TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN

BỘ MÔN TỰ ĐỘNG HOÁ CÔNG NGHIỆP

====o0o====

ĐỒ ÁN TỐT NGHIỆP

Hà nội, 1-2019

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN

BỘ MÔN TỰ ĐỘNG HOÁ CÔNG NGHIỆP

====o0o====

ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI:

Nghiên cứu thiết kế hệ thống điều khiển giống cây trồng trong nông nghiệp.

Giáo viên hướng dẫn : TS. Nguyễn Huy Phương

Sinh viên thực hiện : Dương Đình Quân

Lớp : KT ĐK&TĐH 01 - K58

MSSV : 20133124

Giáo viên duyệt :

Hà nội, 1-2019

LỜI CAM ĐOAN

Em xin cam đoan bản đồ án tốt nghiệp: **Nghiên cứu thiết kế hệ thống điều khiển giống cây trồng trong nông nghiệp** do em tự thiết kế dưới sự hướng dẫn của thầy giáo TS. Nguyễn Huy Phương. Các số liệu và kết quả là hoàn toàn đúng với thực tế.

Để hoàn thành đồ án này em chỉ sử dụng những tài liệu được ghi trong danh mục tài liệu tham khảo và không sao chép hay sử dụng bất kỳ tài liệu nào khác. Nếu phát hiện có sự sao chép em xin chịu hoàn toàn trách nhiệm.

Hà Nội, ngày 04 tháng 12 năm 2018

Sinh viên thực hiện

Dương Đình Quân

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG SỐ LIỆU

DANH MỤC TỪ VIẾT TẮT

IoT	Internet of Things	Mạng lưới vạn vật kết nối Internet
MQTT	Message Queue Telemetry Transport	Truyền gói tin từ xa theo hàng đợi
ROM	Read Only Memory	Bộ nhớ chỉ đọc
PWM	Pulse Width Modulation	Điều chế độ rộng xung
ADC	Analog to Digital Converter	Chuyển đổi tương tự sang số
DMA	Direct Memory Access	Truy cập trực tiếp bộ nhớ
OTA	Over The Air	Cập nhật phần mềm từ xa

LỜI NÓI ĐẦU

Một trong những xu hướng công nghệ nổi bật trong năm 2018 là Internet of Things (viết tắt là IoT -Internet kết nối vạn vật). Theo ước tính của CISCO, đến năm 2020, toàn thế giới sẽ có 50 tỷ thiết bị được kết nối không dây vào mạng lưới IoT. Internet of Things sẽ là xu hướng công nghệ của tương lai. Nắm bắt được nhứng thông tin trên, em đã lên ý tưởng cho đồ án tốt nghiệp về ứng dụng xu hướng công nghệ này.

Trong đồ án tốt nghiệp em đã được giao đề tài vi điều khiển : "**Nghiên cứu thiết kế hệ thống điều khiển giống cây trồng trong nông nghiệp**". Nội dung đồ án gồm các phần cơ bản như sau:

- *Tổng quan hệ thống điều khiển giống cây trồng trong nông nghiệp.*
- *Cấu hình hệ thống điều khiển và giám sát.*
- *Thiết kế phần cứng.*
- *Thiết kế phần mềm.*
- *Kết quả thực nghiệm.*

Sau 3 tháng được sự hướng dẫn tận tình của thầy giáo TS. Nguyễn Huy Phương và các thầy cô trong bộ môn Tự động hoá, đồ án của em đã hoàn thiện. Do thời gian làm đồ án ngắn và khả năng còn hạn chế, chắc chắn đồ án của em còn nhiều thiếu sót. Em rất mong nhận được sự đóng góp của thầy cô và các bạn.

Hà Nội, ngày 01 tháng 01 năm 2019

Sinh viên thực hiện

Dương Đình Quân

Chương 1

GIỚI THIỆU HỆ THỐNG ĐIỀU KHIỂN GIỐNG CÂY TRỒNG TRONG NÔNG NGHIỆP

1.1. Một số thông tin về hệ thống điều khiển giống cây trồng trong nông nghiệp

Hiện nay tại Việt Nam chúng ta đã nghe nói rất nhiều về nông nghiệp thông minh, nhìn dạo quanh trên thị trường Việt Nam cũng có rất nhiều các startup bắt đầu khởi nghiệp với nông nghiệp thông minh, tập trung chủ yếu các chức năng hiện nay là trồng cây thủy canh, giám sát tưới tiêu, tự điều khiển theo thời gian,… có thể điểm qua như Hachi, Greenbot, Mimosatek,..

Em chưa có điều kiện để sử dụng cũng như trải nghiệm các sản phẩm của các startup trên, nhưng với một số tham khảo cóp nhặt từ google em sẽ thiết kế một hệ thống nông nghiệp thông minh đơn giản để điều khiển giống cây trồng trong nông nghiệp có thể ứng dụng được vào cuộc sống.

1.2. Lựa chọn giống cây trồng và quá trình sinh trưởng của giống cây trồng trong hệ thống nông nghiệp.

Trong hệ thống của mình em chọn rau xà lách để thử nghiệm vì là loại rau ngắn ngày, dễ trồng, dễ chăm sóc và đã quen thuộc và người dân phía Bắc. Xà lách là loại cây thân thảo và cũng là tên gọi chung cho một loại rau ăn sống. Rau xà lách có khá nhiều loại như: xà lách mỡ, xà lách xoăn là lớn, xà lách lô tô xanh, xà lách lô tô tím,..

Quá trình sinh trưởng của rau xà lách:

1. Gieo hạt.
    - Hạt ngâm trong nước ấm nhiệt độ 50 - 52^0C trong 1-2 giờ rồi đem gieo hạt.
    - Sau khi gieo xong, phủ một lớp giá thể mỏng lên bề mặt hạt để giữ ẩm .
    - Sau gieo 18 - 20 ngày khi cây được 4 - 6 lá tiến hành đánh dặm tỉa định mật độ cho cây sinh trưởng phát triển.

Lưu ý về mật độ trồng:

- Mật độ trồng cây/ khay: 6 - 8 cây.

- Lượng hạt gieo trồng 15-20 hạt/ khay.
- Nếu tỉ lệ nảy mầm của hạt kém có thể gieo dư ra để bù trừ với lượng 20-30 hạt/ khay trồng.

2. Chăm sóc.

- Trong giai đoạn ươm cây chú ý giữ ẩm cho cây phát triển, chú ý đề phòng sâu ăn lá bằng cách quan sát và bắt bằng tay vào buổi tối và sáng sớm.
- Khi cây có 4 - 6 lá thật (18 - 20 ngày sau gieo), cây khỏe mạnh tiến hành tỉa ra trồng.
- Bỏ cây vào hốc, lấp đất dùng tay ấn nhẹ đất, tưới giữ ẩm 2 lần/ ngày trong tuần đầu tiên để cây bén rễ hồi xanh.
- Chú ý khi trồng tỉa không nên trồng sâu quá hoặc nông quá tránh ảnh hưởng đến sinh trưởng của cây.
- 2 - 3 ngày tưới 1 lần tùy thuộc vào độ ẩm của đất.

3. Thu hoạch.

- Khi cây phát triển tối đa, sau trồng từ 35 - 40 ngày, thì có thể thu hoạch

Lưu ý: Sau trồng 1 - 2 lứa rau, cần bổ sung thêm dinh dưỡng cho cây bằng phân hữu cơ vi sinh với lượng 350-400 g/m2 hoặc 80-100 g/ khay kích thước 40x 60cm. Đất trồng tiến hành xới xáo lại và phơi khô trong 2-3 nắng để diệt nấm bệnh sâu hại.

1.3 Yêu cầu thiết kế chung điều khiển giống cây trồng và những yêu cầu riêng của giống cây được chọn trong hệ thống.

1.3.1 Yêu cầu thiết kế chung điều khiển giống cây trồng

Từ mục đích khi xây dựng đề tài là tự động hóa quá trình trồng cây nên ta thiết kế hệ thống sẽ có các chức năng như : giám sát, tính toán, điều chỉnh môi trường của hệ thống để đối tượng cây trồng có điều kiện phát triển tốt nhất.

Những yếu tố ảnh hưởng đến sự phát triển của cây trồng (xét trong phạm vi là cây rau):

1. Ánh sáng

Ánh sáng ảnh hưởng đến sự phát triển của cây rau, là yếu tố quan trọng nhất cho sự sinh trưởng và phát triển của cây. Ánh sáng mặt trới là nguồn năng lượng duy nhất, vô tận để cây xanh quang hợp, biến các chất vô cơ, nước và khí cacbonic thành hợp chất hữu cơ tích lũy trong lá, hoa, quả, củ… phục vụ cho nhu cầu sống của con người và động vật.

Cường độ ánh sáng cũng ảnh hưởng lớn đến sự phát triển của cây trồng. Dựa vào cường độ ánh sáng , người ta phân rau ra các nhóm: Nhóm yêu cầu cường độ ánh sáng mạnh là bí ngô, cà, cà chua, ốt, đậu. Nhóm yêu cầu cường độ ánh sáng trung bình như bắp cải, cải trắng, cải củ,hành, tỏi. Nhóm yêu cầu cường độ sáng yếu là: xà lách, rau diếp.

2. Nhiệt độ

Nhiệt độ là yếu tố quan trọng trong sinh trưởng và sự phát triển của cây rau. Nhiệt độ chính là yếu tố tạo nên các vùng khí hậu khác nhau trên trái đất, và từ đó sẽ có những loài rau khác nhau phù hợp với khí hậu của từng vùng. Nhiệt độ còn ảnh hưởng đến sự phát triển, sự nở hoa, chất lượng sản phẩm, khả năng bảo quản, thời gian ngủ của hạt và sự ảnh hưởng đến sự phát triển của sâu bệnh trên các loại rau

3. Độ ẩm

Độ ẩm trong không khí, trong đất có tác động đến các giai đoạn sinh trưởng của cây như sự nảy mầm của hạt, sự ra hoa, kết hạt, thời gian chín của quả, chất lượng rau, sản lượng, sinh trưởng sinh dưỡng, phát triển sâu bệnh và bảo quản hạt giống.

4. Đất

Đất là nơi bộ rễ phát triển, giữ chặt cây. Rau cần đất tốt, có chế độ dinh dưỡng cao. Bộ rễ của các loài rau nói chung ăn nông nên tính chịu hạn, chịu nóng kém, do đó đất trồng rau phải là chân đất cao, dễ tiêu nước

Độ PH của đất cũng là một yếu tố quan trọng trong sự sinh trưởng của cây. Môi loại cây sẽ phù hợp với một độ PH riêng để có thể phát triển tốt nhất.

5. Chất khoáng

Rau là cây trồng ngắn ngày nhưng sản lượng lớn, nên rau cần lượng chất dinh dưỡng lớn. Các chất dinh dưỡng này rau lấy từ đất không đủ, nên người trồng rau phải bổ sung bằng các loại phân bón. Dù là rau ăn lá, rau ăn củ, rau ăn quả cung cần đầy đủ các chất dinh dưỡng cơ bản là đạm (N), lân (P), kali (K) và một số nguyên tố vi lượng khác.

6. Nước tưới

Thành phần hóa học trong rau chú yếu là nước, chiếm đến 90%, do đó lượng nước cây cần lấy vào trong tự nhiên là rất lớn. Nước còn là môi trường sống của một số loại rau, là nơi chất khoáng hòa tan được rễ hút vào nuôi cây. Nước cũng là môi trường để pha các loại thuốc trừ sâu bệnh. Nên muốn có năng suất rau cao, cần đảm bảo lượng nước đủ theo yêu cầu của từng loại rau.

Từ những yếu tố đã nêu ở trên, vì thời gian có hạn và trong phạm vi đồ án môn học em đã giải quyết được những vấn đề:

- Hệ thống đo được nhiều thông số của môi trường như nhiệt độ, độ ẩm, độ ẩm của đất, cường độ ánh sáng của môi trường .
- Hệ thống hỗ trợ con người được nhiều công việc như tưới nước, bật đèn sưởi,duy trì độ sáng phù hợp với cây trồng.
- Hệ thống có thể điều khiển, giám sát qua mạng internet.
- Hệ thống có thể hoạt động ổn định và chính xác.

1.3.2 Yêu cầu thiết kế riêng của giống cây xà lách.

Yêu cầu về điều kiện môi trường:

- Nhiệt độ thích hợp cho cây sinh trưởng và phát triển từ 15-25^0C.
- Xà lách ưu ánh sáng dịu, không quá gắt, có thể sử dụng ánh sáng nhân tạo từ đèn huỳnh quang, ánh sáng trung bình từ 10-12 giờ/ngày.
- Độ ẩm đất khoảng 70-80%.
- Xà lách không kén đất, thích hợp đất thoát nước tốt.

1.4. Mở rộng về yêu cầu thời gian thực của hệ thống.

Một vấn đề đặt ra là hệ thống sẽ hoạt động như thế nào khi môi trường thay đổi và qua từng quá trình sinh trưởng của xà lách. Khi điều kiện của môi trường thay đổi như xét thời gian trong một ngày, nếu ta thực hiện tưới nước cho cây xà lách vào buổi trưa và tối sẽ không tốt cho sự phát triển của cây, cây có thể chết hoặc dễ mắc sâu bệnh. Vậy nên cần tưới nước bù độ ẩm cho cây vào buổi sáng hoặc chiều để cây có thể phát triển. Một ví dụ khác là xà lách ưa ánh sáng dịu, ánh sáng trung bình từ 10-12 giờ/ngày, nhưng khi vào mùa đông có thể lượng ánh sáng trong một ngày sẽ không đủ cho cây nên hệ thống cần phải tạo bù lượng ánh sáng còn thiếu để cây có môi trường phát triển tốt nhất.

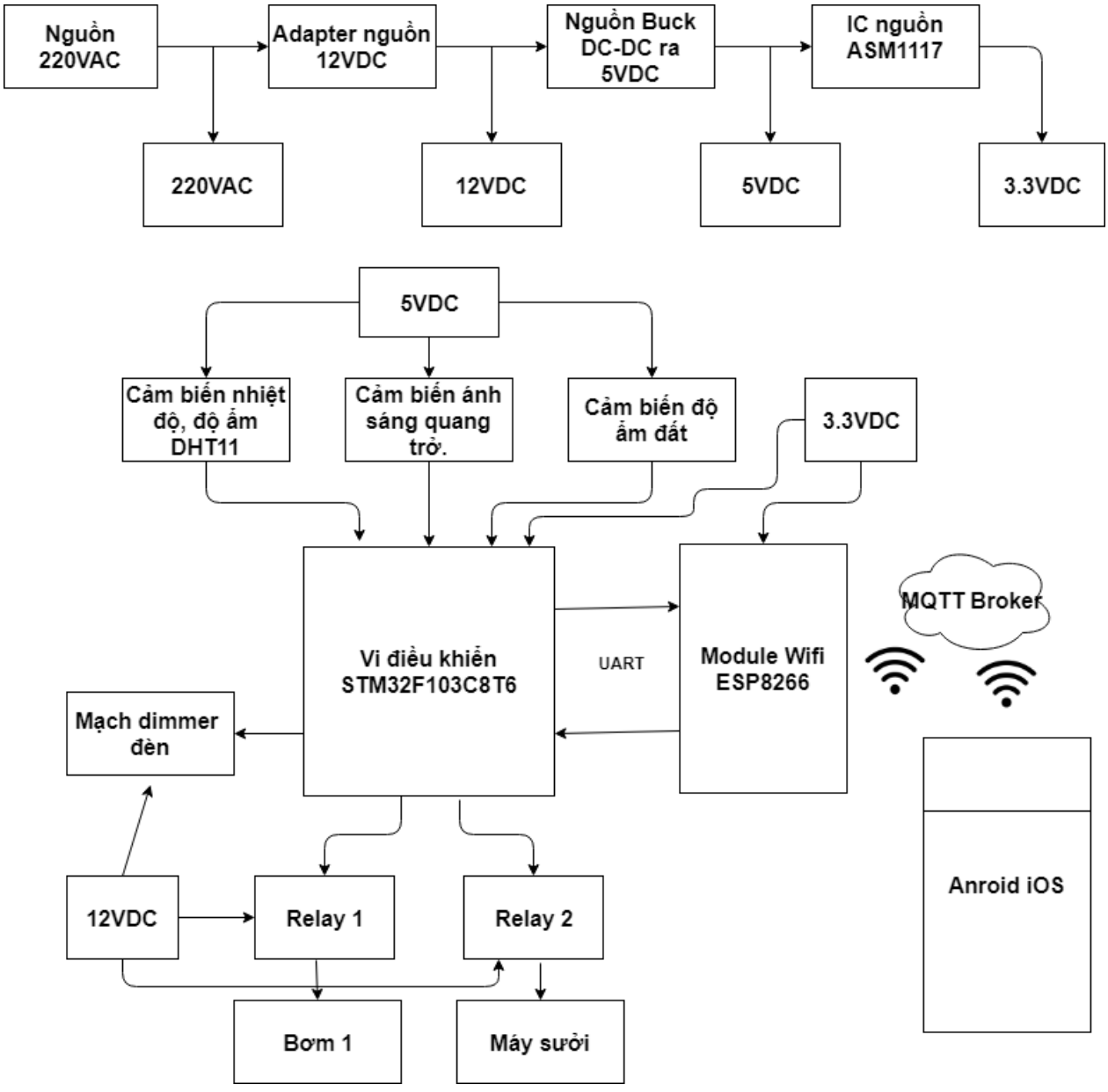
Để đáp ứng được các yếu cầu ở trên thì ta cần tích hợp một hệ thống thời gian thực để hệ thống có thể giám sát, tính toán , điều khiển môi trường cho phù hợp nhất với từng thời kì sinh trưởng của cây.

