChanged,
                  ),
                  new Text(GlobalConfig.NETWORK_OPTION_INTERNET, style: new
TextStyle(fontSize: 16.0,)),
                ],
            ),
            Row(
                mainAxisSize: MainAxisSize.min,
                mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[
                    FlatButton(
                        color: Colors.blue,
                        textColor: Colors.white,
                        disabledColor: Colors.grey,
                        disabledTextColor: Colors.black,
                        padding: EdgeInsets.all(8.0),
                        splashColor: Colors.blueAccent,
                        child: Text('Next', style: new TextStyle(fontSize: 16.
0)),
                        onPressed: () {
                            Navigator.push(context, MaterialPageRoute<void>(
                                builder: (BuildContext context){
                                    return Theme(
                                        data: GlobalConfig.myTheme.copyWith(pl
atform: Theme.of(context).platform),
                                        child: _findWidget(this.widget.deviceN
ame),
                                    );
                                }
                            ));
                        },
                    ),
                ],
            ),
        ]
      )
    );
  }
```

```
  Widget _findWidget(DEVICE_NAME deviceName){

    Widget widget;

    switch(deviceName){
      case DEVICE_NAME.LIGHT:
        widget = new LightWidget(radioValue);
        break;

      case DEVICE_NAME.FAN:
        widget = new FanWidget(radioValue);
        break;

      case DEVICE_NAME.SHUTTER:
        widget = new ShutterWidget(radioValue);
        break;

      default:
        break;

    }

    return widget;
  }

}
```

Code 20 networksOption.dart

5.2.5.2. Entertainment Game Options Class

There are two game options respectively are bomb man and word game in this class to be chosen by user when the user would like to play the game which is chosen.

- entertainmentOption.dart

UI:

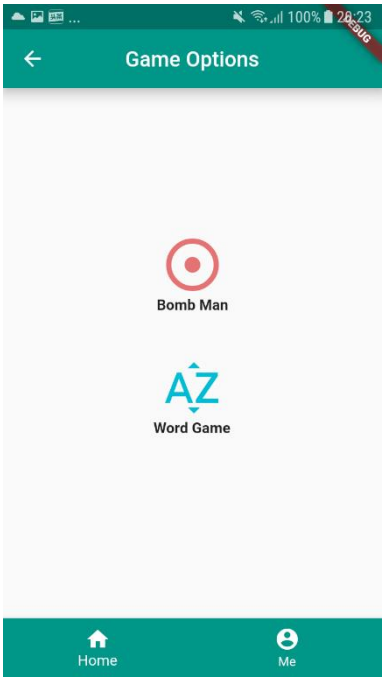

Figure 20 game option screen

Path: flutterapp/lib/control/option/entertainmentOption.dart

Description: when the user would like to play the game can be chosen either the bomb man or the word game.

```
import 'package:flutter/material.dart';
import 'package:flutterapp/common/const/globalConf.dart';
import 'package:flutterapp/home/appHome.dart';

class EntertainmentOptionWidget extends StatefulWidget {
  @override
  State<StatefulWidget> createState() => _EntertainmentOptionState();

}

class _EntertainmentOptionState extends State<StatefulWidget> {
  @override
  Widget build(BuildContext context) {
    return new Scaffold(
        key: new GlobalKey<ScaffoldState>(),
        appBar: AppBar(
                automaticallyImplyLeading: true,
                title: Text("Game Options"),
                elevation: 10.0,
                centerTitle: true,
                backgroundColor: Colors.teal,
        ),
```

```
        body: new Padding(
            padding: const EdgeInsets.all(0.0),
            child: _buildBody(),
        ),
        bottomNavigationBar: BottomNavigationBar(
            currentIndex: 0, // This will be set when a new tab is tapped
            backgroundColor: Colors.teal,
            items: [
                BottomNavigationBarItem(
                    icon: new Icon(Icons.home, color: Colors.white,),
                    title: new Text('Home', style: TextStyle(color: Colors.whi
te),),
                ),
                BottomNavigationBarItem(
                    icon: new Icon(Icons.account_circle, color: Colors.white),
                    title: new Text('Me', style: TextStyle(color: Colors.white
),),
                )
            ],
            onTap: (index) => Navigator.push(context, MaterialPageRoute<void>(
              builder: (BuildContext context){
                  return Theme(
                      data: GlobalConfig.myTheme.copyWith(platform: Theme.of(c
ontext).platform),
                      child: AppHome(),
                  );
              }
            )),
        ),
    );

  }

    Widget _buildBody(){
      return new Column(
        crossAxisAlignment: CrossAxisAlignment.center,
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Row(
            crossAxisAlignment: CrossAxisAlignment.center,
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
            Container(
              padding: EdgeInsets.only(top:3.0),
              child: Icon(
                Icons.adjust,
                size: 60,
                color: Colors.red[300],
```

```
                ),
            ),
          ],),
          Row(
            crossAxisAlignment: CrossAxisAlignment.center,
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
            Container(
              padding: EdgeInsets.only(bottom:38.0),
              child: Text("Bomb Man", style: const TextStyle(fontWeight: FontW
eight.bold, fontSize: 14),),
            ),
          ],),
          Row(
            crossAxisAlignment: CrossAxisAlignment.center,
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
            Container(
              padding: EdgeInsets.only(top:3.0),
              child: Icon(
                Icons.sort_by_alpha,
                size: 60,
                color:Colors.cyan,),
            ),
          ],),
          Row(
            crossAxisAlignment: CrossAxisAlignment.center,
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
            Container(
              padding: EdgeInsets.only(bottom:38.0),
              child: Text("Word Game", style: const TextStyle(fontWeight: Font
Weight.bold, fontSize: 14),),
            ),
          ],),
        ],
      );
    }

}
```

Code 21 entertainmentOption.dart

5.2.6 Android Platform Configuration

Due to this app should be granted the permissions through the mobile phone such as microphone permission and the Internet access permission, so the configuration file on the mobile phone will be changed.

- **AndroidManifest.xml**

Path: flutterapp/android/app/src/main/AndroidManifest.xml

Description: In this case, I am going to focus on the Android OS main configuration named AndroidManifest.xml. There are two configuration items should be added into it that are 'android.permission.RECORD_AUDIO' and 'android.permission.INTERNET'.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.flutterapp">
    <!-- io.flutter.app.FlutterApplication is an android.app.Application that
         calls FlutterMain.startInitialization(this); in its onCreate method.
         In most cases you can leave this as-
is, but you if you want to provide
         additional functionality it is fine to subclass or reimplement
         FlutterApplication and put your custom class here. -->

    <uses-permission android:name="android.permission.RECORD_AUDIO" />

    <!-- Flutter needs it to communicate with the running application
         to allow setting breakpoints, to provide hot reload, etc.
    -->
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:name="io.flutter.app.FlutterApplication"
        android:label="flutterapp"
        android:icon="@mipmap/ic_launcher">
        <activity
            android:name=".MainActivity"
            android:launchMode="singleTop"
            android:theme="@style/LaunchTheme"
            android:configChanges="orientation|keyboardHidden|keyboard|screenS
ize|smallestScreenSize|locale|layoutDirection|fontScale|screenLayout|density|u
iMode"
            android:hardwareAccelerated="true"
            android:windowSoftInputMode="adjustResize">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <!-- Don't delete the meta-data below.
             This is used by the Flutter tool to generate GeneratedPluginRegis
trant.java -->
        <meta-data android:name="flutterEmbedding" android:value="2" />
    </application>
</manifest>
```

Code 22 androidManifest.xml

5.3. Python Scripts

In this project, the devices are controlled by the python scripts that executed on the Raspberry Pi when the user send the message to the Raspberry Pi.

5.3.1. Controller Scripts

The controller script that is an entry point when python scripts program is launched on the Raspberry Pi. In this script, it will run three thread respective for local MQTT broker, cloud MQTT broker, and the doorbell listening. The controller scripts structure diagram below.

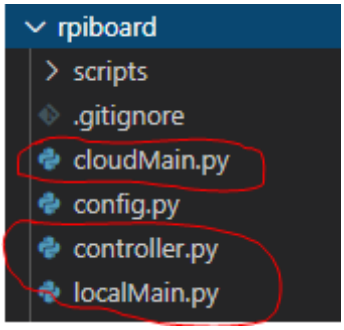

Figure 21 controller python scripts structure

- **controller.py**

Path: rpiboard/controller.py

Description: this is an entry point script that called the localMain.py and cloudMain.py to execute in respectively the threads.

```
#! /usr/bin/python3.7
import sys
sys.path.append('/opt/rpiboard')

from localMain import main as localMain
from cloudMain import main as cloudMain
from scripts.door.bell.control import bellListenning

import _thread

if __name__ == "__main__":

    try:
        _thread.start_new_thread( localMain, () )
        _thread.start_new_thread( cloudMain, () )
        _thread.start_new_thread( bellListenning, () )
    except:
        print ("Error: unable to start thread")
```

