TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN  
& TRUYỀN THÔNG VIỆT- HÀN

**KHOA KỸ THUẬT MÁY TÍNH VÀ ĐIỆN TỬ**

**🙟🕮🙝**

****

**BÁO CÁO**

**ĐỒ ÁN CƠ SỞ 4**

**ĐỀ TÀI:**

**XÂY DỰNG HỆ THỐNG ĐIỀU KHIỂN THIẾT BỊ LÀM MÁT THÔNG QUA SMARTPHONE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| **Sinh viên thực hiện** | **:** | **Trần Văn Thanh** |  | **20CE044** |  |
|  |  | **Trần Huỳnh Anh Nhật** |  | **20CE040** |  |
| **Giảng viên hướng dẫn** | **:** | **TS. Vương Công Đạt** |  |  |  |
| **Lớp** | **:** | **20CE** |  |  |  |

***Đà Nẵng, tháng 12 năm 2022***

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN &

TRUYỀN THÔNG VIỆT - HÀN

**KHOA KỸ THUẬT MÁY TÍNH VÀ ĐIỆN TỬ**



**BÁO CÁO**

**ĐỒ ÁN CƠ SỞ 4**

**ĐỀ TÀI:**

**XÂY DỰNG HỆ THỐNG ĐIỀU KHIỂN THIẾT BỊ LÀM MÁT THÔNG QUA SMARTPHONE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sinh viên thực hiện** | **:** | **Trần Văn Thanh** | **20CE044** |  |  |
|  |  | **Trần Huỳnh Anh Nhật** | **20CE040** |  |  |
| **Giảng viên hướng dẫn** | **:** | **TS. Vương Công Đạt** |  |  |  |
| **Lớp** | **:** | **20CE** |  |  |  |

**Đà Nẵng, tháng 12 năm 2022**

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

**Giảng viên hướng dẫn**

LỜI CẢM ƠN

Chúng em xin trân trọng cảm ơn chân thành đến ban giám hiệu nhà trường, cùng các thầy cô đã tận tình chỉ bảo giảng dạy, cung cấp kiến thức nền tảng tạo điều kiện tốt nhất để chúng em học tập trau dồi.

Đặc biệt em xin chân thành cảm ơn TS. Vương Công Đạt đã nhiệt tình hướng dẫn chúng em trong suốt quá trình xây dựng sản phẩm.

Trong quá trình thực hiện đề tài, chúng em đã rất cố gắng nỗ lực tuy nhiên không tránh khỏi sai sót. Chúng em mong nhận được sự góp ý của thầy cô giáo và từ phía hội đồng nhà trường để có thể thực hiện hoàn chỉnh hơn ở những chủ đề.

*Chúng em xin chân thành cảm ơn*!

MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc122777665)

[MỤC LỤC 2](#_Toc122777666)

[DANH MỤC HÌNH ẢNH 4](#_Toc122777667)

[MỞ ĐẦU 5](#_Toc122777668)

[1. Giới thiệu 5](#_Toc122777669)

[2. Mục tiêu của đề tài 5](#_Toc122777670)

[3. Nội dung và kế hoạch thực hiện 5](#_Toc122777671)

[4. Bố cục báo cáo 5](#_Toc122777672)

[CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI 7](#_Toc122777673)

[1.1. Giới thiệu về đề tài 7](#_Toc122777674)

[1.1.1. Đặt vấn đề 7](#_Toc122777675)

[1.1.2. Ý tưởng 7](#_Toc122777676)

[1.2. Một số linh kiện quan trọng trong mạch điều khiển 7](#_Toc122777677)

[1.2.1. ESP8266 NODE MCU 7](#_Toc122777678)

[1.2.2. PIC16F887-PU 8](#_Toc122777679)

[1.2.3. Relay 9](#_Toc122777680)

[1.2.4. Cảm biến nhiệt đó LM35 9](#_Toc122777681)

[1.3. Kết Chương 1 10](#_Toc122777682)

[CHƯƠNG 2. PHÂN TÍCH HỆ THỐNG 11](#_Toc122777683)

[2.1. Sơ Đồ Khối 11](#_Toc122777684)

[2.2. Sơ đồ mạch và danh sách linh kiện 11](#_Toc122777685)

[2.3. Cách thức hoạt động của hệ thống 12](#_Toc122777686)

[2.4. Một số kỹ thuật quan trọng 13](#_Toc122777687)

[2.4.1. Giao Thức MQTT 13](#_Toc122777688)

[2.4.2. Giao Tiếp UART 14](#_Toc122777689)

[2.4.3. Bộ chuyển đổi tương tự - số 15](#_Toc122777690)

[2.5. Quy trình làm và thiết kế mạch thủ công 16](#_Toc122777691)

[CHƯƠNG 3. THIẾT KẾ HỆ THỐNG 19](#_Toc122777692)

[3.1. Vẽ Sơ đồ mạch trên phần mềm Altium 19](#_Toc122777693)

[3.2. Xây dựng sản phẩm trên board đồng 19](#_Toc122777694)

[3.3. Source code cho hệ thống 22](#_Toc122777695)

[3.4. Thiết lập phần mềm 22](#_Toc122777696)

[3.5. Kết chương 3 23](#_Toc122777697)

[KẾT LUẬN 24](#_Toc122777698)

[TÀI LIỆU THAM KHẢO 25](#_Toc122777699)

[PHỤ LỤC 26](#_Toc122777700)

[Phụ lục 1: Chương trình cho PIC16F887 26](#_Toc122777701)

[Phụ lục 2: Chương trình cho module ESP8266 29](#_Toc122777702)

DANH MỤC HÌNH ẢNH

[Hình 1.1. ESP8266 Node MCU 8](#_Toc122777703)

[Hình 1.2. PIC16F887 8](#_Toc122777704)

[Hình 2.1. Sơ đồ khối 11](#_Toc122777705)

[Hình 2.2. Sơ đồ nguyên lý 12](#_Toc122777706)

[Hình 2.3. Sơ đồ thuật toán 13](#_Toc122777707)

[Hình 2.4. Vị trí của MQTT trong mô hình IoT 14](#_Toc122777708)

[Hình 2.5. Mạch nguyên lý 17](#_Toc122777709)

[Hình 2.6. Mạch PCB 17](#_Toc122777710)

[Hình 2.7. Mạch in trên PDF 18](#_Toc122777711)

[Hình 3.1. Chọn linh kiện trên Altium 19](#_Toc122777712)

[Hình 3.2. Sơ đồ mạch nguyên lý 19](#_Toc122777713)

[Hình 3.3. Sản phẩm hoàn chỉnh 22](#_Toc122777714)

[Hình 3.4. Topic trên Adafruit 23](#_Toc122777715)

[Hình 3.5. Câu lệnh điều khiển trên IFTTT 23](#_Toc122777716)

MỞ ĐẦU

## 1. Giới thiệu

Thế giới đang bước vào thời đại của Internet vạn vật (IoT), với sự xuất hiện của thiết bị thông minh, nhà thông minh. Nhưng mọi thứ không dừng lại ở đó. Các ứng dụng công nghệ đến từ Internet vạn vật (IoT) hứa hẹn mang đến những giá trị to lớn cho con người. Hầu hết mọi thiết bị công nghệ trên thế giới đều có ít nhất là một bản mạch trong đó. Vì vậy việc biết và xây dựng một hệ thống mạch là một điều tất yếu của mọi sinh viên ngành kỹ thuật.

**2. Mục tiêu của đề tài**

Mục tiêu của đề tài thiết bị sẽ có thể bật tắt thiết bị làm mát thông qua một vài thiết bị thông minh.

Ngoài ra đề tài còn giúp sinh viên hiểu thêm về đặc điểm cấu tạo, chức năng, nguyên lý hoạt động của các loại linh kiện điện tử, làm quen với các loại cảm biến, rèn luyện kỹ năng đội nhóm, kỹ năng thuyết trình, kỹ năng tra cứu thông tin, kỹ năng thiết kế và xây dựng một hệ thống mạch.