Chương 2

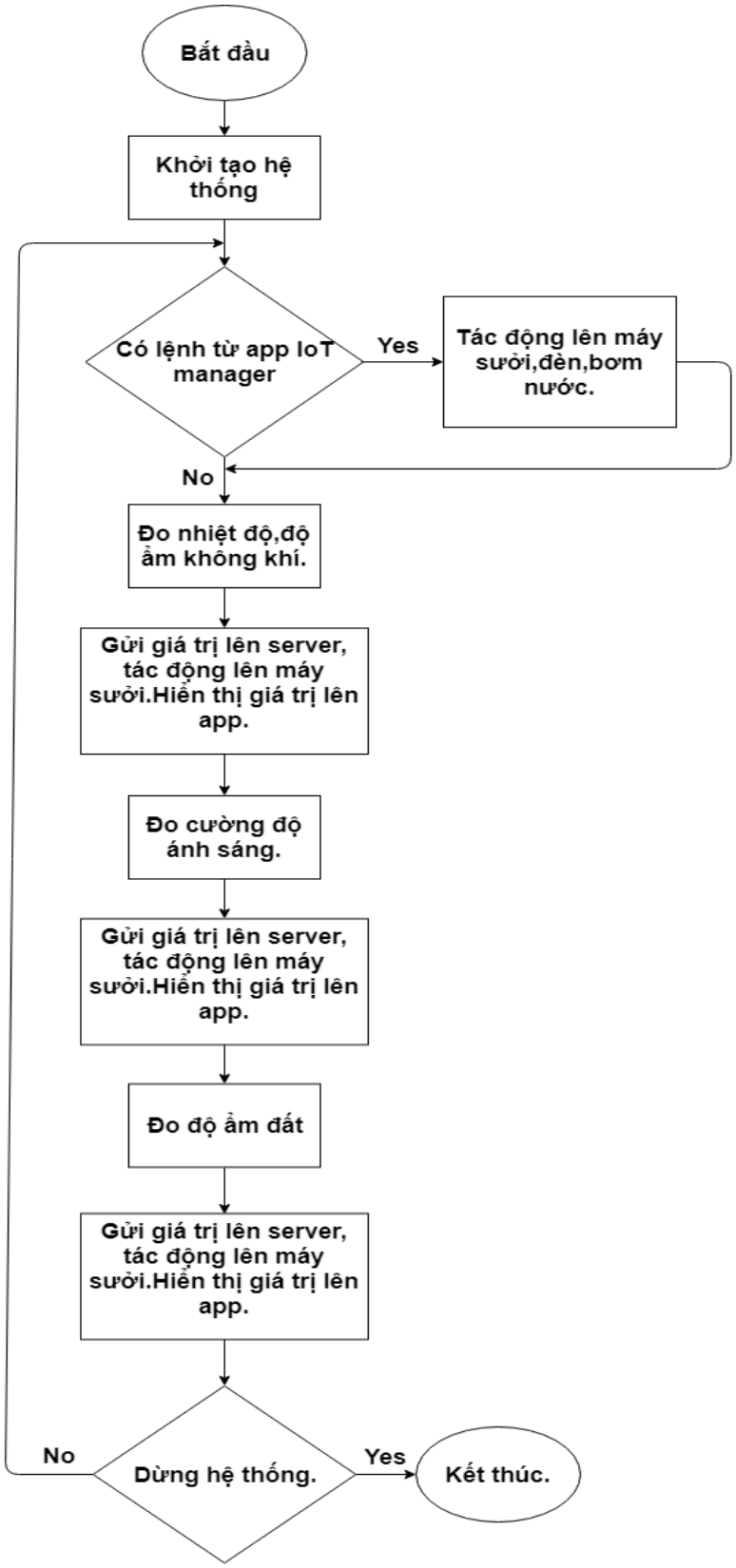
CẤU HÌNH HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT

2.1. Sơ đồ khối và lưu đồ hệ thống điều khiển giống cây trồng trong nông nghiệp

Theo các yêu cầu thiết kế ở chương 1, hệ thống được thiết kế như sau:



Hình 2.1.Sơ đồ khối thiết kế hệ thống điều khiển giống cây trồng.



Hình 2.2. Lưu đồ thiết kế hệ thống điều khiển giống cây trồng.

Hệ thống có chức năng điều khiển và giám sát hệ thống nông nghiệp từ xa bằng ứng dụng IOT Manager trên điện thoại chạy hệ điều hành Android (hoặc iOS).

Hệ thống bao gồm các khối chính là:

Đối tượng: Các khay trồng rau

Vi điều khiển STM32F103C8T6 :

- Thu thập thông tin từ các cảm biến
- Gửi thông tin kết quả đo đến module wifi ESP8266
- Nhận lệnh từ ứng dụng Android để tác động lên các thiết bị chấp hành.

Module Wifi ESP8266:

- Module wifi và điện thoại giao tiếp với nhau qua cầu nối trung gian Broker trong giao thức MQTT.
- Nhận thông tin từ vi điều khiển để gửi đến Broker
- Gửi thông tin nhận được từ Broker xuống vi điều khiển
- Thay đổi giao diện điều khiển và hiển thị của ứng dụng IoT manager

Điện thoại chạy hệ điều hành Android/iOS đã cài đặt ứng dụng IoT Manager:

- Chứa giao diện hiển thị và giao diện điều khiển
- Nhận dữ liệu từ broker và hiển thị lên màn hình ứng dụng
- Ra lệnh cho vi điều khiển bằng cách gửi tin lên Broker

Các cảm biến, module đo:

- Nhiệt độ
- Ánh sáng
- Độ ẩm không khí
- Độ ẩm đất

Các thiết bị, cấu chấp hành:

- Mạch điều chỉnh độ sáng tối của đèn chiếu sáng
- Bơm nước
- Máy sưởi

Khối nguồn:

- Nguồn xoay chiều: 220V
- Nguồn một chiều: 12V, 5V, 3.3V

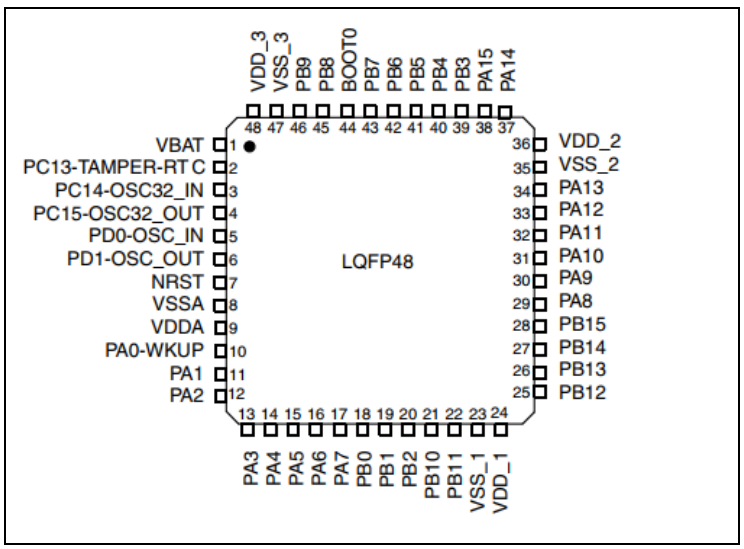
Từng thành phần của hệ thống sẽ được giới thiệu trong chương này.

2.2. Giới thiệu về vi điều khiển STM32F103C8T6

Vi điều khiển STM32F103C8T6 là họ vi điều khiển 32 bit của hãng STMicroelectronics.

Dưới đây là thông tin chung về sản phẩm:

- Lõi : ARM 32 bit Cortex-M3
- Bộ nhớ : 64-128 Kb Flash, 20 Kb SRAM
- Tần số hoạt động lên tới 72 Mhz
- DAC : không có
- ADC : 2×12 bit, tần số lấy mẫu 1Mhz
- DMA : Điều khiển 7 kênh DMA
- Timer : 7 bộ, 16 bit
- Các chuẩn giao tiếp : 2xI2C, 2xSPI, CAN, 3xUSART, USB 2.0 full-speed.
- Kiểu chân : QFPN48, LQFP48



Hình 2.3. Vi điều khiển STM32F103C8T6

Những tính năng này của vi điều khiển STM32F103C8T6 thích hợp cho một loạt các ứng dụng như điều khiển động cơ, kiểm soát các ứng dụng nâng cao, thiết bị y tế và thiết bị cầm tay, máy tính và thiết bị ngoại vi chơi game, GPS, máy quét , hệ thống báo động, ứng dụng công nghiệp, PLC, biến tần, hệ thống liên lạc video...

2.3. Giới thiệu hệ thống thời gian thực RTC

RTC (Real time clock) là bộ thời gian thực được cung cấp cho chúng ta thời gian giống như một chiếc đồng hồ thông thường.

So với các loại module hiện có trên thị trường như DS3231, DS1307… chúng ta phải dùng thêm IC để đọc được dữ liệu thời gian về ngày, tháng, năm, giờ, phút, giây và đa số các loại IC này đều sử dụng giao thức I2C để đọc/ghi dữ liệu. Còn đối với chip STM32F103C8T6 đã tích hợp sẵn một bộ thời gian thực.

Ưu và nhược điểm khi sử dụng bộ RTC trong chip STM32F103C8T6:

Ưu điểm: Không phải tốn chi phí cho bất kì IC RTC nào vì đã được tích hợp sẵn, tiết kiệm diện tích thiết kế mạch.

Nhược điểm: Bộ RTC trong chip STM32F103C8T6 sử dụng từ Clock từ các bộ LSI, LSE, HSE. Nếu sử dụng LSI làm bộ nguồn Clock thì đây là bộ clock nội và sai số tầm khoảng 1%, vì vậy trong quá trình hoạt động thì khi chúng ta đọc thời gian sẽ bị sai lệch (có thể lưu ý khắc phục được)

Việc của chúng ta chỉ cần tìm hiểu và sử dụng chứ không cần bận tâm đến phần cứng nữa. Một số ứng dụng chính mà bộ RTC mang lại là làm đồng hồ, mạch kiểm soát thời gian, báo thức, bộ đếm…Bộ RTC này sử dụng timer độc lập, tách biệt với các bộ timer khác. Việc cài đặt thời gian, đọc thời gian cũng trở nên dễ dàng bằng cách tác động trực tiếp vào thanh ghi.

Nguồn clock cấp cho bộ RTC hoạt động có thể được sử dụng 1 trong 3 nguồn sau:

HSE : sử dụng thạch anh ngoài tốc độ cao 62.5 KHz, từ 8MHz sẽ được chia 128 lần để ra tần số 62.5Khz

LSI RC: sử dụng bộ giao động RC nội tốc độ 40KHz.

LSE : sử dụng thạch anh ngoài tốc độ thấp 32.768KHz.

Thạch anh ngoài giúp bộ MCU hoạt động ổn định hơn so với bộ giao động RC nội (sai số 1%). Khi cần backup data khi mất nguồn trên chân VDD thì cần có 2 điều kiện là sử dụng thạch anh ngoài và có điện áp trên chân VBAT.

Các chức năng cơ bản của bộ RTC:

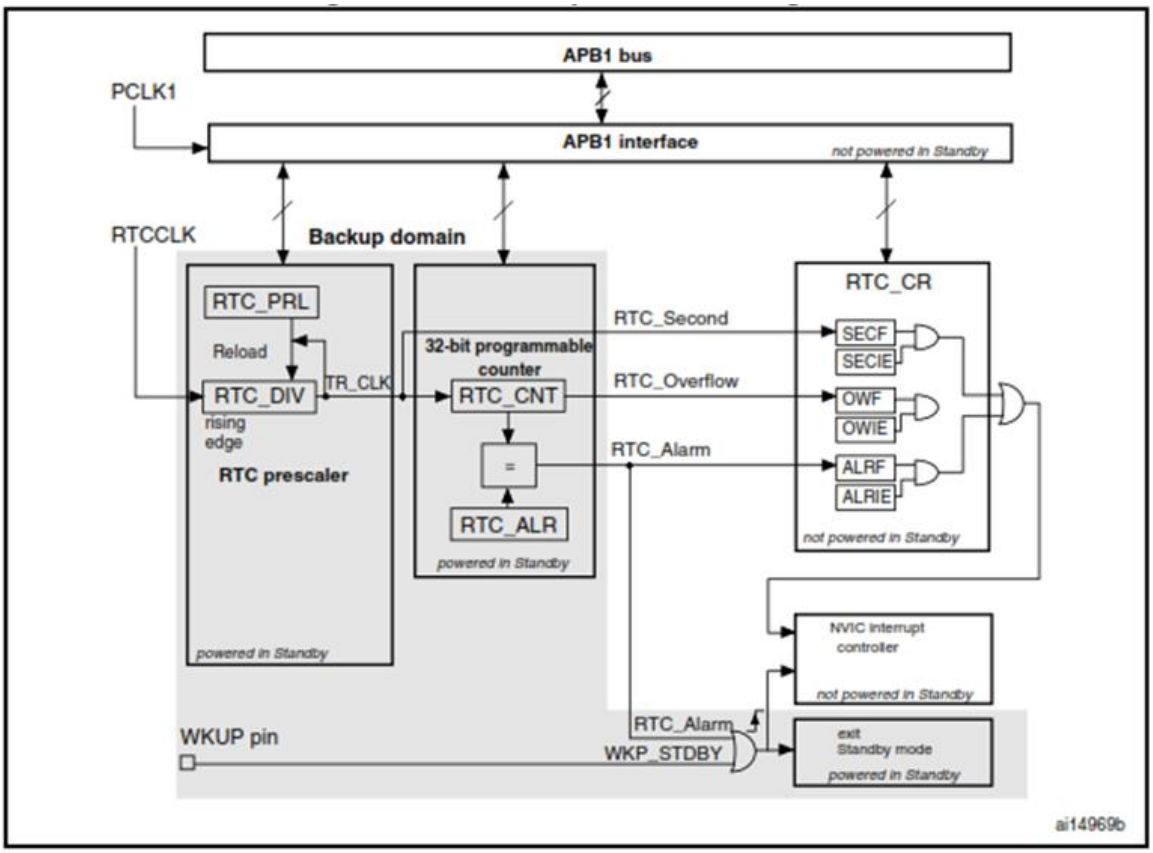
Bộ chia clock lên đến 20 bit, giúp bộ RTC hoạt động chính xác.

Độ phân giải của timer RTC lên đến 32 bit – tức là 2^32 giây mới tràn và cần reset lại.

3 nguồn clock source có thể được sử dụng.

2 loại Reset RTC riêng biệt.

Có các ngắt hỗ trợ là : ngắt alarm, ngắt mỗi giây, ngắt tràn bộ đếm.



Hình 2.4. Sơ đồ khối RTC

Bộ RTC ở đây sẽ gồm 2 phần chính:

Đầu tiên, APB1 Interface được sử dụng để giao tiếp với APB1 bus. APB1 Interface giúp cho Core có thể đọc ghi dữ liệu đến các thanh ghi trong bộ RTC thông qua APB1 bus. Ngoài ra, APB1 interface sẽ được APB1 bus clock trong quá trình giao tiếp dữ liệu.

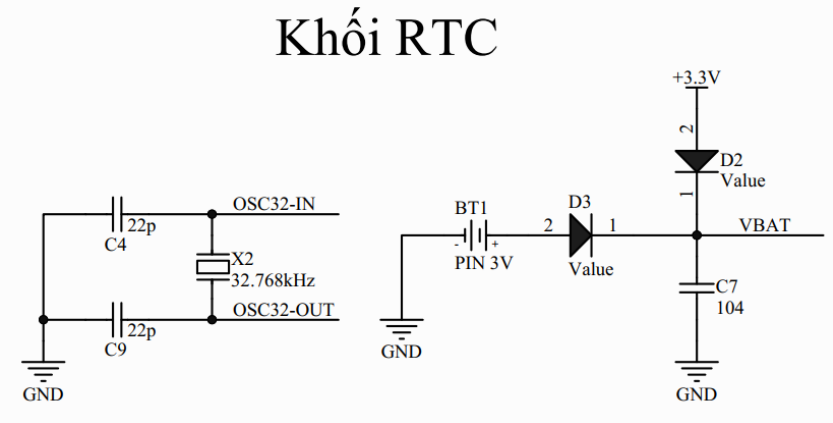
Tiếp theo, các khối RTC được chia làm 2 phần chính:

Khối đầu tiên, RTC prescaler sau khi chúng ta cấp clock cho RTC thì ở đây nó sẽ qua bộ RTC_DIV để chia tần, và chúng ta có thể lập trình tạo ra tần số lên đến 1Hz (1s).

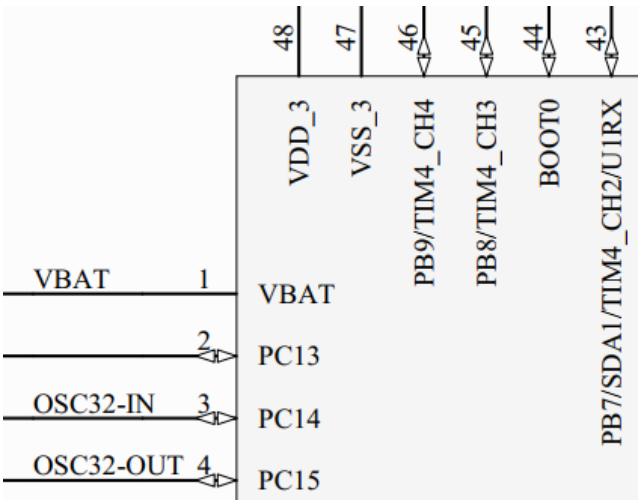
Sau đó xung TR_CLK sẽ được cấp vào khối 32 bit programmable counter, giá trị trong RTC_CNT theo với tần số TR_CLK cấp vào, RTC_CNT sẽ được so sánh với giá trị định sẵn trong RTC_ALR để tạo ra ngắt đánh thức hệ thống dậy ở chế động Standby mode(chế độ tiết kiệm năng lượng)

Thiết kế phần cứng cho hệ thống RTC

Khối RTC được thiết kế với thạch anh ngoài tốc độ thấp 32.768KHz và sử dụng nguồn pin 3V để đảm bảo RTC vẫn hoạt động khi mạch điện mất nguồn.



Hình 2.5. Thiết kế khối RTC



Hình 2.6. Kết nối với các chân trên vi điều khiển STM32F103C8T6

2.4. Module Wifi ESP8266 ESP-12E

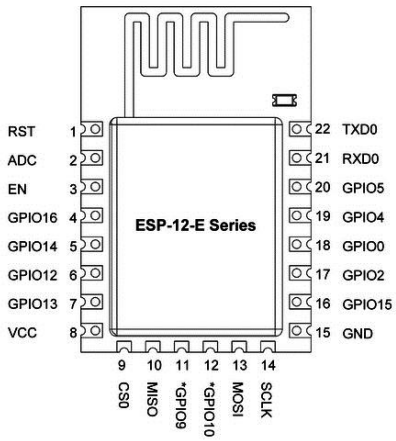
2.4.1. Các thông số cơ bản

Chip ESP8266 được phát triển bởi Espressif để cung cấp giải pháp giao tiếp Wifi cho các thiết bị IoT. Hiện tại dòng chip ESP8266 của Espressif đang được sử dụng rất phổ biến trong các thiết bị giao tiếp với smartphone hay web server thông qua Wifi nhờ giá thành rẻ, module nhỏ gọn và đặc biệt dễ sử dụng thông qua firmware có sẵn từ nhà sản xuất.

Hình 2.7. Module wifi ESP8266 12E

Do không hỗ trợ bộ nhớ Flash nên các board sử dụng ESP8266 phải gắn thêm chip Flash bên ngoài và thường là Flash SPI để ESP8266 có thể đọc chương trình ứng dụng với chuẩn SDIO hoặc SPI. ESP8266 được tích hợp vi điều khiển 32-bit Tensilica, RF balun, khuếch đại công suất, các ngoại vi cơ bản, antenna switches, khuếch đại nhận nhiễu thấp (low noise), bộ lọc và các modules quản lý nguồn. Tensilica L106 là dòng chip tiết kiệm năng lượng, tập lệnh đơn giản 16-bit RSIC, tốc độ xung nhịp clock cao nhất là 160 MHz. Nếu hệ thống hoạt động với hệ thống thời gian thực (RTOS) và Wi-Fi stack thì ta có khoảng 80% khả năng xử lý cho ứng dụng người dùng.

ESP8266 hoạt động với dải nhiệt độ khá rộng -40°C to +125°C, có thể hoạt động tốt trong môi trường công nghiệp. Với sự tích hợp cao, dòng chip này hoạt động với rất ít linh kiện ngoài làm tăng độ tin cậy, chặt chẽ và ổn định cao.