Code 23 controller.py

- **localMain.py**

Path: rpiboard/localMain.py

Description: this script is running as an MQTT client to subscribe/publish the message to/from in which the local router has already installed the MQTT broker.

```
#! /usr/bin/python3.7
import sys
sys.path.append('/opt/rpiboard')

from config import MQTT_CLIENT_IDENTIFIER, MQTT_BROKER_ADDRESS
from scripts.cctv.control import cctvOn, cctvOff
from scripts.light.control import lightOn, lightOff
from scripts.fan.control import fanOff, fanSpeed
from scripts.shutter.control import shutterMove, shutterStop
from scripts.door.bell.control import bellListenning
from scripts.door.lock.control import lock, unlock
from scripts.door.camera.control import cameraOn, cameraOff
from scripts.atmosphere.indoor.control import atmosphereDataRead as indoorAtmo
sphereDataRead
from scripts.atmosphere.outdoor.control import atmosphereDataRead as outdoorAt
mosphereDataRead

import paho.mqtt.client as mqtt
import time


print("creating new instance")
client = mqtt.Client(MQTT_CLIENT_IDENTIFIER) #create new instance

print("connecting to broker")
client.connect(MQTT_BROKER_ADDRESS) #connect to broker

def on_message(client, userdata, message):
    status = str(message.payload.decode("utf-8"))
    print("message received " , status)
    print("message topic=",message.topic)
    print("message qos=",message.qos)
    print("message retain flag=",message.retain)

    if message.topic=='topic/camera/cctv':
        if status=='on':
            print("turn camera on")
            cctvOn()
```

```
        else:
            print("turn camera off")
            cctvOff()
    elif message.topic=='topic/light/control':
        if status=='on':
            print('turn light on by WIFI')
            lightOn()
        else:
            print('turn light off by WIFI')
            lightOff()
    elif message.topic=='topic/fan/control':
        if status=='off':
            print('turn fan off')
            fanOff()
        elif status.isnumeric():
            print('fan speed adjust')
            fanSpeed(int(status))
    elif message.topic=='topic/shutter/control':
        if status=='stop':
            print('shutter stop')
            shutterStop()
        else:
            print('shutter move to ', status)
            shutterMove(status)
    elif message.topic=='topic/door/camera/control':
        if status=='on':
            print("turn door camera on")
            cameraOn()
        else:
            print("turn door camera off")
            cameraOff()
    elif message.topic=='topic/door/lock/control':
        if status=='lock':
            print("door locked")
            lock()
        else:
            print("door unlocked")
            unlock()
    elif message.topic=='topic/indoor/measurement/read':
        if status=='on':
            print("indoor measurement reading switch on")
            result = indoorAtmosphereRead()
            atmosphere_publish("topic/indoor/measurement/data", result)
    elif message.topic=='topic/outdoor/measurement/read':
        if status=='on':
            print("outdoor measurement reading switch on")
            result = outdoorAtmosphereRead()
            atmosphere_publish("topic/outdoor/measurement/data", result)
```

```
client.on_message=on_message #attach function to callback


########################################
def main():

    client.loop_start() #start the loop
    print("Subscribing to camera topic","topic/camera/cctv by WIFI")
    client.subscribe("topic/camera/cctv")
    print("Subscribing to ligh topic","topic/light/control by WIFI")
    client.subscribe("topic/light/control")
    print("Subscribing to fan topic","topic/fan/control by WIFI")
    client.subscribe("topic/fan/control")
    print("Subscribing to shutter topic","topic/shutter/control by WIFI")
    client.subscribe("topic/shutter/control")
    print("Subscribing to door camera topic","topic/door/camera/control by WIF
I")
    client.subscribe("topic/door/camera/control")
    print("Subscribing to door lock topic","topic/door/lock/control by WIFI")
    client.subscribe("topic/door/lock/control")
    print("Subscribing to indoor temperature and humidity read topic","topic/i
ndoor/measurement/read by WIFI")
    client.subscribe("topic/indoor/measurement/read")
    print("Subscribing to outdoor temperature and humidity read topic","topic/
outdoor/measurement/read by WIFI")
    client.subscribe("topic/outdoor/measurement/read")

    time.sleep(24*60*60) # wait
    print('time up to stop')
    client.loop_stop() #stop the loop
```

Code 24 localMain.py

- **cloudMain.py**

Path: rpiboard/cloudMain.py

Description: this script is running as an MQTT client to subscribe/publish the message to/from where the AWS ActiveMQ is cloud MQTT broker.

```
#! /usr/bin/python3.7

import sys
sys.path.append('/opt/rpiboard')
```

```
from scripts.light.control import lightOn, lightOff
import paho.mqtt.client as mqtt
from config import AWS_ACTIVEMQ_HOST, AWS_ACTIVEMQ_USER, AWS_ACTIVEMQ_PASSWORD
, AWS_ACTIVEMQ_PORT

import time

def on_message(client, userdata, message):
    status = str(message.payload.decode("utf-8"))
    print("message received " , status)
    print("message topic=",message.topic)
    print("message qos=",message.qos)
    print("message retain flag=",message.retain)

    if message.topic=='topic/light/control':
        if status=='on':
            print('light on by INTERNET')
            lightOn()
        else:
            print('light off by INTERNET')
            lightOff()
    elif message.topic=='topic/fan/control':
        if status=='off':
            print('turn fan off by INTERNET')
            fanOff()
        elif status.isnumeric():
            print('fan speed adjust by INTERNET')
            fanSpeed(int(status))
    elif message.topic=='topic/shutter/control':
        if status=='stop':
            print('shutter stop by INTERNET')
            shutterStop()
        else:
            print('shutter move to ', status, 'by INTERNET')
            shutterMove(status)

def main():

    print('connecting ',AWS_ACTIVEMQ_HOST)
    client = mqtt.Client('RPi')
    client.username_pw_set(AWS_ACTIVEMQ_USER, AWS_ACTIVEMQ_PASSWORD)
    client.on_message=on_message
    client.tls_set('/home/pi/cert/AmazonRootCA1.pem')
    client.connect(AWS_ACTIVEMQ_HOST, port=AWS_ACTIVEMQ_PORT)
    print("Subscribing to ligh topic","topic/light/control by INTERNET")
    client.subscribe(topic='topic/light/control')
    print("Subscribing to fan topic","topic/fan/control by INTERNET")
```

```
    client.subscribe("topic/fan/control")
    print("Subscribing to shutter topic","topic/shutter/control by INTERNET")
    client.subscribe("topic/shutter/control")
    client.loop_start() #start the loop
    time.sleep(24*60*60) # wait
    client.loop_stop() #stop the loop
```

Code 25 cloudMain.py

5.3.2. Device Control Scripts

The each of those device control scripts defined under the scripts folder is executed through the Raspberry Pi. Currently, there is a set of devices is controlled such as table light, fan, shutter, cctv camera, door access system devices and temperature & humidity sensors. Those device control scripts will be described in detail.

The scripts folder structure diagram below.

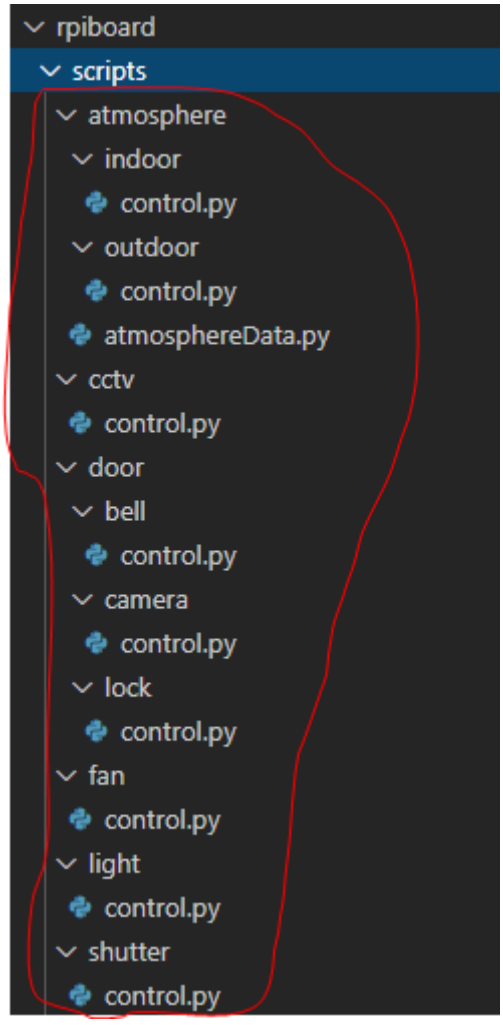

Figure 22 device control scripts structure

5.3.2.1. Light Control Script

This script controls the table light in the real world through the signal sent by the Raspberry Pi.

- **light/control.py**

Path: rpiboard/scripts/light/control.py

Description: The script maintains the status of the table light to turn it on or off.

```
#! /usr/bin/python3.7

import RPi.GPIO as GPIO
from config import LIGHT_RELAY_CHANNEL

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)
GPIO.setup(LIGHT_RELAY_CHANNEL, GPIO.OUT)

def lightOn():
    GPIO.output(LIGHT_RELAY_CHANNEL, GPIO.HIGH)

def lightOff():
    GPIO.output(LIGHT_RELAY_CHANNEL, GPIO.LOW)
    #GPIO.cleanup()
```

Code 26 light/control.py

5.3.2.2. Fan Control Script

This script controls the fan in real world through the signal sent by the Raspberry Pi.

- **fan/control.py**

Path: rpiboard/scripts/fan/control.py

Description: The script maintains the status of the fan to turn it on or off, even adjust its running speed.

```
import RPi.GPIO as GPIO
from config import FAN_PWM_CHANNEL

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)
GPIO.setup(FAN_PWM_CHANNEL, GPIO.OUT)
```

```
pwm=GPIO.PWM(FAN_PWM_CHANNEL, 100)

speed=35

isRunning=0

def start():
    pwm.start(0)
    isRunning=1

def fanOn():
    start()
    fanSpeed(speed)

def fanOff():
    speed=0
    isRunning=0
    pwm.stop()

def fanSpeed(speed):
    if isRunning==0:
        print('isRunining :', isRunning)
        start()
    pwm.ChangeDutyCycle(speed)
    print(speed)
```

Code 27 fan/control.py

5.3.2.3. Shutter Control Script

This script controls the shutter in the real world through the signal sent by the Raspberry Pi.

