**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN ĐIỆN**

**BỘ MÔN THIẾT BỊ ĐIỆN – ĐIỆN TỬ**

****

**BÁO CÁO ĐỒ ÁN TỐT NGHIỆP TUẦN 15**

**Đề tài: Thiết kế thiết bị đóng cắt 4 kênh sử dụng WiFi**

**Sinh viên: Nguyễn Tuấn Anh**

**MSSV: 20173616**

**Học phần: Đồ Án Tốt Nghiệp Cử Nhân**

**Mã học phần: EE**

**Mã lớp:**

**Giảng viên hướng dẫn: TS. Nguyễn Văn Ánh**

**HÀ NỘI, 2020**

---------------------------------------------------------------------------------------------------

Mục lục

[Lời cảm ơn 3](#_Toc59025374)

[Tóm tắt nội dung báo cáo 3](#_Toc59025375)

[Chương 1. Tổng quan về thiết bị điện thông minh 4](#_Toc59025376)

[1.1. Giới thiệu chung về thiết bị điện thông minh 4](#_Toc59025377)

[1.2. Một số thiết bị điện thông minh trong thực tế 4](#_Toc59025378)

[Chương 2. Cấu tạo phần cứng 4](#_Toc59025379)

[2.1. Giới thiệu các linh kiện sử dụng 4](#_Toc59025380)

[2.2. Sơ đồ phần cứng 6](#_Toc59025381)

[2.3. Thiết kế mạch 6](#_Toc59025382)

[2.3.1. Sơ đồ nguyên lý 6](#_Toc59025383)

[2.3.2. Mạch in 7](#_Toc59025384)

[Chương 3. Thiết kế phần mềm 10](#_Toc59025385)

[3.1. Giới thiệu một số phần mềm sử dụng trong quá trình thiết kế phần mềm 10](#_Toc59025386)

[3.2. Giới thiệu một số giao thức truyền thông sử dụng trong thiết bị điện thông minh 10](#_Toc59025387)

[3.3. Giới thiệu giao thức truyền thông MQTT 10](#_Toc59025388)

[3.4. Lưu đồ thuật toán cho Vi điều khiển 12](#_Toc59025389)

[3.5. Chương trình Vi điều khiển 12](#_Toc59025390)

[3.6. Lưu đồ thuật toán cho Phần mềm 19](#_Toc59025391)

[3.7. Chương trình Phần mềm 19](#_Toc59025392)

[Chương 4. Mô hình và kết quả thu được 21](#_Toc59025393)

[Kết luận 21](#_Toc59025394)

[Tài liệu tham khảo 22](#_Toc59025395)

**Danh mục bảng biểu**

**Danh mục hình vẽ**

# Lời cảm ơn

Để hoàn thành bản báo cáo Thực tập kĩ thuật, em xin tỏ lòng biết ơn sâu sắc đến thầy TS. Nguyễn Văn Ánh đã tận tình hướng dẫn trong suốt quá trình tìm hiều, nghiên cứu và làm Đồ án tốt nghiệp cử nhân.

Em xin chân thành cảm ơn các thầy cô trong Viện Điện, Trường Đại học Bách Khoa Hà Nội đã tận tình truyền đạt những kiến thức để em có thể tự tin hoàn thành bản Đồ án này.

Cuối cùng, em xin kính chúc quý thầy, cô dồi dào sức khỏe và thành công trong sự nghiệp cao quý.

Em xin chân thành cảm ơn!