Hình 2.8. Các GPIO của module wifi ESP8266 12E

Module ESP8266 là mạch được thiết kế với các thành phần khác như Antenna, bộ nhớ Flash, mạch nguồn, và các linh kiện phụ trợ khác. Dựa trên kích thước và số lượng chân ra, có nhiều phiên bản cho Module này: ESP-01, ESP-02, ESP-03… nhưng phổ biến nhất là dòng ESP-12. Dưới đây là một số thông số của ESP-12E:

- Dải tần số 2.4GHz-2.5GHz
- Điện áp vận hành 3.0-3.6V, dòng điện 80mA
- Sử dụng 32-bit MCU core có tên là Tensilica
- Kích thước nhỏ 16mm*24mm*3mm
- Hỗ trợ giao thức wifi 802.11 b/g/n
- Có đầy đủ các ngoại vi chuẩn để giao tiếp như 17 GPIO, 1 Slave SDIO, 3 SPI, 1 I2C, 1 I2S, 2 UART, 2 PWM
- Có shield chống nhiễu và PCB anten on-board

Thông số phần mềm:

- Chế độ Wi-fi: station/softAP/softAP+ station
- Mã hóa WEP/TKIP/AES với bảo mật WPA/WPA2
- Cập nhật firmware qua chân UART/OTA qua network/ WebServer/Cloud server MQTT
- Hỗ trợ các giao thức mạng như Ipv4, TCP/UDP/HTTP/FTP

- Hỗ trợ người dùng sử dụng các lệnh AT, Cloud Server, Application trên Android/iOS để làm việc với ESP-12E

2.4.2. Các mô hình lập trình

Có nhiều mô hình lập trình cho ESP8266, một trong số đó là sử dụng các thư viện từ cộng đồng phát triển. Cộng đồng phát triển của ESP8266 cũng góp phần tạo ra các thư viện cũng như môi trường lập trình khác thân thiện hơn giúp cho nhiều người không chuyên về ngôn ngữ C vẫn có thể tiếp cận và sử dụng module ESP8266 một cách dễ dàng hơn. Hai ngôn ngữ đang hỗ trợ mạnh mẽ nhất trong cộng đồng là MicroPython và Adruino.

Bộ thư viện Arduino cho ESP8266 hỗ trợ các hàm chuẩn để làm việc với Digital IO, Analog IO, Software I2C, UART, timer. Ngoài ra, bộ thư viện còn hỗ trợ các hàm API của riêng ESP8266 như các hàm Wifi, các hàm để lấy thông tin phần cứng trên ESP, các hàm để làm việc với files/folders, và các hàm để thực hiện cả OTA.

Ưu điểm: sử dụng được ngôn ngữ cấp cao (Python, Arduino), không cần sử dụng vi điều khiển ngoài để hỗ trợ ESP8266.

Nhược điểm: bị giới hạn bởi bộ thư viện được cung cấp, khả năng tối ưu phần cứng, tốc độ truyền dữ liệu không cao

2.4.3. Lập trình ESP8266 sử dụng giao thức MQTT

Với ESP8266, các thiết bị IoT có thể dễ dàng kết nối đến TCP socket của một MQTT Broker từ đó mở ra khả năng trao đổi dữ liệu giữa hàng ngàn thiết bị IoT với nhau qua mô hình cloud server. Hiện nay, có khá nhiều thư viện mã nguồn mở chúng ta có thể sử dụng cho dự án của mình một cách nhanh chóng mà không cần thời gian đọc hiểu chi tiết giao thức MQTT. Dưới đây là các thư viện mã nguồn mở phổ biến như sau:

- Thư viện open source Pubsubclient cho MQTT Client để lập trình trên Arduino
- Thư viện open source cho MQTT Broker
- Thư viện open source cho MQTT Client để lập trình với SDK của ESP8266 trên môi trường Eclipse

Mỗi thư viện đều có tài liệu và code mẫu hướng dẫn cách sử dụng các hàm API cho việc kết nối server, publish(), subscribe(), unsubscribe() và đăng ký callback(). Do đó, chúng ta có thể dễ dàng tích hợp khả năng truyền dữ liệu qua cloud server giữa các thiết bị ESP8266 một cách nhanh chóng và dễ dàng với MQTT. Các bộ thư viện open

source này được cộng đồng phát triển tiếp tục hoàn thiện. Trong đồ án này, tác giả sử dụng thư viện PubSubClient và lập trình trên Arduino vì thư viện này đơn giản và dễ hiểu.

Các hàm API cơ bản và thông dụng nhất của thư viện PubSubClient có thể được liệt kê như sau:

- Hàm PubSubClient(Client &client) để khởi động bộ thư viện. Chúng ta sẽ tạo 1 biến kiểu Client và pass địa chỉ biến đó vào hàm này để khởi động bộ thư viện.
- Hàm setServer(<domain name của Broker>, <TCP port>) để thiết lập địa chỉ của Broker server và port TCP sẽ kết nối đến
- Hàm setCallback(<callback>) để đăng ký hàm callback sẽ được gọi khi thư viện nhận được giá trị mới từ Broker cho dữ liệu đã được subscribe
- Hàm connect(<client_name>) để bắt đầu quá trình kết nối đến Broker server. Hàm này sẽ đợi đến khi kết nối thành công hoặc timeout; giá trị trả về của hàm là kiểu boolean với True là kết nối thành công, False là kết nối không thành công
- Hàm disconnect() để chủ động ngắt kết nối đến Broker server
- Hàm publish(<topic>, <value>) để gửi giá trị <value >cho dữ liệu <topic> lên Broker server
- Hàm subscribe(<topic>) để đăng ký nhận giá trị mới của dữ liệu <topic> từ Broker server
- Hàm connnected() để kiểm tra trạng thái kết nối đến Broker server (trả về True là kết nối thành công)
- Hàm loop() là hàm xử lý các tác vụ theo giao thức MQTT. Chúng ta sẽ gọi hàm này liên tục trong hàm loop() của Arduino

2.5. Module cảm biến nhiệt độ, độ ẩm DHT21

2.5.1. Các thông số cơ bản

DHT21/AM2301 là mạch tích hợp tích hợp cảm biến độ ẩm điện dung và cảm biến nhiệt độ có độ chính xác cao. Sản phẩm có độ chống nhiễu và độ chính xác cao, ngõ ra tín hiệu số và được giao tiếp với Vi điều khiển 8-bit thông qua 1 dây duy nhất(SDA)

giúp tiết kiệm chân Vi điều khiển. Thiết kế nhỏ gọn, tiêu thụ điện năng thấp, khoảng cách truyền dẫn tín hiệu lên đến 20m giúp việc lắp đặt dễ dàng hơn. Điện năng tiêu thụ cực thấp, khoảng cách truyền dẫn, hiệu chuẩn hoàn toàn tự động, sử dụng các cảm biến độ ẩm điện dung, hoàn toàn hoán đổi cho nhau, thiết bị đo nhiệt độ chính xác cao.

Hình 2.9. Module cảm biến nhiệt độ, độ ẩm

Thông số kỹ thuật:

- Kích thước: 58.8 x 26.7 x 13.8 (mm)
- Điện áp hoạt động: 3.3 - 5V
- Dòng tiêu thụ: 300 uA
- Model: AM2301
- Độ phân giải chính xác: 0.1
- Khoảng đo độ ẩm: 0-99.9% RH
- Khoảng đo nhiệt độ: -40 ℃ ~ 80 ℃
- Sai số độ ẩm: ± 3% RH
- Sai số nhiệt độ: ± 0.5 ℃
- Chuẩn giao tiếp: 1 dây (1 wire).

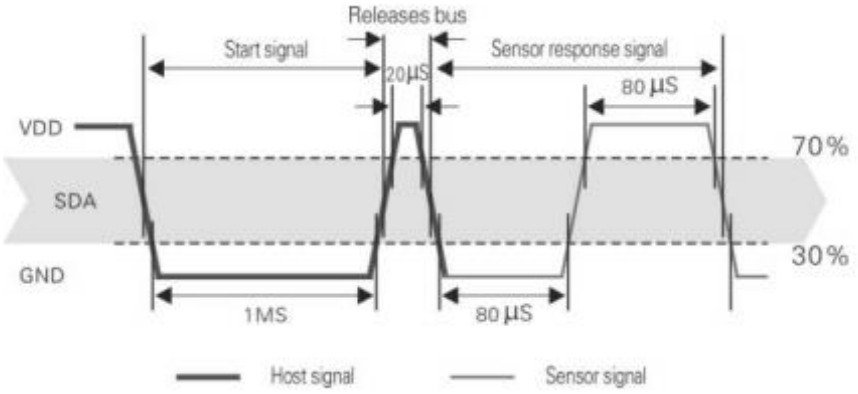
2.5.2. Nguyên lý hoạt động.

Để có thể giao tiếp với DHT21 theo chuẩn 1 chân vi xử lý thực hiện theo 2 bước:

- Gửi tín hiệu muốn đo (Start) tới chân DHT21, sau đó DHT21 xác nhận lại.

- Khi đã giao tiếp được với DHT21, cảm biến sẽ gửi lại 5 byte dữ liệu và nhiệt độ đo được.

Bước 1: Gửi tín hiệu start



Hình 2.10. Gửi tín hiệu start và tín hiệu phản hồi từ sensor.

- MCU thiết lập chân DATA là Output, kéo chân DATA xuống 0 trong khoảng thời gian lớn hơn 800ms. Khi đó DHT21 sẽ hiểu MCU muốn đo giá trị nhiệt độ và độ ẩm.
- MCU đưa chân DATA lên 1, sau đó thiết lập lại là chân đầu vào.
- Sau khoảng 20µs DHT21 sẽ kéo chân DATA xuống thấp. Nếu lớn hơn 20µs mà chân DATA không được kéo xuống thấp nghĩa là không giao tiếp được với DHT21.
- Chân DATA sẽ ở mức thấp 80µs sau đó nó được DHT21 kéo lên cao trong 80µs. Bằng việc giám sát chân DATA, MCU có thể biết được có giao tiếp với DHT21 không. Nếu tín hiệu đo được ở DHT21 lên cao, khi đó hoàn thiện quá trình giao tiếp của MCU với DHT21.

Bước 2: Đọc giá trị trên DHT21.

DHT21 sẽ trả về giá trị nhiệt độ và độ ẩm dưới dạng 5 byte, được miêu tả như sau:

- Byte 1: Byte cao của giá trị độ ẩm.
- Byte 2: Byte thấp của giá trị độ ẩm.
- Byte 3: Byte cao của giá trị nhiệt độ.
- Byte 4: Byte thấp của giá trị nhiệt độ.
- Byte 5: Kiểm tra tổng.

Nếu Byte 5 (8 bit) = (Byte 1 + Byte 2 + Byte 3 + Byte 4) thì giá trị độ ẩm và nhiệt độ là chính xác, nếu sai thì kết quả đo không có ý nghĩa.

Ví dụ tính toán dữ liệu từ chân DATA

Ví dụ 1 : Nhận 5 byte dữ liệu.

0000 0010	1001 0010	0000 0001	0000 1101	1010 0010
Byte cao độ ẩm	Byte thấp độ ẩm	Byte cao nhiệt độ	Byte thấp nhiệt độ	Byte kiểm tra tổng

Tính toán :

0000 0010 + 1001 0010 +0000 0001 + 0000 1101 = 1010 0010 (Byte kiểm tra tổng)

(Nếu tổng 4 byte không bằng byte kiểm tra tổng thì kết quả nhận được không chính xác, yêu cầu nhận lại dữ liệu.)

Dữ liệu nhận được là chính xác:

Độ ẩm : 0000 0010 1001 0010 = 0292H (Hệ hexa) = 2 × 256 + 9 × 16 + 2 = 658

=> Độ ẩm = 65,8% Rh

Nhiệt độ : 0000 0001 0000 1101 = 10DH (Hệ hexa) = 1 × 256 + 0 × 16 + 13 = 269

=> Nhiệt độ = 26,9 ℃

Trường hợp đặc biệt : Khi nhiệt độ dưới 0℃

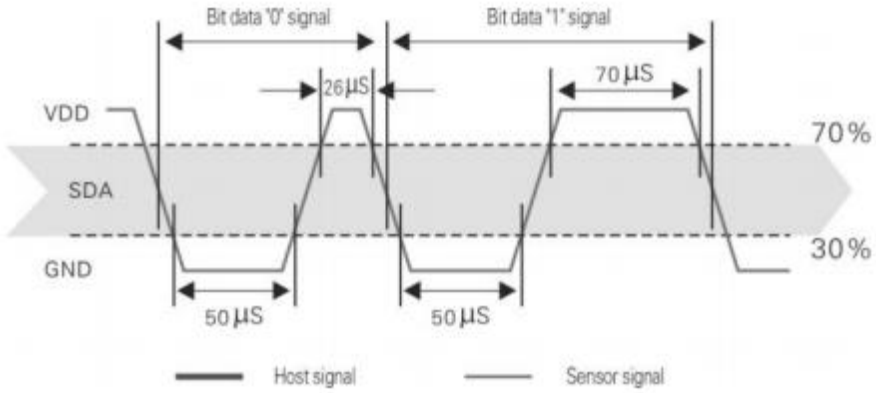
Ví dụ 2: -10.1 ℃ . Được biểu thị là 1000 0000 0110 0101

Nhiệt độ : 00000000 0000 0110 0101 = 0065H (Hệ hexa) 6 × 16 +5 = 101

=> Nhiệt độ. = -10,1 ℃

Bước 3: Đọc dữ liệu.

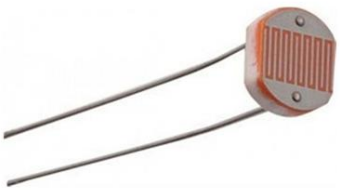
Sau khi giao tiếp được với DHT21, DHT21 sẽ gửi liên tiếp 40 bit 0 hoặc 1 về MCU, tương ứng với 5 byte kết quả của nhiệt độ và độ ẩm.



Hình 2.11. Thời gian xác định bit gửi về MCU.

Sau khi tín hiệu được đưa về 0, ta đợi chân DATA của MCU được DHT21 kéo lên 1. Nếu chân DATA là 1 trong khoảng 26-28µs thì là bit 0, còn nếu tồn tại 70µs thì là bit 1. Do đó trong lập trình ta lập trình bắt sườn lên của chân DATA, sau đó delay 50µs. Cứ thế ta đọc được các bit tiếp theo.

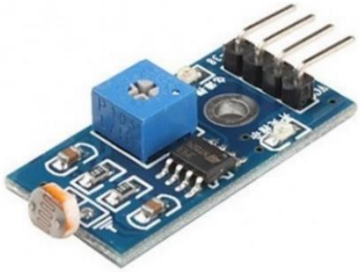
2.6. Module đo cường độ ánh sáng dùng quang trở



Hình 2.12. Một quang trở trên thị trường

Quang trở là điện trở có trị số càng giảm khi được chiếu sáng càng mạnh. Điện trở tối (khi không được chiếu sáng hoặc ở trong bóng tối) thường trên 1MΩ, trị số này giảm rất nhỏ có thể dưới 100Ω khi được chiếu sáng mạnh.

Nguyên lý làm việc của quang trở là khi ánh sáng chiếu vào chất bán dẫn (có thể là Cadmium sulfide – CdS, Cadmium selenide – CdSe) làm phát sinh các điện tử tự do, độ dẫn điện tăng lên và làm giảm điện trở của chất bán dẫn. Các đặc tính điện và độ nhạy của quang điện trở phụ thuộc vào vật liệu dùng trong chế tạo.

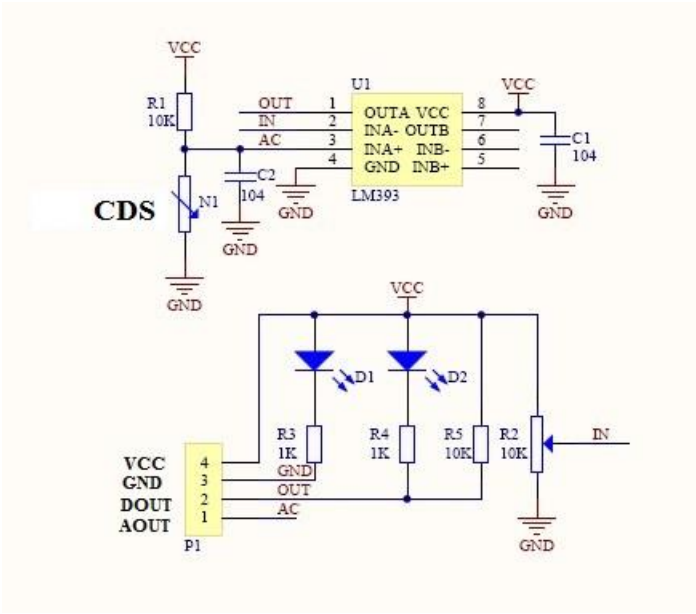


Hình 2.13. Module cảm biến quang trở CDS5537

Module bao gồm 1 quang trở CDS 5MM 5537, IC so sánh LM393, biến trở điều chỉnh độ nhạy của quang trở, các led báo hiệu có nguồn và báo thay đổi mức logic ở chân OUTA, các điện trở, các tụ điện 104.

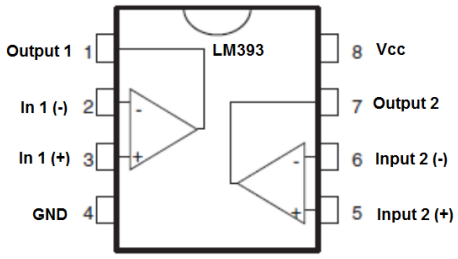
Thông số của quang trở CDS 5537:

- Điện áp tối đa: 150 V một chiều
- Nhiệt độ hoạt động: -30 độ C đến 70 độ C
- Đỉnh quang phổ: 540nm
- Điện trở khi có ánh sáng (10Lx) : 16→50 kΩ
- Điện trở khi không có ánh sáng : nhỏ nhất là 2MΩ
- Thời gian đáp ứng: 20 ms



Hình 2.14. Sơ đồ nguyên lý module cảm biến quang trở CDS5537

Module có hàng jump 4 chân bao gồm: chân cấp nguồn cho module VCC, chân đất GND, chân ra tín hiệu số DOUT, chân ra tín hiệu tương tự AOUT. Chân DOUT là chân đầu ra Output 1 của một khối so sánh trong LM393.



Hình 2.15. IC so sánh LM393

Khi không có ánh sáng hoặc ánh sáng yếu rọi vào quang trở, điện trở quang trở rất lớn, điện áp rơi trên nó lớn (gần bằng VCC). Nếu điện áp này lớn hơn điện áp tham chiếu lấy từ biến trở thì Output 1 có giá trị logic ở mức cao, kéo theo DOUT cũng ở mức logic cao và ngược lại. Chân DOUT được sử dụng khi điều khiển on/off hoặc ngắt ngoài. Chân AOUT được đưa vào các bộ ADC rời hoặc khối ADC trong vi điều khiển nhằm tính toán giá trị cường độ ánh sáng.

2.7. Module đo độ ẩm đất.

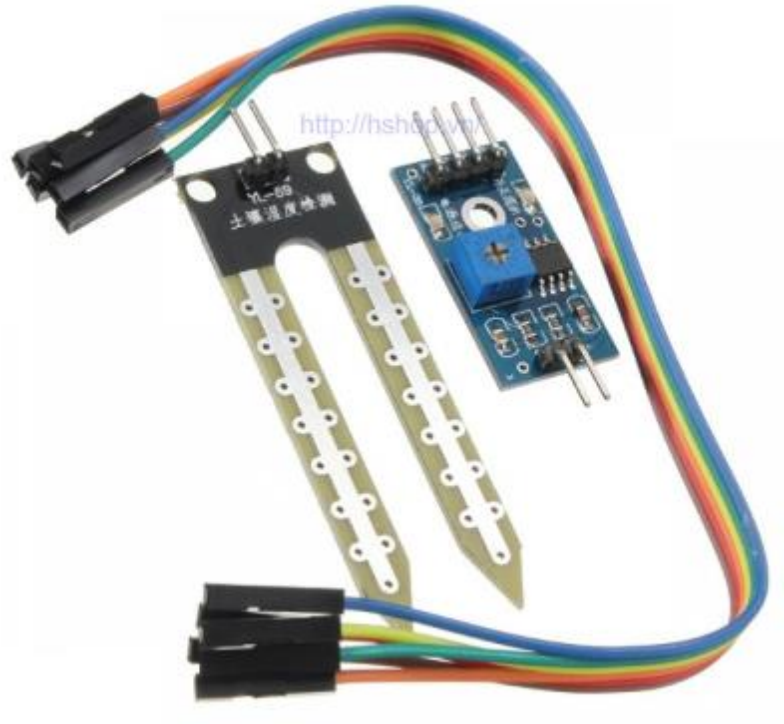
Cảm biến độ ẩm đất thường được sử dụng trong các mô hình tưới nước tự động, vườn thông minh,..., cảm biến giúp xác định độ ẩm của đất qua đầu dò và trả về giá trị Analog, Digital qua 2 chân tương ứng để giao tiếp với Vi điều khiển để thực hiện vô số các ứng dụng khác nhau.