- **shutter/control.py**

Path: rpiboard/scripts/shutter/control.py

Description: The script maintains the status of the shutter to move it to right or left, even stop it.

```
import RPi.GPIO as GPIO
from config import SHUTTER_DIR_CHANNEL_1, SHUTTER_PWM_CHANNEL_2
import time

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)
GPIO.setup(SHUTTER_DIR_CHANNEL_1, GPIO.OUT)
GPIO.setup(SHUTTER_PWM_CHANNEL_2, GPIO.OUT)
```

```
speed=70;

pwm=GPIO.PWM(SHUTTER_PWM_CHANNEL_2, speed)

def speedup(direction):
    GPIO.output(SHUTTER_DIR_CHANNEL_1, direction)
    pwm.ChangeDutyCycle(speed)

def shutterMove(direction):
    pwm.start(0)
    if direction=='left':
        speedup(GPIO.LOW)
    else:
        speedup(GPIO.HIGH)

def shutterStop():
    pwm.start(0)
    pwm.stop()
```

Code 28 shutter/control.py

5.3.2.4. CCTV Control Script

This script controls the CCTV camera in the real world through the signal sent by the Raspberry Pi.

- **cctv/control.py**

Path: rpiboard/scripts/cctv/control.py

Description: : The script maintains the status of the CCTV camera to turn it on or off. In this script, it will issue the Linux command to maintain the camera to turn it on or off.

```
#! /usr/bin/python3.7
import os

def cameraOn():
    os.system("nohup mjpg_streamer -i 'input_uvc.so -d /dev/video1 -
r 640x480 -f 20' -o 'output_http.so -p 8080 -w /tmp/www' &")

def cameraOff():
    os.system('pkill mjpg_streamer')
```

Code 29 cctv/control.py

5.3.2.5 Door Access Control Scripts

The door access control scripts contain a set of the devices scripts respectively are doorbell, door camera, door lock that will be described in detail next sections.

- **bell/control.py**

Path: rpiboard/scripts/door/bell/control.py

Description: The script maintains the status of the doorbell to keep it beeps or quiet through the signal sent by the Raspberry Pi.

```
#! /usr/bin/python3.7

import RPi.GPIO as GPIO
from config import DOORBELL_CHANNEL, DOORBUTTON_CHANNEL
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(DOORBELL_CHANNEL, GPIO.OUT)
GPIO.output(DOORBELL_CHANNEL, GPIO.HIGH)

GPIO.setup(DOORBUTTON_CHANNEL, GPIO.IN, pull_up_down=GPIO.PUD_UP)

#That buzzer is triggered by low-level
def bellOn():
    GPIO.output(DOORBELL_CHANNEL, GPIO.LOW)

def bellOff():
    GPIO.output(DOORBELL_CHANNEL, GPIO.HIGH)

#hen the door button was pressed then let the bell beep.
def bellListenning():
    while True:
        time.sleep(1)
        if GPIO.input(DOORBUTTON_CHANNEL) == False:
            bellOn()
        else:
            bellOff()

def bellCallBack(button_callback):
    GPIO.add_event_detect(DOORBUTTON_CHANNEL,GPIO.RISING,callback=button_callb
ack) # Setup event on pin 10 rising edge
```

Code 30 bell/control.py

- **camera/control.py**

Path: rpiboard/scripts/door/camera/control.py

Description: The script maintains the status of the door camera to turn it on or off through the signal sent by the Raspberry Pi.

```
#! /usr/bin/python3.7
```

```
import os

def cameraOn():
    os.system("nohup mjpg_streamer -i 'input_raspicam.so -d /dev/video0' -
o 'output_http.so -p 8088 -w /tmp/www' &")

def cameraOff():
    os.system('pkill mjpg_streamer')
```

Code 31 camera/control.py

- **lock/control.py**

Path: rpiboard/scripts/door/lock/control.py

Description: The script maintains the status of the door lock to lock or unlock it through the signal sent by the Raspberry Pi.

```
#! /usr/bin/python3.7

import RPi.GPIO as GPIO
from config import LOCK_CHANNEL

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)
GPIO.setup(LOCK_CHANNEL, GPIO.OUT)

def lock():
    GPIO.output(LOCK_CHANNEL, GPIO.LOW)

def unlock():
    GPIO.output(LOCK_CHANNEL, GPIO.HIGH)
```

Code 32 lock/control.py

5.3.2.6 Temperature & Humidity Sensor Scripts

The temperature and humidity sensor scripts is designed to control the atmosphere sensor that collects the environment measurement then received by the Raspberry Pi.

- **atmosphereData.py**

Path: rpiboard/scripts/atmosphere/atmosphereData.py

Description: The script collects the temperature and humidity measurement by the sensor in real-time.

```
#! /usr/bin/python3.7
import Adafruit_DHT
```

```
def read(DHT_SENSOR, DHT_PIN):
    humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)

    if humidity is not None and temperature is not None:
        print("Temp={0:0.1f}*C  Humidity={1:0.1f}%".format(temperature, humidi
ty))
        return "{0:0.1f}|{1:0.1f}".format(temperature, humidity)
    else:
        print("Failed to retrieve data from the sensor")
```

Code 33 atmosphereData.py

- **indoor/control.py**

Path: rpiboard/scripts/atmosphere/indoor/control.py

Description: The script is indoor sensor that collect the indoor environment temperature and humidity measurement called the atmosphereData.py script.

```
#! /usr/bin/python3.7
import Adafruit_DHT
from config import INDOOR_DHT_PIN
from scripts.atmosphere.atmosphereData import read

INDOOR_DHT_SENSOR = Adafruit_DHT.AM2302

def atmosphereDataRead():
    return read(INDOOR_DHT_SENSOR, INDOOR_DHT_PIN)
```

Code 34 indoor/control.py

- **outdoor/control.py**

Path: rpiboard/scripts/atmosphere/outdoor/control.py

Description: The script is outdoor sensor that collect the outdoor environment temperature and humidity measurement called the atmosphereData.py script.

```
#! /usr/bin/python3.7
import Adafruit_DHT
from config import OUTDOOR_DHT_PIN
from scripts.atmosphere.atmosphereData import read
import time

OUTDOOR_DHT_SENSOR = Adafruit_DHT.AM2302
```

```
def atmosphereDataRead():
    return read(OUTDOOR_DHT_SENSOR, OUTDOOR_DHT_PIN)
```

Code 35 outdoor/control.py

5.3.3. Global Configuration Script

This script is a configuration that defined the constant used in whole python scripts.

- **config.py**

Path: rpiboard/config.py

Description: The constants further on pins of the Raspberry Pi and AWS ActiveMQ broker information are defined in this script used other scripts.

```
DOORBELL_CHANNEL=5
DOORBUTTON_CHANNEL=15
LOCK_CHANNEL=23

FAN_PWM_CHANNEL=12

LIGHT_RELAY_CHANNEL=21

SHUTTER_DIR_CHANNEL_1=4
SHUTTER_PWM_CHANNEL_2=13

INDOOR_DHT_PIN=17
OUTDOOR_DHT_PIN=27

MQTT_BROKER_ADDRESS="192.168.8.1"
MQTT_CLIENT_IDENTIFIER="RPi"

AWS_ACTIVEMQ_HOST="b-ddb6ba7b-55f1-4ad2-b3c9-7754a11843ac-1.mq.eu-west-
1.amazonaws.com"
AWS_ACTIVEMQ_USER="mqtt"
AWS_ACTIVEMQ_PASSWORD="1qaz@WSX3edc"
AWS_ACTIVEMQ_PORT=8883
```

Code 36 config.py

5.4. Local MQTT Broker

This is MQTT broker in the home local network is responsible for communication between the Raspberry Pi and the mobile app. It is a program installed in the local router. The communication process below.

