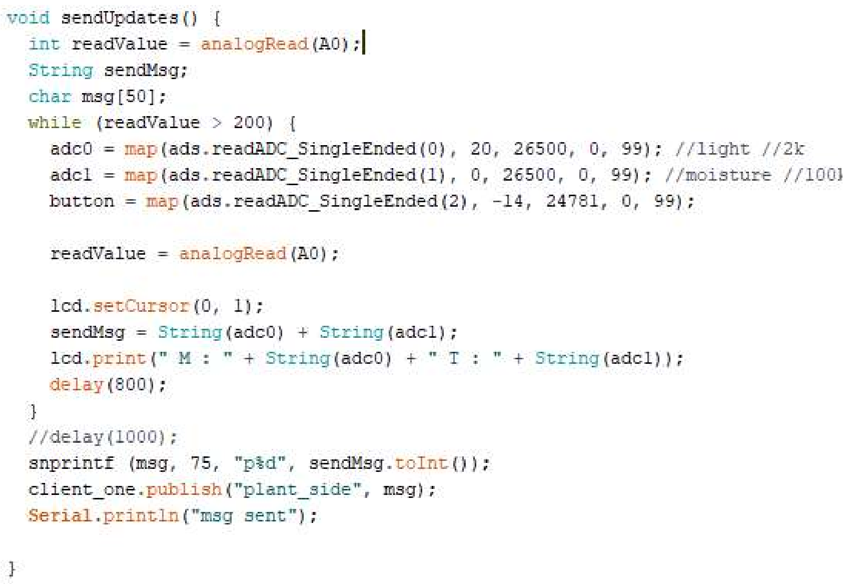
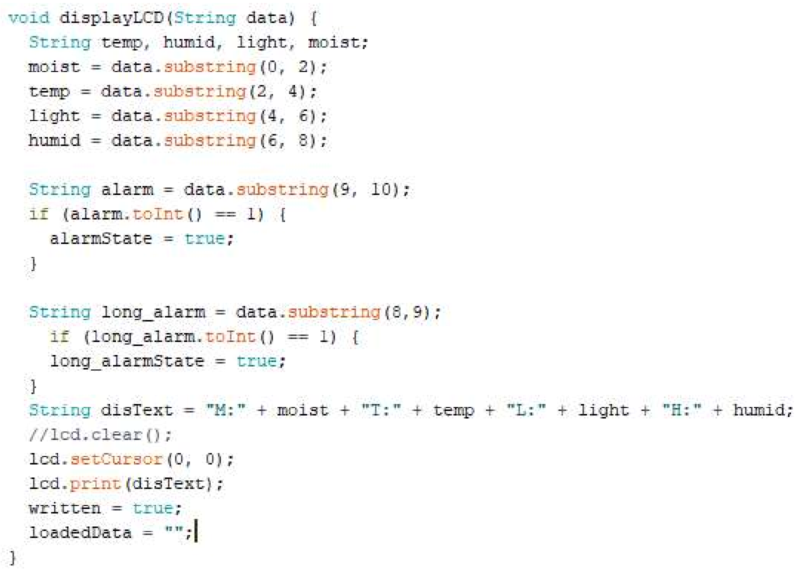
User side Device

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

#include <LiquidCrystal.h>

LiquidCrystal lcd(D0, D3, D4, D6, D7, D8);

#include <Wire.h>
#include <Adafruit_ADS1015.h>

Adafruit_ADS1115 ads;
int16_t adc0, adc1, button;

const char* ssid = "Dialog 4G";
const char* password = "E3344662904";
const char* mqtt_server = "broker.hivemq.com";

WiFiClient espClient;
PubSubClient client_one(espClient);
PubSubClient client_two(espClient);


String loadedData;
String loadedData_2;
boolean written = true;
boolean edittable = false;
//data format THLS



unsigned long butDecTime = 0;
boolean butDecState = false;
boolean butDeclastState = false;

boolean alarmState = false;
boolean long_alarmState = false;



void setup() {
  Serial.begin(115200);
  setup_wifi();
  client_one.setServer(mqtt_server, 1883);
  client_one.setCallback(callback);

  ads.setGain(GAIN_ONE);
  lcd.begin(16, 2);
  ads.begin();

  pinMode(A0, INPUT);
  pinMode(D5, OUTPUT);
}
```

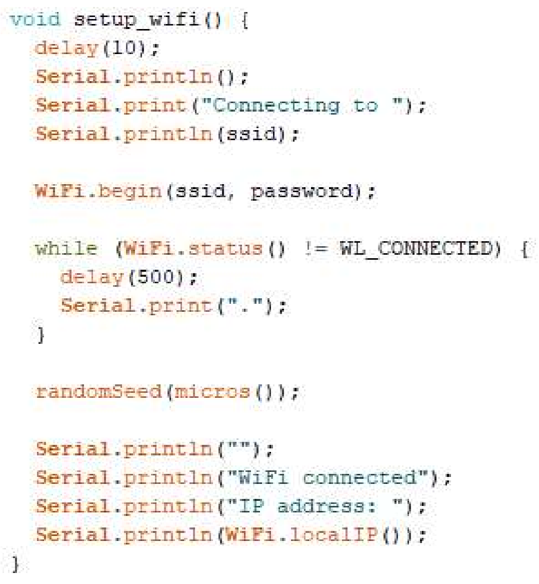
```
void loop() {
  if (!client_one.connected()) {
    reconnect();
  }
  client_one.loop();
  int readValue = analogRead(A0);
  if (readValue > 200) {
    written = false;
    lcd.setCursor(15, 1);
    lcd.print("*");
    sendUpdates();
    lcd.setCursor(15, 1);
    lcd.print(" ");
  } else {
    if (written == false) {
      displayLCD(loadedData);
    }
  }