Nguyên lý làm việc của sensor cảm biến là khi môi trường có độ ẩm lớn làm phát sinh các điện tử tự do, độ dẫn điện tăng lên và làm giảm điện trở của chất bán dẫn. Các đặc tính điện và độ nhạy của sensor cảm biến trở phụ thuộc vào vật liệu dùng trong chế tạo

Thông số kỹ thuật:

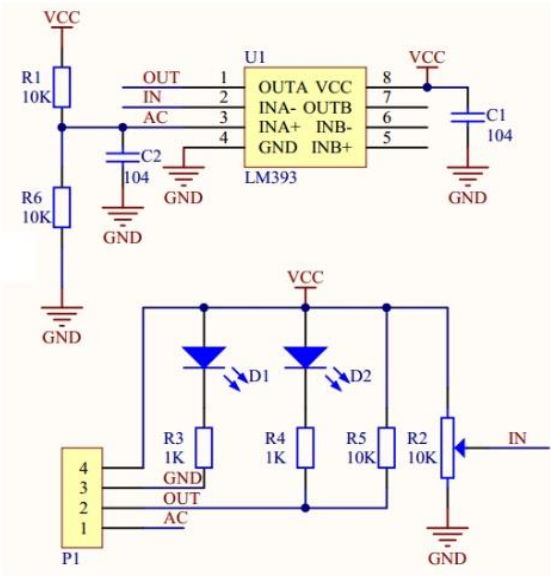
- Điện áp hoạt động: 3.3~5VDC
- Tín hiệu đầu ra:
    - Analog: theo điện áp cấp nguồn tương ứng.
    - Digital: High hoặc Low, có thể điều chỉnh độ ẩm mong muốn bằng biến trở thông qua mạch so sánh LM393 tích hợp.Kích thước: 3 x 1.6cm.

- Kích thước : 3 x 1.6cm
- Nguồn 5VDC
- DO đầu ra tín hiệu số
- AO đầu ra tín hiệu tương tự



Hình 2.16. Module cảm biến độ ẩm đất

Với tín hiệu trả về là tín hiệu tương tự, người lập trình hoàn toàn có thể sử dụng module ADC của vi điều khiển để đọc và tính toán giá trị độ ẩm đất theo hướng dẫn của hãng.



Hình 2.17. Sơ đồ nguyên lý module cảm biến độ ẩm đất

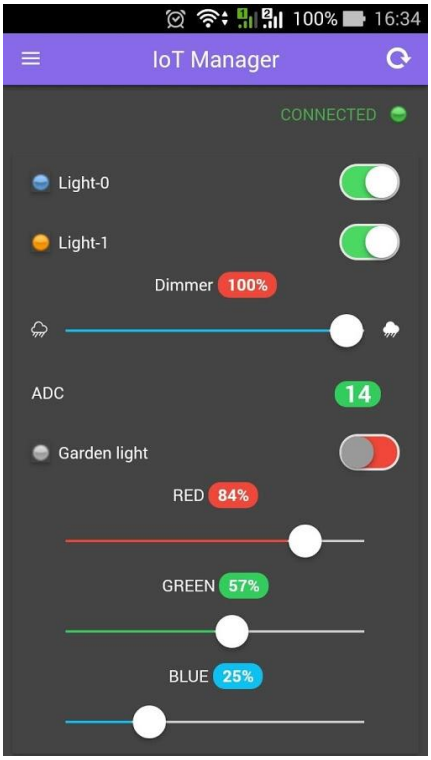
Về cơ bản nguyên lý hoạt động của module cảm biến độ ẩm đất khá giống với module cảm biến ánh sang gồm 4 chân : chân cấp nguồn cho module VCC, chân đất GND, chân ra tín hiệu số DOUT, chân ra tín hiệu tương tự AOUT. Chân DOUT được sử dụng khi điều khiển on/off hoặc ngắt ngoài. Chân AOUT được đưa vào các bộ ADC rời hoặc khối ADC trong vi điều khiển nhằm tính toán giá trị cường độ ánh sáng.

2.8. Ứng dụng IoT Manager trên hệ điều hành Android

IoT Manager hỗ trợ hệ điều hành Android và iOS, được viết bởi Victor Brutskiy, đây là phần mềm cho phép người lập trình có thể thay đổi giao diện qua mã nguồn mở trong môi trường Arduino IDE. IoT Manager hỗ trợ các thiết bị như Arduino, ESP8266, Raspberi Pi; các ngôn ngữ lập trình C, C++, python, NodeJS. Bên cạnh đó, phần mềm còn hỗ trợ người dùng phát triển các ứng dụng IoT bằng giao thức MQTT.

Người dùng có thể thêm/xóa các tiện ích điều khiển (widget) như toggle, badge, power-button, range , fillgauge, display-value, simple-btn, anydata, steel cho các giao diện hiển thị và điều khiển. Bằng việc kết nối với mạng Internet, IoT Manager có thể quản lý các thiết bị từ xa qua giao thức MQTT.

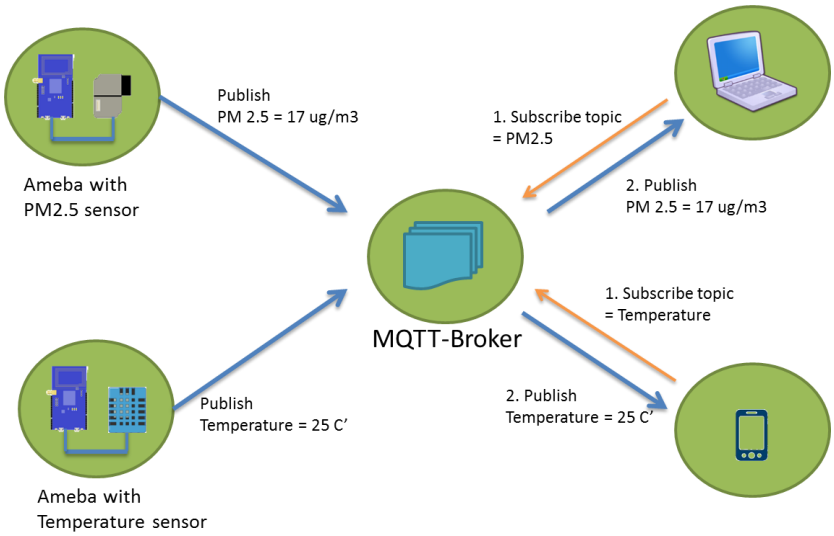
IoT Manager được tải về và cài đặt trên Google Play đối với hệ điều hành Android, hoặc AppStore đối với hệ điều hành iOS.



Hình 2.18. Giao diện ứng dụng IoT Manager

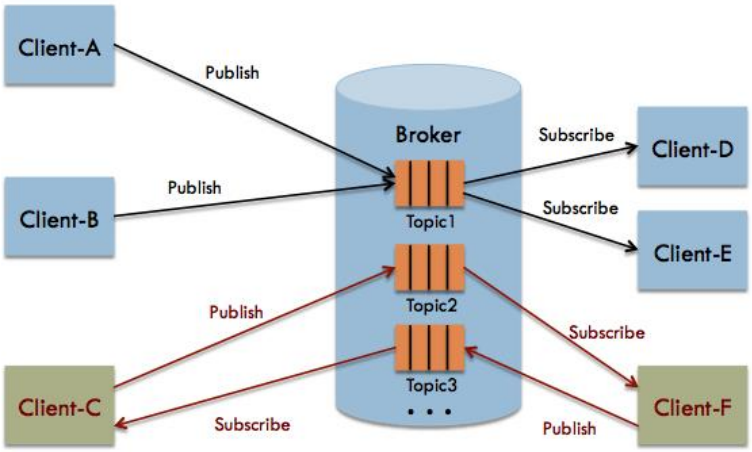
2.9. Giao thức MQTT

MQTT (Message Queuing Telementary Transport) là một giao thức gửi gói tin dạng publish/subscribe sử dụng cho các thiết bị Internet of Things.



Hình 2.19. Một ứng dụng của giao thức của MQTT

MQTT hoạt động với độ tin cậy cao, băng thông thấp và khả năng được sử dụng trong mạng lưới không ổn định, vì vậy đây là giao thức rất thích hợp để xây dựng hệ thống thông minh có khả năng giám sát, điều khiển từ xa. Ngoài ra, với cơ chế Publish/Subscribe thông qua Broker trung gian giúp hệ thống giải quyết được vấn đề đồng bộ khi điều khiển từ nhiều thiết bị khác nhau mà không sợ xung đột hoặc bị ngược trạng thái điều khiển.



Hình 2.20. Cơ chế hoạt động của MQTT

Về cơ bản, các thiết bị IoT sẽ đóng vai trò Client và đăng ký với dữ liệu muốn gửi lên hoặc dữ liệu nó muốn nhận được. Nhiệm vụ của Broker là thiết bị trung gian nhận dữ liệu truyền lên từ các thiết bị Client và gửi dữ liệu đến các Client muốn nhận tương ứng. Mô hình Broker – Client tương tự như mô hình tổng đài điện thoại. Tuy nhiên, MQTT không phải là mô hình Server-Client vì tất cả thiết bị đều ngang hàng và đều có thể gửi dữ liệu lẫn nhau thông qua cầu nối Broker.

Các thiết bị Client có thể đăng ký nhận dữ liệu hoặc ngừng nhận dữ liệu từ Broker trong thời gian hoạt động tùy yêu cầu của ứng dụng. Ngoài ra các thiết bị Client gửi dữ liệu cũng không cần đăng ký trước loại dữ liệu mà nó sẽ gửi. Do đó, giao thức MQTT được coi là mô hình sao và cho phép nhiều thiết bị trao đổi dữ liệu khá linh hoạt cả đối với thiết bị gửi dữ liệu và thiết bị nhận dữ liệu.

Luồng làm việc của giao thức MQTT có thể được mô tả một cách đơn giản như sau:

- Kết nối đến một Broker

- Đăng ký loại dữ liệu muốn nhận (nếu thiết bị có nhu cầu nhận dữ liệu)
- Gửi dữ liệu lên Broker (khi thiết bị có dữ liệu và muốn gửi tới các thiết bị khác)

Các khái niệm cơ bản của giao thức MQTT như sau:

- Broker: là thiết bị trung gian nhận dữ liệu từ các Client muốn gửi và gửi dữ liệu đó tới các Client muốn nhận
- Client : là thiết bị IoT muốn gửi/nhận dữ liệu trong network
- Topic: tượng trưng cho loại dữ liệu mà các thiết bị Client gửi/nhận thông qua MQTT. Topic là một chuỗi UTF-8 text tượng trưng cho tên loại dữ liệu.
- Publish: là bước gửi dữ liệu từ một thiết bị Client đến Broker. Trong bước này, thiết bị IoT sẽ xác định topic (là loại dữ liệu muốn gửi) và giá trị của topic đó
- Subscribe: là bước đăng ký nhận dữ liệu từ Broker của một thiết bị Client . Trong bước này, thiết bị IoT sẽ xác định loại dữ liệu mà nó muốn nhận. Khi Broker nhận được loại dữ liệu này từ một thiết bị Client khác, nó sẽ gửi dữ liệu này tới thiết bị Client đã đăng ký nhận
- Unsubscribe: là bước thông báo với Broker là thiết bị Client không muốn tiếp tục nhận dữ liệu nữa

Sau khi hiểu được các khái niệm cơ bản, người lập trình có thể sử dụng các thư viện mã nguồn mở cho MQTT Client và MQTT Broker vì chương trình sử dụng thư viện MQTT Client khá đơn giản chỉ sử dụng vài hàm API như sau:

- Tạo kết nối TCP đến server chạy code MQTT Broker
- Gọi hàm **publish()** để gửi dữ liệu lên Broker
- Gọi hàm **subscribe()** để đăng ký nhận dữ liệu từ Broker (hoặc **unsubscribe()** để ngừng nhận dữ liệu từ Broker)
- Đăng ký hàm **callback()** của ứng dụng để được gọi khi thư viện MQTT Client nhận được dữ liệu từ Broker

Hiện nay có một số Broker được xây dựng sẵn như HiveMQ Broker, Mosquito Broker, ThingMQ, ThingStudio… Trong thiết kế hệ thống, dịch vụ CloudMQTT chạy trên nền tảng Mosquito Broker được sử dụng.

Chương 3
THIẾT KẾ PHẦN CỨNG

3.1. Khối nguồn

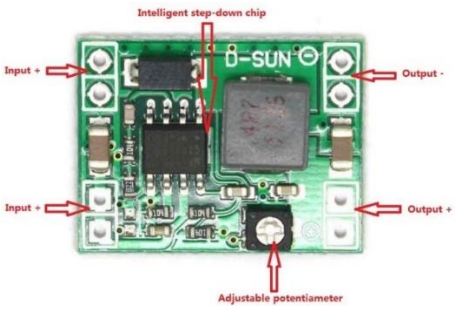
Trong thiết kế, một adapter nguồn 12V-3A cung cấp nguồn cho toàn bộ mạch điều khiển. Ngoài ra, một số thiết bị cần nguồn 220V xoay chiều thì được cấp riêng từ lưới điện dân dụng.

Hình 3.1. Adapter nguồn 12V cho mạch điều khiển

Trên board mạch điều khiển, các nguồn 5V và 3,3Vđược thiết kế cho vi điều khiển và các ngoại vi. Để tạo nguồn 5V, tác giả sử dụng nguồn Buck DC-DC Mini, điều chỉnh điện áp ra bằng biến trở để được 5V ở đầu ra.

Thông số:

- Sử dụng IC MP1584
- Kích thước nhỏ gọn: 22x17 mm
- Điện áp đầu vào: 4.5 – 28V
- Điện áp đầu ra: 0.8 – 20V, dòng điện ra lên tới 3A
- Hiệu suất 96%

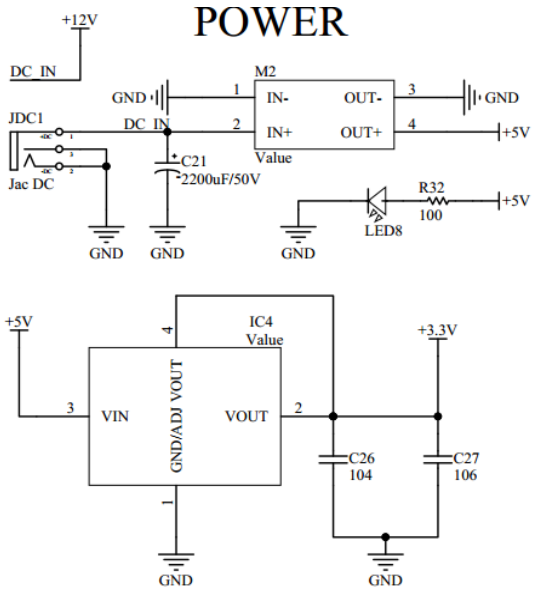


Hình 3.2. Module nguồn buck 3A mini

Nguồn 3,3V được tạo ra nhờ sử dụng IC AMS1117-3.3V, kiểu chân dán SOT223, dòng điện ra là 1A, điện áp đầu vào 5V lấy từ nguồn Buck.

Hình 3.3. IC nguồn 3.3V

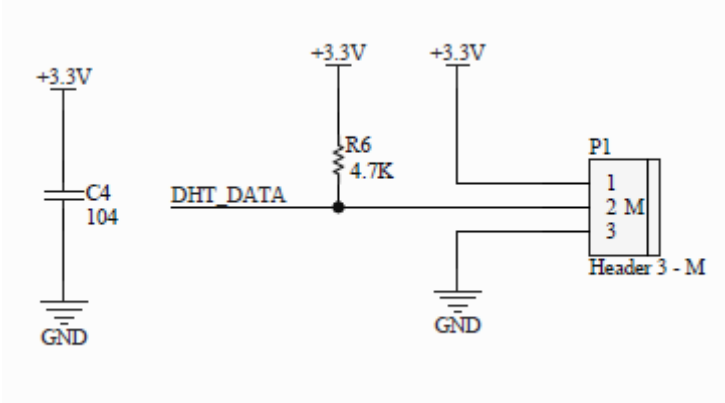
Thiết kế chi tiết được mô tả bằng hình ảnh dưới đây:



Hình 3.4. Thiết kế khối nguồn của mạch điều khiển

3.2. Khối đo nhiệt độ, độ ẩm hệ thống điều khiển nông nghiệp.

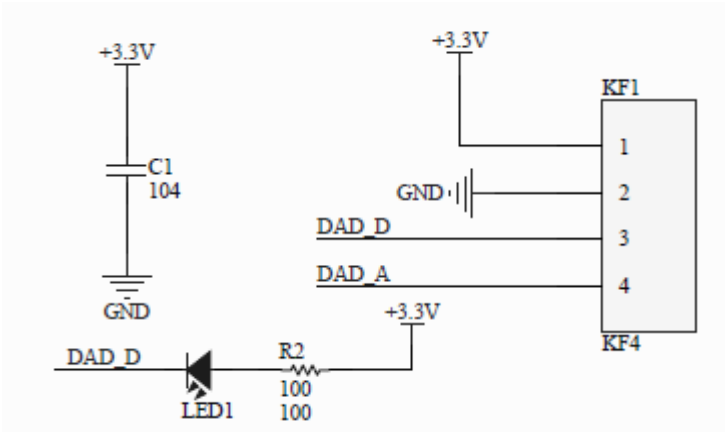
Module DHT21 được sử dụng để đo nhiệt độ, độ ẩm của hệ thống nông nghiệp. Vi điều khiển đọc nhiệt độ cảm biến qua giao tiếp số một dây (One Wire), trong đó chân DHT_DATA được nối đến chân PA9 của STM32F103C8T6. Nguồn cấp cho module là nguồn 3.3V một chiều.



Hình 3.5. Thiết kế khối đo nhiệt độ, độ ẩm DHT21.

3.3. Khối đo độ ẩm đất của hệ thống nông nghiệp.

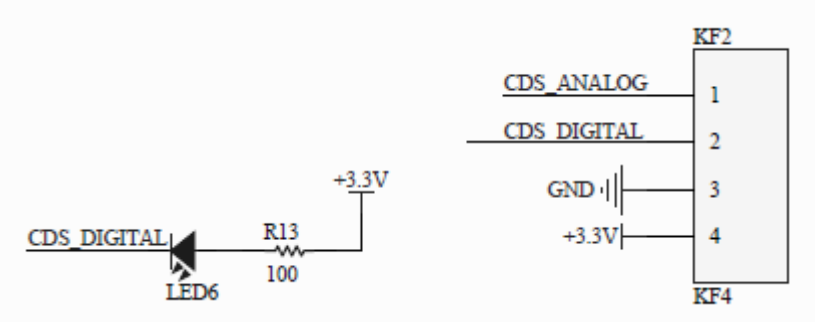
Trong thiết kế sử dụng một cảm biến để đo độ ẩm đất của hệ thống nông nghiệp. Chân dữ liệu (DAD_A) của cảm biến được nối đến chân ADC0 của vi điều khiển. Nguồn cấp cho cảm biến là nguồn 3.3V một chiều.