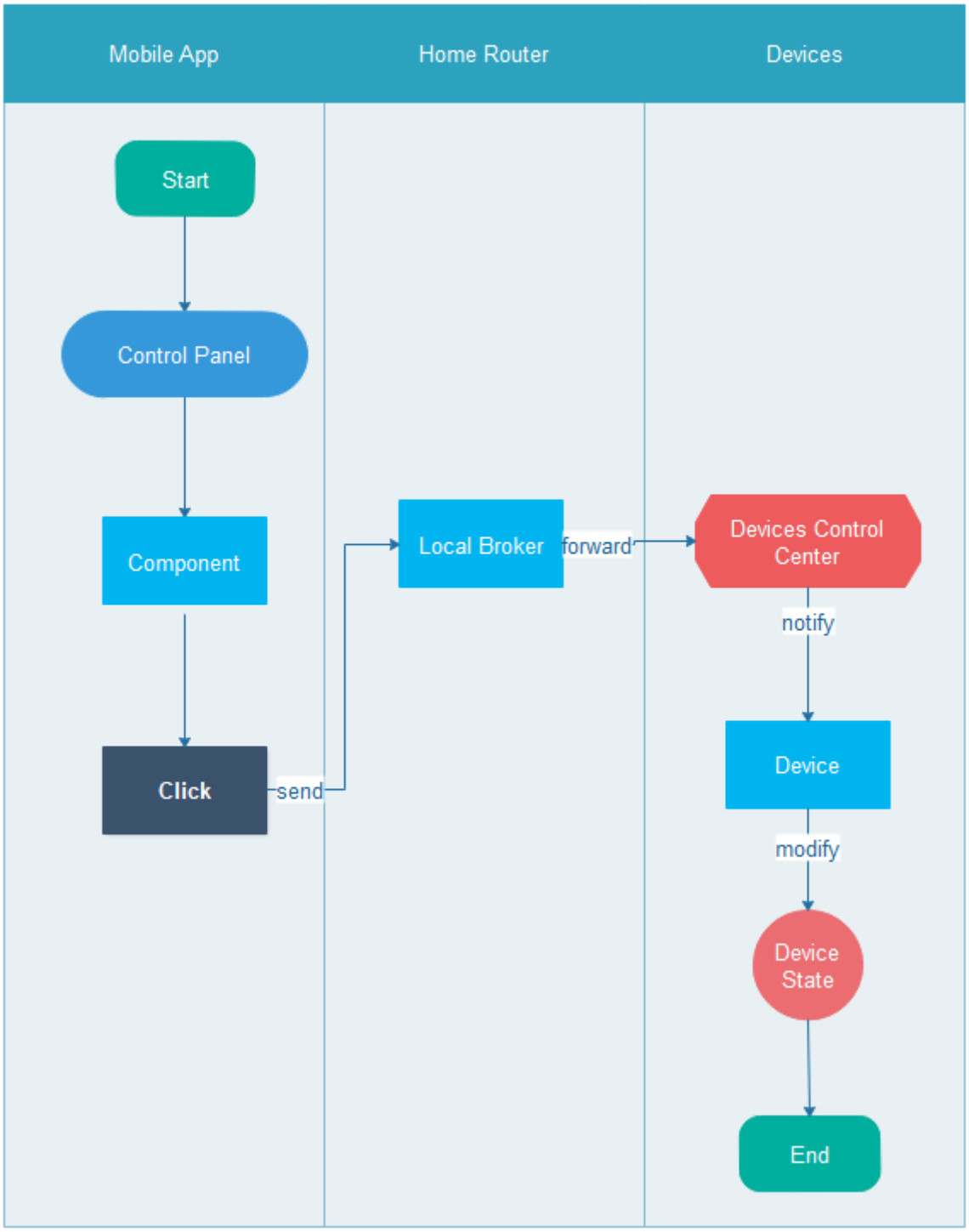

Figure 23 local MQTT broker communication

Further on the local MQTT broker install, please read the section Local MQTT Broker

5.5. Cloud MQTT Broker

This is MQTT broker is responsible for communication across the Internet between the Raspberry Pi and the mobile app. It is an AWS cloud service named ActiveMQ. The communication process below.

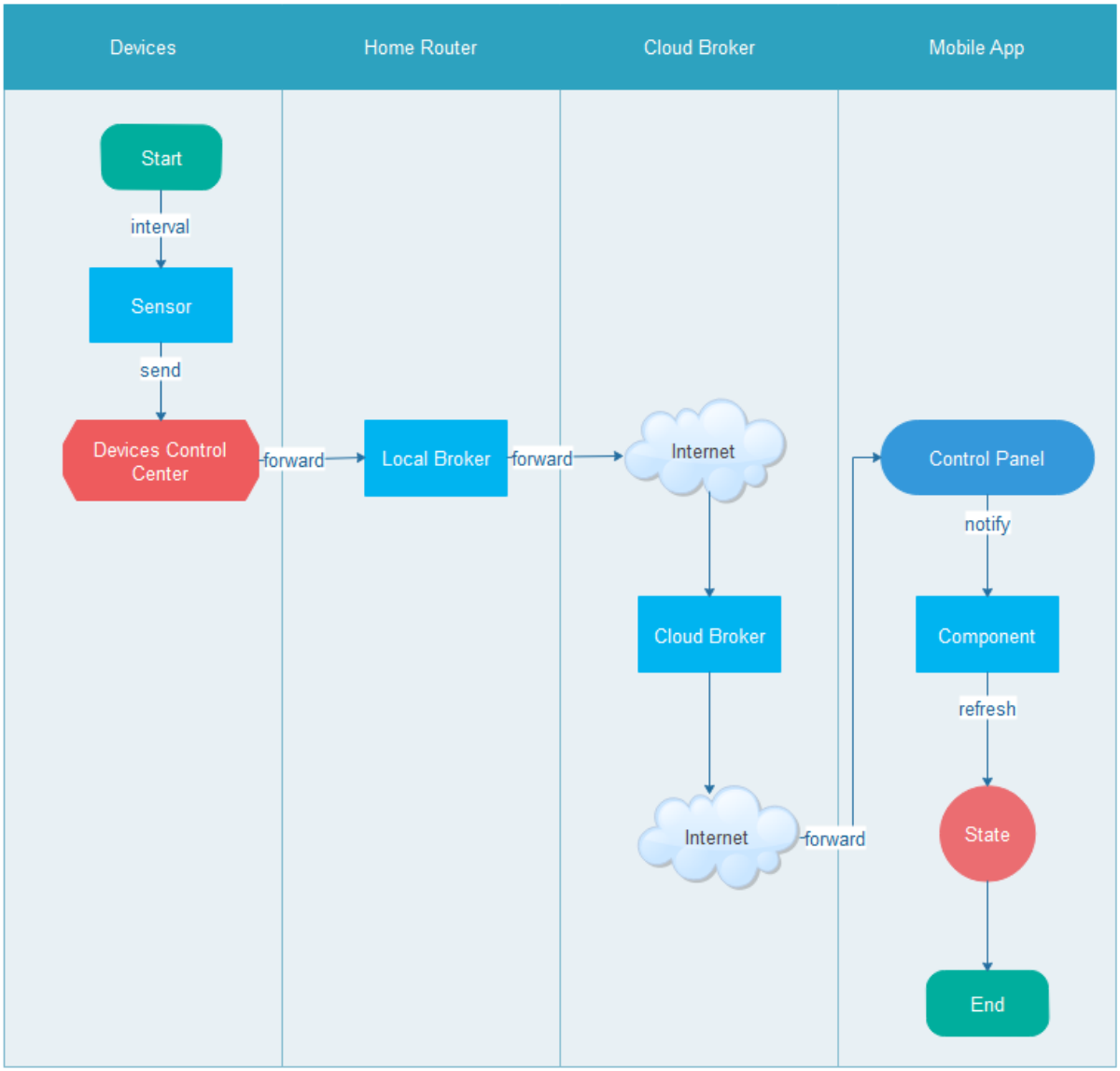

Figure 24 cloud MQTT broker communication

Further on the local MQTT broker install, please read the section AWS ActiveMQ

6. Application Deploy

5.2. Python Script

The python script will be executed by Raspberry Pi, So You should install the Python runtime environment on the Raspberry Pi.

5.2.2. Install Python Environment

1. Install python3.7, to run:

```
sudo apt-get update
sudo add-apt-repository ppa:deadsnakes/ppa
```

```
sudo apt-get install python3.7
```

2. Install pip3, to run:

```
sudo apt-get install python3-pip
```

5.2.3. Install Dependency Lib

1. install RPi.GPIO module, to run:

```
pip3 install RPi.GPIO
```

2. install paho.mqtt, to run:

```
pip3 install paho-mqtt
```

3. install Adafruit-DHT, to run:

```
pip3 install Adafruit_DHT
```

5.2.4. Download Source Code

1. Using git to clone the project code, to run:

```
git clone https://github.com/hellolingxh/HomeAutomation.git
```

2. Change the dicrectory from HomeAutomation folder to rpiboard folder.

5.2.5. Run

1. under the rpiboard folder to run the python script:

```
python3.7 controller.py &
```

5.3. Mjpeg-steamer

The Mjpeg-streamer is a driver on the Raspberry Pi to drive the USB camera and Raspberry Pi camera to turn on or off. In this section will introduce how to install it on the Raspberry Pi.

5.3.1. Mjpeg-streamer Driver Installation

1. Clone the source code from Github, to run the command:

```
git clone https://github.com/jacksonliam/mjpg-streamer.git
```

2. Install the compiler

```
sudo apt-get install gcc g++
sudo apt-get install cmake
```

3. Change the directory into the mjpg-streamer-experimental folder under the source code folder
   `cd $mjpg-streamer_home/mjpg-streamer-experimental`

4. Compile the source code under the mjpg-streamer-experimental folder
   `make`
   `sudo make install`

5. Create the temp folder if it is not existed
   `mkdir /tmp/www`

6. Run the command after plugin a camera into the Raspberry Pi
   `mjpeg_streamer -I 'input-uvc.so' -o 'output-http.so -w /tmp/www'`

5.4. Flutter App

In this section will introduce the Flutter framework environment installation.

5.4.1. Install Android SDK

The easy way to install the Android SDK is to just install the Android Studio.

If you want to install the Android SDK, please visit here:
https://developer.android.com/studio#downloads

In this project, The Android SDK version is 28.0.3.

5.4.2. Install Flutter SDK

For the Flutter framework beginner, the flutter official website provides tutorial to guide the developer to install the Flutter SDK. Please visit here:

https://flutter.dev/docs/get-started/install

In this project, the flutter SDK version is 1.15.18

5.4.1. Download Source Code

1. clone the project code from the Github, to run:
   `git clone https://github.com/hellolingxh/HomeAutomation.git`
2. change the directory to the flutterapp folder.

5.4.2. Run

1. Your Android mobile phone should connect to the computer which installed the flutter runtime environment through the USB cable.
2. under the flutterapp folder, you can run the command to verify the Flutter runtime environment. To run:
   `flutter doctor`

3. execute the run command on terminal to install the flutter app into the Android mobile phone. To run:
   `flutter run`

5.4.3. Flutter App Permission

Due to the app listen to the householder voice to control the light, so in this app, we need to grant the microphone permission to the app.