  if (long_alarmState == true) {
    Serial.println("long alarm on");
    digitalWrite(D5, HIGH);
    delay(1000);
    digitalWrite(D5, LOW);
    delay(1000);
    digitalWrite(D5, HIGH);
    delay(1000);
    digitalWrite(D5, LOW);
    delay(1000);
    digitalWrite(D5, HIGH);
    delay(1000);
    digitalWrite(D5, LOW);
    delay(1000);
    long_alarmState = false;;
  } else {
    if (alarmState == true) {
      Serial.println("Alarm On");
      digitalWrite(D5, HIGH);
      delay(50);
      digitalWrite(D5, LOW);
      delay(50);
      alarmState = false;
    }
  }
  delay(800);
}
```



```
void sendUpdates() {
  int readValue = analogRead(A0);
  String sendMsg;
  char msg[50];
  while (readValue > 200) {
    adc0 = map(ads.readADC_SingleEnded(0), 20, 26500, 0, 99); //light //2k
    adc1 = map(ads.readADC_SingleEnded(1), 0, 26500, 0, 99); //moisture //100
    button = map(ads.readADC_SingleEnded(2), -14, 24781, 0, 99);

    readValue = analogRead(A0);

    lcd.setCursor(0, 1);
    sendMsg = String(adc0) + String(adc1);
    lcd.print(" M : " + String(adc0) + " T : " + String(adc1));
    delay(800);
  }
  //delay(1000);
  snprintf (msg, 75, "p%d", sendMsg.toInt());
  client_one.publish("plant_side", msg);
  Serial.println("msg sent");

}
```



```
void displayLCD(String data) {
  String temp, humid, light, moist;
  moist = data.substring(0, 2);
  temp = data.substring(2, 4);
  light = data.substring(4, 6);
  humid = data.substring(6, 8);

  String alarm = data.substring(9, 10);
  if (alarm.toInt() == 1) {
    alarmState = true;
  }

  String long_alarm = data.substring(8,9);
    if (long_alarm.toInt() == 1) {
    long_alarmState = true;
  }
  String disText = "M:" + moist + "T:" + temp + "L:" + light + "H:" + humid;
  //lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(disText);
  written = true;
  loadedData = "";
}
```



```
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    loadedData += String((char)payload[i]);
    Serial.print(String((char)payload[i]));
  }
  written = false;
}

void reconnect() {
  while (!client_one.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client_one.connect(clientId.c_str()))  {
      Serial.println("connected");
      client_one.subscribe("node_red");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client_one.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

<u>Mobile application</u>

main.dart

```
import 'package:flutter/material.dart';
import 'package:iotapp/MQTTAppState.dart';
import 'package:iotapp/designView.dart';
import 'package:iotapp/settingsPage.dart';
import 'package:iotapp/MQTTManager.dart';
import 'package:provider/provider.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (_)=>MQTTAppState(),
      child:MaterialApp(
        debugShowCheckedModeBanner: false,
        title: "Flutter_view",
        home: Dashboard(),
      ),

    );
  }
}
```

MQTTAppState.dart

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

enum MQTTAppConnectionState{connected ,disconnected, connecting}
class MQTTAppState extends ChangeNotifier{
  MQTTAppConnectionState _appConnectionState = MQTTAppConnectionState.disconnected;
  String _receivedText="";

  void setReceivedText(String text){
    _receivedText=text;
    notifyListeners();
  }

  void setAppConnectionState(MQTTAppConnectionState state){
    _appConnectionState=state;
    notifyListeners();
  }