Hình 3.6. Cảm biến độ ẩm đất.

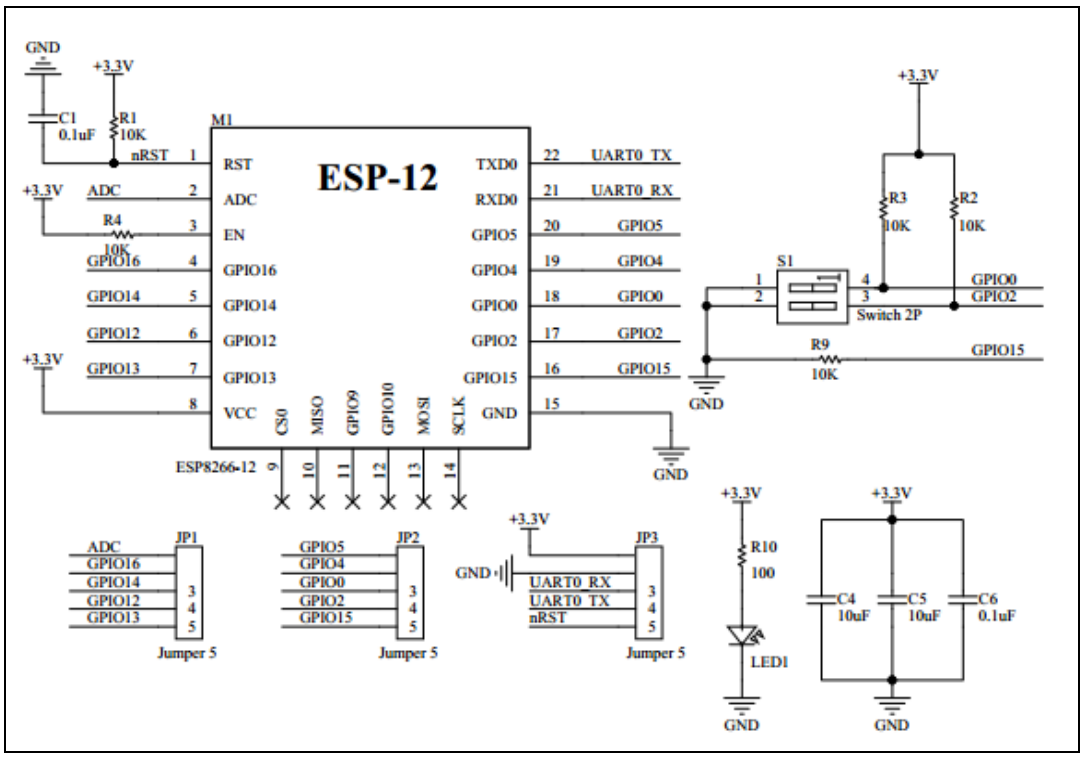
3.4. Khối đo cường độ ánh sang của hệ thống nông nghiệp.

Module cảm biến ánh sáng quang trở CDS5537 được sử dụng để đo cường độ ánh sáng. Module sử dụng nguồn 5V. Module có hai chân tín hiệu ra là chân đầu ra tương tự (CDS_ANALOG) và chân đầu ra số (CDS_DIGITAL). Chân đầu ra tương tự được nối đến chân ADC1 của vi điều khiển qua một mạch chia áp. Mạch chia đảm bảo không gây quá áp cho chân của vi điều khiển khi điện áp từ đầu ra tương tự của module có thể lên đến 5V.



Hình 3.7. Thiết kế khối đo cường độ ánh sáng

3.5. Thiết kế phần cứng cho module wifi ESP8266



Hình 3.10. Thiết kế khối module wifi ESP8266

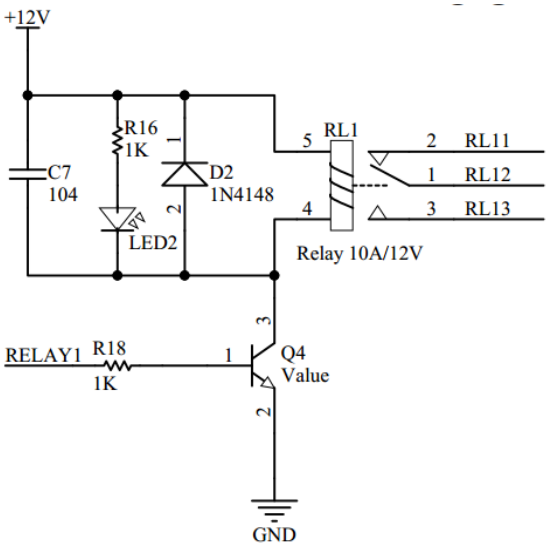
3.6. Khối Relay điều khiển đóng cắt các thiết bị

Khối Relay được thiết kế với 2 Relay, dùng để đóng cắt các thiết bị như bơm tưới nước, hệ thống sưởi mô hình nông nghiệp hoặc các thiết bị chỉ cần điều khiển bật, tắt khác.

Các thông số của Relay:

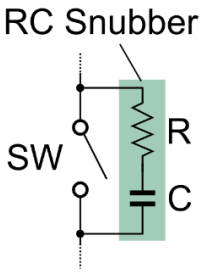
- Điện áp điều khiển: 12V một chiều
- Loại 5 chân, một tiếp điểm thường mở, một tiếp điểm thường đóng
- Dòng điện định mức của tải 10A, điện áp định mức của tải: 250 VAC

- Tuổi thọ: tuổi thọ cơ khí 10^7 lần đóng cắt, tuổi thọ điện 10^5 lần đóng cắt



Hình 3.15. Thiết kế một mạch điều khiển Relay

Trên thực tế, người thiết kế cần sử dụng mạch snubber RC nối song song với tiếp điểm của Relay nhằm mục đích để dập hồ quang khi đóng/cắt và tăng tuổi thọ của tiếp điểm. Mạch snubber RC bao gồm mạch nối tiếp điện trở và tụ điện. Thông thường, giá trị của điện trở R = 100Ω, giá trị tụ điện là C = 100 nF.



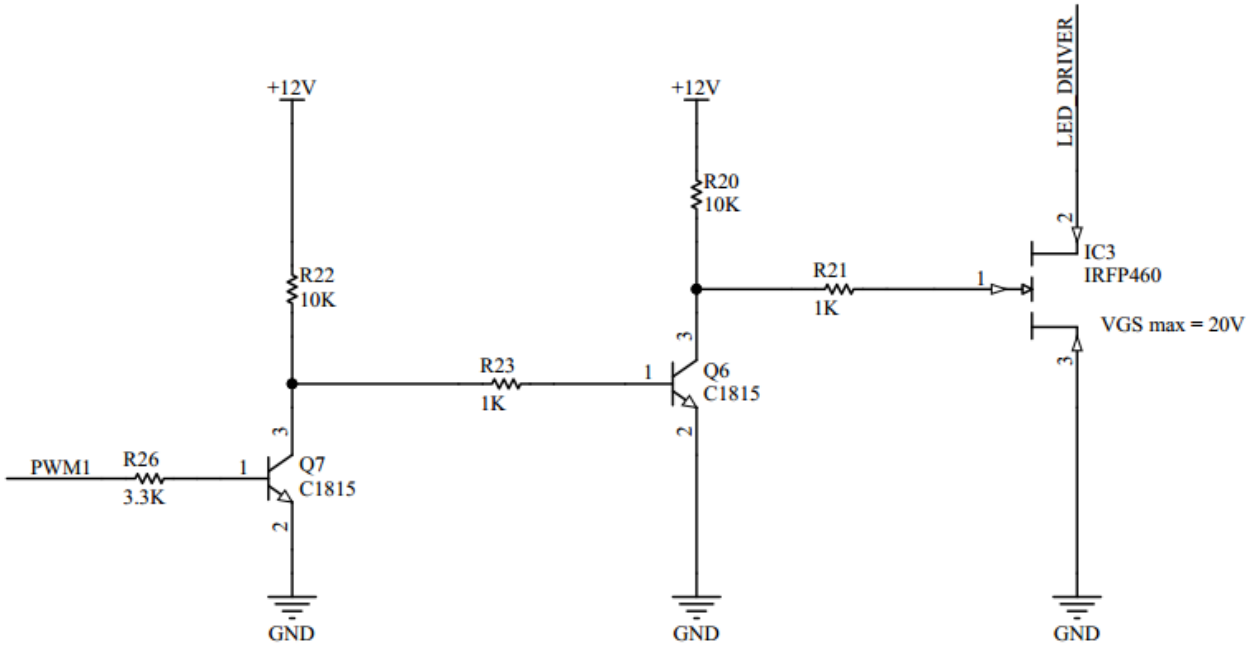
Hình 3.16. Mạch Snubber

Ngoài ra, còn nhiều khối khác chưa sử dụng nên tác giả không cho vào báo cáo. Những khối này được thiết kế khi cần nâng cấp sản phẩm sau này.

3.7. Thiết kế mạch Dimmer đèn

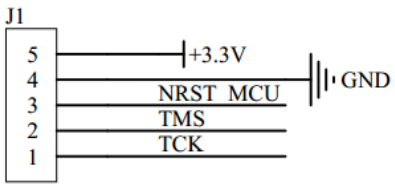
Khối Dimmer đèn là khối điều chỉnh độ sáng tối của đèn chiếu sáng mô hình nông nghiệp, sử dụng khối PWM trong vi điều khiển STM32. Trong thiết kế sử dụng đèn led ánh sáng trắng, điện áp 12VDC. Khối này sử dụng một mosfet IRFP460 và 2 transistor NPN C1815. Mosfet IRFP460 là mosfet công suất chịu được dòng điện và điện áp lớn lên tới 20A, 500V, C1815 là transistor được sử dụng phổ biến trong các ứng dụng điều chế

độ rộng xung PWM. Mạch dimmer đèn led được điều khiển bởi chân PWM1 của vi điều khiển STM32F103C8T6.



3.8. Thiết kế các khối phụ trợ

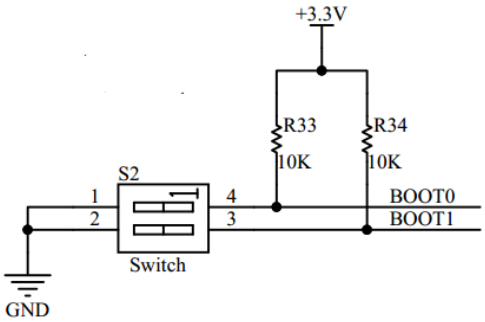
Khối mạch nạp: Vi điều khiển được nạp chương trình bằng mạch nạp STLink Mini version2 với chuẩn SWD. Khi nạp code cho STM32 chỉ cần 4 chân là 3.3V, GND, TMS, TCK. Ngoài ra, STM32 còn được nạp và debug bằng kit STM32F4 DISCOVERY trong môi trường lập trình "IAR Embedded Workbench® for ARM" version 5.3.



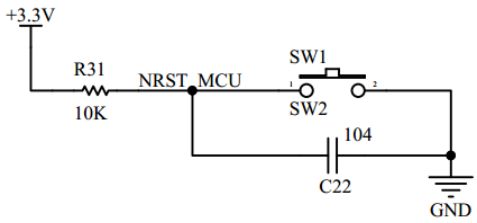
Hình 3.11. Thiết kế khối mạch nạp STLink Mini

Khối boot, reset:

- Khối reset: Dùng để khởi động lại vi điều khiển khi cần thiết
- Khối boot: Dùng để cấu hình khi nạp chương trình cho vi điều khiển. Với lựa chọn mạch nạp ở trên, chân BOOT0 và BOOT1 nối xuống GND.

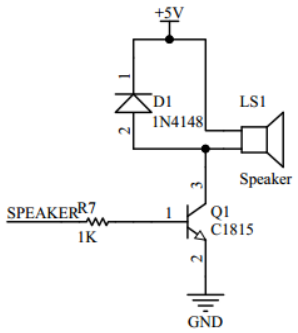


Hình 3.12. Thiết kế khối Boot cho STM32



Hình 3.13. Thiết kế khối Reset cho STM32

Khối còi: Sử dụng còi điện áp 5V, dùng để tạo hiệu ứng khi có kết nối của module wifi ESP8266 với Broker hoặc có lệnh từ điện thoại gửi xuống.



Hình 3.14. Thiết kế còi tạo hiệu ứng âm thanh

Chương 4

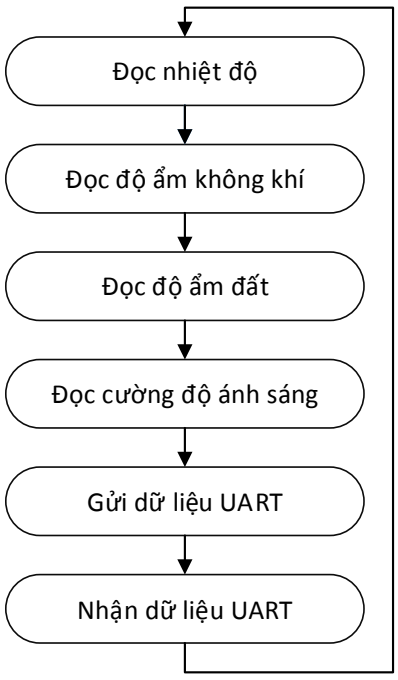
THIẾT KẾ PHẦN MỀM

4.1. Lập trình cho vi điều khiển STM32F103C8T6

STM32F103C8T6 được lập trình bằng phần mềm "IAR Embedded Workbench® for ARM" version 5.3, thư viện lập trình cho vi điều khiển này là CMSIS. Để nạp code và debug mạch, tác giả dùng kit STM32F4 DISCOVERY của STMicroelectronics. Các hướng dẫn nạp và debug được trình bày ở phần phụ lục.

KIT STM32F4 DISCOVERY tích hợp vi điều khiển chính: STM32F407VGT6 32-bit ARM Cotex-M4F. Kit cho phép mạch nạp và debug theo chuẩn SWD và có đầy đủ tính năng của một mạch nạp STLINK/V2.

Vi điều khiển thực hiện liên tục các công việc, các công việc được thực hiện định kỳ và quản lý bằng Timer. Tức là sau những khoảng thời gian xác định mỗi công việc lại được thực hiện.

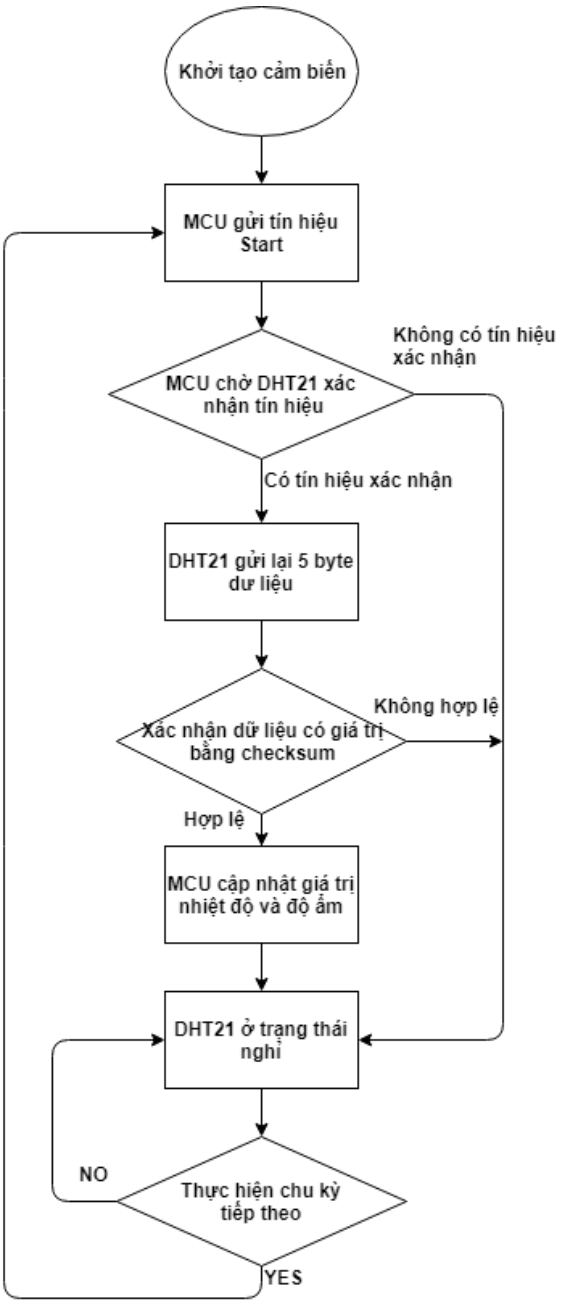


Hình 4.1. Lưu đồ thuật toán cho vi điều khiển STM32

4.1.1. Lập trình đo nhiệt độ, độ ẩm của hệ thống nông nghiệp.

Để có thể giao tiếp với DHT21 theo chuẩn 1 chân vi xử lý thực hiện theo 2 bước:

- Gửi tín hiệu muốn đo (Start) tới chân DHT21, sau đó DHT21 xác nhận lại.
- Khi đã giao tiếp được với DHT21, cảm biến sẽ gửi lại 5 byte dữ liệu và nhiệt độ đo được.

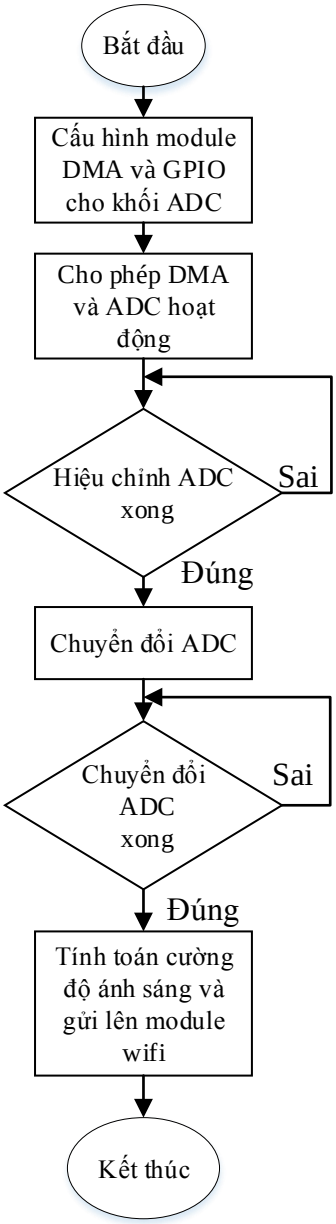


Hình 4.3. Lưu đồ thuật toán đo nhiệt độ, độ ẩm

4.1.2. Lập trình đo cường độ ánh sáng của hệ thống nông nghiệp.

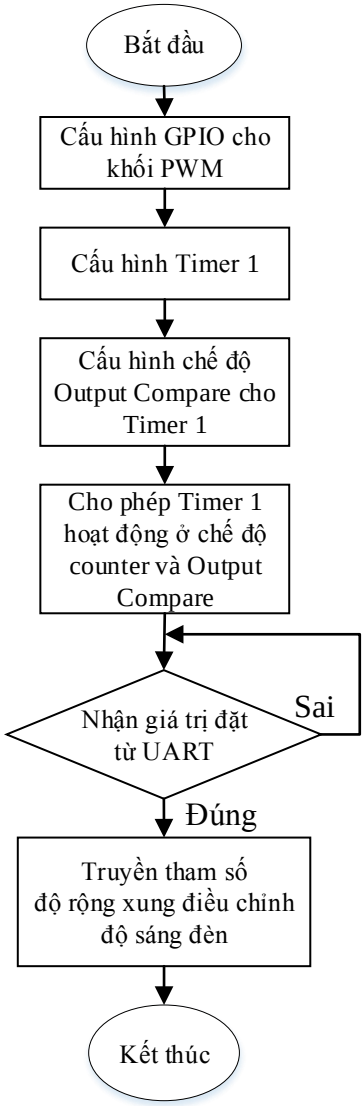
Thực hiện đo cường độ ánh sáng bằng các sử dụng khối ADC và DMA của STM32F103C8T6.