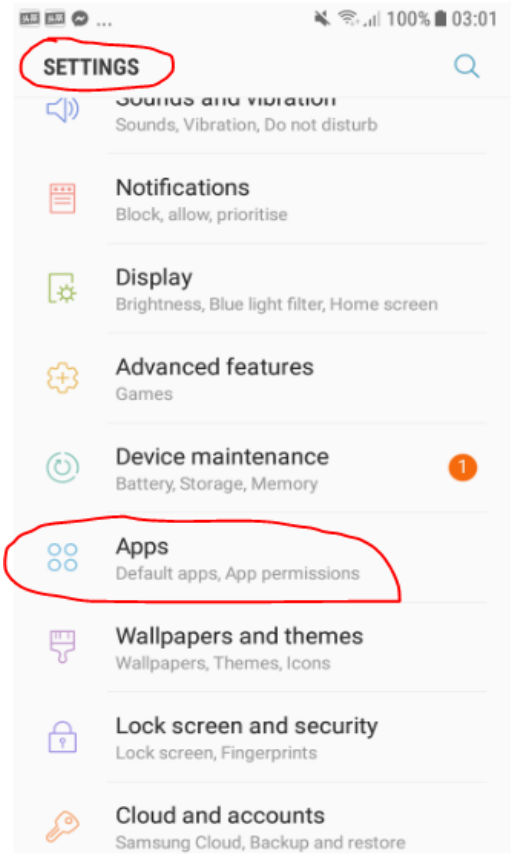

Figure 25 grant permission screen 1

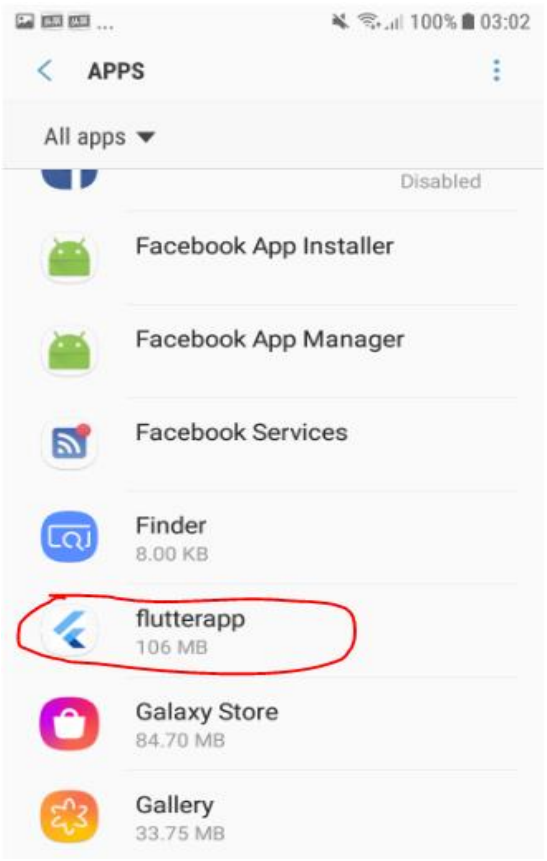

Figure 26 grant permission screen 2

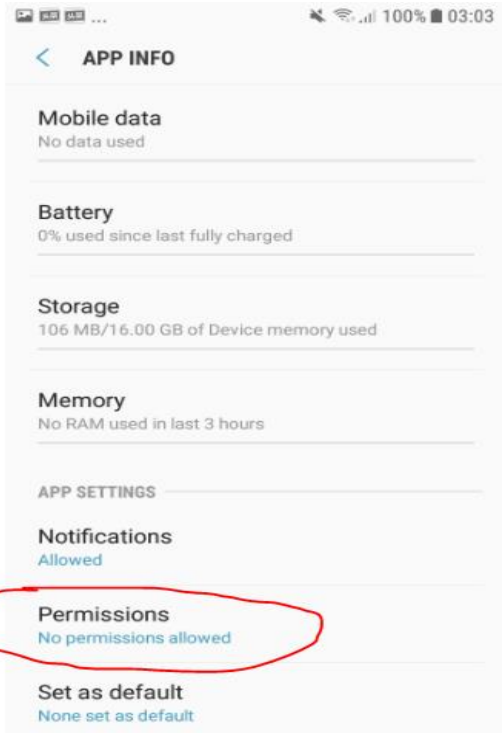

Figure 27 grant permission screen 3

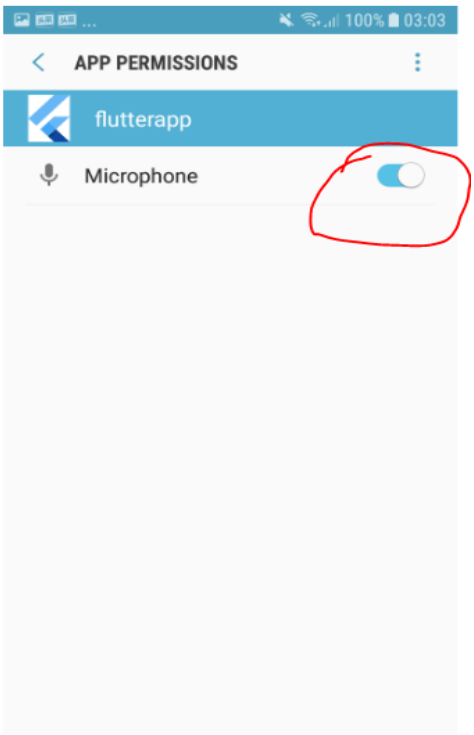

Figure 28 grant permission screen 4

5.5. Local MQTT Broker

In this section introduce the local MQTT broker installation.

5.5.1. Flashed OpenWRT Firmware

1. Visit the OpenWRT official website
   URL: https://openwrt.org/

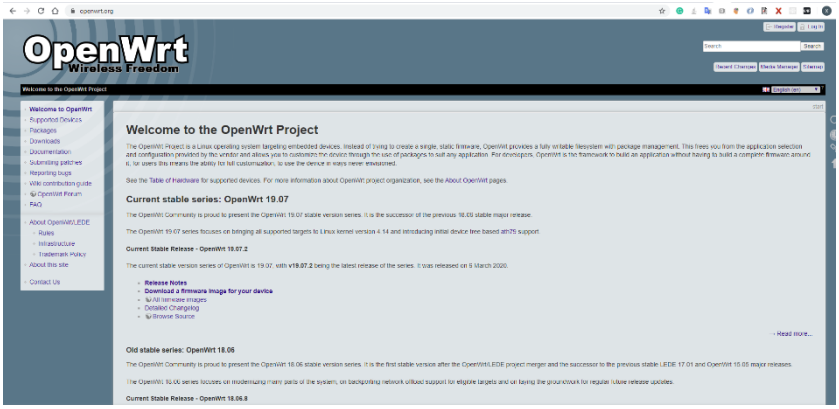

Figure 29 openWRT screen 1

2. Search your router brand and model on the official website
   You need to make sure your router supports flashing the openWRT firmware.
   In this project, I am using a router brand called GL.iNet and the model is GL-MT300N V2.