  String get getReceivedText => _receivedText;
  MQTTAppConnectionState get getAppConnectionState => _appConnectionState;
}
```

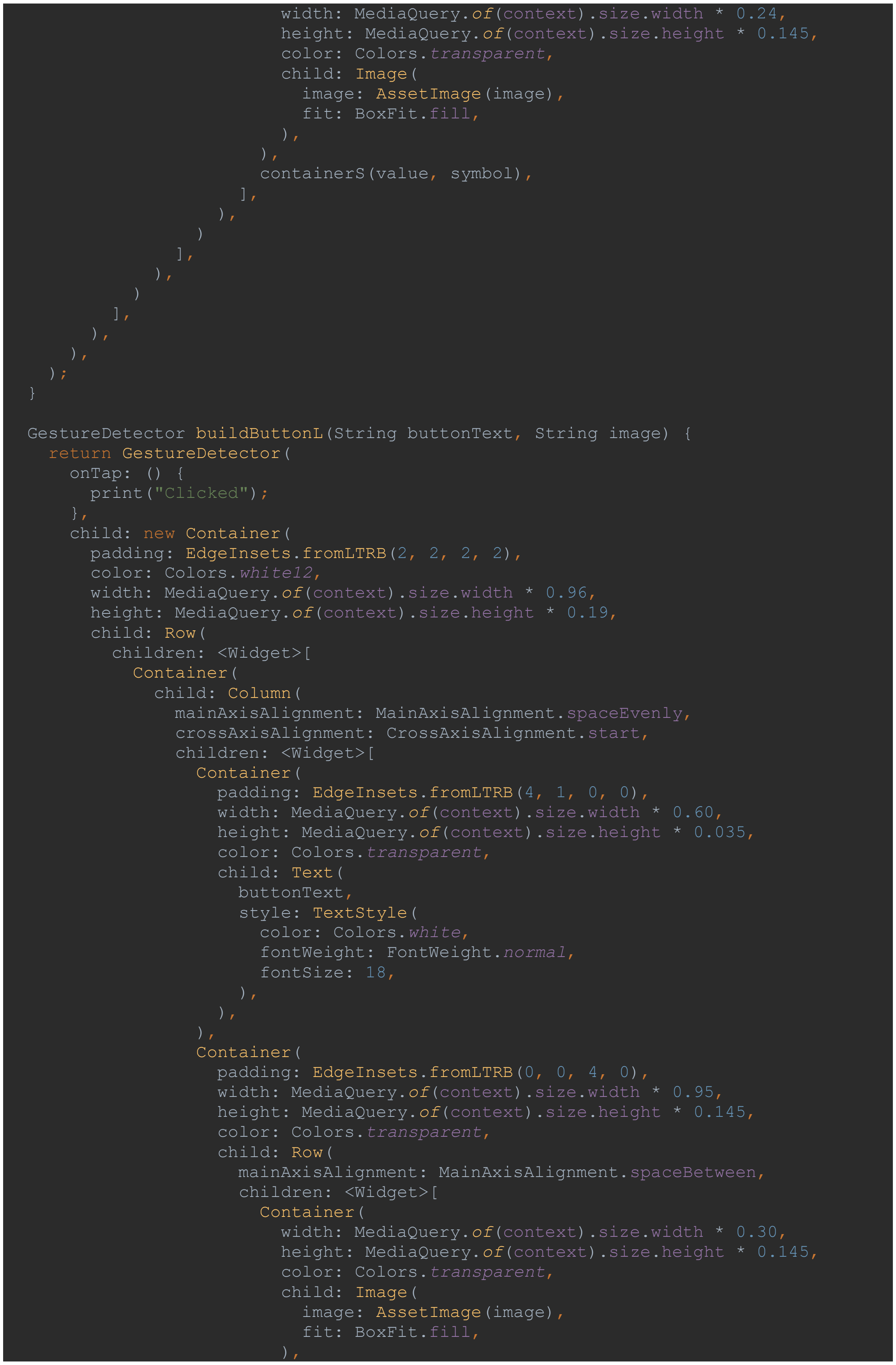
MQTTManager.dart

```
import 'package:flutter/material.dart';
import 'package:mqtt_client/mqtt_client.dart';
import 'package:iotapp/MQTTAppState.dart';
import 'package:flutter/cupertino.dart';

class MQTTManager{

  //private instance of client
  MQTTAppState _currentState;
  MqttClient _client;
  String _identifier;
  String _host;
  String _topic;

  //constructor
  MQTTManager({@required String host,@required String topic,@required String
identifier,@required MQTTAppState state}):
        _identifier=identifier, _host=host,_topic=topic,_currentState=state;

  void initializeMQTTClient(){
    _client=MqttClient(_host, _identifier);
    _client.port=1883;
    _client.keepAlivePeriod = 20;
    _client.onDisconnected = onDisconnected;
    _client.secure=false;
    _client.logging(on: true);

    //ADD the successful connection callback
    _client.onConnected=onConnected;
    _client.onSubscribed=onSubscribed;

    final MqttConnectMessage connMess =MqttConnectMessage()
        .withClientIdentifier(_identifier)
        .withWillTopic('willtopic')
        .withWillMessage('My will message')
        .startClean()
        .withWillQos(MqttQos.atLeastOnce);
    print('EAMPLE::Mosquitto client connecting.....');
    _client.connectionMessage = connMess;
  }

  void connect() async{
    assert(_client!=null);
    try{
      print('EXAMPLE:: Mosquitto start client connecting....');
      _currentState.setAppConnectionState(MQTTAppConnectionState.connecting);
      await _client.connect();
    }on Exception catch (e){
      print('EXAMPLE:: client exception - $e');
      disconnect();
    }
  }
  void disconnect(){
    print('Disconnected');
    _client.disconnect();
  }

  void publish(String message){
    final MqttClientPayloadBuilder builder = MqttClientPayloadBuilder();
    builder.addString(message);
    _client.publishMessage(_topic, MqttQos.exactlyOnce, builder.payload);
  }


  void onSubscribed(String topic){
    print('EXAMPLE::Subscription confirmed for topic $topic');
  }

  void onDisconnected(){
    print('EXAMPLE::onDisconnected client callback - client disconnection');
    if(_client.connectionStatus.returnCode==MqttConnectReturnCode.solicited){
      print('EXAMPLE::onDisconnected callback is solicited, this is correct');
    }
    _currentState.setAppConnectionState(MQTTAppConnectionState.disconnected);
  }

  void onConnected(){
    _currentState.setAppConnectionState(MQTTAppConnectionState.connected);
    print('EXAMPLE::Mosquito client connected...');
    _client.subscribe(_topic, MqttQos.atLeastOnce);
    _client.updates.listen((List<MqttReceivedMessage<MqttMessage>> c){
      final MqttPublishMessage recMess = c[0].payload;
      final String pt =
      MqttPublishPayload.bytesToStringAsString((recMess.payload.message));
      _currentState.setReceivedText(pt);
      print(
          ' EXAMPLE::change notification:: topic is <${c[0].topic}>, payliad is <--
$pt-->');

      print('');