DMA (Direct memory access) là một kỹ thuật chuyển dữ liệu từ bộ nhớ đến ngoại vi hoặc từ ngoại vi đến bộ nhớ mà không yêu cầu đến sự thực thi của CPU, có nghĩa là CPU sẽ hoàn toàn độc lập với ngoại vi được hỗ trợ DMA mà vẫn đảm bảo ngoại vi thực hiện công việc được giao, tùy vào từng loại ngoại vi mà DMA có sự hỗ trợ khác nhau.



Hình 4.4. Lưu đồ thuật toán đo cường độ ánh sáng

4.1.3. Lập trình điều chỉnh độ sáng đèn bằng mạch Dimmer



Hình 4.5. Lưu đồ thuật toán điều khiển độ sáng đèn bằng mạch dimmer

4.1.4. Chương trình gửi dữ liệu UART lên module wifi

Kết thúc mỗi quá trình đo nhiệt độ, độ ẩm, cường độ ánh sáng, độ ẩm đất vi điều khiển sẽ gửi ngay giá trị đo lên module wifi qua giao tiếp UART.

Cấu trúc gói tin gửi dữ liệu UART:

Start	Command	Data	Stop

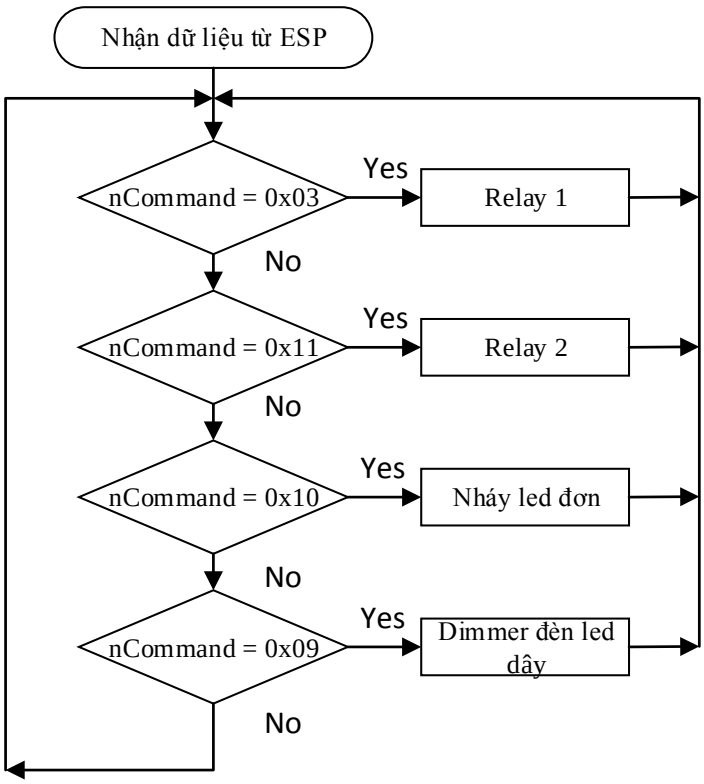
- Start Byte: Là Byte bắt đầu của khung truyền, độ dài 1 byte
- Command Byte: Có độ dài 1 byte, chứa mã lệnh điều khiển
- Data Byte: Dữ liệu gửi đi, có độ dài 1 byte

- Stop Byte: Là Byte cuối của khung truyền, độ dài 1 byte

Start Byte giúp bên nhận biết được byte đầu tiên của khung truyền. Command Byte cho phép bên nhận xác định gói tin áp dụng cho đối tượng nào nhiệt độ, độ pH, cường độ ánh sáng, hay mức nước. Data Byte chứa kết quả đo của các đối tượng trên. Stop Byte giúp bên nhận xác định đã kết thúc khung truyền. Sau khi nhận đủ các byte của khung truyền, bên nhận sẽ xử lý các công việc tương ứng.

4.1.5. Chương trình nhận dữ liệu UART từ module wifi

Vi điều khiển STM32F103C8T6 giao tiếp với Module wifi ESP8266 qua giao tiếp UART. Việc xử lý sự kiện nhận dữ liệu UART là với từng mã lệnh nhận được vi điều khiển sẽ tác động đến một thiết bị chấp hành. Các thiết bị chấp hành bao gồm rơ le để bật tắt các thiết bị, mạch dimmer điều chỉnh độ sáng của hệ thống nông nghiệp . Ngoài ra, trong thiết kế còn sử dụng đèn led đơn và còi để tạo hiệu ứng báo trạng thái của hệ thống.



Hình 4.6. Lưu đồ thuật toán nhận dữ liệu UART của vi điều khiển

Đối với việc xử lý các lệnh từ module wifi gửi xuống, với từng mã lệnh vi điều khiển sẽ thực hiện một công việc tương ứng.

- Relay 1, Relay 2,: mỗi lần nhận được lệnh thì sẽ thay đổi trạng thái bật/tắt của relay 1, relay 2.
- Nháy led đơn: Bật/tắt đèn led 5 lần khi module wifi kết nối thành công đến broker.
- Dimmer đèn led dây: Nhận lệnh để điều chỉnh độ sáng tối đèn hệ thống, dữ liệu ở Data Byte

Cấu trúc gói tin nhận UART:

Start	Length	Address	Command	Data	Stop

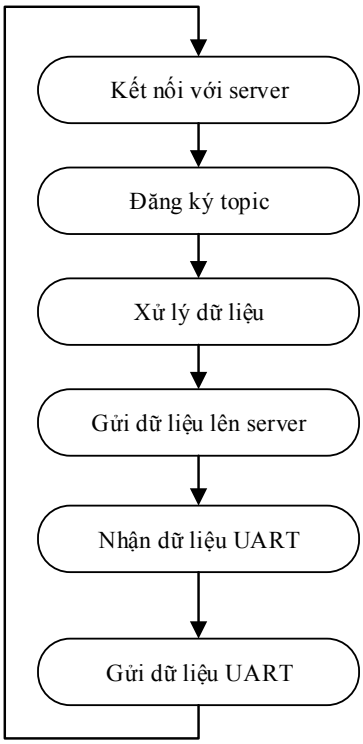
- Start Byte: Là Byte bắt đầu của khung truyền, độ dài 1 byte
- Length Byte: Có độ dài 2 byte, quy định tổng độ dài của Address, Command, Data
- Address Byte: Có độ dài 1 byte, địa chỉ cất gói tin
- Command Byte: Có độ dài một byte, chứa mã lệnh điều khiển
- Data Byte: Dữ liệu nhận được
- Stop Byte: Là Byte cuối của khung truyền, độ dài 1 byte

Cơ chế bóc tách gói tin UART nhận được:

Thực hiện theo cơ chế bộ đệm quay vòng, để sử dụng được bộ đệm quay vòng cần khai báo hai bộ đệm độc lập (có độ dài 255 byte). Bộ đệm thứ nhất là bộ đệm chỉ dùng để nhận dữ liệu trong ngắt UART. Bộ đệm thứ 2 dùng copy dữ liệu từ bộ đệm thứ nhất, và bóc tách gói tin. Khi có chênh lệch về chỉ số (số phần tử) của hai bộ đệm, thì thực hiện sao chép dữ liệu từ bộ đệm thứ nhất sang bộ đệm thứ 2. Tham chiếu đến các byte của bộ đệm thứ 2 bằng cách tham chiếu đến từng phần tử của mảng.

4.2. Lập trình cho module Wifi ESP826

Module wifi ESP8266 thực hiện gửi các gói tin lên Broker, đăng ký các gói tin từ Broker, xử lý các sự kiện sự kiện truyền/nhận dữ liệu UART với vi điều khiển. Ngoài ra, module wifi còn có nhiệm vụ xây dựng giao diện của ứng dụng IoT Manager. Phần mềm Arduino IDE, vesion 1.85 được sử dụng để lập trình cho ESP8266.



Hình 4.7. Lưu đồ thuật toán của ESP8266

ESP8266 kết nối được tới mạng wifi nhờ bước cài đặt tên wifi (SSID) và mật khẩu (password) trong mã nguồn. Sau đó, ESP8266 sẽ kết nối đến Broker. Nếu kết nối thành công với Broker thì module wifi sẽ đăng ký các Topic để nhận các gói tin từ Broker. Sau khi nhận được gói tin, ESP sẽ tiến hành xử lý gói tin và thực hiện gửi dữ liệu xuống vi điều khiển hoặc xử lý các tác vụ trên module. ESP sẽ gửi lại trạng thái công việc vừa thực hiện lên Broker. ESP còn xử lý các tác vụ nhận dữ liệu UART từ vi điều khiển và gửi dữ liệu UART cho vi điều khiển. Các hàm được sử dụng khi lập trình như sau:

- client.connect(): ESP kết nối đến Broker
- client.subscribe():ESP đăng ký với Broker các Topic muốn nhận dữ liệu
- client.set_callback(callback): Xử lý các gói tin nhận được từ Broker
- client.publish(): Gửi dữ liệu lên các Topic của Broker
- ReadUart(): Lấy dữ liệu UART do vi điều khiển gửi lên
- WriteUart(): Gửi dữ liệu UART xuông vi điều khiển

4.2.1 Truyền dữ liệu UART xuống STM32F103C8T6

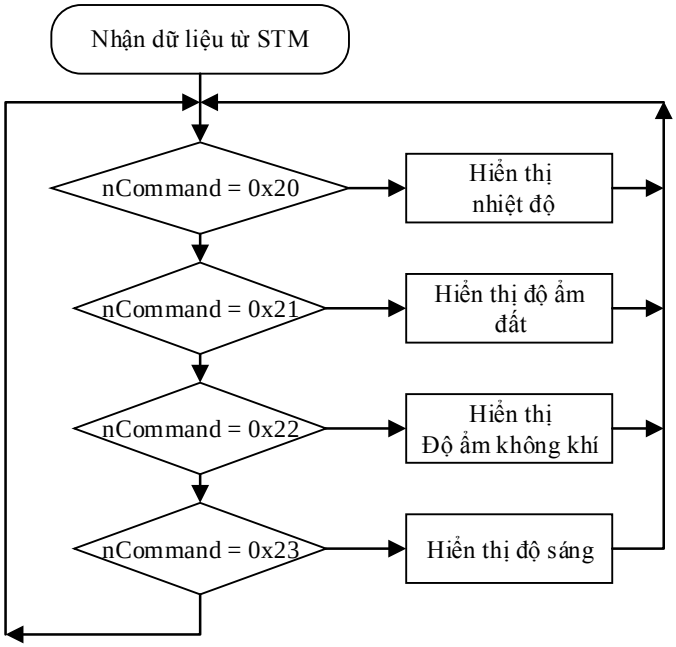
Mỗi gói tin gửi xuống STM32 được quy định theo các byte như sau:

Start	Length	Address	Command	Data	Stop

Trong đó, Start Byte và Stop Byte có độ dài 1 byte, được sử dụng để phục vụ bóc tách gói tin .Length có độ dài 2 byte, chứa thông tin về độ dài của các byte Address, Command, Data cộng lại. Data Byte có thể có nhiều hơn 1 byte, tuy nhiên trong phạm vi đồ án tác giả không sử dụng byte Data, các gói tin gửi xuống STM có giá trị Data=0x66, Length = 0x03. Command là byte cần lưu ý vì tương ứng với mỗi Command Byte gửi xuống STM, STM sẽ thực hiện một lệnh tương ứng. Các gói tin được định nghĩa dưới dạng mảng có 6 phần tử. Trong môi trường lập trình Arduino IDE, câu lệnh Serial.write() được sử dụng để gửi dữ liệu ra chân Tx của ESP8266.

4.2.2. Nhận dữ liệu UART từ STM32F103C8T6

Việc xử lý sự kiện nhận dữ liệu UART của module wifi cũng tương tự như STM, với từng mã lệnh, module wifi sẽ gửi thông số đo nhiệt độ, mức nước, độ pH, cường độ ánh sáng lên Broker.



Hình 4.8. Lưu đồ thuật toán nhận dữ liệu UART của ESP8266

Trong môi trường lập trình Arduino IDE, câu lệnh Serial.read() được sử dụng để lấy dữ liệu từ chân Rx ở cổng truyền tin nối tiếp UART của ESP8266.

4.3. Lập trình thay đổi giao diện phần mềm IoT Manager

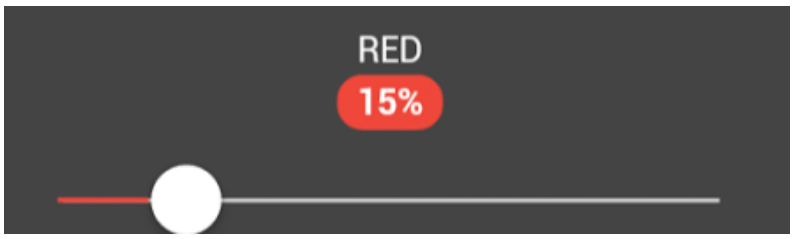
IoT Manager được lập trình viên người Nga Victor Brutskiy phát triển trên cả hai hệ điều hành Android và iOS. IoT Manager cho phép người dùng thay đổi giao diện điều khiển và hiển thị. Việc lập trình thay đổi giao diện được thực hiện đồng thời với lập trình cho module wifi trên phần mềm Arduino IDE 1.8.5.

4.3.1. Các đối tượng cơ bản trên giao diện ứng dụng IoT Manager

- Nút nhấn Toggle thay đổi trạng thái: Dùng để điều khiển bật tắt thiết bị

Hình 4.9. Nút nhấn thay đổi trạng thái

- Thanh trượt Range điều chỉnh giá trị: Dùng để thay đổi linh hoạt giá trị của một đối tượng



Hình 4.10. Thanh trượt thay đổi giá trị

- Đối tượng Badge:Cho phép hiển thị giá trị số

Hình 4.11. Đối tượng hiển thị giá trị số

Trên đây là 3 đối tượng được sử dụng trong đồ án, đây là các đối tượng đơn giản và dễ sử dụng. Ngoài ra, IoT Manager còn hỗ trợ nhiều đối tượng khác cho người lập trình lựa chọn.

4.3.2. Thêm đối tượng trên giao diện ứng dụng IoT Manager

Thêm đối tượng trên giao diện của ứng dụng IoT Manager được thực hiện như sau:

➢ Cấu hình số lượng Widgets, khai báo mảng các thông số của device

```
const  int      nWidgets = 14;
String stat      [nWidgets];
String sTopic    [nWidgets];
String color     [nWidgets];
String style     [nWidgets];
String badge     [nWidgets];
String widget    [nWidgets];
String descr     [nWidgets];
String page      [nWidgets];
String pageId    [nWidgets];
String thing_config[nWidgets];
String id        [nWidgets];
int    pin       [nWidgets];
int    defaultVal  [nWidgets];
bool   inverted    [nWidgets];
```

- Id: ID của đối tượng hiển thị

- Page: Định nghĩa tên trang mà đối tượng chứa trong đó
- PageID: Định nghĩa ID của trang
- Descr: Tên hiển thị của đối tượng trên ứng dụng
- Widget: Loại biểu tượng điều khiển
- Pin: Khai báo chân GPIO của ESP8266 làm chân chức năng (nếu có).
- defaultVal: Giá trị logic mặc định ("1" hoặc "0") hoặc giá trị nguyên dương
- inverted: Định nghĩa True cho phép đảo trạng thái của đối tượng
- sTopic: Chuỗi prefix+deviceID+descr
- stat: Là trạng thái của đối tượng
- thing_config: là chuỗi ký tự mang thông tin của đối tượng
- style: Màu hiển thị của đối tượng thanh trượt Range
- color: Là màu hiển thị của đối tượng nút nhấn toggle trên ứng dụng
- badge: Là mẫu hiển thị dạng số trên ứng dụng

➢ Khởi tạo các thông số ban đầu của đối tượng

➢ Khai báo chuỗi thing_config[] để phục vụ cho việc gửi dữ liệu khởi tạo lên MQTT Broker:

Cấu trúc của một chuỗi thing_config[] chung cho các đối tượng:

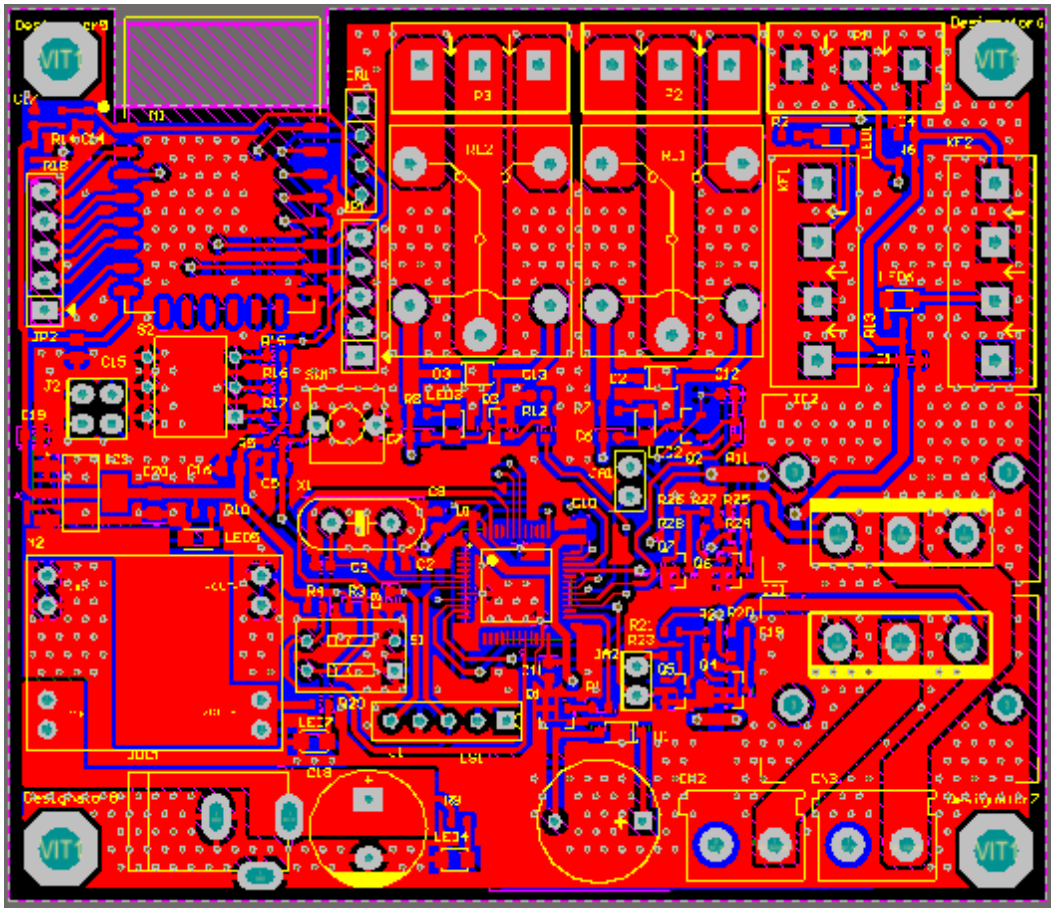
thing_config[0] = "{\"id\":\"" + id[] + "\",\"page\":\"" + page[] + "\",\"descr\":\"" + descr[] + "\",\"widget\":\"" + widget[] + "\",\"topic\":\"" + sTopic[]+…}"

Ngoài ra, đối tượng còn có thêm style[], badge[], color[] ở cuối chuỗi tùy theo từng đối tượng cụ thể.