**3. Nội dung và kế hoạch thực hiện**

|  |  |
| --- | --- |
| **Nội dung** | **Kế hoạch thực hiện** |
| Lên ý tưởng và tìm nguồn tham khảo | Thành viên trong nhóm sẽ tự nghĩ ý tưởng trong một tuần và tìm ý tưởng từ nhiều nguồn sau đó cùng nhau thống nhất đề tài |
| Tìm nguồn linh kiện | Tham khảo giá trên các shop thuộc các trang thương mại điện tử và tìm shop tại địa phương dựa trên các tiêu chí: Giá cả - Độ tin cậy – Tính đa dạng |
| Làm đề cương đồ án và tiếp nhận ý kiến của giảng viên | Liên hệ giảng viên hướng dẫn để lắng nghe góp ý về đồ án, hướng phát triển, …. |
| Vẽ sơ đồ mạch và bắt đầu xây dựng | Vẽ sơ đồ mạch |
| Chuẩn bị bài thuyết trình và báo cáo chi tiết bảo vệ đồ án |  |

**4. Bố cục báo cáo**

Sau phần *Mở đầu*, báo cáo được trình bày trong ba chương, cụ thể như sau:

Chương 1. Tổng quan về đề tài và giới thiệu các linh kiện chính có trong mạch.

Chương 2. Phân tích về sơ đồ mạch, nguyên lý hoạt động của mạch, phân tích. chức năng.

Chương 3. Xây dựng ở chương này.

Cuối cùng là *Kết luận*, *Tài liệu tham khảo* và *Phụ lục* liên quan đến đề tài.

# TỔNG QUAN VỀ ĐỀ TÀI

## Giới thiệu về đề tài

### Đặt vấn đề

Internet of Things là một vấn đề cốt lõi, quan trọng trong cuộc cách mạng công nghiệp 4.0. Internet of Things (IoT) là một mạng gồm các hệ thống nhúng (thiết bị) có kết nối với internet, cho phép chúng kết nối và tương tác với các thiết bị nhúng khác, dịch vụ và con người trên quy mô lớn. IoT được ứng dụng phổ biến trong hệ thống nhà thông minh liên quan đến an ninh, tiết kiệm năng lượng, chăm sóc sức khỏe, sự hiện đại, cải thiện chất lượng cuộc sống.

### Ý tưởng

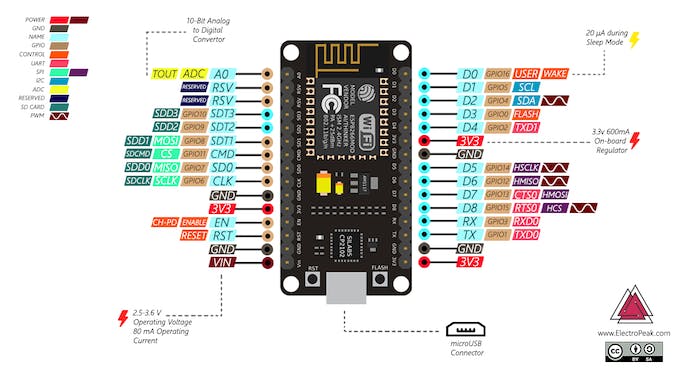
Nhằm mục đích tiết kiệm năng lượng điện, và tăng tính hiện đại trong mỗi ngôi nhà, nhóm đặt ra ý tưởng xây dựng hệ thống điều khiển thông minh cho thiết bị làm mát. Thiết bị làm mát sẽ được bật/tắt tự động tùy vào nhiệt độ môi trường hoặc có thể do con người điều khiển thông qua ứng dụng GoogleHome. Hệ thống giúp người dùng có thể theo dõi nhiệt độ trong chính ngôi nhà của mình. Hệ thống bao gồm mạch điều khiển thiết bị, thiết bị làm mát, phần mềm theo dõi nhiệt độ và điều khiển bằng giọng nói (đã được phát triển).

## Một số linh kiện quan trọng trong mạch điều khiển

### ESP8266 NODE MCU

#### ESP8266 NODE MCU là gì?

NodeMCU là một nền tảng nguồn mở dựa trên ESP8266 có thể kết nối các đối tượng và cho phép truyền dữ liệu qua Wi-Fi. Ngoài ra, bằng cách cung cấp một số tính năng quan trọng nhất của vi điều khiển như GPIO, PWM, ADC, v.v., nó có thể tự mình giải quyết nhiều nhu cầu của các dự án.



Hình .. ESP8266 Node MCU

#### Đặc điểm chung

Dễ sử dụng.

Khả năng lập trình với ngôn ngữ Arduino IDE hoặc IUA.

Có sẵn dưới dạng điểm truy cập hoặc trạm.

Có thể thực hiện được trong các ứng dụng API hướng sự kiện.

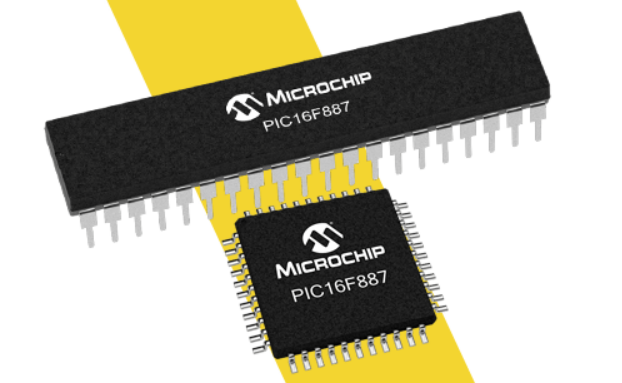
Có anten bên trong.

Chứa 13 chân GPIO, 10 kênh PWM, I2C, SPI, ADC, UART và 1-Wire.

### PIC16F887-PU

#### PIC16F887-PU là gì?

NodeMCU là một nền tảng nguồn mở dựa trên ESP8266 có thể kết nối các đối tượng và cho phép truyền dữ liệu bằng giao thức Wi-Fi. Ngoài ra, bằng cách cung cấp một số tính năng quan trọng nhất của vi điều khiển như GPIO, PWM, ADC, v.v., nó có thể tự mình giải quyết nhiều nhu cầu của các dự án.



Hình .. PIC16F887

#### Thông số kỹ thuật

Bộ nhớ chương trình: 8 (KB)

Tốc độ CPU (MIPS): 5

Byte RAM: 368

EEPROM (byte): 256

Thiết bị ngoại vi giao tiếp: Mô-đun giao tiếp nối tiếp UART, I2C, SPI, CCP

Thiết bị ngoại vi PWM: ĐCSTQ

Hẹn giờ: 2 x 8-bit, 1x16-bit

ADC: 14 ch, 10-bit

Điện áp hoạt động (V): 2 đến 5,5

Số chân I/O: 35

### Relay

Relay là một công tắc hoạt động bằng điện. Nó bao gồm một tập hợp các đầu nối đầu vào cho một hoặc nhiều tín hiệu điều khiển và một tập hợp các đầu nối tiếp điểm vận hành. Công tắc có thể có bất kỳ số lượng liên hệ nào ở nhiều dạng liên hệ, chẳng hạn như tạo liên hệ, ngắt liên hệ hoặc kết hợp của chúng.

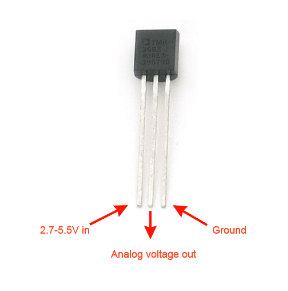


Hình .3. Relay

### Cảm biến nhiệt độ LM35

#### Cảm biến nhiệt độ LM35 là gì?

LM35 là một cảm biến nhiệt độ tương tự tích hợp có đầu ra điện tỷ lệ với độ C. Cảm biến LM35 không yêu cầu bất kỳ hiệu chuẩn hoặc cắt tỉa bên ngoài nào để cung cấp độ chính xác điển hình. Trở kháng đầu ra thấp, đầu ra tuyến tính và hiệu chuẩn vốn có chính xác của LM35 làm cho việc kết nối với mạch đọc hoặc mạch điều khiển trở nên đặc biệt dễ dàng.



Hình .4. Cảm biến LM35

#### Tính năng của cảm biến nhiệt độ LM35

Hiệu chuẩn trực tiếp theo độ C (C).

Tuyến tính ở hệ số tỷ lệ 10,0 mV/°C.

Có thể đảm bảo độ chính xác 0,5°C (ở a25°C).

Được xếp hạng cho toàn bộ phạm vi -55°C đến 150°C.

Thích hợp cho các ứng dụng từ xa.

Chi phí thấp do cắt tỉa ở mức wafer.

Hoạt động từ 4 đến 30 volt.

Ít hơn 60 mA cống hiện tại

Tự sưởi ấm thấp, 0,08°C thấm khí.

Phi tuyến tính chỉ 0,25°C điển hình.