# Tóm tắt nội dung báo cáo

Chương 1. Tổng quan về thiết bị điện thông minh

Chương 2. Cấu tạo phần cứng

Chương 3. Thiết kế phần mềm

Chương 4. Mô hình và kết quả thu được

# Chương 1. Tổng quan về thiết bị điện thông minh

## Giới thiệu chung về thiết bị điện thông minh

## Một số thiết bị điện thông minh trong thực tế

# Chương 2. Cấu tạo phần cứng

## Giới thiệu các linh kiện sử dụng

* + 1. Vi điều khiển ESP8266

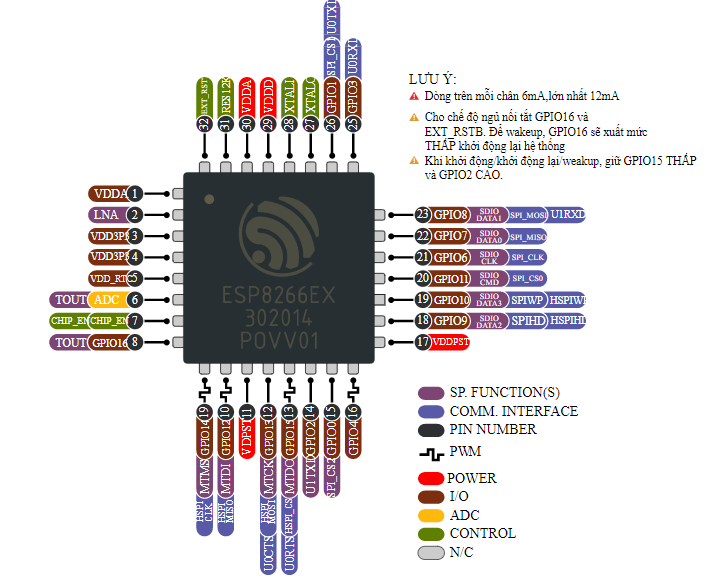
ESP8266 là dòng chip tích hợp Wi-Fi 2.4Ghz có thể lập trình được, rẻ tiền được sản xuất bởi một công ty bán dẫn Trung Quốc: Espressif Systems.

Được phát hành đầu tiên vào tháng 8 năm 2014, đóng gói đưa ra thị trường dạng Mô dun ESP-01, được sản xuất bởi bên thứ 3: AI-Thinker. Có khả năng kết nối Internet qua mạng Wi-Fi một cách nhanh chóng và sử dụng rất ít linh kiện đi kèm. Với giá cả có thể nói là rất rẻ so với tính năng và khả năng ESP8266 có thể làm được.

ESP8266 có một cộng đồng các nhà phát triển trên thế giới rất lớn, cung cấp nhiều Module lập trình mã mở giúp nhiều người có thể tiếp cận và xây dựng ứng dụng rất nhanh.

Hiện nay tất cả các dòng chip ESP8266 trên thị trường đều mang nhãn ESP8266EX, là phiên bản nâng cấp của ESP8266.

Sơ đồ chân của ESP8266



Thông số phần cứng:

* 32-bit RISC CPU: Tensilica Xtensa LX106 running at 80 MHz
* Hổ trợ Flash ngoài từ 512KiB đến 4MiB
* 64KBytes RAM thực thi lệnh
* 96KBytes RAM dữ liệu
* 64KBytes boot ROM
* Chuẩn wifi EEE 802.11 b/g/n, Wi-Fi 2.4 GHz Tích hợp TR switch, balun, LNA, khuếch đại công suất và matching network Hổ trợ WEP, WPA/WPA2, Open network
* Tích hợp giao thức TCP/IP
* Hổ trợ nhiều loại anten
* 16 chân GPIO
* Hổ trợ SDIO 2.0, UART, SPI, I²C, PWM,I²S với DMA
* 1 ADC 10-bit
* Dải nhiệt độ hoạt động rộng : -40C ~ 125C
  + 1. Triac
    2. IC lái Triac
    3. Module chuyền đổi nguồn AC/DC

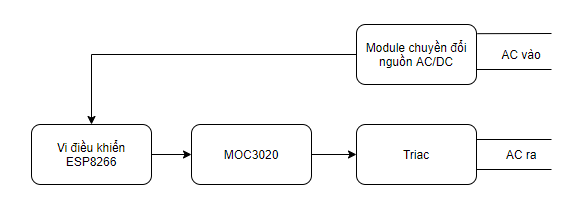
Module nguồn AC-DC Hi-Link HLK-PM01 5VDC 3W có thiết kế nhỏ gọn với vỏ bọc nhựa an toàn, chuyên nghiệp, được sử dụng để chuyển nguồn xoay chiều AC sang 5VDC công suất tối đa 3W cấp cho thiết bị, module được sản xuất bởi hãng Hi-Link chuyên về các module nguồn được sử dụng trong công nghiệp với độ bền, chống nhiễu tốt và độ an toàn cao.