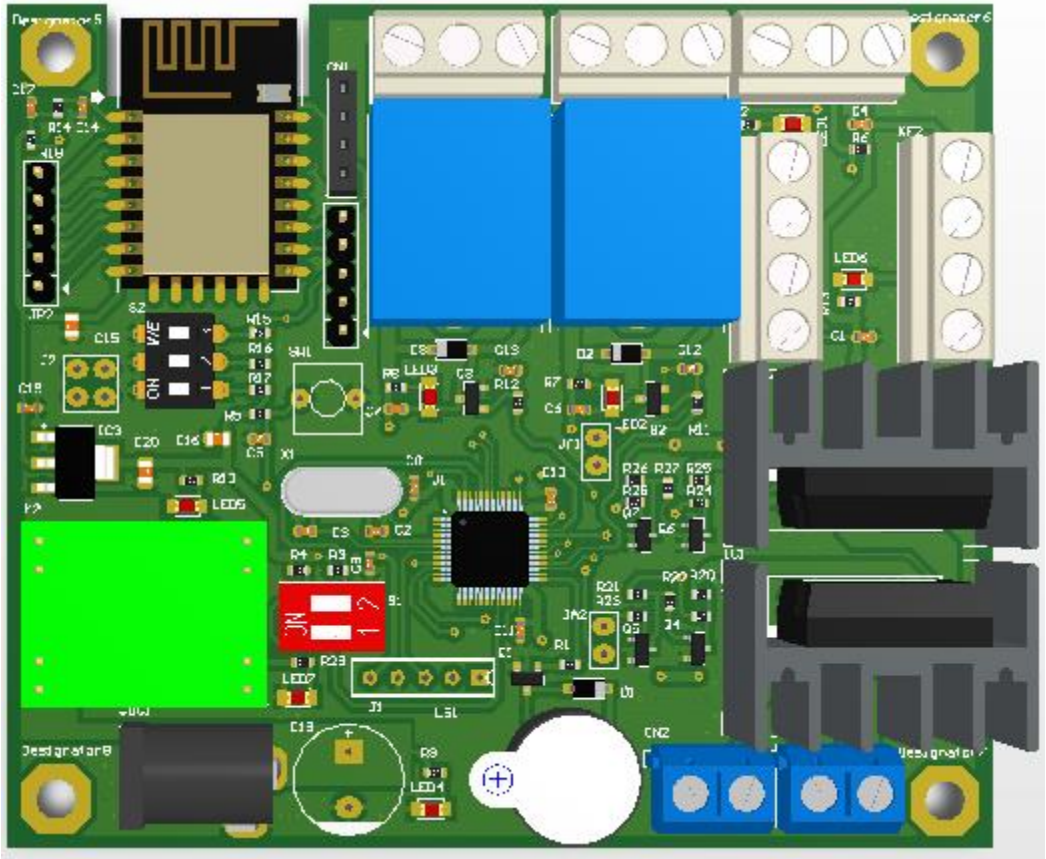
Chương 5
KẾT QUẢ THỰC NGHIỆM

5.1. Kết quả thiết kế trên phần mềm Altium

Mạch mạch nguyên lý và mạch in PCB của sản phẩm được thiết kế trên phần mềm Altium Designer version 15.0.14. Mạch in PCB được thiết kế 2 lớp, gia công tại Tuấn Cường Technology.



Hình 5.1. Hình vẽ 2D của mạch in PCB trên Atium

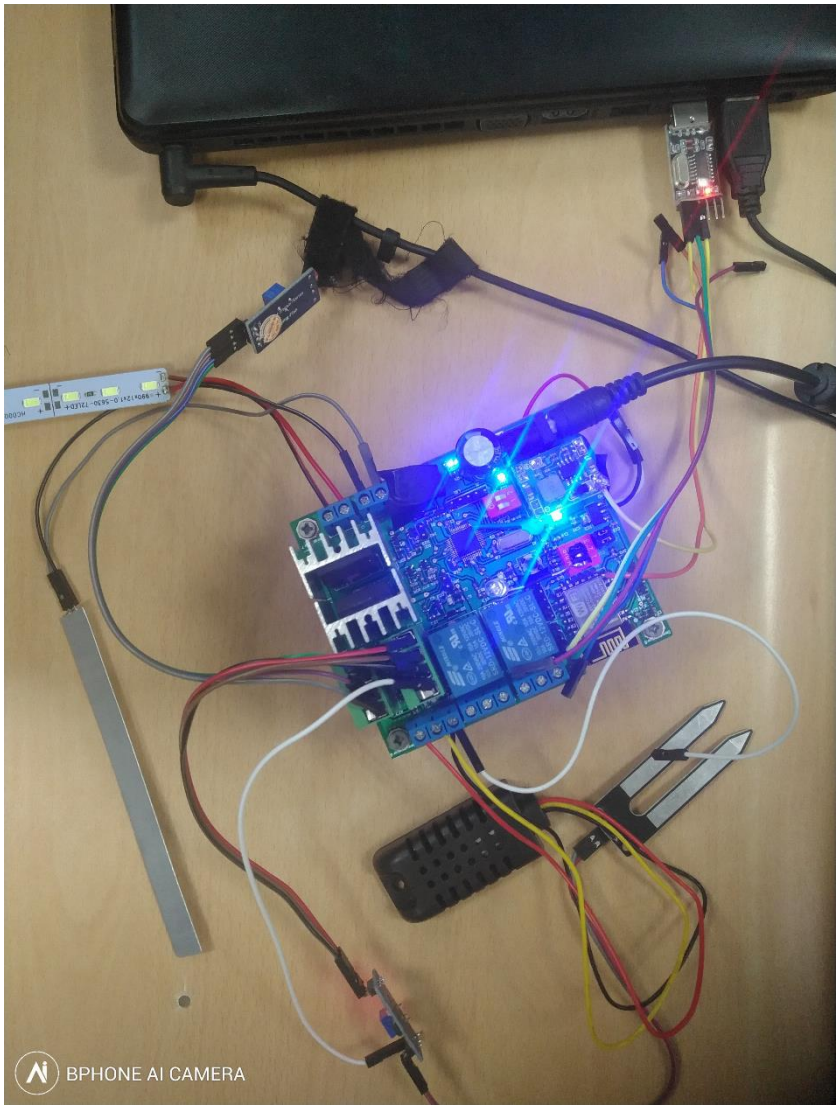


Hình 5.2. Hình vẽ 3D của mạch in PCB trên Atium

5.2. Kết quả vận hành thực tế

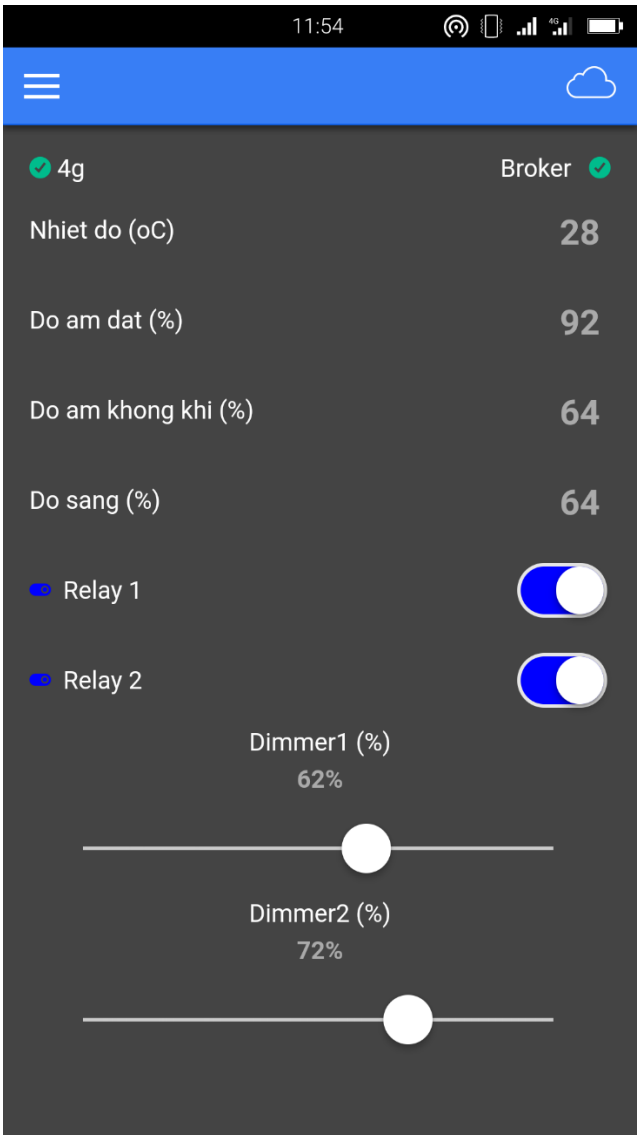
- Hệ thống hoạt động khá ổn định
- Đo được các thông số nhiệt độ, độ ẩm ,độ ẩm đất, cường độ ánh sáng
- Hiển thị được thông số của hệ thống nông nghiệp lên giao diện ứng dụng IoT Manager
- Điều khiển được các thiết bị Relay, mạch dimmer đèn chiếu sáng qua giao diện của ứng dụng IoT Manager

Do thời gian và khả năng còn hạn chế, các kết quả đo trong thực nghiệm còn có sai số. Bên cạnh đó, giao diện ứng dụng còn đơn giản, chưa bắt mắt. Với khả năng tài chính có hạn, tác giả chưa trang bị hết các thiết bị như các loại máy bơm, máy sưởi cho hệ thống. Tuy nhiên, tác giả đã dùng đèn led để khắc phục vấn đề trên, khi thiết bị được bật đèn led hiển thị sẽ sáng và ngược lại. Dưới đây là một số hình ảnh của sản phẩm hệ thống điều khiển nông nghiệp của đồ án.



Hình 5.3. Sản phẩm hệ thống điều khiển nông nghiệp

Giao diện hiển thị trên màn hình điện thoại:



Hình 5.4. Giao diện điều khiển và giám sát.

KẾT LUẬN

Quá trình thực hiện đồ án: "**Nghiên cứu thiết kế hệ thống điều khiển giống cây trồng trong nông nghiệp**", em đã học hỏi được nhiều điều cũng như tổng hợp lại nhiều kiến thức đã được học.

Em xin chân thành cảm ơn sự chỉ bảo nhiệt tình của thầy giáo **TS. Nguyễn Huy Phương**, các thầy cô giáo trong bộ môn Tự động hoá công nghiệp đã giúp em hoàn thành bản đồ án này. Em đã thu được những kết quả cụ thể như sau:

Đã thực hiện:

- Thiết kế phần cứng và phần mềm cho hệ thống điều khiển giống cây trồng trong nông nghiệp.
- Các module đo và thiết bị chấp hành hoạt động ổn định, tương đối chính xác.
- Thay đổi được giao diện của ứng dụng IoT Manager có sẵn trên Android.
- Điều khiển và giám sát hệ thống nông nghiệp từ xa qua ứng dụng IoT Manager.

Định hướng mở rộng:

- Cải thiện tính năng của mạch để phù hợp với điều kiện sinh trưởng của từng loại cây riêng biệt.
- Đóng hộp bo mạch thành một sản phẩm hoàn chỉnh.
- Lập trình một ứng dụng mới thay cho việc sử dụng một ứng dụng có sẵn.
- Tối ưu tốc độ đáp ứng của hệ thống

Do thời gian có hạn và năng lực bản thân còn hạn chế nên những kết quả của em cần phải cải thiện nhiều, em rất mong được sự chỉ dạy và đóng góp ý kiến của thầy cô và các bạn.

Em xin chân thành cảm ơn.

Hà Nội, ngày 01 tháng 01 năm 2019

Sinh viên thực hiện

Dương Đình Quân

TÀI LIỆU THAM KHẢO

[1] R. Krishnan, *Switched Reluctance Motor Drives*, CRC Press LLC, 2001.

[2] T.J.E. Miller, *Switched Reluctance Motors and Their Control*, Magna Physics, Oxford, 1992.

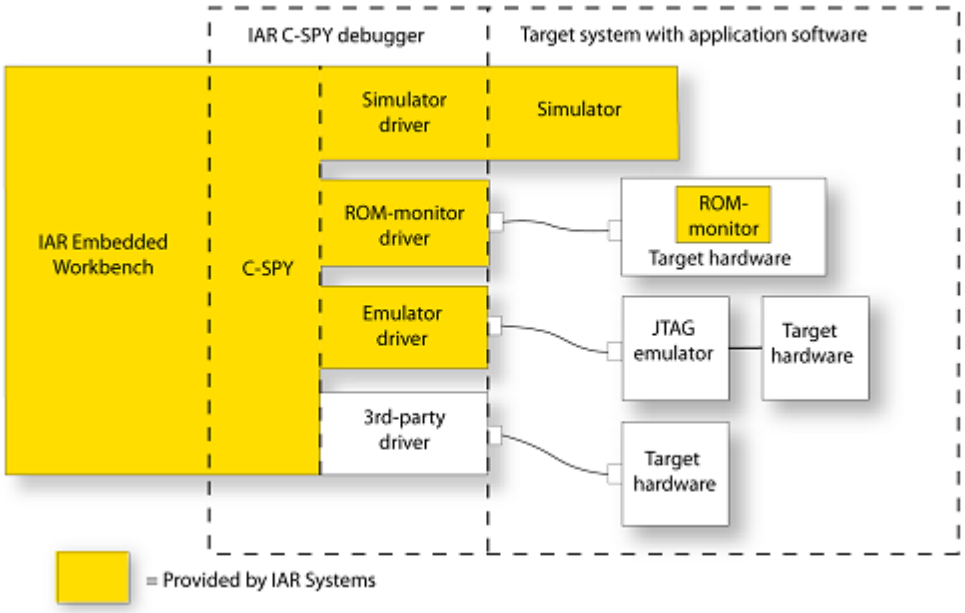
[3] Husain, I. and M. Ehsani, "Torque ripple Minimization in Switched Reluctance Motor Drives by PWM Control", *IEEE Trans. on Power Electronics*, Vol. 11, No. 1, 1996, pp. 83-88.

[4] C. Dufour, J. Bélanger, S. Abourida, V. Lapointe, "FPGA-Based Real-Time Simulation of Finite-Element Analysis Permanent Magnet Synchronous Machine Drives", *Proceedings of the 38th Annual IEEE Power Electronics Specialists Conference (PESC '07),* Orlando, Florida, USA, June 17-21, 2007, pp. 530-537.

PHỤ LỤC

P1. Hướng dẫn gỡ lỗi phần mềm bằng IAR và KIT nạp STM32F4DISCOVERY

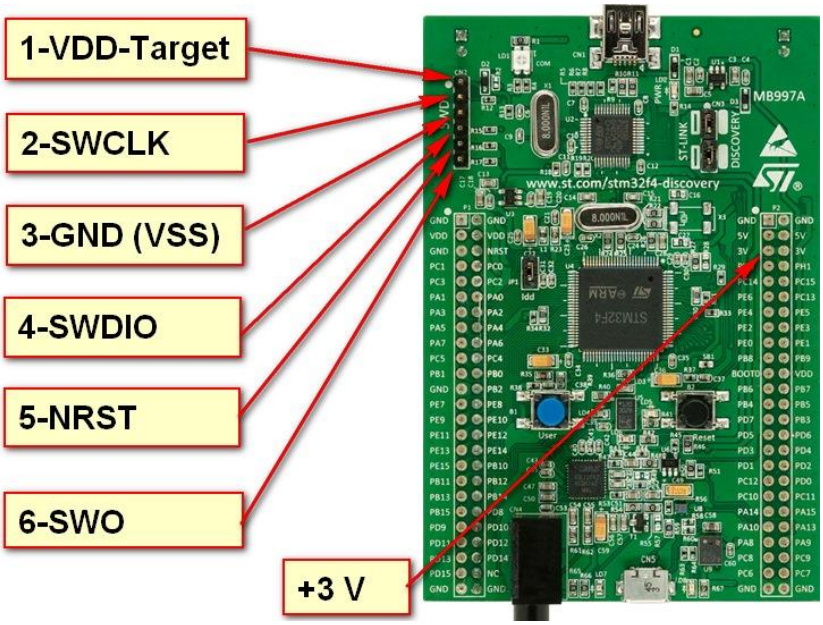
Khi thiết kế phần mềm cho một dự án nào đó. Quá trình gỡ lỗi phần mềm đòi hỏi rất nhiều thời gian và công sức. Do đó để có thể thực hiện nhanh quá trình gỡi lỗi cần có các công cụ đi kèm. Dưới đây em xin giới thiệu phần mềm IAR for Arm và KIT nạp STM32F4 DISCOVERY.



Hình P1.1. Sơ đồ khối của IAR và cơ chế debug

Phần mềm IAR hỗ trợ đọc giá trị của biến, địa chỉ của biến, đọc flash, ROM, chạy từng dòng lệnh nhờ cơ chế truy xuất tất cả dữ liệu từ thanh ghi của vi điều khiển lên máy tính thông qua mạch debug.

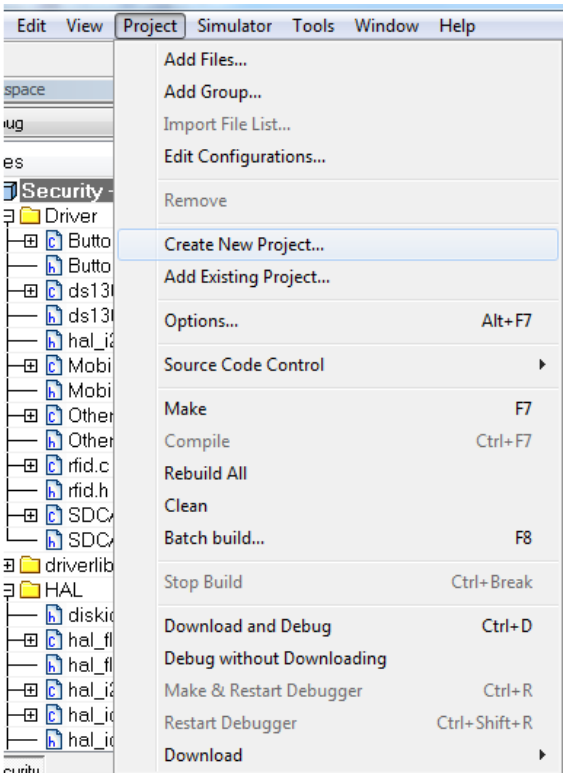
KIT STM32F4 DISCOVERY tích hợp vi điều khiển chính: STM32F407VGT6 32-bit ARM Cotex-M4F. Kit cho phép mạch nạp và debug theo chuẩn SWD và có đầy đủ tính năng của một mạch nạp STLINK/V2.