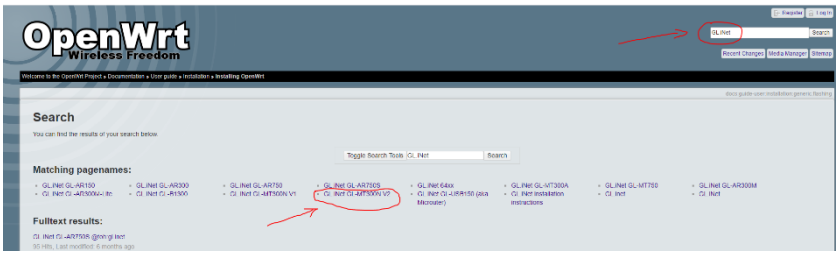

Figure 30 openWRT screen 2

3. Download the OpenWRT firmware from the official website

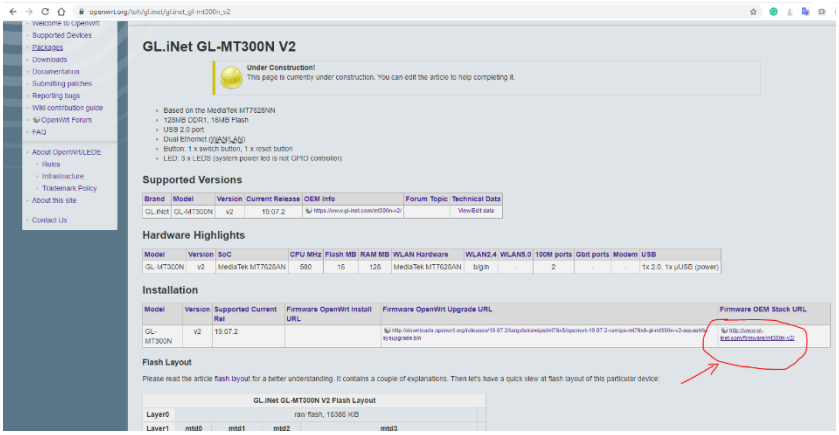

Figure 31 openWRT screen 3

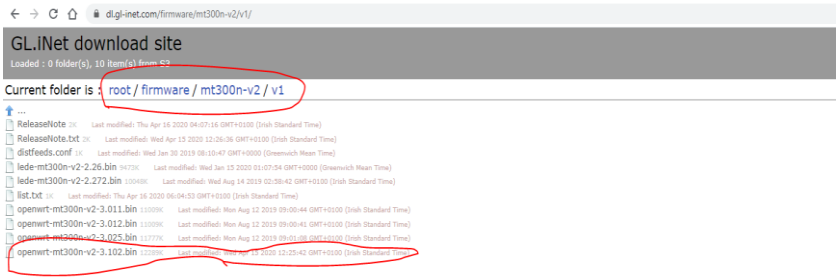

Figure 32 openWRT screen 4

4. Installation
   Please read the guide which URL is https://openwrt.org/toh/gl.inet/installation

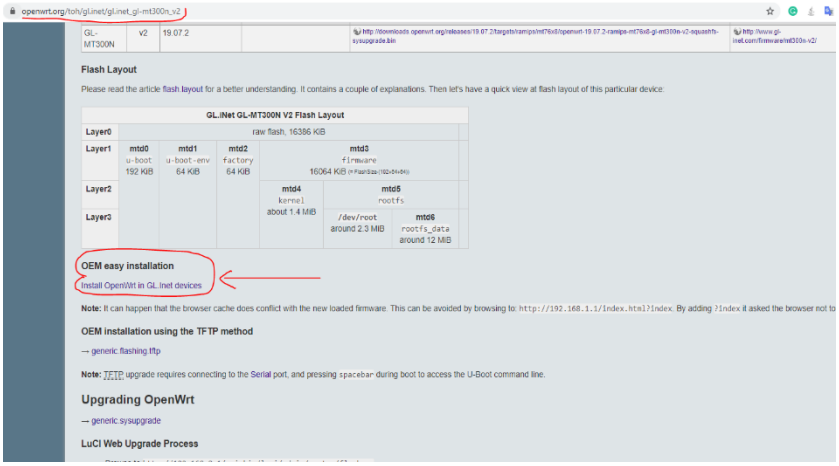

Figure 33 openWRT screen 5

5. The Router LAN interface connect to Computer through the Ethernet cable.

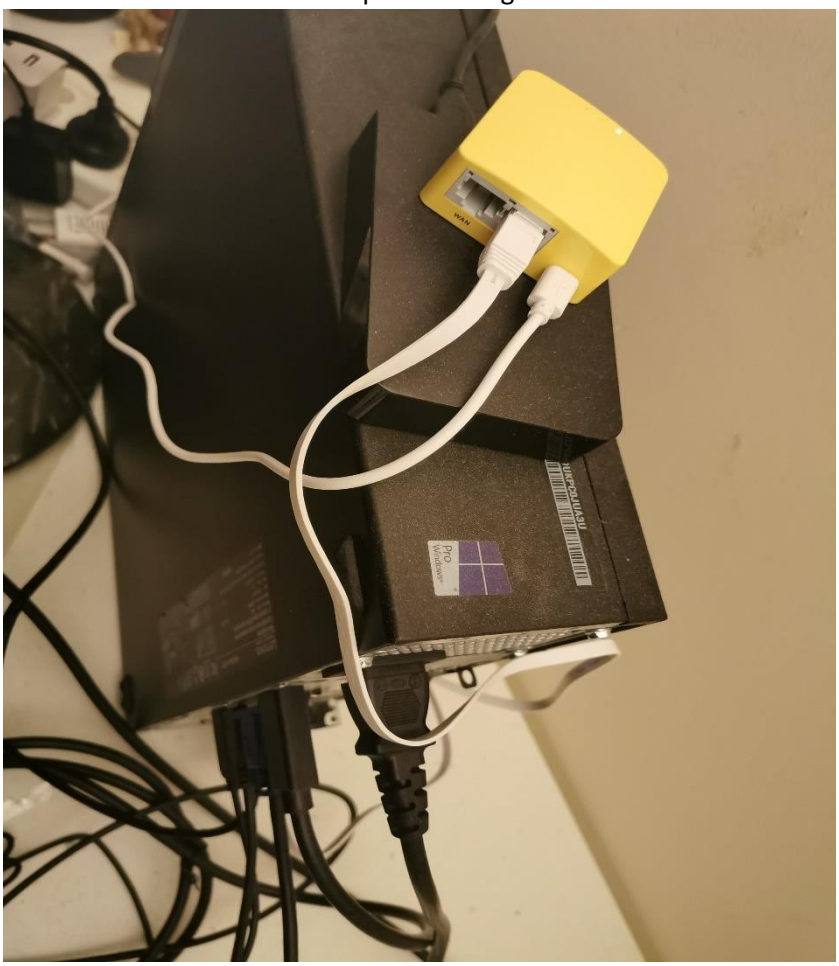

Figure 34 openWRT screen 6

6. Set your computer IP Address which the router connects to

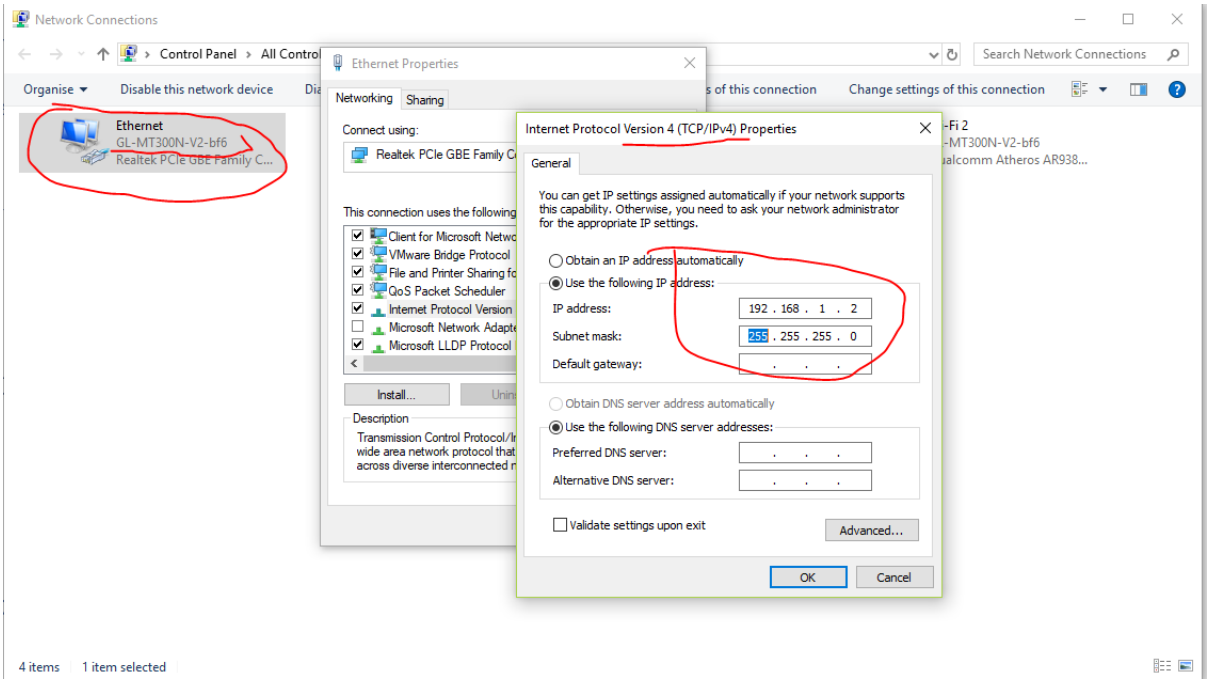

Figure 35 openWRT screen 7

7. Visit the Router console from your browser
   Press and hold the Reset button firmly first, and then power on your device. Must obey the first and then procedure and utilities the LEDs will start blinking maybe 5 times. Then visit the router console which URL is http://192.168.1.1/

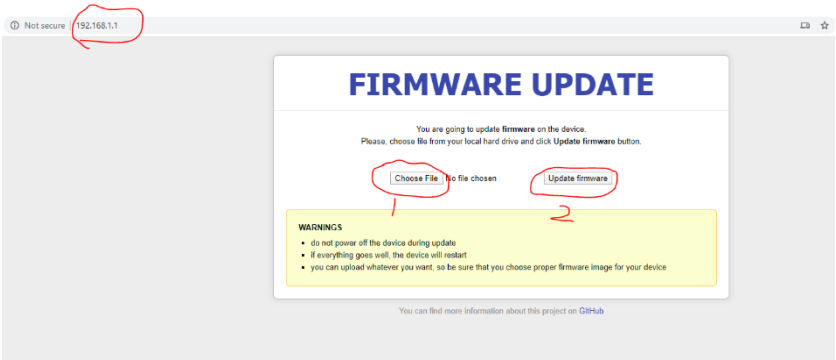

Figure 36 openWRT screen 8

8. Flash the OpenWRT firmware
   Click the button called ‘Choose File’ and then click the button which called ‘Update firmware’.

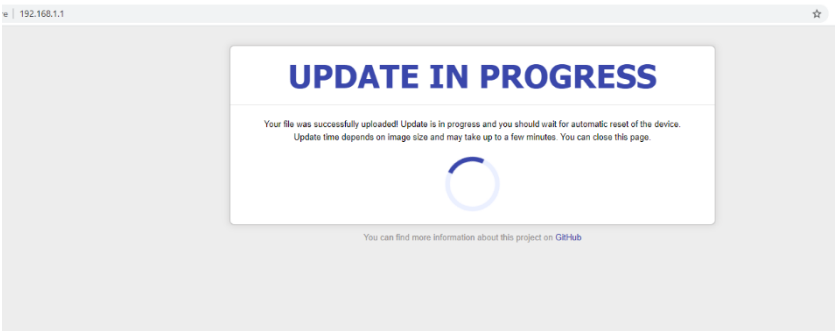

Figure 37 openWRT screen 9

9. It will be done should wait for a few minutes. If that finished, you could visit the router console which URL is http://192.168.8.1 through Wi-Fi