Thông số kĩ thuật:

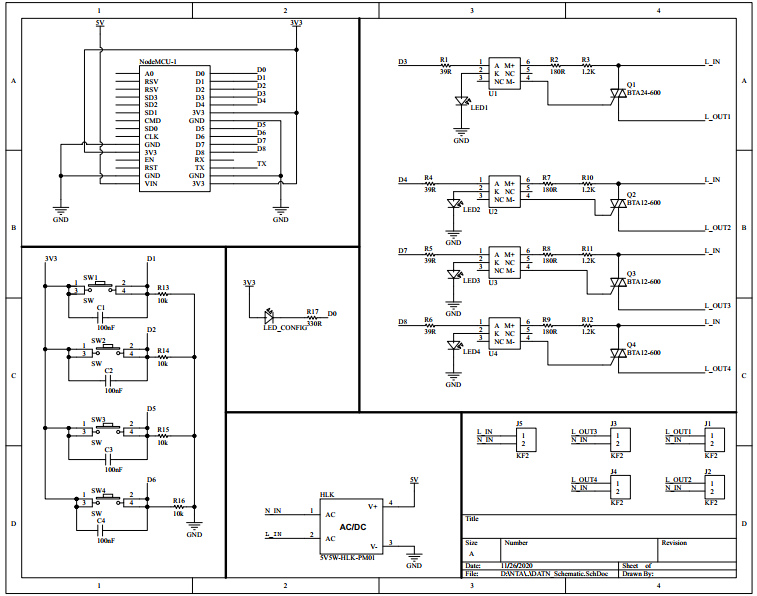
* Điện áp ngõ vào: 100~240 VAC/50~60 Hz
* Điện áp ngõ ra: 5 VDC
* Công suất trung bình: 3W
* Răng cưa và nhiễu nhỏ
* Bảo vệ quá tải và ngắn mạch
* Hiệu suất cao
  + 1. Các linh kiện thụ động khác
* Điện trở
* Tụ điện
* Đèn LED
* Nút bấm