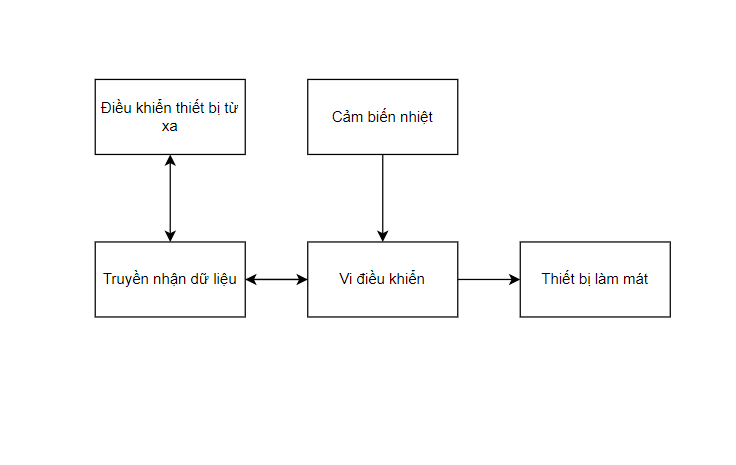
Đầu ra trở kháng thấp, 0,1Ω cho tải 1 mA.

## Kết Chương 1

Thông qua tìm hiểu về cấu tạo, chức năng nguyên lý hoạt động của từng loại linh kiện, các thành phần cấu thành nên mạch. Ta đã hiểu được những khái niệm về các loại công nghệ và chức năng của chúng làm cơ sở để phân tích thiết kế và xây dựng sản phẩm hoàn chỉnh sẽ được trình bày trong chương tiếp theo.

# PHÂN TÍCH HỆ THỐNG

## Sơ Đồ Khối



Hình .. Sơ đồ khối

Điều khiển thiết bị từ xa: trong hệ thống sử dụng điện thoại thông minh làm thiết bị điều khiển từ xa chính, ứng dụng google home sẽ kết nối với IFTTT, người dùng sẽ dùng google assistant để ra lệnh Bật/Tắt.

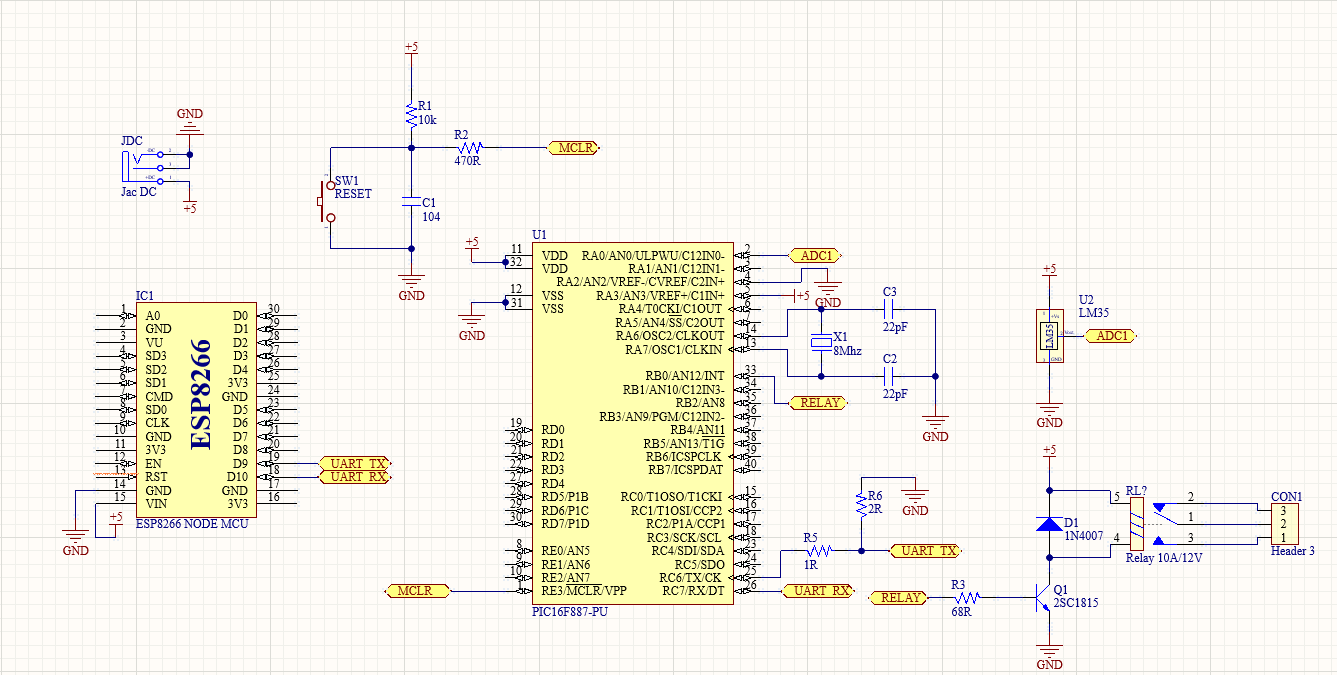
Cảm biến nhiệt : sử dụng cảm biến nhiệt độ LM35 để thu thập dữ liệu liên tục , thông qua cảm biến để hệ thống tự động hoạt động và gửi thông tin lên Adafruit.

Truyền nhận dữ liệu: Sử dụng ESP8266 để kết nối wifi, nhận tín hiệu điều khiển từ smartphone bằng giao thức MQTT, gửi tín hiệu cho vi điều khiển PIC16F887 bằng giao tiếp UART và ngược lại nhận dữ liệu nhiệt độ từ PIC16F887 sau đó gửi lên web.

Vi Điều Khiển: sử dụng PIC16F887 để điều khiển bật tắt relay khi nhận được tín hiệu điều khiển và xử lý dữ liệu từ LM35.

Thiết Bị Làm Mát: hệ thống sử dụng quạt để mô phỏng hoạt động.

## Sơ đồ mạch và danh sách linh kiện



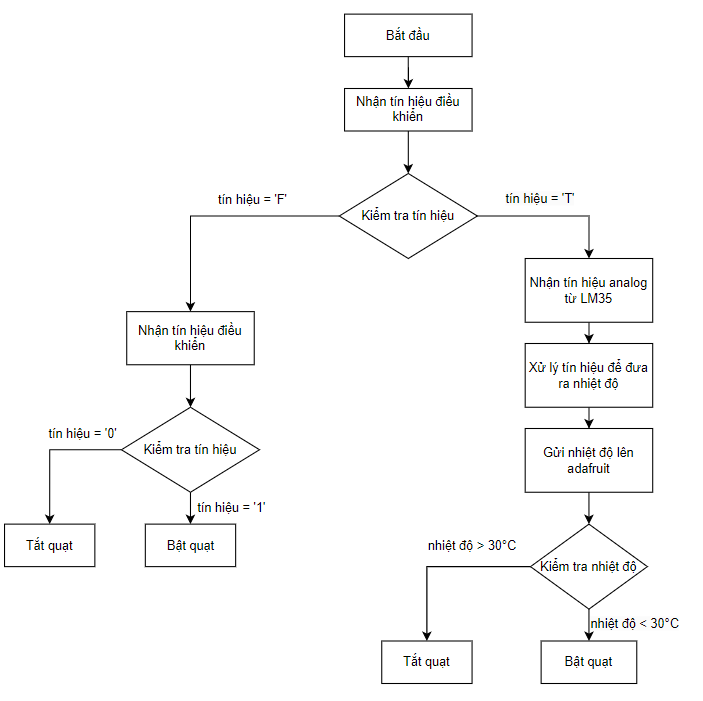
Hình .. Sơ đồ nguyên lý

Các linh kiện sử dụng:

* ESP8266 NODE MCU
* PIC16F887-PU
* Resistor
* Relay
* LM35 sensor
* Switch
* Thạch anh
* Transistor 2SC1815
* Tụ điện
* Diode 1N4007

## Cách thức hoạt động của hệ thống

Hệ thống sử dụng vi điều khiển PIC16F887 để kích hoạt relay nhằm bật/tắt quạt. Từ thiết bị smartphone, thông qua ứng dụng google home chúng có thể ra lệnh điều khiển để quạt hoạt động ở chế độ tự động dựa vào nhiệt độ môi trường hoặc hoạt động nhờ bật tắt qua smartphone. Tín hiệu điều khiển sẽ được gửi đến vi điều khiển PIC thông qua module wifi ESP8266. Ở chế độ tự động, nhờ vào cảm biến LM35 mà hệ thống có thể đo được nhiệt độ của môi trường xung quanh sau đó sẽ thực hiện đóng relay nếu nhiệt độ môi trường lớn hơn 30°C và ngược lại, ngắt relay nếu nhiệt độ nhỏ hơn 30°C. Ở chế độ tự động, nhiệt độ môi trường sẽ được gửi lên Adafruit, nhờ đó mà người dung có thể theo dõi nhiệt độ trong phòng một cách thường xuyên.