    });
    print(
        'EXAMPLE:: onConnected client callBack = Client connection was
successful');
  }




}
```

designView.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:iotapp/MQTTAppState.dart';
import 'package:iotapp/MQTTManager.dart';
import 'package:iotapp/settingsPage.dart';
import 'package:iotapp/configSettings.dart';


class Dashboard extends StatefulWidget {

  @override
  _DashboardState createState() => _DashboardState();
}
```

```
class _DashboardState extends State<Dashboard> {
  double tempTreashold = 0.0;
  double moistTreashold = 0.0;

  String hostName="broker.hivemq.com";
  String topic="plant_side";

  MQTTManager manager_pub;
  MQTTAppState currentAppState;
  MQTTAppState currentAppState_pub;

  String humid, temp, light, mos;

  Widget send_button(MQTTAppConnectionState state){
    return RaisedButton(
      //onPressed: (){publishMessage("asas");},//(tempTreashold.toInt()).toString()
+ (moistTreashold.toInt()).toString()
      textColor: Colors.black,
      elevation: 8,
      color: Colors.teal,
      colorBrightness: Brightness.dark,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(6),
      ),
      child: Text(
        "Set Values",
        style: TextStyle(
          fontSize: 18,
          fontWeight: FontWeight.w600,
        ),
      ),
      onPressed: (){publishMessage("P"+(tempTreashold.toInt()).toString() +
(moistTreashold.toInt()).toString());},
    );
  }

  void publishMessage(String text){
    final message = text;
    manager_pub.publish(message);
  }

  Widget containerS(String value, String symbol) {
    return (new Container(
      width: MediaQuery.of(context).size.width * 0.19,
      height: MediaQuery.of(context).size.height * 0.135,
      color: Colors.transparent,
      child: Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: <Widget>[
          Container(
            width: MediaQuery.of(context).size.width * 0.18,
            height: MediaQuery.of(context).size.height * 0.06,
            color: Colors.transparent,
            child: Center(
              child: Text(
                value,
                style: TextStyle(
                  color: Colors.teal,
                  fontWeight: FontWeight.normal,
                  fontSize: 40,
                ),
              ),
            ),
          ),
          Container(
            width: MediaQuery.of(context).size.width * 0.18,
            height: MediaQuery.of(context).size.height * 0.06,
            color: Colors.transparent,
            child: Center(
              child: Text(
                symbol,
                style: TextStyle(
                  color: Colors.teal,
                  fontWeight: FontWeight.normal,
                  fontSize: 35,
                ),
              ),
            ),
          )
        ],
      ),
    ));
  }
  void configureAndConnect() {
    manager_pub = MQTTManager(
        host: "broker.hivemq.com",
        topic: "plant_side",
        identifier: "ENTC1",
        state: currentAppState_pub
    );
    manager_pub.initializeMQTTClient();
    manager_pub.connect();
  }

  GestureDetector buildButtonS(
      String buttonText, String image, String value, String symbol) {
    return GestureDetector(
      onTap: () {
        print("Clicked");
      },
      child: new Container(
        padding: EdgeInsets.fromLTRB(2, 2, 2, 2),
        color: Colors.white12,
        width: MediaQuery.of(context).size.width * 0.46,
        height: MediaQuery.of(context).size.height * 0.19,
        child: Row(
          children: <Widget>[
            Container(
              child: Column(
                mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                crossAxisAlignment: CrossAxisAlignment.start,
                children: <Widget>[
                  Container(
                    padding: EdgeInsets.fromLTRB(4, 1, 0, 0),
                    width: MediaQuery.of(context).size.width * 0.26,
                    height: MediaQuery.of(context).size.height * 0.035,
                    color: Colors.transparent,
                    child: Text(
                      buttonText,
                      style: TextStyle(
                        color: Colors.grey,
                        fontWeight: FontWeight.normal,
                        fontSize: 18,
                      ),
                    ),
                  ),
                  Container(
                    padding: EdgeInsets.fromLTRB(0, 0, 4, 0),
                    width: MediaQuery.of(context).size.width * 0.45,
                    height: MediaQuery.of(context).size.height * 0.145,
                    color: Colors.transparent,
                    child: Row(
                      mainAxisAlignment: MainAxisAlignment.spaceBetween,
                      children: <Widget>[
                        Container(
```