## Sơ đồ phần cứng

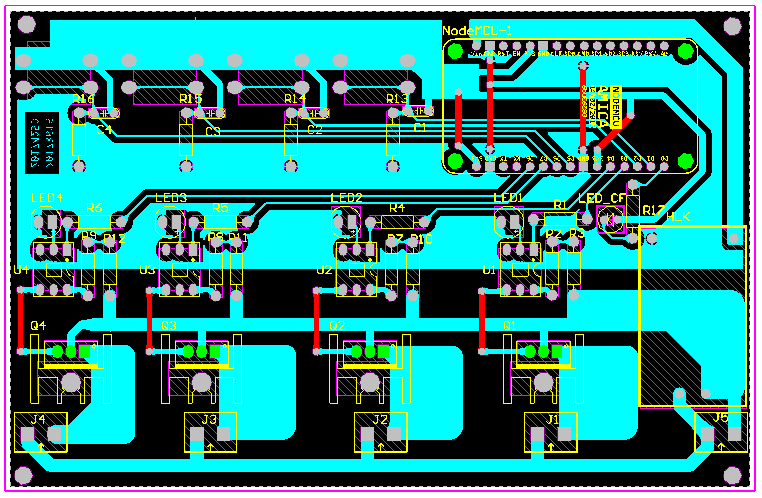


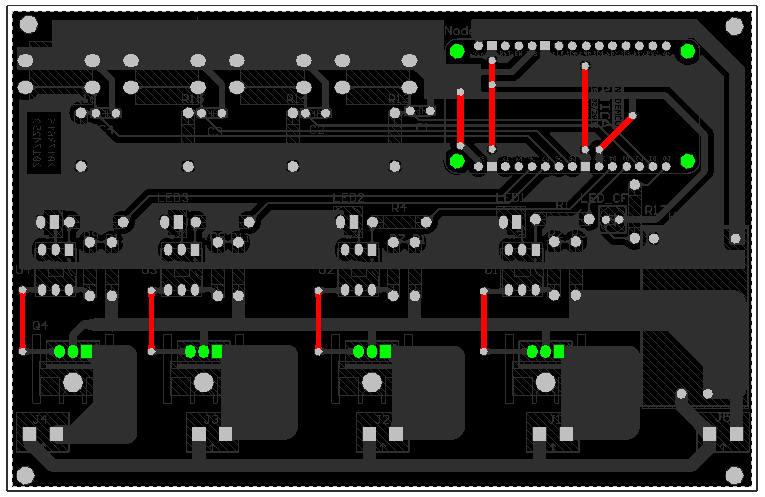
## Thiết kế mạch

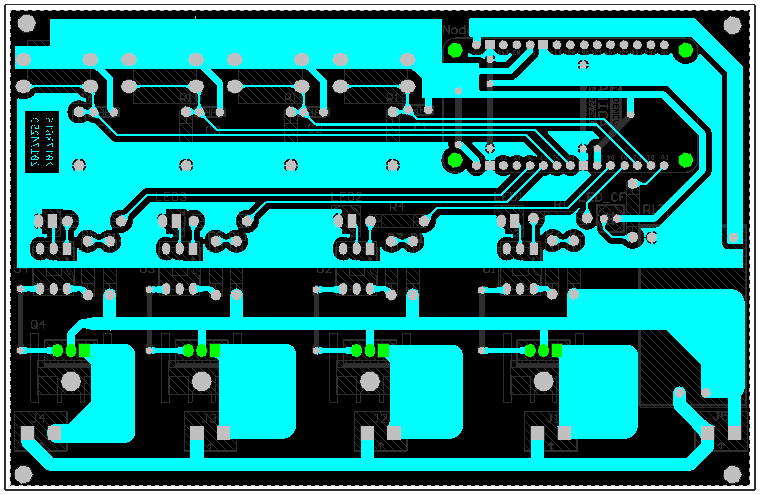
### Sơ đồ nguyên lý

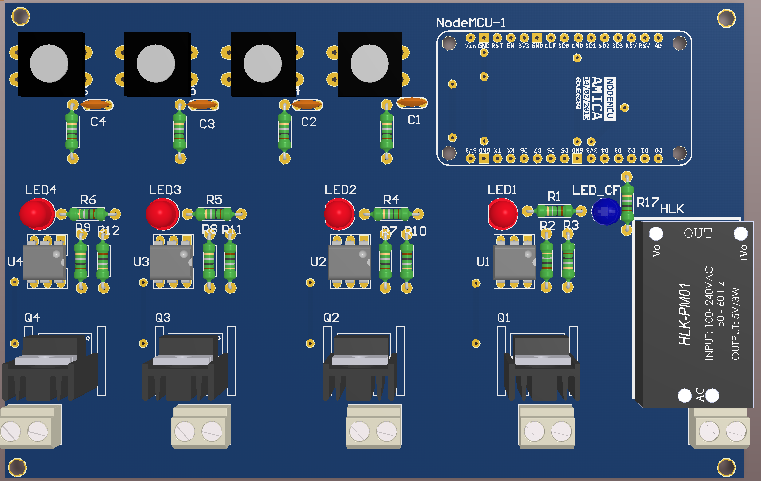


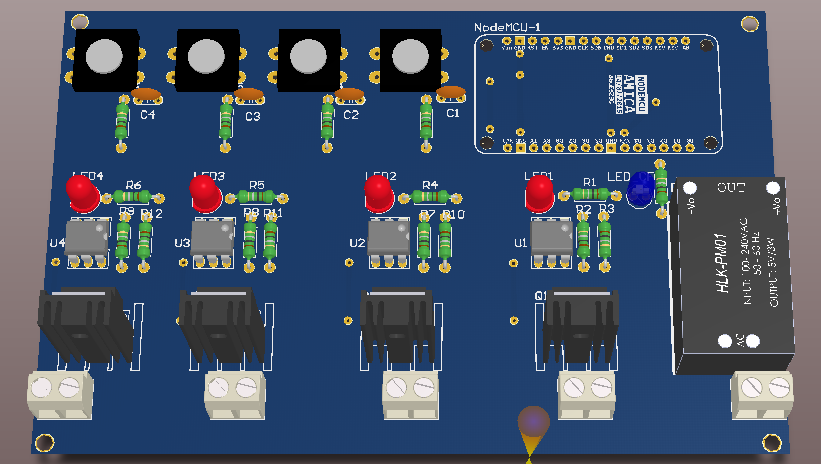
### Mạch in











# Chương 3. Thiết kế phần mềm

## Giới thiệu một số phần mềm sử dụng trong quá trình thiết kế phần mềm

## Giới thiệu một số giao thức truyền thông sử dụng trong thiết bị điện thông minh

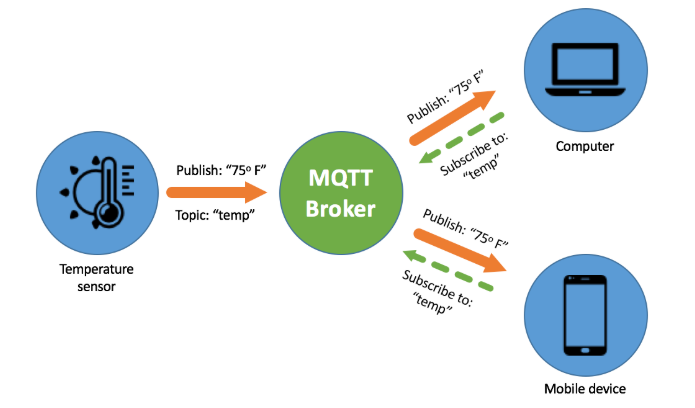
Trong IoT việc giao tiếp giữa các thiết bị với nhau là một trong các yếu tố cơ bản, từ đó dẫn đến yêu cầu phải có một giao thức chung để các thiết bị với phần cứng khác nhau có thể giao tiếp với nhau. Các giao thức truyền dữ liệu phổ biến trong lĩnh vực thiết bị điện thông minh hiện nay là CoAP, MQTT, AMQP, DSS,…