Hình .. Sơ đồ thuật toán

Hệ thống có 2 chế độ hoạt động:

Điều khiển thông qua smart phone: lệnh sẽ truyền từ điện thoại thông minh thông qua ứng dụng googlehome , tín hiệu sẽ được gửi đến ESP8266 từ đây tín hiệu sẽ được đưa đến Pic16F887 để bật quạt (tín hiệu đến 1), tắt quạt (tín hiệu đến 0 ).

Tự động: Thông qua cảm biến nhiệt độ LM35, hệ thống nhận tín hiệu, từ đó sẽ tự động bật quạt khi nhiệt độ quá 30ºC và tắt quạt khi dưới 30ºC. Đồng thời nhiệt độ sẽ được cập nhật trên adafruit.

## Một số kỹ thuật quan trọng

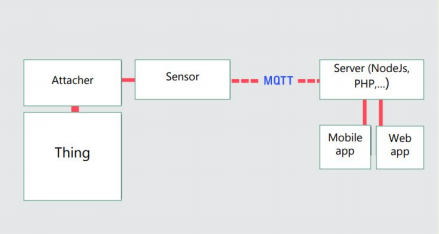
### Giao Thức MQTT

#### Giao Thức MQTT là gì?

MQTT (Message Queuing Telemetry Transport) là giao thức truyền thông điệp (message) theo mô hình publish/subscribe (cung cấp / thuê bao), được sử dụng cho các thiết bị IoT với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định.

#### Ưu Điểm Của Giao Thức MQTT

* Truyền thông tin hiệu quả hơn.
* Tăng khả năng mở rộng.
* Giảm đáng kể tiêu thụ băng thông mạng.
* Rất phù hợp cho điều khiển và do thám.
* Tối đa hóa băng thông có sẵn.
* Chi phí thấp.
* Rất an toàn, bảo mật.
* Được sử dụng trong các ngành công nghiệp dầu khí, các công ty lớn như Amazon, Facebook, ....
* Tiết kiệm thời gian phát triển.
* Giao thức publish/subscribe thu thập nhiều dữ liệu hơn và tốn ít băng thông hơn so với giao thức cũ.



Hình .. Vị trí của MQTT trong mô hình IoT

### Giao Tiếp UART

#### Giao Tiếp UART là Gì?

UART (Universal synchronous asynchronous receiver transmitter ) là một ngoại vi cơ bản của STM32 sử dụng 2 chân Rx và Tx để nhận và truyền dữ liệu.

UART truyền dữ liệu không đồng bộ, có nghĩa là không có tín hiệu để đồng bộ hóa đầu ra của các bit từ UART truyền đến việc lấy mẫu các bit bởi UART nhận. Thay vì tín hiệu đồng bộ, UART truyền thêm các bit start và stop vào gói dữ liệu được chuyển. Các bit này xác định điểm bắt đầu và điểm kết thúc của gói dữ liệu để UART nhận biết khi nào bắt đầu đọc các bit.

#### Các Thông Số Cơ Bản Tron Truyền Nhận UART

Baund rate (tốc độ baund): Khoảng thời gian dành cho 1 bit được truyền. Phải được cài đặt giống nhau ở gửi và nhận. Một số Baud Rate thông dụng: 9600, 38400, 115200, 230400,…

Frame (khung truyền): Khung truyền quy định về số bit trong mỗi lần truyền.

Start bit: là bit đầu tiên được truyền trong 1 Frame. Báo hiệu cho thiết bị nhận có một gói dữ liệu sắp đc truyền đến. Bit bắt buộc.

Data: dữ liệu cần truyền. Bit có trọng số nhỏ nhất LSB được truyền trước sau đó đến bit MSB.

Parity bit: kiểm tra dữ liệu truyền có đúng không.

Stop bit: là 1 hoặc các bit báo cho thiết bị rằng các bit đã được gửi xong. Thiết bị nhận sẽ tiến hành kiểm tra khung truyền nhằm đảm bảo tính đúng đắn của dữ liệu. Bit bắt buộc.

### Bộ chuyển đổi tương tự - số

#### Giới thiệu

Các tín hiệu trong tự nhiên thường gặp như ánh sáng, nhiệt độ, âm thanh,… đều tồn tại dưới dạng tín hiệu tương tự (analog), tức là tín hiệu liên tục và có các khoảng chia vô hạn. Mà trong vi điều khiển chỉ có khái niệm số, cấu trúc từ nhân đến bộ nhớ hoạt động dựa trên các Transistor chỉ có 2 mức logic 0 và 1. Vậy làm thế nào để vi điều khiển hiểu và đọc được tín hiệu này? Đó chính là ứng dụng của bộ chuyển đổi tương tự - số (Analog to Digital Converter - ADC).

Trong bộ ADC, ta cần quan tâm đến 2 yếu tố:

* Độ phân giải (resolution): số bit cần thiết để chứa giá trị chuyển đổi ở ngõ ra. Ví dụ với vi điều khiển PIC, ta dùng 10bit để chứa giá trị, tương đương với 1024 giá trị từ 0-1023.
* Thời gian lấy mẫu (sampling time): thời gian giữa 2 lần số hóa của bộ chuyển đổi.

#### Ứng dụng ADC đọc nhiệt độ từ LM35 thông PIC16F887

Vi điều khiển PIC16F887 có bộ chuyển đổi ADC 10bit đa hợp 14 kênh. Các kênh AN0 đến AN13 là 14 ngõ vào của 14 kênh ADC đưa đến mạch đa hợp.

Bộ ADC 10 bit thì ta có độ phân giải là:

Resolution = =

Mà cảm biến LM35 có độ phân giải là 10mV/°C nên giá trị điện áp ra từ cảm biến:

Vout = 10\*temp (mV)

Giá trị này chính là điện áp tham chiếu của ADC. Như vậy, giá trị nhiệt độ sẽ là:

temp = = = =

## Quy trình làm và thiết kế mạch thủ công

Vẽ sơ đồ nguyên lý mạch (schematic) trên phần mềm Altium Designer

Giao diện thiết kế, quản lý và chỉnh sửa thân thiện, dễ dàng biên dịch, quản lý file, quản lý phiên bản cho các tài liệu thiết kế.

Hỗ trợ mạnh mẽ cho việc thiết kế tự động, đi dây tự động theo thuật toán tối ưu, phân tích lắp ráp linh kiện. Hỗ trợ việc tìm các giải pháp thiết kế hoặc chỉnh sửa mạch, linh kiện, netlist có sẵn từ trước theo các tham số mới.