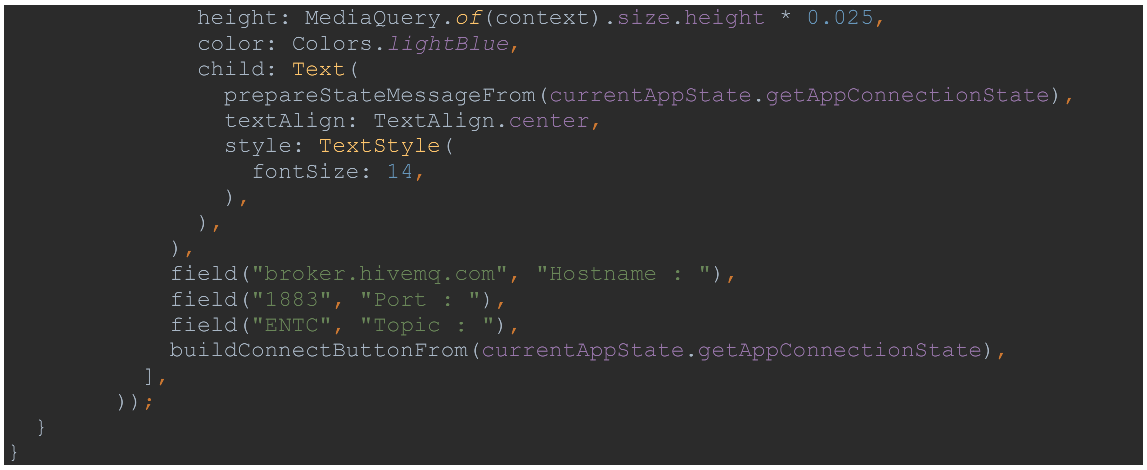


```
                        width: MediaQuery.of(context).size.width * 0.24,
                        height: MediaQuery.of(context).size.height * 0.145,
                        color: Colors.transparent,
                        child: Image(
                          image: AssetImage(image),
                          fit: BoxFit.fill,
                        ),
                      ),
                      containerS(value, symbol),
                    ],
                  ),
                )
              ],
            ),
          )
        ],
      ),
    ),
  );
}

GestureDetector buildButtonL(String buttonText, String image) {
  return GestureDetector(
    onTap: () {
      print("Clicked");
    },
    child: new Container(
      padding: EdgeInsets.fromLTRB(2, 2, 2, 2),
      color: Colors.white12,
      width: MediaQuery.of(context).size.width * 0.96,
      height: MediaQuery.of(context).size.height * 0.19,
      child: Row(
        children: <Widget>[
          Container(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              crossAxisAlignment: CrossAxisAlignment.start,
              children: <Widget>[
                Container(
                  padding: EdgeInsets.fromLTRB(4, 1, 0, 0),
                  width: MediaQuery.of(context).size.width * 0.60,
                  height: MediaQuery.of(context).size.height * 0.035,
                  color: Colors.transparent,
                  child: Text(
                    buttonText,
                    style: TextStyle(
                      color: Colors.white,
                      fontWeight: FontWeight.normal,
                      fontSize: 18,
                    ),
                  ),
                ),
                Container(
                  padding: EdgeInsets.fromLTRB(0, 0, 4, 0),
                  width: MediaQuery.of(context).size.width * 0.95,
                  height: MediaQuery.of(context).size.height * 0.145,
                  color: Colors.transparent,
                  child: Row(
                    mainAxisAlignment: MainAxisAlignment.spaceBetween,
                    children: <Widget>[
                      Container(
                        width: MediaQuery.of(context).size.width * 0.30,
                        height: MediaQuery.of(context).size.height * 0.145,
                        color: Colors.transparent,
                        child: Image(
                          image: AssetImage(image),
                          fit: BoxFit.fill,
                        ),
                      ),
                      Container(
                        width: MediaQuery.of(context).size.width * 0.60,
                        height: MediaQuery.of(context).size.height * 0.135,
                        color: Colors.white10,
                      ),
                    ],
                  ),
                ),
              ],
            ),
          )
        ],
      ),
    ),
  );
}

Widget custom_slider_temp() {
  return SliderTheme(
    data: SliderTheme.of(context).copyWith(
      trackHeight: 8,
      overlayColor: Colors.white30,
      minThumbSeparation: 50,
      rangeThumbShape: RoundRangeSliderThumbShape(
          enabledThumbRadius: 10, disabledThumbRadius: 10),
    ),
    child: Slider(
      label: tempTreashold.abs().toString(),
      value: tempTreashold,
      min: 0,
      max: 99.0,
      onChanged: (val) {
        setState(() {
          tempTreashold = val;
        });
      },
    ),
  );
}

Widget custom_slider_moist() {
  return SliderTheme(
    data: SliderTheme.of(context).copyWith(
      trackHeight: 8,
      overlayColor: Colors.white30,
      minThumbSeparation: 50,
      rangeThumbShape: RoundRangeSliderThumbShape(
          enabledThumbRadius: 10, disabledThumbRadius: 10),
    ),
    child: Slider(
      label: moistTreashold.abs().toString(),
      value: moistTreashold,
      min: 0,
      max: 99.0,
      onChanged: (val) {
        setState(() {
          moistTreashold = val;
        });
      },
    ),
  );
}
```

```dart
Widget slider_row(Widget name, String text) {
    String value_of_slider;
    if (text == "Threshold for Soil-moisture") {
      value_of_slider = (moistTreashold.toInt()).toString();
    } else {
      value_of_slider = (tempTreashold.toInt()).toString();
    }
    return Row(
      mainAxisAlignment: MainAxisAlignment.spaceAround,
      children: <Widget>[
        Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            Container(
              padding: EdgeInsets.fromLTRB(10, 0, 0, 0),
              width: MediaQuery.of(context).size.width * 0.60,
              height: 25,
              color: Colors.transparent,
              child: Text(
                text,
                style: TextStyle(color: Colors.teal, fontSize: 18),
              ),
            ),
            Row(
              children: <Widget>[
                Container(
                  width: MediaQuery.of(context).size.width * 0.78,
                  height: 35,
                  color: Colors.transparent,
                  child: name,
                ),
                Container(
                  padding: EdgeInsets.fromLTRB(20, 3, 5, 5),
                  width: MediaQuery.of(context).size.width * 0.19,
                  height: 35,
                  color: Colors.transparent,
                  child: Text(
                    value_of_slider,
                    style: TextStyle(
                      fontSize: 25,
                      color: Colors.blue,
                    ),
                  ),
                ),
              ],
            ),
          ],
        ),
      ],
    );
  }

  @override
  Widget build(BuildContext context) {
    final appState = Provider.of<MQTTAppState>(context);
    final appState_pub = Provider.of<MQTTAppState>(context);
    currentAppState = appState;
    currentAppState_pub = appState_pub;
    String recText = currentAppState.getReceivedText;

    //_configureAndConnect();

    if (recText != "") {
      if(recText.substring(0,1)=="P"){
        humid=humid;
        temp=temp;
        light=light;
        mos=mos;
      }else {
        humid = recText.substring(0, 2);
        temp = recText.substring(2, 4);
        light = recText.substring(4, 6);
        mos = recText.substring(6);
      }
    } else {
      humid = "??";
      temp = "??";
      mos = "??";
      light = "??";
    }
    //print(recText);
    return Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        backgroundColor: Colors.white10,
        title: Text(
          'Dashboard',
          style: TextStyle(
            color: Colors.white,
          ),
        ),
        centerTitle: true,
      ),
      body: Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: <Widget>[
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceAround,
            children: <Widget>[
              buildButtonS("Moisture", 'assets/tap.png', humid, "%"),
              buildButtonS("Temperature", 'assets/temp.png', temp, "C"),
            ],
          ),
          //buildButtonL("Humidity Temperature Plots", 'assets/plot.png'),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceAround,
            children: <Widget>[
              buildButtonS("Light", 'assets/light.png', light, "%"),
              buildButtonS("Humidity", 'assets/humidity.png', mos, "%"),//
            ],
          ),
          slider_row(custom_slider_temp(), "Threshold for Temperature"),
          slider_row(custom_slider_moist(), "Threshold for Soil-moisture"),
          SizedBox(
            width: 150,
            height: 40,
            child: send_button(currentAppState_pub.getAppConnectionState),
          ),
          SizedBox(
            width: 50,
            height: 50,
            child: RaisedButton(
              //onPressed: (){_configureAndConnect();}
              onPressed : (){configureAndConnect();}
               ),
          )

        ],
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () => Navigator.of(context).push(
          MaterialPageRoute(
            builder: (context) => Settings(),
          ),
        ),
        backgroundColor: Colors.white30,
        child: Icon(Icons.settings),
      ),
    );
  }
}
```