* Giao thức CoAP: Giao thức được thiết kế cho các thiết bị nhúng, về cơ bản giao thức CoAP tương tự như HTTP nhưng các gói tin nhỏ hơn, chủ yếu là giao thức 1-1 để truyền trạng thái thông tin giữa client và server
* Giao thức AMQP: Giao thức AMQP được thiết kế với các gói tin được định hướng, xếp hàng và định tuyến có độ tin cậy và bảo mật cao
* Giao thức DDS: Giao thức truyền tin được thiết kế cho giao tiếp máy-máy, hiệu năng cao, tương thích tốt giữa các phần cứng và phần mềm khác nhau, thời gian thực, có khả năng mở rộng tốt
* Giao thức MQTT: MQTT là giao thức truyền thông theo mô hình Publish/Subscribe.

## Giới thiệu giao thức truyền thông MQTT

MQTT là giao thức truyền thông theo mô hình Publish/Subscribe, sử dụng băng thông thấp, có độ tin cậy cao, có khả năng hoạt động trong môi trường đường truyền không ổn định.

* + 1. Mô hình của giao thức MQTT



Mô hình gồm 2 phần chính là là Broker (với vai trò máy chủ) và Client (máy khách)

* Broker đóng vai trò máy chủ đảm nhận vai trò nhận và chuyển các bản tin. Broker có thể cài đặt trên các máy tính nhúng, hoặc sử dụng Broker Cloud
* Client có thể là các thiết bị chấp hành, các cảm biến, cloud nhận dữ liệu, các thiết bị điện thông minh, …
* Client của MQTT gửi và nhận bản tin dựa trên mô hình Topic. Để nhận bản tin của 1 Topic, Client sẽ Subscribe (đăng kí) Topic đó và khi có bản tin được Publish (xuất bản) đến Topic đó Broker sẽ tự động gửi bản tin đến Client.
* Broker nhận tin nhắn từ các Client gửi đến các Topic, chuyển tiếp bản tin hoặc lưu lại. Ngoài ra Broker cũng đảm nhận thêm một số tính năng khác như bảo mật, …
* QoS: Mức độ tin cậy khi gửi bản tin. Có 3 mức QoS: 0,1 và 2

+ QoS 0: Broker/Client gửi bản tin đúng 1 lần

+ QoS 1: Broker/Client gửi bản tin cho đến khi có xác nhận từ bên nhận

+ QoS 2: Broker/Client gửi bản tin và đảm bảo có xác nhận từ cả 2 phía

* Retain: Bản tin có flag Retain sẽ được Broker lưu lại tại Topic và chuyển đến các Client mới Subscribe Topic đó.
* Birth/Death/LWT:

+ Birth là bản tin được gửi tới các Client khi có thiết bị mới kết nối

+ Death là bản tin được gửi khi có thiết bị mất kết nối

+ LWT là bản tin được cài đặt bới Client, sẽ được gửi tới Topic chỉ định khi Client đó mất kết nối

* Mức độ bảo mật: 1 lớp xác thực bằng ID và mật khẩu khi các Client kết nối với Broker. Mức độ bảo mật có thể tăng thêm bằng các giải pháp bảo mật ở tầng mạng
  + 1. Tại sao lại lựa chọn giao thức MQTT
* Ưu điểm:

+ Yêu cầu băng thông thấp

+ Thích hợp với các thiết bị nhúng

+ Chi phí triển khai thấp

+ Dễ nâng cấp hệ thống

+ Các client dễ kết nối lại khi bị mất kết nối

+ Có các tính năng giúp duy trì kết nối và phát hiện khi các node bị mất kết nối

* Nhược điểm:

+ Sử dụng Broker nên khi Broker gặp sụ cố sẽ cả hệ thống sẽ mất kết nối. Để đảm bảo kết nối thông suốt cần có Broker backup