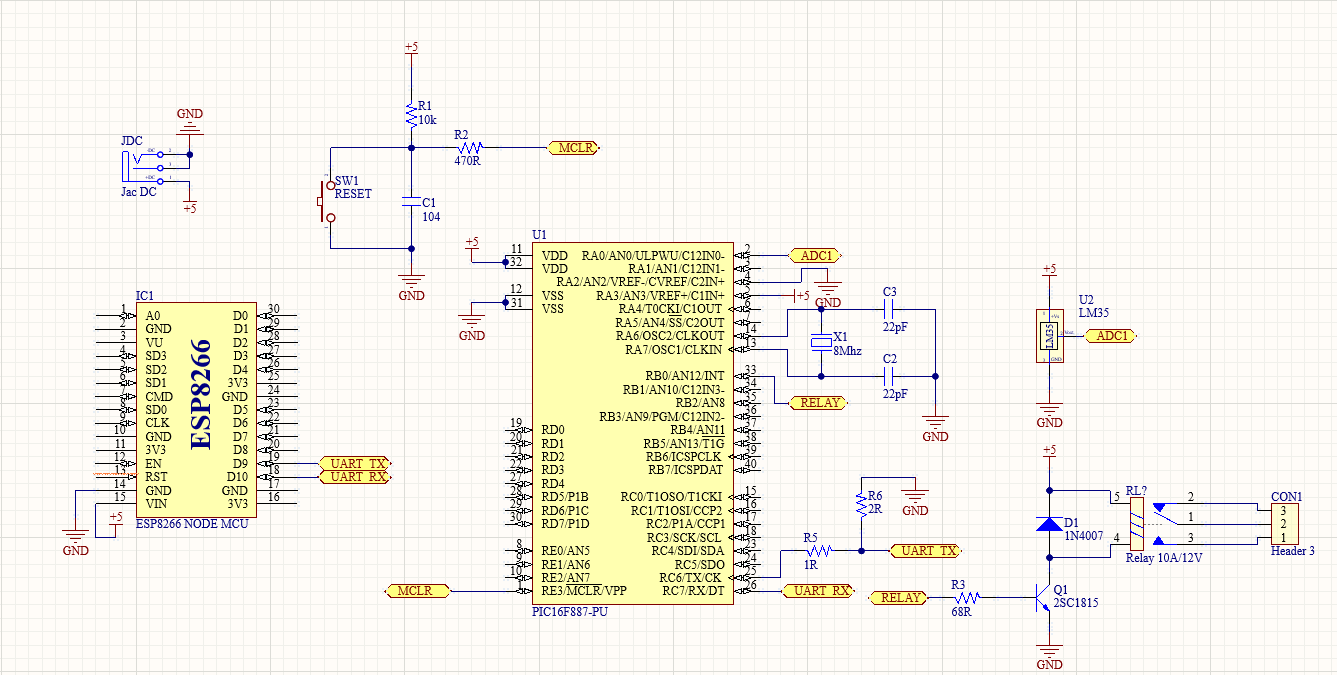
Mở, xem và in các file thiết kế mạch dễ dàng với đầy đủ các thông tin linh kiện, netlist, dữ liệu bản vẽ, kích thước, số lượng…

Hệ thống các thư viện linh kiện phong phú, chi tiết và hoàn chỉnh bao gồm tất cả các linh kiện nhúng, số, tương tự…

Đặt và sửa đối tượng trên các lớp cơ khí, định nghĩa các luật thiết kế, tùy chỉnh các lớp mạch in, chuyển từ schematic sang PCB, đặt vị trí linh kiện trên PCB.

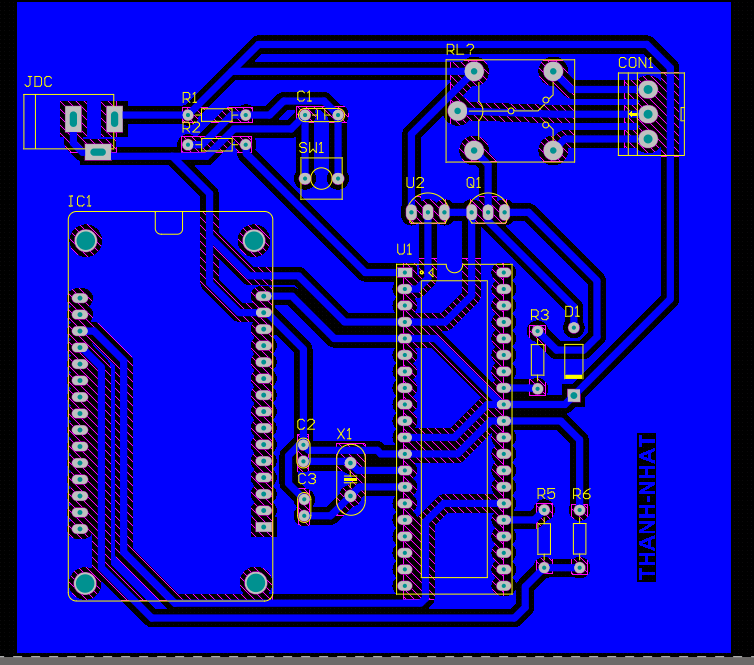
Mô phỏng mạch PCB 3D, đem lại hình ảnh mạch điện trung thực trong không gian 3 chiều, hỗ trợ MCAD-ECAD, liên kết trực tiếp với mô hình STEP, kiểm tra khoảng cách cách điện, cấu hình cho cả 2D và 3D.

Hỗ trợ thiết kế PCB sang FPGA và ngược lại.



Hình .. Mạch nguyên lý

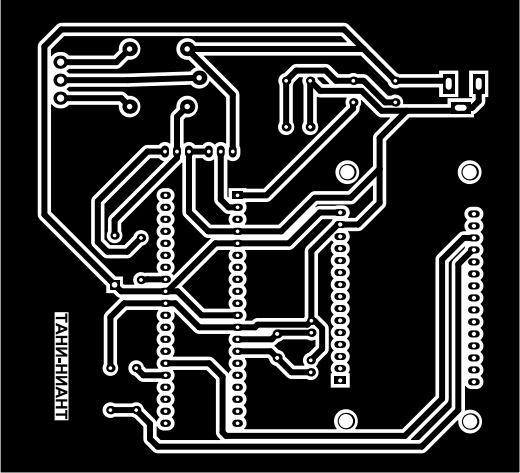
Chuyển từ sơ đồ nguyên lý sang sơ đồ mạch in:



Hình .. Mạch PCB

Sau khi thiết kế xong mạch in chúng xuất mạch in ra định dạng file PDF rồi tiến hành in mạch ra giấy với tỷ lệ 1:1 hoặc 100%.

Đây là mạch in sau khi xuất ra định dạng PDF:



Hình .. Mạch in trên PDF

Thi công mạch in

**Bước 1: Chuẩn bị dụng cụ**

**Bước 2: Cắt Board đồng theo kích thước của mạch in**

**Bước 3: Ủi mạch**

**Bước 4: Ngâm mạch**

**Bước 5: Khoan lỗ**

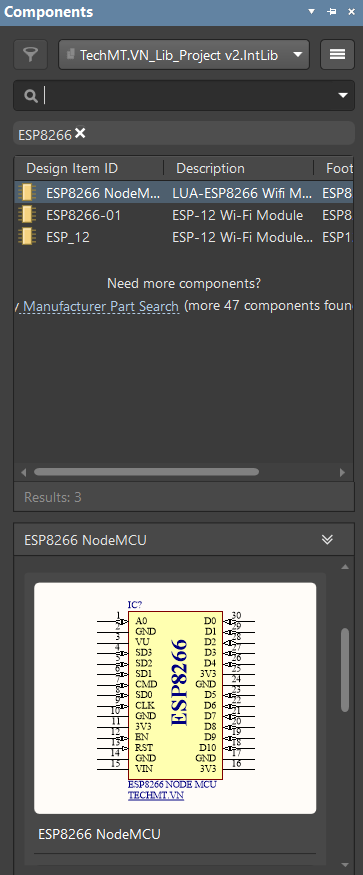
**Bước 6: Hàn mạch**

**Bước 7: Kiểm tra và chạy thử mạch**

# THIẾT KẾ HỆ THỐNG

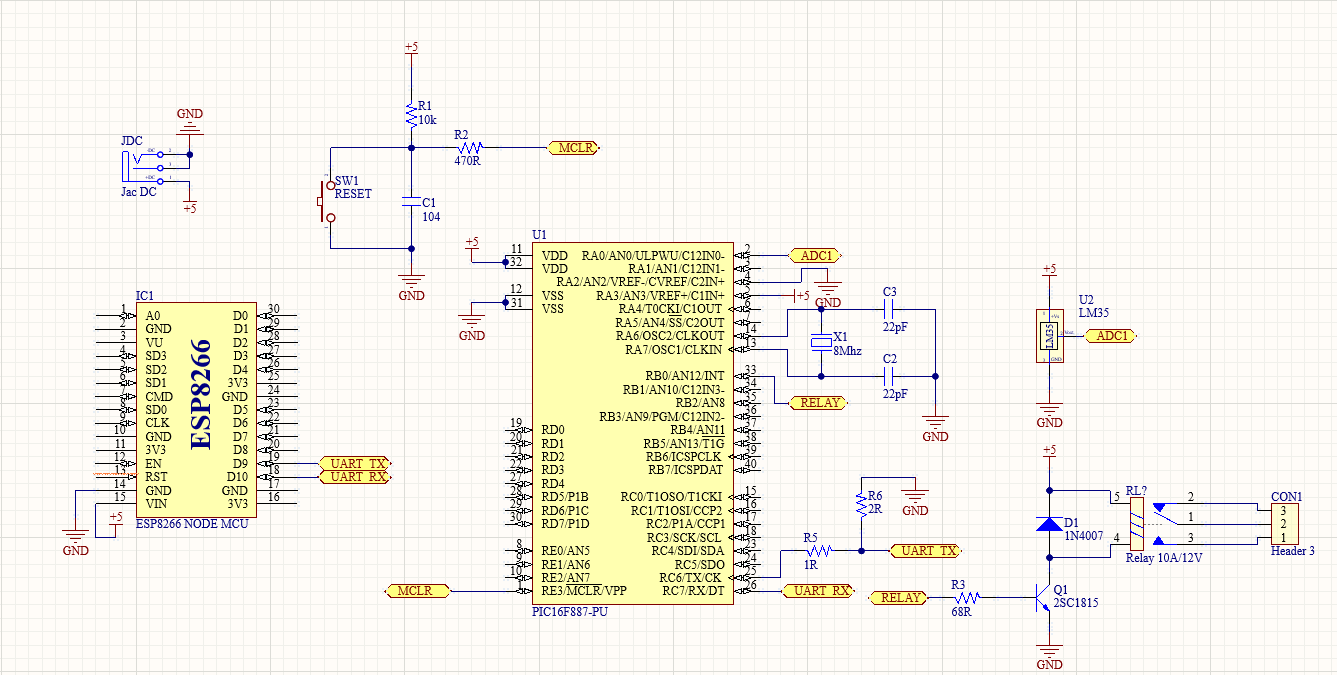
## Vẽ Sơ đồ mạch trên phần mềm Altium

Chọn các linh kiện cần thiết ở trên trong mục components.



Hình .. Chọn linh kiện trên Altium

Sắp xếp các linh kiện và nối dây, chúng ta đặt linh kiện vào nhưng vị trí thuận tiện và nối các chân lại với nhau.



Hình .. Sơ đồ mạch nguyên lý

## Xây dựng sản phẩm trên board đồng

**Bước 1**: **Chuẩn bị dụng cụ**

1. Thiếc hàn (chì hàn)
2. Mỏ hàn chì
3. Giấy nhám (giấy ráp)
4. Bút lông dầu
5. Mạch in
6. Kềm cắt chân linh kiện
7. Dao cắt mạch
8. Board Đồng (phíp đồng).
9. Bàn ủi
10. Máy khoan mạch
11. Thước kẻ
12. Thuốc ngâm mạch (bột sắt FeCl3)

**Bước 2**: **Cắt Board đồng theo kích thước của mạch in**

Sau đó bạn các bạn dùng giấy nhám chà thật sạch board đồng rồi dùng cồn lau cho board đồng thật sạch.

**Bước 3: Ủi mạch**

Đây là công đoạn cần sức lực và cũng khá khó khăn đấy, đòi hỏi bạn phải có chút kinh nghiệm thì mới thành công được.

* Bạn áp phần có hình mạch in trên giấy vào mặt có đồng của board đồng, căn chỉnh sao cho khớp, sau đó tận dụng phần thừa của giấy in mạch để dán cố định giấy với board đồng (có thể dùng băng keo giấy để cố định)
* Sau đó bạn ủi sơ qua toàn bộ bề mặt cần ủi để cho giấy định hình tiếp xúc hoàn toàn với board đồng.
* Tiếp theo bạn dùng mũi và cạnh của bàn ủi tập trung ủi các góc và cạnh của board mạch cần ủi vì góc và cạnh của board mạch là nơi khó ủi nhất và nhiệt khó tập trung ở những nơi đó khó nhất, vùng trung tâm thì bạn ủi 2-3 lần là dính hết rồi.
* Thời gian ủi tùy thuộc vào từng loại board to nhỏ và kinh nghiệm của mỗi người.

Bạn ủi mạch khoảng 5-10 phút cho đến khi nào lớp mực in trên giấy bám hết vào mặt đồng của board đồng.

Sau đó đợi mạch nguội ta dùng tay nhẹ nhang bóc 1 ít ra xem nếu mạch mực chưa bám xuống hết ta cố định lại và ủi lại khoảng 3 phút.

Sau khi ủi xong mạch ta dùng bút lông dầu tô lại những chỗ mực in chưa ăn xuống hết.

**Bước 4: Ngâm mạch**

Pha bột sắt ( FeCl3) với nước theo tỉ lệ 100g bột sắt pha với 250ml nước.

Sau đó ngâm mạch trong dung dịch FeCl3 khoảng 10 phút cho đến khi lớp đồng được ăn mòn hết thì lấy mạch ra rửa với nước cho sạch.

Một số lưu ý:

* Bột sắt pha với nước với tỷ lệ vừa phải đừng loãng quá điều đó sẽ làm mạch bị ăn mòn lâu hoặc pha quá nhiều sẽ làm mạch in tróc hết.
* Vì thuốc rửa mạch có hại cho da nên bạn hãy cẩn thận hạn chế bị dính vào người, tốt nhất là nên đeo găng tay và kính bảo hộ khi thực hiện. Nếu không may bị dính vào tay thì hãy nhanh chóng rửa nhanh bằng nước sạch.

Mạch in sau khi ngâm và rửa sạch bằng nước.

**Bước 5: Khoan lỗ**

Dùng giấy nhám làm sạch hết lớp mực trên mạch. Sau đó tiến hành khoan mạch.

Một số lưu ý khi khoan mạch:

* Bạn nên đặt board cần khoan trên một cuốn vở, khi bạn lỡ khoan sâu quá thì mũi khoan xuyên vào giấy không bị dính mũi khoan, đồng thời bảo vệ mặt bàn.
* Khi khoan cố gắng đặt mũi khoan vuông góc với board mạch.
* Đặt mũi khoan vào những lỗ chấm nhỏ có tác dụng cố định đầu khoan.
* Bạn có thể tập khoan trên những tấm board hư để làm quen với việc khoan mạch, sau khi đã thành thạo thì khoan trên mạch cần làm.
* Chọn mũi khoan phù hợp với linh kiện.

**Bước 6**: **Hàn mạch**

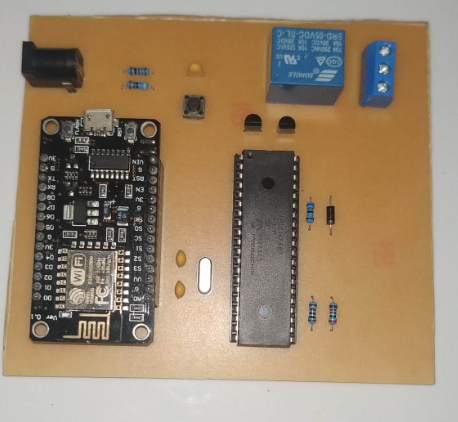
Hàn từ linh kiện thấp nhất đến cao nhất. Mạch này có 1 jump, ta hàn jump đầu tiên. Sau đó hàn tiếp điện trở, đế IC, tụ điện, điện trở, nút nhấn…

Sau khi hàn xong, các bạn nên quét một lớp nhựa thông pha xăng thơm lên mạch để bảo vệ mạch, nếu không thì chỉ sau 1 vài ngày đường đồng bị oxy hoá đen thui, dẫn điện kém và nhìn board rất xấu.

**Bước 7**: **Kiểm tra và chạy thử mạch**

* Kiểm tra kĩ các đường mạch các chânIC xem có bị dính chân hay chập mạch hay không.
* Cấp nguồn và chạy thử mạch

Và đây sản phẩm của mình đã làm thành công trong nhiều ngày nổ lực.