5.5.2. Install MQTT Broker

1. Visit the router console URL http://192.168.8.1 via browser to install the MQTT plug-in.

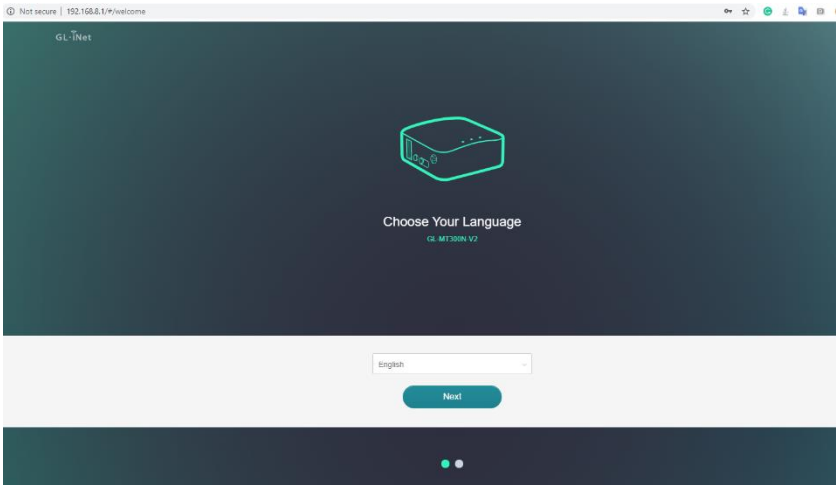

Figure 38 local MQTT broker installation screen 1

2. Let the router connect to Internet

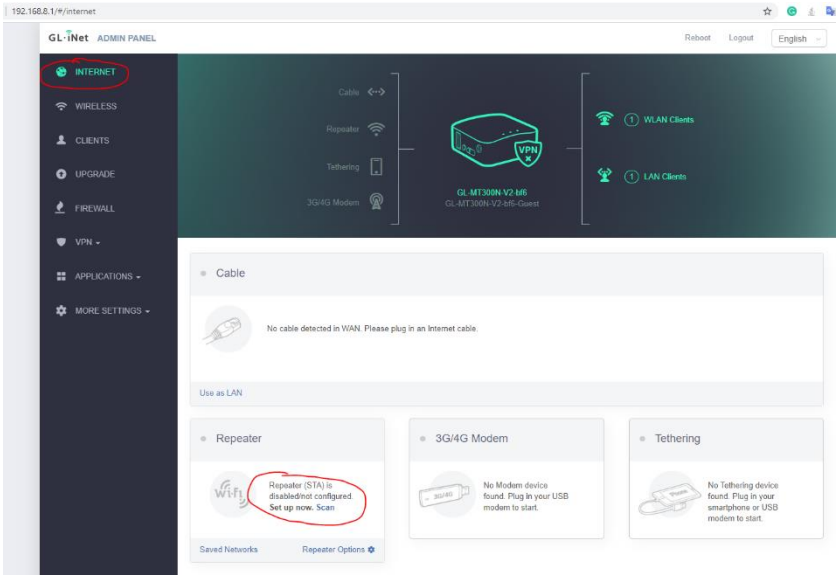

Figure 39 local MQTT broker installation screen 2

3. Join to your home router or connects to ISP directly.

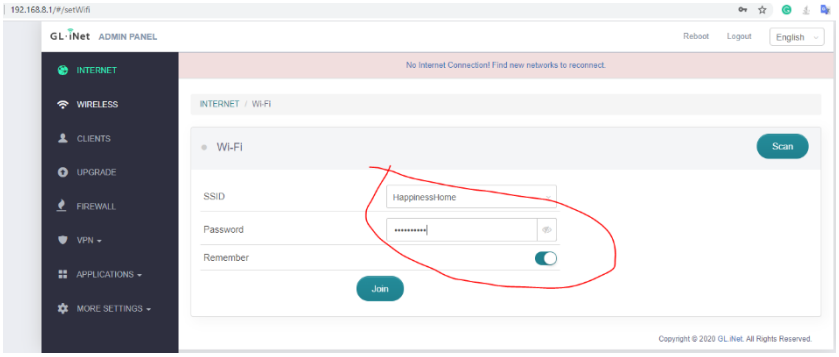

Figure 40 local MQTT broker installation screen 3

4. To install an implementation version of the MQTT that called Mosquitto MQTT

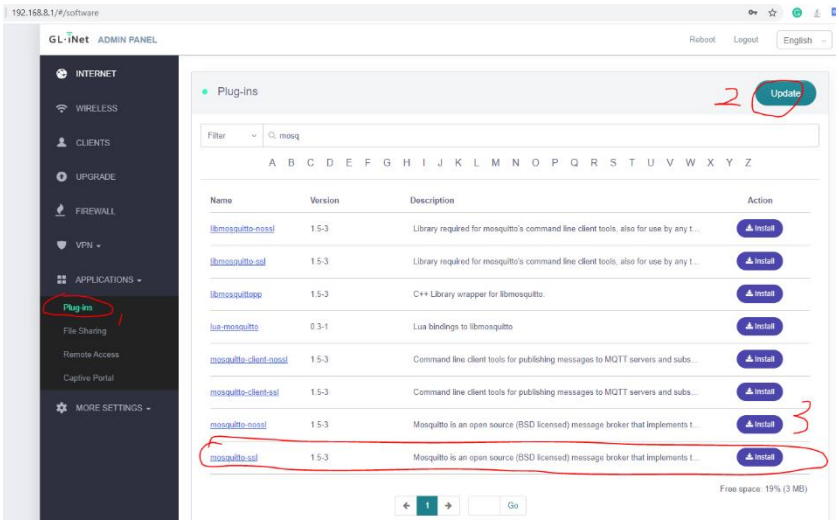

Figure 41 local MQTT broker installation screen 4

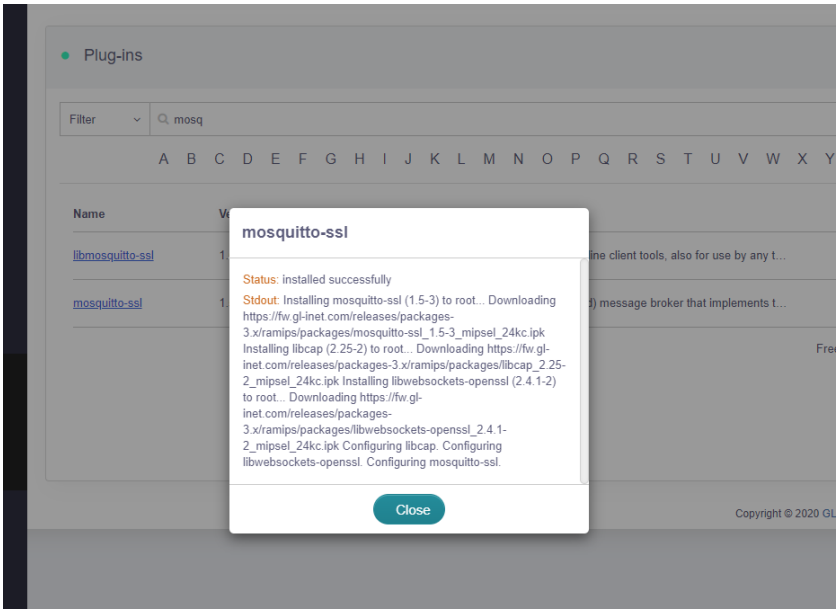

Figure 42 local MQTT broker installation screen 5

5.6. AWS ActiveMQ

In this section introduce the AWS ActiveMQ configuration and management.

5.6.1. AWS ActiveMQ configuration

This screen displays the AWS ActiveMQ broker I created configuration information in detail.

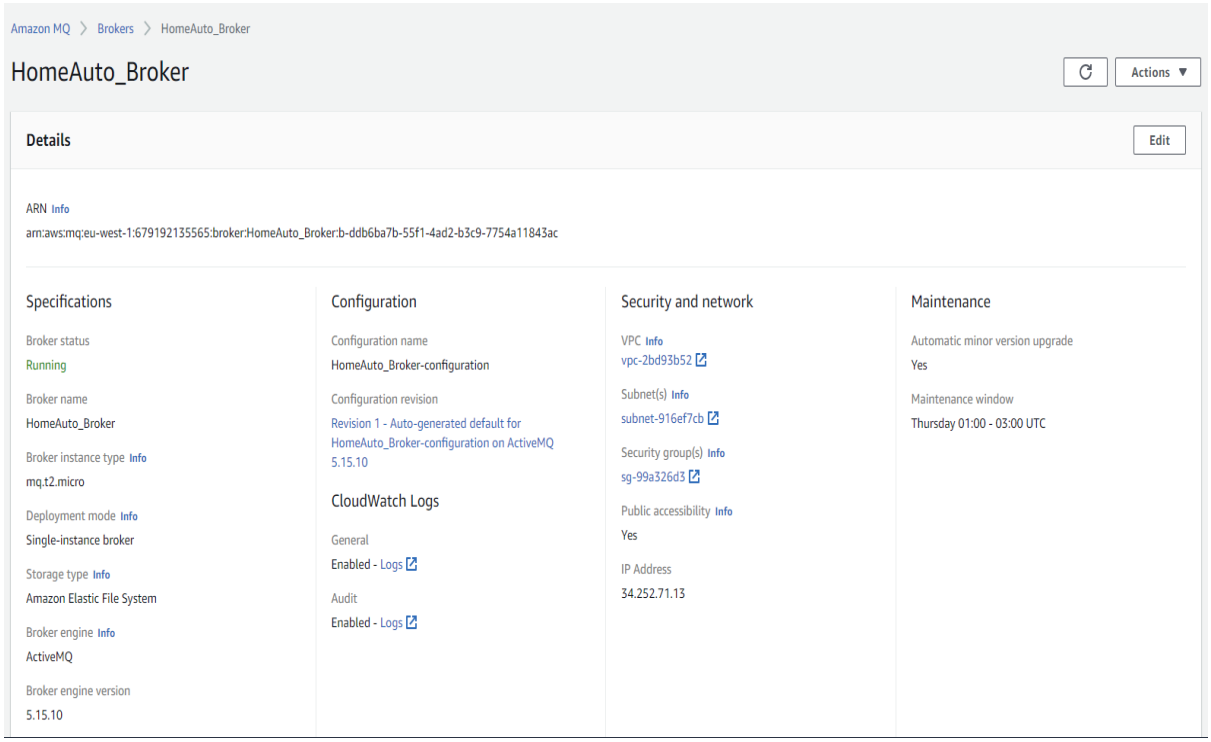

Figure 43 AWS ActiveMQ configuration screen

5.6.2. ActiveMQ console

The AWS ActiveMQ also provided a console that used to monitoring the traffic of the messages which published and subscribed.