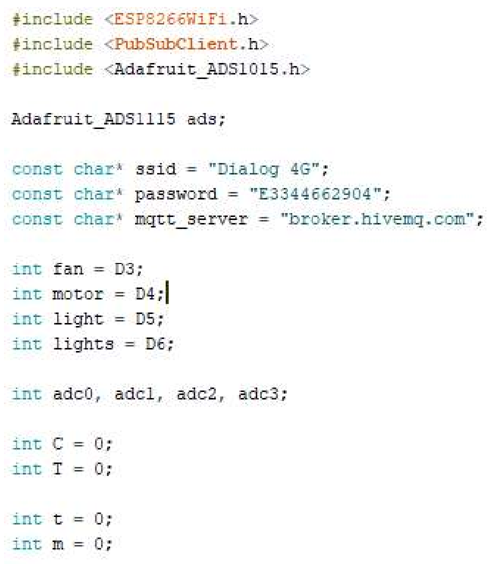
settingsPage.dart

```
import 'dart:io' show Platform;

import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:iotapp/MQTTAppState.dart';
import 'package:iotapp/MQTTManager.dart';
import 'package:iotapp/configSettings.dart';

class Settings extends StatefulWidget {
  @override
  _SettingsState createState() => _SettingsState();
}

class _SettingsState extends State<Settings> {
  MQTTAppState currentAppState;
  MQTTManager manager;
  String hostName;
  String portName;
  String topic;




  void publishMessage(String text){
    final message =text;
    manager.publish(message);
  }

  Material field(String hint, String labelText) {
    return Material(
      child: new Container(
        child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Container(
              padding: EdgeInsets.fromLTRB(4, 2, 0, 2),
              width: MediaQuery.of(context).size.width * 0.98,
              height: MediaQuery.of(context).size.height * 0.09,
              color: Colors.transparent,
              child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: <Widget>[
                  Container(
                    padding: EdgeInsets.fromLTRB(18, 16, 0, 2),
                    width: MediaQuery.of(context).size.width * 0.28,
                    height: MediaQuery.of(context).size.height * 0.085,
                    color: Colors.transparent,
                    child: Text(
                      labelText,
                      style: TextStyle(
                        color: Colors.black,
                        fontSize: 18,
                      ),
                    ),
                  ),
                  Container(
                    width: MediaQuery.of(context).size.width * 0.69,
                    height: MediaQuery.of(context).size.height * 0.085,
                    color: Colors.transparent,
                  child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: <Widget>[
                      new TextField(
                        decoration: new InputDecoration(
                          hintText: hint,
                        ),
                        onChanged: (String str) {
                          setState(() {
                            if (labelText == "Hostname : ") {
                              hostName = str;
                              config().hostName=str;
                            } else if (labelText == "Port : ") {
                              portName = str;
                              config().portName=str;
                            } else if (labelText == "Topic : ") {
                              topic = str;
                              config().topic=str;
                            }
                          });
                        },
                      ),
                    ],
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    ),
  );
}

String prepareStateMessageFrom(MQTTAppConnectionState state) {
  String status;
  switch (state) {
    case MQTTAppConnectionState.connected:
      status = "Connected";
      break;
    case MQTTAppConnectionState.connecting:
      status = "Connecting";
      break;
    case MQTTAppConnectionState.disconnected:
      status = "Disconnected";
      break;
  }
  return status;
}

void _disconnect() {
  manager.disconnect();
}

void _configureAndConnect() {
  manager = MQTTManager(
      host: hostName,
      topic: topic,
      identifier: "ENTC",
      state: currentAppState);
  manager.initializeMQTTClient();
  manager.connect();
}

Widget buildConnectButtonFrom(MQTTAppConnectionState state) {
  return Row(
    mainAxisAlignment: MainAxisAlignment.spaceAround,
    children: <Widget>[
      Container(
        padding: EdgeInsets.fromLTRB(10, 0, 10, 0),
        width: MediaQuery.of(context).size.width,
        height: MediaQuery.of(context).size.height * 0.07,
        color: Colors.transparent,
        child: Row(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: <Widget>[
            SizedBox(
              width: 180,
              height: 40,
              child: RaisedButton(
                color: Colors.green,
                child: Text(
                    'Connect',
                  style: TextStyle(
                    fontSize: 16
                  ),
                ),
                onPressed: state == MQTTAppConnectionState.disconnected
                    ? _configureAndConnect
                    : null,
              ),
            ),
            SizedBox(
              width: 180,
              height: 40,
              child: RaisedButton(
                  color: Colors.red,
                  child: Text(
                      'Disconnect',
                    style: TextStyle(
                      fontSize: 16
                    ),
                  ),
                  onPressed: state == MQTTAppConnectionState.connected
                      ?_disconnect
                      :null
              ),
            )
          ],
        ),

      )
    ],
  );
}

@override
Widget build(BuildContext context) {
  final appState = Provider.of<MQTTAppState>(context);
  currentAppState = appState;
  return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: Colors.black,
        centerTitle: true,
        title: Text(
          "Settings",
          style: TextStyle(
            color: Colors.white,
          ),
        ),
      ),
      body: Column(
        children: <Widget>[
          Container(
            width: MediaQuery.of(context).size.width,
```