Hình .. Sản phẩm hoàn chỉnh

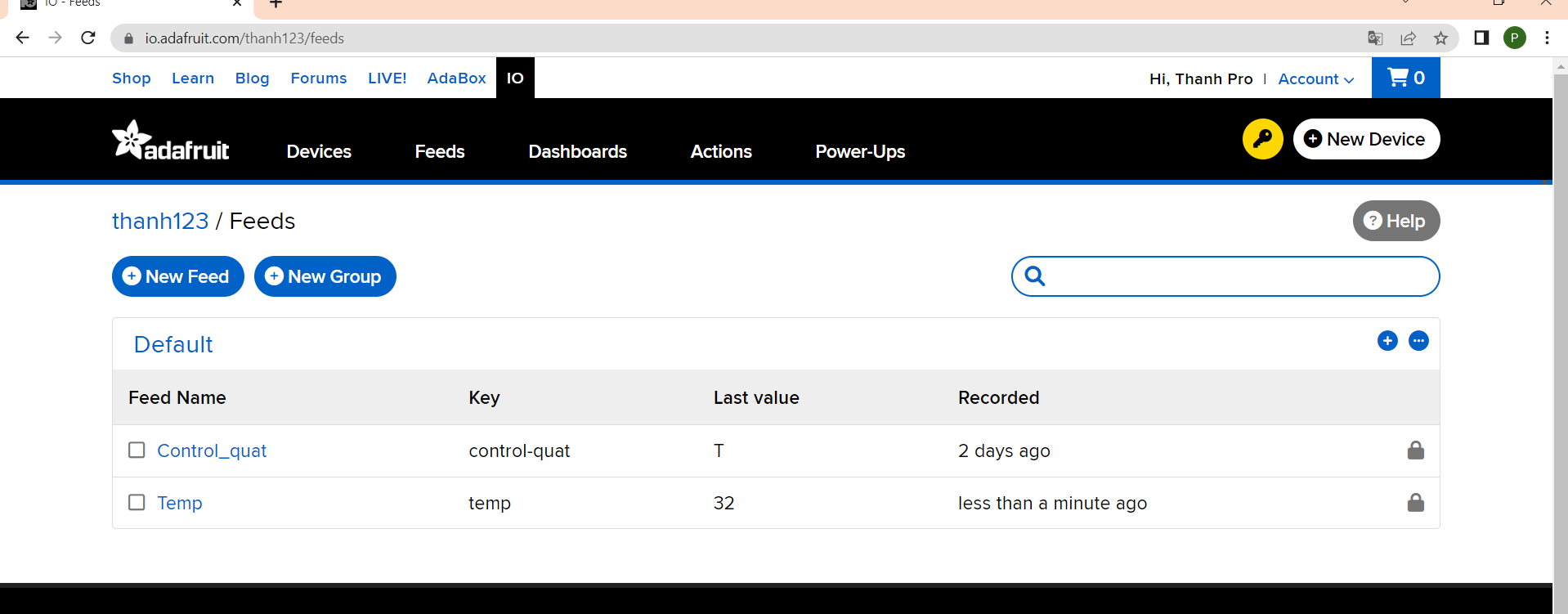
## Source code cho hệ thống

Sử dụng ngôn ngữ lập trình C để viết chương trình cho PIC16F887, sử dụng MPLAP IDE để viết chương trình. Sử dụng ngôn ngữ C++ để viết chương trình cho module ESP8266 với Arduino IDE.

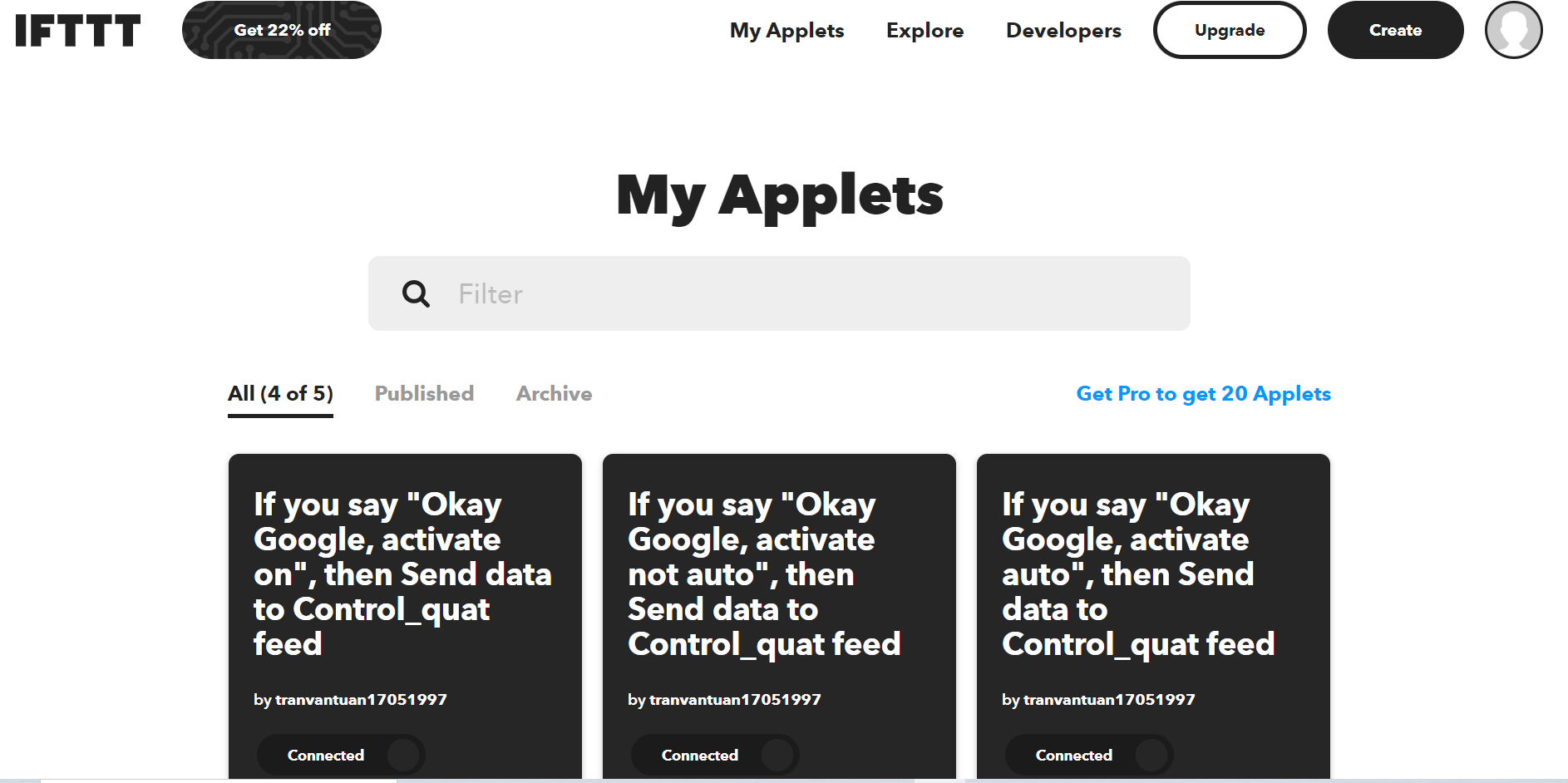
Phần chương trình được trình bày ở phần Phụ lục.

## Thiết lập phần mềm

Tạo tài khoản trên <https://ifttt.com/> sau đó thiết lập kết nối với GoogleHome,đồng thời tạo một tài khoản trên <https://www.adafruit.com/>. Trên Adafruit tiến hành tạo các topic để nhận/truyền đi các lệnh điều khiển từ GoogleHome và lưu trữ dữ liệu về nhiệt độ. Sau đó thiết lập các câu lệnh trên IFTTT để điều truyền lệnh điều khiển tới Adafruit.



Hình .. Topic trên Adafruit



Hình .. Câu lệnh điều khiển trên IFTTT

## Kết chương 3

Trong chương này ta đã cho chúng ta thấy quy trình chuẩn bị và làm một bo mạch thủ công từ thiết kế cho đến hoàn thành mạch qua đó chúng ta học được các bước làm mạch thủ công.

KẾT LUẬN

* **Kết quả đạt được**

**+ Về hệ thống**

Hệ thống đã có thể điều khiển được thiết bị bằng giọng nói cũng như bằng công tắt vật lý và ứng dụng trên điện thoại thông minh. Tuy nhiên còn chưa ổn định.

**+ Về kiến thức**

Sản phẩm còn giúp các thành viên trong nhóm hiểu thêm về đặc điểm cấu tạo, chức năng, nguyên lý hoạt động của các loại linh kiện điện tử, làm quen với các loại cảm biến, rèn luyện kĩ năng đội nhóm, kĩ năng thuyết trình, kĩ năng tra cứu thông tin, kĩ năng thiết kế và xây dựng một hệ thống IOT.