+ Số lượng Client bị giới hạn bởi cấu hình của Broker

+ Cần sử dụng các biện pháp bên ngoài để tăng tính bảo mật cho hệ thống

## Lưu đồ thuật toán cho Vi điều khiển

## Chương trình Vi điều khiển

|  |
| --- |
| #include <ESP8266WiFi.h>  #include <PubSubClient.h>  const char\* ssid = "nta3100";  const char\* password = "20173616";  const char\* mqtt\_server = "broker.hivemq.com";  const int \_10sec = 10\* 1000;  const int \_1min = 60 \* 1000;  const int \_20sec = 20\*1000;  WiFiClient espClient;  PubSubClient client(espClient);  unsigned long lastUpdate=0;  byte led\_cf = 16; //D0  byte bt1 = 5; //D1  byte bt2 = 4; //D2  byte bt3 = 14; //D5  byte bt4 = 12; //D6  byte out1 = 0; //D3  byte out2 = 2; //D4  byte out3 = 13; //D7  byte out4 = 15; //D8  bool out1\_state = LOW, out2\_state = LOW, out3\_state = LOW, out4\_state = LOW;  bool long\_pressed = false, check\_long\_press = false;  bool pressed = false;  int long\_press\_time = 0, bt\_flag = 0;  long mqtt\_rec, wifi\_rec;  // ham doc nut an  void read\_bt(void);  // ham thay doi trang thai relay  void change\_output(void);  // ham set up wifi  void setup\_wifi(int numper\_try);  // ham nhan msg mqtt  void callback(char\* topic, byte\* payload, unsigned int length);  // ham reconnect server mqtt  void reconnect(int number\_try);  // ham update trang thay hien tai cua switch  void update\_state(void);  // ham xu li nut an giu  void long\_press(void);  // ham isr  ICACHE\_RAM\_ATTR void isr\_pressed(void){  noInterrupts();  delayMicroseconds(1000);  if(digitalRead(bt1) == 1)  {  out1\_state = !out1\_state;  bt\_flag = 1;  }  if(digitalRead(bt2) == 1)  {  out2\_state = !out2\_state;  bt\_flag = 2;  }  if(digitalRead(bt3) == 1)  {  out3\_state = !out3\_state;  bt\_flag = 3;  }  if(digitalRead(bt4) == 1)  {  out4\_state = !out4\_state;  bt\_flag = 4;  }  change\_output();  digitalWrite(led\_cf, HIGH);  long\_press\_time = millis();  check\_long\_press = true;  while(check\_long\_press)  {  switch(bt\_flag)  {  case 1:  if(digitalRead(bt1) == 0) check\_long\_press = false;  break;  case 2:  if(digitalRead(bt2) == 0) check\_long\_press = false;  break;  case 3:  if(digitalRead(bt3) == 0) check\_long\_press = false;  break;  case 4:  if(digitalRead(bt4) == 0) check\_long\_press = false;  break;  default:  break;  }  if(millis() - long\_press\_time > \_10sec)  {  long\_pressed = true;  break;  }  }  interrupts();  }  void setup() {  // put your setup code here, to run once:  // setup wifi  Serial.begin(115200);  pinMode(led\_cf, OUTPUT);  digitalWrite(led\_cf, HIGH);  pinMode(bt1, INPUT);  attachInterrupt(digitalPinToInterrupt(bt1), isr\_pressed, RISING);  pinMode(bt2, INPUT);  attachInterrupt(digitalPinToInterrupt(bt2), isr\_pressed, RISING);  pinMode(bt3, INPUT);  attachInterrupt(digitalPinToInterrupt(bt3), isr\_pressed, RISING);  pinMode(bt4, INPUT);  attachInterrupt(digitalPinToInterrupt(bt4), isr\_pressed, RISING);  pinMode(out1, OUTPUT);  digitalWrite(out1, LOW);  pinMode(out2, OUTPUT);  digitalWrite(out2, LOW);  pinMode(out3, OUTPUT);  digitalWrite(out3, LOW);  pinMode(out4, OUTPUT);  digitalWrite(out4, LOW);  setup\_wifi(3);  client.setServer(mqtt\_server, 1883);  client.setCallback(callback);  }  void loop() {  // put your main code here, to run repeatedly:  // if(pressed == true)  // {  // read\_bt();  // }  // if(check\_long\_press == true)  // {  // switch(bt\_flag)  // {  // case 1:  // if(digitalRead(bt1) == 0) check\_long\_press = false;  // break;  // case 2:  // if(digitalRead(bt2) == 0) check\_long\_press = false;  // break;  // case 3:  // if(digitalRead(bt3) == 0) check\_long\_press = false;  // break;  // case 4:  // if(digitalRead(bt4) == 0) check\_long\_press = false;  // break;  // default:  // break;  // }  // if(millis() - long\_press\_time > \_10sec)  // {  // long\_pressed = true;  // }  // }  // if(long\_pressed == true)  // {  // long\_press();  // }  if(millis() - lastUpdate > \_20sec)  {  lastUpdate = millis();  if(client.connected())  {  update\_state();  }  }    if((WiFi.status() != WL\_CONNECTED) && (millis() - wifi\_rec < \_1min))  {  wifi\_rec = millis();  setup\_wifi(3);  digitalWrite(led\_cf, HIGH);  }  if((WiFi.status() == WL\_CONNECTED) && (millis() - mqtt\_rec < \_1min) && !client.connected())  {  mqtt\_rec = millis();  reconnect(3);  digitalWrite(led\_cf, HIGH);  }  client.loop();  }  void read\_bt(void)  {  pressed = false;  check\_long\_press = true;  long\_press\_time = millis();  delay(100);  if(digitalRead(bt1) == 1)  {  out1\_state = !out1\_state;  bt\_flag = 1;  }  if(digitalRead(bt2) == 1)  {  out2\_state = !out2\_state;  bt\_flag = 2;  }  if(digitalRead(bt3) == 1)  {  out3\_state = !out3\_state;  bt\_flag = 3;  }  if(digitalRead(bt4) == 1)  {  out4\_state = !out4\_state;  bt\_flag = 4;  }  change\_output();  }  void change\_output(void)  {  digitalWrite(out1, out1\_state);  digitalWrite(out2, out2\_state);  digitalWrite(out3, out3\_state);  digitalWrite(out4, out4\_state);  update\_state();  }  void setup\_wifi(int number\_try)  {  int try\_times = 0;  delay(10);  // We start by connecting to a WiFi network  Serial.println();  Serial.print("Connecting to ");  Serial.println(ssid);  WiFi.begin(ssid, password);  while ((WiFi.status() != WL\_CONNECTED) && (try\_times < number\_try)) {  digitalWrite(led\_cf, LOW);  delay(500);  digitalWrite(led\_cf, HIGH);  delay(500);  Serial.print(".");  }  randomSeed(micros());  if(WiFi.status() == WL\_CONNECTED)  {  Serial.println("");  Serial.println("WiFi connected");  Serial.println("IP address: ");  Serial.println(WiFi.localIP());  }  else  {  Serial.println("WiFi not connected");  }  }  void reconnect(int number\_try)  {  int try\_times = 0;  while((!client.connected()) && (try\_times < number\_try))  {  Serial.print("Attemp mqtt connection");  String clientID = "esp8266-datn";  if(client.connect(clientID.c\_str()))  {  Serial.println("connected");  client.publish("status/datnta", "connected");  // client.subscribe("status/datnta");  client.subscribe("cmd/datnta");  }  else  {  Serial.print("fail, rc=");  Serial.print(client.state());  Serial.print("try again in 5sec");  try\_times++;  digitalWrite(led\_cf, LOW);  delay(1000);  digitalWrite(led\_cf, HIGH);  delay(1000);  }  }  }  void callback(char\* topic, byte\* payload, unsigned int length)  {  for (int i = 0; i < length; i++) {  Serial.print((char)payload[i]);  }  Serial.println();  if((char)payload[0] == '/')  {  if((char)payload[1] == '1')  {  if((char)payload[3] == '1') out1\_state = HIGH;  else out1\_state = LOW;  }  else if ((char)payload[1] == '2')  {  if((char)payload[3] == '1') out2\_state = HIGH;  else out2\_state = LOW;  }  else if ((char)payload[1] == '3')  {  if((char)payload[3] == '1') out3\_state = HIGH;  else out3\_state = LOW;  }  else if ((char)payload[1] == '4')  {  if((char)payload[3] == '1') out4\_state = HIGH;  else out4\_state = LOW;  }  change\_output();  update\_state();  }  }  void update\_state(void)  {  if(out1\_state == 1)  {  client.publish("status/datnta","1/1");  }  else  {  client.publish("status/datnta","1/0");  }    if(out2\_state == 1)  {  client.publish("status/datnta","2/1");  }  else  {  client.publish("status/datnta","2/0");  }    if(out3\_state == 1)  {  client.publish("status/datnta","3/1");  }  else  {  client.publish("status/datnta","3/0");  }    if(out4\_state == 1)  {  client.publish("status/datnta","4/1");  }  else  {  client.publish("status/datnta","4/0");  }  }  void long\_press(void)  {  long\_pressed = false;  digitalWrite(led\_cf, LOW);  delayMicroseconds(\_1min \* 1000);  } |