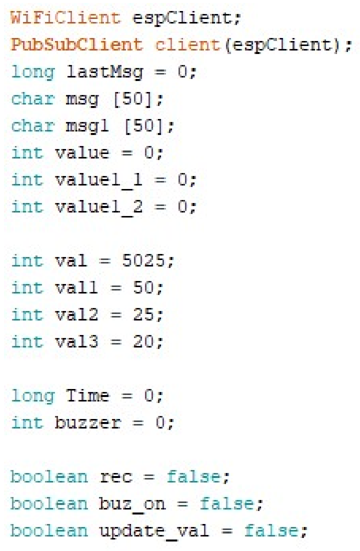


```
              height: MediaQuery.of(context).size.height * 0.025,
              color: Colors.lightBlue,
              child: Text(
                prepareStateMessageFrom(currentAppState.getAppConnectionState),
                textAlign: TextAlign.center,
                style: TextStyle(
                  fontSize: 14,
                ),
              ),
            ),
            field("broker.hivemq.com", "Hostname : "),
            field("1883", "Port : "),
            field("ENTC", "Topic : "),
            buildConnectButtonFrom(currentAppState.getAppConnectionState),
          ],
        ));
  }
}
```

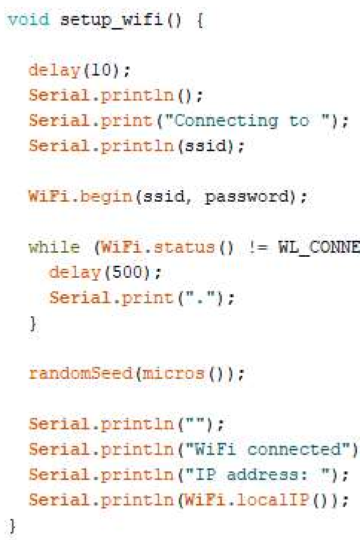
<u>Green house side</u>



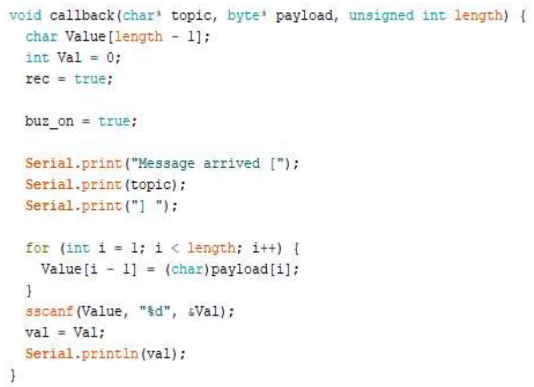
```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Adafruit_ADS1015.h>

Adafruit_ADS1115 ads;

const char* ssid = "Dialog 4G";
const char* password = "E3344662904";
const char* mqtt_server = "broker.hivemq.com";

int fan = D3;
int motor = D4;
int light = D5;
int lights = D6;

int adc0, adc1, adc2, adc3;

int C = 0;
int T = 0;

int t = 0;
int m = 0;
```



```
WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg [50];
char msg1 [50];
int value = 0;
int value1_1 = 0;
int value1_2 = 0;

int val = 5025;
int val1 = 50;
int val2 = 25;
int val3 = 20;

long Time = 0;
int buzzer = 0;

boolean rec = false;
boolean buz_on = false;
boolean update_val = false;
```



```
void setup_wifi() {

  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```