TÀI LIỆU THAM KHẢO

* **Ebook**

1. Home Automation Using ESP8266 (4th Edition)
2. Hands-On Internet of Things with Blynk by [**Pradeeka Seneviratne**](https://www.kobo.com/us/en/author/pradeeka-seneviratne)

* **Video**

1. Water tank level monitoring system with Nodemcu and Blynk application - [ESP8266 Project] (<https://www.youtube.com/watch?v=Iuxl0k4lnqA&t=264s>)
2. How to use Blynk app with ESP8266 | Step by step instructions (https://www.youtube. com/watch?v=oernrRksrR4)

* **Website**

1. Arduino – Home ([https: //www. arduino. cc/](https://www.arduino.cc/))

PHỤ LỤC

**Phụ lục 1: Chương trình cho PIC16F887**

/\*

\* File: newmain.c

\* Author: tranv

\*

\* Created on July 7, 2022, 2:11 PM

\*/

#pragma config FOSC = HS

#pragma config WDTE = OFF

#pragma config PWRTE = OFF

#pragma config BOREN = ON

#pragma config LVP = OFF

#pragma config CPD = OFF

#pragma config WRT = OFF

#pragma config CP = OFF

#include <xc.h>

#define \_XTAL\_FREQ 8000000

#include <stdbool.h>

#include <stdio.h>

#include <pic16f887.h>

bool on\_off = true;

bool check\_auto = true;

ADC\_Init(){

ADCON0 =0x41;

ADFM = 1;

VCFG0 = 0;

VCFG1 = 0;

}

void UART\_init(){

TRISC7 = 1; //RX->input

TRISC6 = 0; //TX->output

SPBRG = 51; // F/(16\*baudrate)-1

BRG16 = 0;

BRGH = 1;// baudrate toc do cao

SYNC = 0; // che do khong dong bo

SPEN = 1; // truyen du lieu noi tiep

CREN = 1;

TXEN = 1; // cho phep truyen du lieu

RX9 = 0;

TX9 = 0; // truyen 8bit

}

float ADC\_read(){

CHS3 = 0;

CHS2 = 0;

CHS1 = 0;

CHS0 = 0;

ADCON0bits.GO\_DONE = 1; // bat dau chuyen doi ADC

while (ADCON0bits.GO\_DONE); //cho chuyen doi ket thuc

return ((ADRESH << 8) + ADRESL); // return gia tri

}

void UART\_sendChar(char bt)

{

while (TXIF==0); //lam tre he thong khi bo dem TX dang trong

TXREG = bt; // ghi gia tri truyen vao thanh ghi TXREG

//

}

void UART\_sendString(char\* str)

{

while (\*str)

{

UART\_sendChar(\*str++);

}

}

void Auto(){

float temp;

int temp2;

char temp3[10],\*p;

float value = ADC\_read();

temp = (value\*5000)/(1023\*10);

if(temp>30) RB0 = 1;

else if(temp<30) RB0 = 0;

temp2 = (int)temp;

sprintf(temp3, "%d", temp2);

p = temp3;

UART\_sendString(p);

}

int interup = 0;

void \_\_interrupt() ISR(){

if(RCIF == 1){

if(OERR==1){//kiem tra loi tran

CREN = 0;

CREN = 1;

}

if(RCREG == 'T'){

T0IE = 1;

check\_auto = true;

}

else if(RCREG == 'F'){

T0IE = 0;

check\_auto = false;

}

else if(check\_auto == false){

if(RCREG == '1') on\_off = true;

else if(RCREG == '0') on\_off = false;

}

RCIF=0;

}

if(T0IF == 1){

TMR0 = 5;

T0IF = 0;

interup++;

if(interup==5000){

Auto();

interup = 0;

}

}

}

void main(void) {

\_\_delay\_ms(3000);

GIE = 1; //Ngat toan cuc

PEIE = 1; //Ngat ngoai vi

RCIE = 1; // Bat ngat nhan UART

RCIF = 0; // trang thai thanh ghi rcreg

PSA = 0; // gan Prescaler cho Timer0

PS2 = 0;

PS1 = 1;

PS0 = 0; // chon bo chia 8 PS2-PS1

T0CS = 0;

T0IF = 0;

T0IE = 1;

TRISB = 0;

TRISD = 0;

PORTD = 0xff;

RB0=1;

UART\_init();

ADC\_Init();

while(1){

if(check\_auto == true){

// float value = ADC\_read();

// temp = (value\*5000)/(1023\*10);

//// if(temp>30) RB0 = 1;

//// else if(temp<30) RB0 = 0;

// temp2 = (int)temp;

// sprintf(temp3, "%d", temp2);

// p = temp3;

// UART\_sendString(p);

}

else if(check\_auto == false){

if(on\_off == true) RB0 = 1;

else if(on\_off == false)RB0 = 0;

}

}

return;

}

## Phụ lục 2: Chương trình cho module ESP8266

#include <ESP8266WiFi.h>

#include "Adafruit\_MQTT.h"

#include "Adafruit\_MQTT\_Client.h"

#include<SoftwareSerial.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* WiFi Access Point \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define WLAN\_SSID       "Thanhcoder"

#define WLAN\_PASS       "88888888"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Adafruit.io Setup \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define AIO\_SERVER      "io.adafruit.com"

#define AIO\_SERVERPORT  1883                   // use 8883 for SSL

#define AIO\_USERNAME    "thanh123"

#define AIO\_KEY         "aio\_yflm55hm9sog25UKVKVTqtCIXAKj"

/\*\*\*\*\*\*\*\*\*\*\*\* Global State (you don't need to change this!) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Create an ESP8266 WiFiClient class to connect to the MQTT server.

WiFiClient client;

// or... use WiFiClientSecure for SSL

//WiFiClientSecure client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.

Adafruit\_MQTT\_Client mqtt(&client, AIO\_SERVER, AIO\_SERVERPORT, AIO\_USERNAME, AIO\_KEY);

// Setup a feed called 'photocell' for publishing.

// Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>

Adafruit\_MQTT\_Publish publish\_Temp = Adafruit\_MQTT\_Publish(&mqtt, AIO\_USERNAME "/feeds/Temp");

// Setup a feed called 'onoff' for subscribing to changes.

Adafruit\_MQTT\_Subscribe onoffbutton = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/Control\_quat");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Sketch Code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Bug workaround for Arduino 1.6.6, it seems to need a function declaration

// for some reason (only affects ESP8266, likely an arduino-builder bug).

void MQTT\_connect();

void setup() {

//   pinMode(D11, OUTPUT);

// digitalWrite(D11, LOW);   // turn the LED on (HIGH is the voltage level)

  Serial.begin(9600);

  delay(10);

  Serial.println(F("Adafruit MQTT demo"));

  // Connect to WiFi access point.

  Serial.println(); Serial.println();

  Serial.print("Connecting to ");

  Serial.println(WLAN\_SSID);

  WiFi.begin(WLAN\_SSID, WLAN\_PASS);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println();

  Serial.println("WiFi connected");

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

  // Setup MQTT subscription for onoff feed.

  mqtt.subscribe(&onoffbutton);

}

uint32\_t x=0;

char value[10], value1;

void loop() {

  // Ensure the connection to the MQTT server is alive (this will make the first

  // connection and automatically reconnect when disconnected).  See the MQTT\_connect

  // function definition further below.

  MQTT\_connect();

  // this is our 'wait for incoming subscription packets' busy subloop

  // try to spend your time here

  Adafruit\_MQTT\_Subscribe \*subscription;

  while ((subscription = mqtt.readSubscription(5000))) {

    if (subscription == &onoffbutton)

    {

      Serial.write((char \*)onoffbutton.lastread);

      // Serial.print(F("Got: "));

      // Serial.println((char \*)onoffbutton.lastread);

     if (!strcmp((char\*) onoffbutton.lastread, "1"))

      {

        //Active low logic

        digitalWrite(16, HIGH);

      }

    if (!strcmp((char\*) onoffbutton.lastread, "0"))

      {

        //Active low logic

        digitalWrite(10, LOW);

      }

    }

  }

  int i = 0;

  while(Serial.available()>0){

    value1 = Serial.read();

    value[i] = value1;

    Serial.println(value1);

    i++;

  }

  if(i>0){

    char \*temp;

    temp = value;

    publish\_Temp.publish(temp);

  }

         //Publish to Adafruit

  // ping the server to keep the mqtt connection alive

  // NOT required if you are publishing once every KEEPALIVE seconds

  /\*

  if(! mqtt.ping()) {

    mqtt.disconnect();

  }

  \*/

}

// Function to connect and reconnect as necessary to the MQTT server.

// Should be called in the loop function and it will take care if connecting.

void MQTT\_connect() {

  int8\_t ret;

  // Stop if already connected.

  if (mqtt.connected()) {

    return;

  }

  Serial.print("Connecting to MQTT... ");

  uint8\_t retries = 3;

  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected

       Serial.println(mqtt.connectErrorString(ret));

       Serial.println("Retrying MQTT connection in 5 seconds...");

       mqtt.disconnect();

       delay(5000);  // wait 5 seconds

       retries--;

       if (retries == 0) {

         // basically die and wait for WDT to reset me

         while (1);

       }

  }

  Serial.println("MQTT Connected!");

}