## Lưu đồ thuật toán cho Phần mềm

## Chương trình Phần mềm

Chương trình chính:

|  |
| --- |
| import kivy  from kivy.app import App  from kivy.uix.label import Label  from kivy.uix.gridlayout import GridLayout  from kivy.uix.button import Button  from kivy.uix.widget import Widget  from kivy.properties import ObjectProperty  from kivy.properties import StringProperty  from kivy.clock import Clock, mainthread  import paho.mqtt.client as mqtt  # MQTT protocol  topicSubscribe = ["status/datnta"]  topicPublish = "cmd/datnta"  mqtt\_server = "broker.hivemq.com"  mqtt\_port = 1883  buttonState = [0,0,0,0]  # user = ""  # pw = ""  def on\_connect(client, userdata, flags, rc):  print("Connected with result code "+str(rc))  for topic in topicSubscribe:  client.subscribe(topic)  # client.username\_pw\_set(user, pw)  # client = mqtt.Client()  # App GUI and logic  class MyGrid(Widget):  button1 = ObjectProperty(None)  button2 = ObjectProperty(None)  button3 = ObjectProperty(None)  button4 = ObjectProperty(None)  @mainthread  def update(self, button\_id, state):  if button\_id == '1':  if state == '1':  self.button1.text = "Button 1 ON"  self.button1.background\_color = [240, 255, 0, 1]  buttonState[0] = 1  elif state == '0':  self.button1.text = "Button 1 OFF"  self.button1.background\_color = [255, 255, 255, 1]  buttonState[0] = 0  if button\_id == '2':  if state == '1':  self.button2.text = "Button 2 ON"  buttonState[1] = 1  elif state == '0':  self.button2.text = "Button 2 OFF"  buttonState[1] = 0  if button\_id == '3':  if state == '1':  self.button3.text = "Button 3 ON"  buttonState[2] = 1  elif state == '0':  self.button3.text = "Button 3 OFF"  buttonState[2] = 0  if button\_id == '4':  if state == '1':  self.button4.text = "Button 4 ON"  buttonState[3] = 1  elif state == '0':  self.button4.text = "Button 4 OFF"  buttonState[3] = 0  def btn(self, bt\_number):  DATNApp().sendCmd(bt\_number)    class DATNApp(App):  client = mqtt.Client()  def on\_message(self, client, userdata, message):  data = message.payload.decode("utf-8")  self.root.update(data[0], data[2])  def build(self):  self.client.on\_message = self.on\_message  self.client.connect(mqtt\_server, mqtt\_port, 60)  self.client.loop\_start()  self.client.subscribe("status/datnta")  return MyGrid()  def sendCmd(self, bt\_number):  str\_temp = "/" + str(bt\_number) + "/"  if buttonState[int(bt\_number) - 1] == 0:  str\_temp += '1'  self.client.publish(topicPublish, str\_temp)  else:  str\_temp += '0'  self.client.publish(topicPublish, str\_temp)  if \_\_name\_\_ == "\_\_main\_\_":  DATNApp().run() |

Chương trình giao diện

|  |
| --- |
| <MyGrid>:  button1: button1  button2: button2  button3: button3  button4: button4  GridLayout:  cols: 2  size: root.width, root.height  Button:  id: button1  text: "Button 1"  color: [0,0,0,1]  font\_size: 50  on\_press: root.btn(1)  Button:  id: button2  text: "Button 2"  color: [0,0,0,1]  font\_size: 50  on\_press: root.btn(2)  Button:  id: button3  text: "Button 3"  color: [0,0,0,1]  font\_size: 50  on\_press: root.btn(3)  Button:  id: button4  text: "Button 4"  color: [0,0,0,1]  font\_size: 50  on\_press: root.btn(4) |

# Chương 4. Mô hình và kết quả thu được

# Kết luận

Trong quá trình thực hiện do kiến thức còn ít ỏi nên có thể gặp nhiều sai sót, em mong được thầy góp ý để bài làm của em trở nên hoàn thiện hơn.

Em xin chân thành cảm ơn.

# Tài liệu tham khảo