```
void callback(char* topic, byte* payload, unsigned int length) {
  char Value[length - 1];
  int Val = 0;
  rec = true;

  buz_on = true;

  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");

  for (int i = 1; i < length; i++) {
    Value[i - 1] = (char)payload[i];
  }
  sscanf(Value, "%d", &Val);
  val = Val;
  Serial.println(val);
}
```

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      client.subscribe("plant_side");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

```
int control() {

  //margines for errors
  int margine1 = 2;
  int margine2 = 10;

  String Val = String(val);
  val1 = (Val.substring(0, 2)).toInt();
  val2 = (Val.substring(2, 4)).toInt();
  Serial.print(val1);
  Serial.print(val2);
  Serial.println(val3);

  int desiredMoisture = val1;
  int desiredTemp = val2;
  int desiredLightlevel = val3;

  int Temprature = (ads.readADC_SingleEnded(0)) / 85.158;
  int Moisture = (99 - map(ads.readADC_SingleEnded(1), 0, 26400, 0, 89));
  int Lightlevel = map(ads.readADC_SingleEnded(2), 0, 24600, 10, 99);
  int Humidity = map(ads.readADC_SingleEnded(3), 0, 24600, 10, 99);

  if (Temprature <= desiredTemp) {
    digitalWrite(fan, LOW);
    T = 0;
  }



  if (Temprature <= (desiredTemp - margine1)) {
    digitalWrite(light, HIGH);
    digitalWrite(fan, LOW);
    T = 0;
  }

  if (Temprature > desiredTemp) {
    T = 1;
  }

  if (Temprature >= (desiredTemp + margine1)) {
    digitalWrite(light, LOW);
    digitalWrite(fan, HIGH);
    T = 1;
  }
  if ((Temprature > (desiredTemp - margine1)) && (Temprature < (desiredTemp + margine1))) {
    t = 1;
  }
  else {
    t = 0;
  }

  if (Moisture <= (desiredMoisture - margine2)) {
    digitalWrite(motor, HIGH);
  }



  if (Moisture >= desiredMoisture) {
    digitalWrite(motor, LOW);
  }

  if (Moisture > (desiredMoisture + margine2) && T == 1) {
    digitalWrite(motor, LOW);
    digitalWrite(fan, HIGH);
  }

  if ((Moisture > (desiredMoisture - margine2)) && (Moisture < (desiredMoisture + margine2))) {
    m = 1;
  }
  else {
    m = 0;
  }

  if (Lightlevel < desiredLightlevel) {
    digitalWrite(lights, HIGH);
  }

  if (Lightlevel >= desiredLightlevel) {
```

```
  String Msg = (String)Moisture + (String)Temprature + (String)Lightlevel +  (String)Humidity;
  String Msg1_1 = (String)Moisture + (String)Temprature + (String)Lightlevel +  (String)Humidity ;
  String Msg1_2 = (String)val1 + (String)val2 + (String)buzzer;
  value1_1 = Msg1_1.toInt();
  value1_2 = Msg1_2.toInt();
  value = Msg.toInt();

  Serial.print("Moisture = ");
  Serial.print(Moisture);
  Serial.print(" , Temprature = ");
  Serial.print(Temprature);
  Serial.print(" , Lightlevel = ");
  Serial.print(Lightlevel);
  Serial.print(" , Humidity = ");
  Serial.println(Humidity);

  return value;
}
```

```
 void setup() {
   pinMode(fan, OUTPUT);
   pinMode(motor, OUTPUT);
   pinMode(light, OUTPUT);
   pinMode(lights, OUTPUT);
   Serial.begin(115200);
   ads.setGain(GAIN_ONE);
   setup_wifi();
   ads.begin();
   client.setServer(mqtt_server, 1883);
   client.setCallback(callback);
   digitalWrite(motor, LOW);
   digitalWrite(fan, LOW);
   digitalWrite(lights, LOW);
   digitalWrite(light, LOW);
 }
```

```
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  if (rec == true) {
    Time = millis();
    rec = false;
  }
  if (update_val == true) {
    control();
    update_val = false;
  } else {
    buzzer = 0;
    control();

  }

  long duration = millis() - Time;
  if ((duration > 10000) && t == 1 && m == 1 && buz_on == true) {
    buzzer = 0;
    Serial.println("buzzer off");
    Time = millis();
    buz_on = false;
    update_val = true;
  }
  else if ((duration > 10000) && t == 0 && m == 0 && buz_on == true) {
    buzzer = 1;
    Serial.println("buzzer on");
    Time = millis();
    buz_on = false;
    update_val = true;
  }
  snprintf (msg, 75, "%d", value);
  snprintf (msg1, 100, "%d%d", value1_1, value1_2);
  client.publish("node_rec_val", msg1);
  client.publish("app", msg);

  delay(2000);
}
```