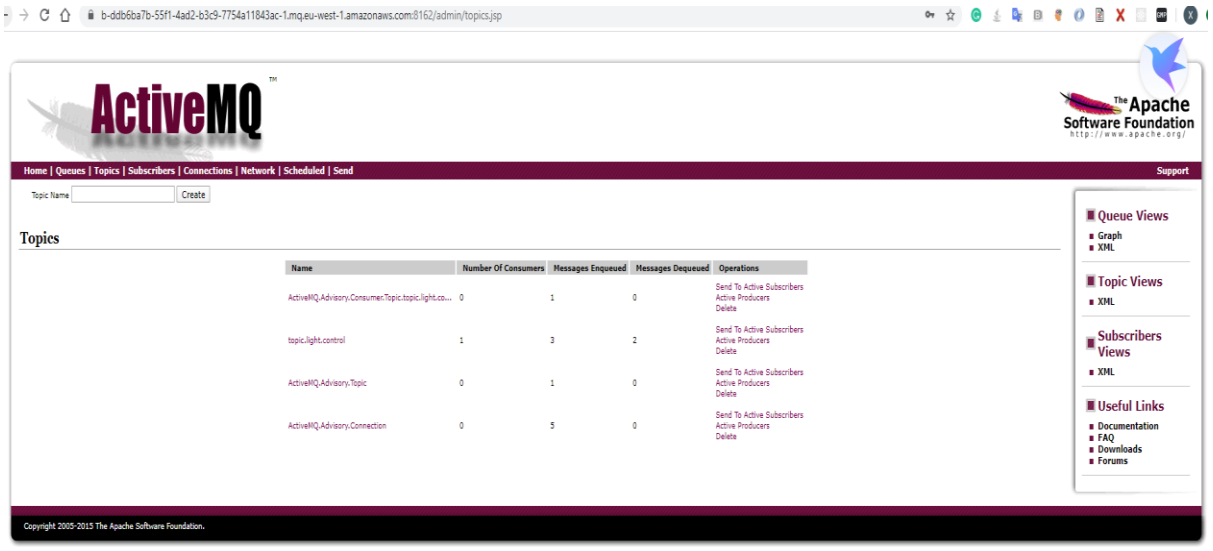

Figure 44 AWS ActiveMQ console screen

5.6.3. Launch an ActiveMQ Broker

This section introduces how to launch an ActiveMQ broker on the AWS console.

1. Login into the AWS console to search the ActiveMQ service then create a broker

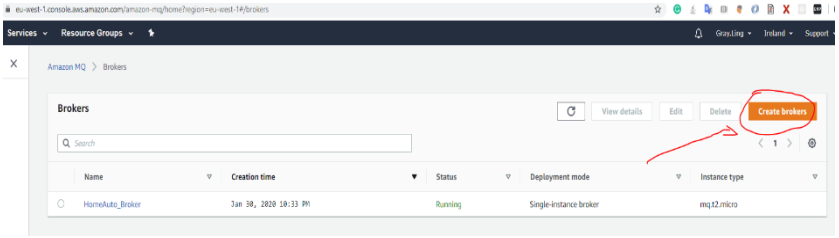

Figure 45 launch an ActiveMQ broker screen

2. Configure the broker which you want to create

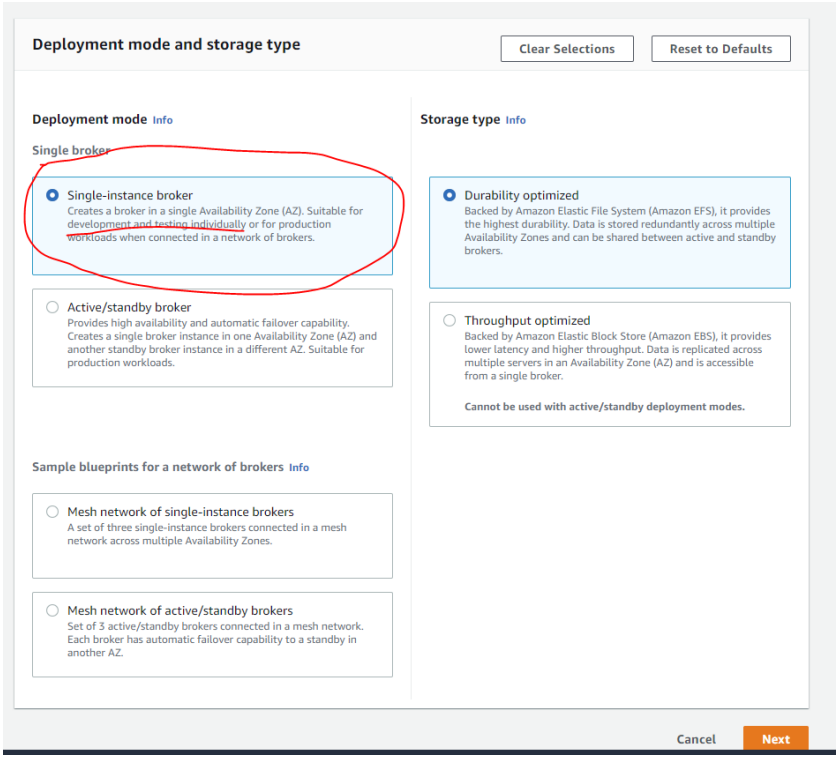

Figure 46 configure a broker screen 1

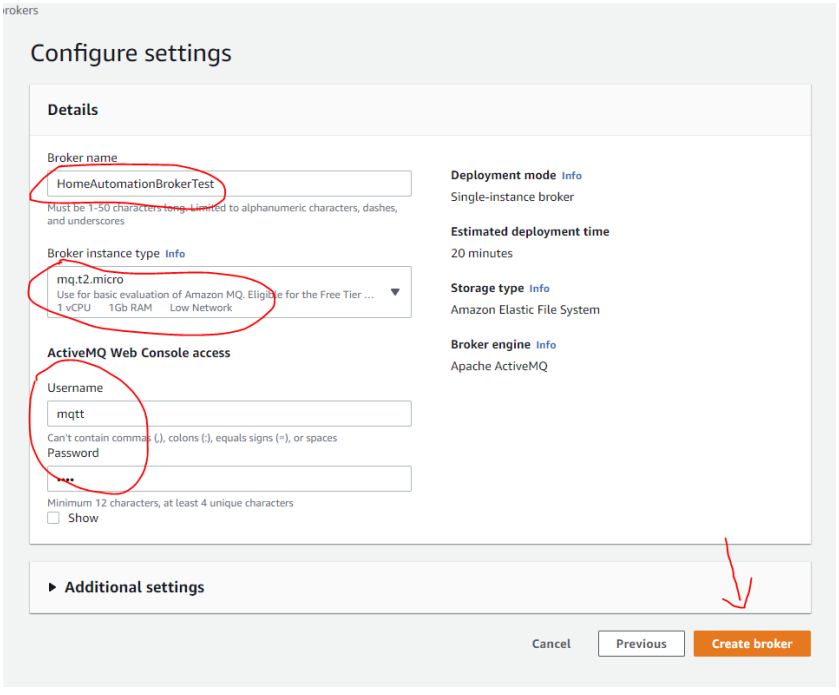

Figure 47 configure a broker screen 2

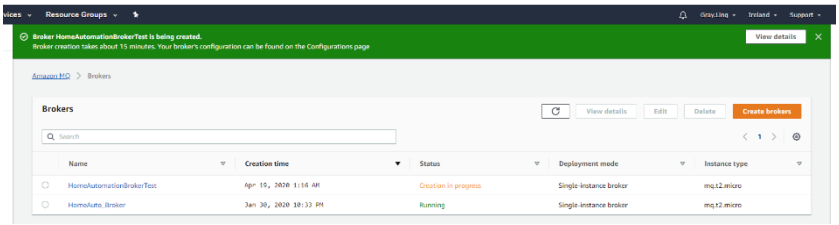

Figure 48 configure a broker screen 3

3. When the broker is running status, then you can see the broker MQTT request method

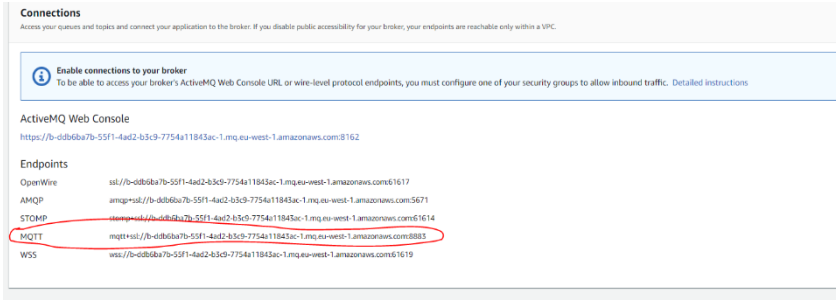

Figure 49 ActiveMQ broker information screen

4. Also, you can click the ActiveMQ web console to see the message of topic which is enqueued or dequeued.

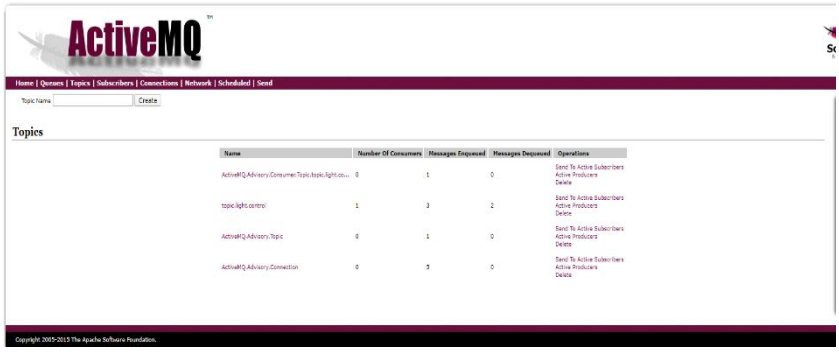

Figure 50 AWS ActiveMQ console monitoring screen

5. For security, you need to get the certificate keys, please visit here.

https://docs.aws.amazon.com/iot/latest/developerguide/create-device-certificate.html