Hình P1.2. KIT STM32F4 DISCOVERY

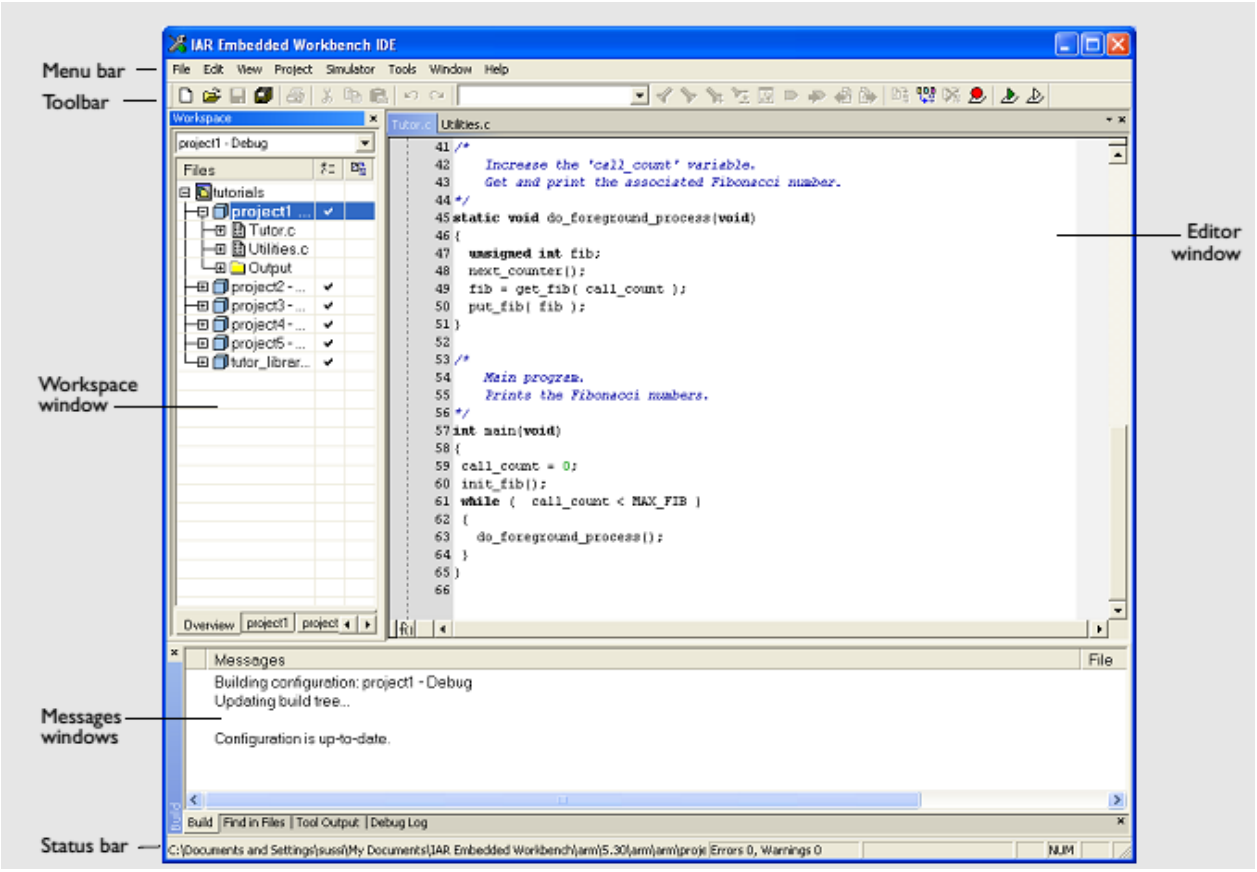
Hướng dẫn sử dụng IAR for Arm version 5.3

Để tạo một project mới kích vào project và chọn Creat New Project :



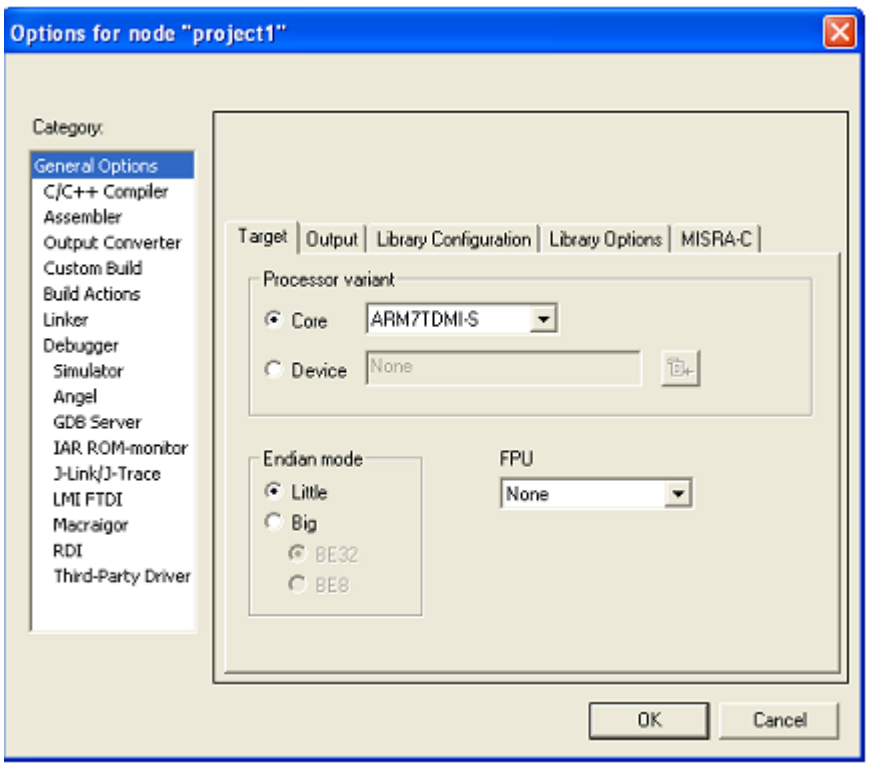
Hình P1.3. Tạo project mới bằng IAR

Sau khi tạo một project mới người lập trình có thể viết code trực tiếp trên cửa sổ lệnh :



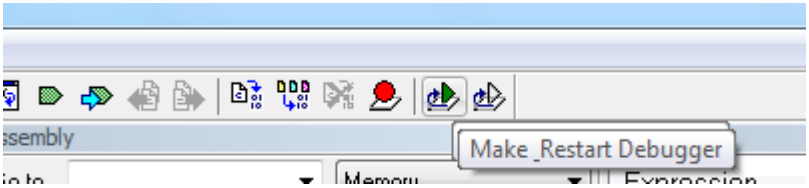
Hình P1.4. Cửa sổ làm việc của IAR

Để có thể nạp được code cho chip và gỡ lỗi trên mạch ta cần chọn đúng loại chíp:



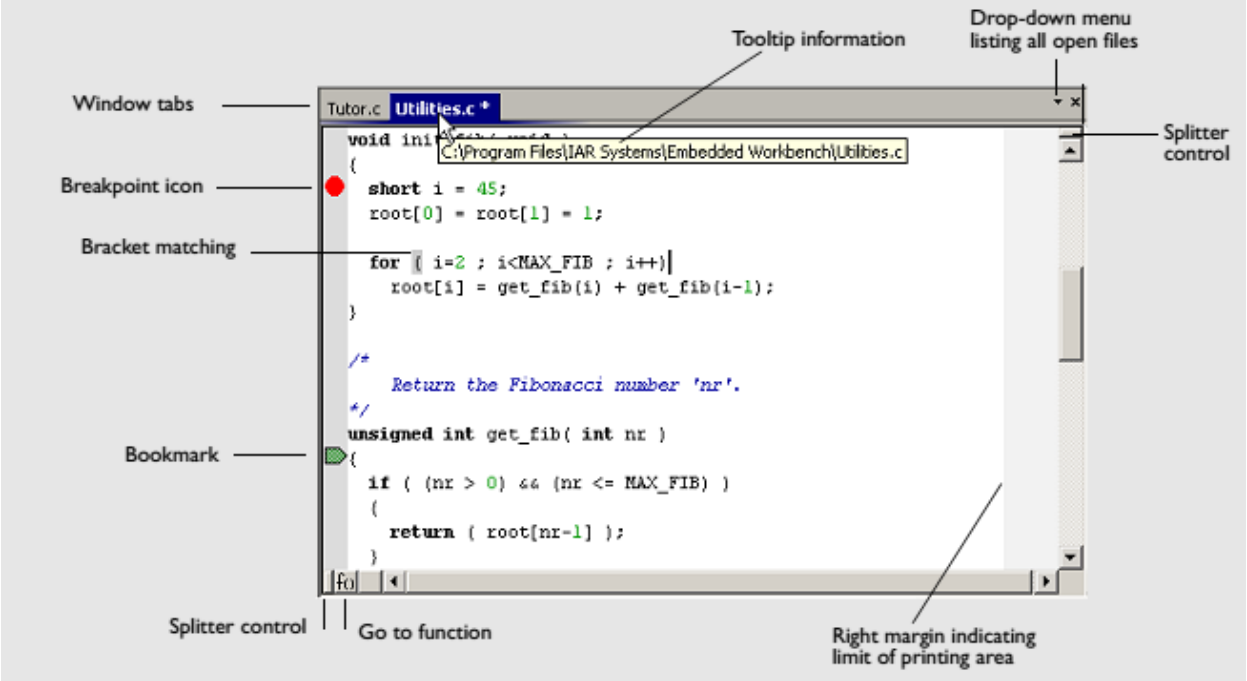
Hình P1.5. Cửa sổ cài đặt loại chip

Kích chuột trái vào biểu tượng Make Restart Debugger để nạp chương trình :



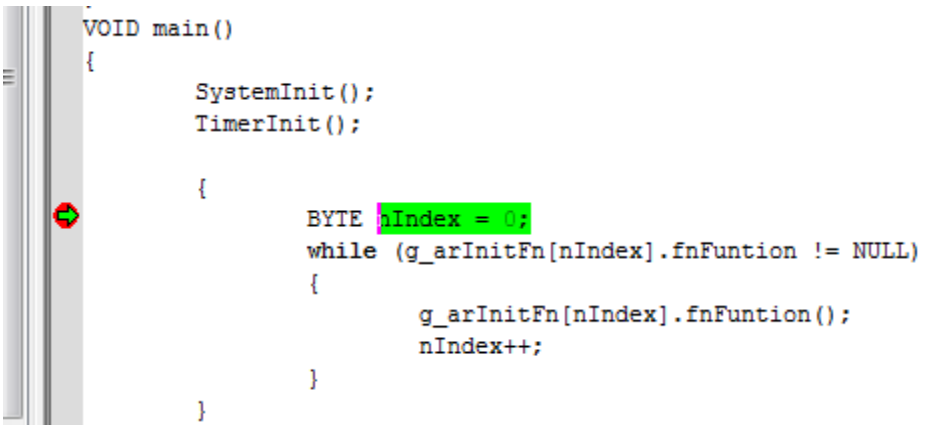
Hình P1.6. Thanh công cụ nạp chương trình

IAR hỗ trợ cơ chế xuất tất cả dữ liệu từ thanh ghi trong chip qua mạch nạp lên máy tính mà người lập trình có thể chạy từng dòng lệnh bằng cách đặt breakpoint :



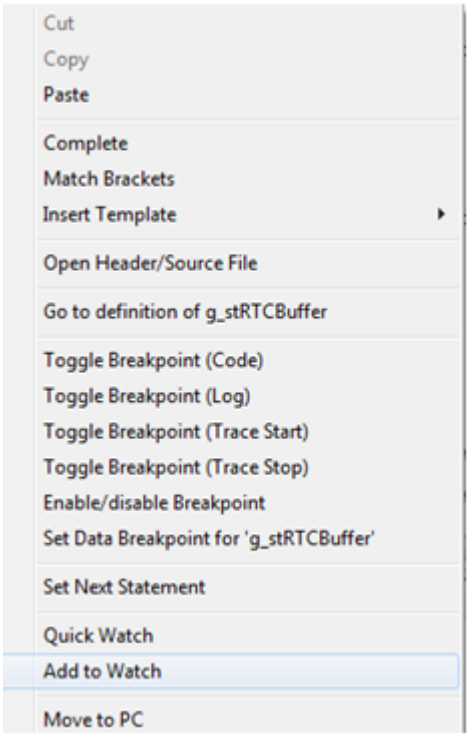
Hình P1.7. Breakpoint trong cửa số lệnh

Để đặt được breakpoint cần nạp chương trình trước và đưa chuột đến dòng lệnh và kích đúp chuột tại đầu dòng lệnh. Chương trình sẽ dừng lại tại những vị trí đặt breakpoint :



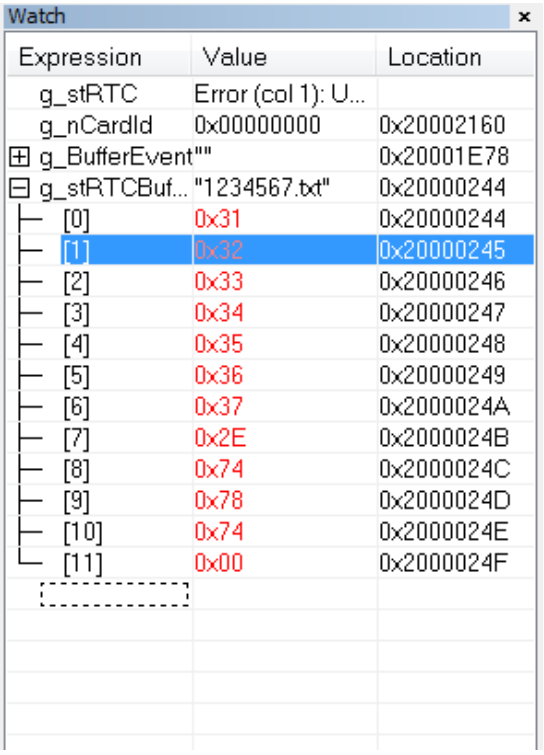
Hình P1.8. Lệnh được chạy đến breakpoint

Khi chương trình dừng lại tại vị trí breakpoint ta có thể xem được giá trị của biến nào đó bằng cách kích chuột phải vào biến đó và chọn Add to Watch :



Hình P1.9. Thao tác để xem giá trị của biến

Một cửa sổ watch xuất hiện ở bên phải màn hình cho phép xem chi tiết giá trị và địa chỉ của biến:



Hình P1.10. Cửa sổ xem giá trị của biến

P2. Code chương trình của đồ án

P2.1. Mã nguồn của vi điều khiển STM32F103C8T6

- Code chương trình đo nhiệt độ,độ ẩm của cảm biến DHT21:

```
VOID DHT21_Init()
{
   SetDirPin(DHT21_PORT, DHT21_PIN, 0);
}

BOOL DHT21_GetData(PDHT21DATA pData)
{
   BYTE arBuffer[5] = { 0 };
   BYTE byChecksum;
   WORD nTick;

   SetDirPin(DHT21_PORT, 0, DHT21_PIN);
   SetPin(DHT21_PORT, 0 , DHT21_PIN);

   DHT_DELAY_US(250);

   SetPin(DHT21_PORT, DHT21_PIN, 0);
   SetDirPin(DHT21_PORT, DHT21_PIN, 0);
   DHT_DELAY_US(15);

   if (GetPin(DHT21_PORT, DHT21_PIN))
      return FALSE;

   nTick = GetTickCount();
   while ((!GetPin(DHT21_PORT, DHT21_PIN)) & DHT_TIMEOUT(nTick));

   DHT_DELAY_US(10);

   if (!GetPin(DHT21_PORT, DHT21_PIN))
      return FALSE;

   nTick = GetTickCount();
   while (GetPin(DHT21_PORT, DHT21_PIN) & DHT_TIMEOUT(nTick));

   for (BYTE nIndex = 0; nIndex < 5; nIndex++)
   {
      for (BYTE nCount = 0; nCount < 8; nCount++)
      {
         nTick = GetTickCount();
         while ((!GetPin(DHT21_PORT, DHT21_PIN)) & DHT_TIMEOUT(nTick));

         DHT_DELAY_US(10);

         if (GetPin(DHT21_PORT, DHT21_PIN))
         {
            arBuffer[nIndex] |= ( 1 << (7 - nCount));
            nTick = GetTickCount();
            while (GetPin(DHT21_PORT, DHT21_PIN) & DHT_TIMEOUT(nTick));
         }
      }
   }
```

```
    byChecksum = (arBuffer[0] + arBuffer[1] + arBuffer[2] + arBuffer[3]);
    if (byChecksum != arBuffer[4])
        return FALSE;

    pData->nTemperature  = ((arBuffer[2] * 256 + arBuffer[3]) / 10);
    pData->nHumidity = ((arBuffer[0] * 256 + arBuffer[1]) / 10);

    SetPin(DHT21_PORT, DHT21_PIN, 0);
    g_Temp = (BYTE)(pData->nTemperature);
    g_Humi = (BYTE)(pData->nHumidity);
    return TRUE;
}
```

- Code chương trình đo cường độ ánh sáng:

```
VOID InitADC()
{
    ADC_InitTypeDef ADC_InitStructure;
    DMA_InitTypeDef DMA_InitStructure;
     GPIO_InitTypeDef GPIO_InitStructure;

    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1 | RCC_APB2Periph_AFIO, ENABLE);

    /* DMA channel1 configuration ----------------------------------------------*/
    DMA_DeInit(DMA1_Channel1);
    DMA_InitStructure.DMA_PeripheralBaseAddr = ADC1_DR_ADD;
    DMA_InitStructure.DMA_MemoryBaseAddr = (DWORD)g_arADC;
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
    DMA_InitStructure.DMA_BufferSize = ADC_CHANEL_NUMBER;
    DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
    DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
    DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
    DMA_InitStructure.DMA_Priority = DMA_Priority_High;
    DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
    DMA_Init(DMA1_Channel1, &DMA_InitStructure);

    /* Enable DMA1 channel1 */
    DMA_Cmd(DMA1_Channel1, ENABLE);

    /* ADC1 configuration ------------------------------------------------------*/
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
    ADC_InitStructure.ADC_ScanConvMode = ENABLE;
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_NbrOfChannel = ADC_CHANEL_NUMBER;
    ADC_Init(ADC1, &ADC_InitStructure);

    /* ADC1 regular channel15 configuration */
    ADC_RegularChannelConfig(ADC1, ADC_Channel_0, 1, ADC_SampleTime_239Cycles5);
```

```
        ADC_RegularChannelConfig(ADC1, ADC_Channel_1, 2, ADC_SampleTime_239Cycles5);

        ADC_DMACmd(ADC1, ENABLE);
        ADC_Cmd(ADC1, ENABLE);              /* Enable ADC1 */

        /* Enable ADC1 reset calibaration register */
        ADC_ResetCalibration(ADC1);
        /* Check the end of ADC1 reset calibration register */
        while (ADC_GetResetCalibrationStatus(ADC1));

        /* Start ADC1 calibaration */
        ADC_StartCalibration(ADC1);
        /* Check the end of ADC1 calibration */
        while (ADC_GetCalibrationStatus(ADC1));
        ADC_SoftwareStartConvCmd(ADC1, ENABLE);    /*  */
         StartLongTimer(3, ReadADCProcess, NULL);
    }

    WORD ReadADC(BYTE nChannel)
    {
       if (nChannel <= 2)
       {
            return g_arADC[nChannel - 1];
       }

       else
          return 0;
    }
    VOID ReadADCProcess(PVOID pDATA)
    {

            g_arADC_0 = ReadADC(1);
            g_arADC_1 = ReadADC(2);
            g_Light = (BYTE)(100-100*g_arADC_1/4096);
            g_Doamdat = (BYTE)100*g_arADC_0/4096;
            if(g_Doamdat<60) SetPin(RELAY1_PORT, RELAY1_PIN, 0);//1
            else SetPin(RELAY1_PORT, 0, RELAY1_PIN);//0
            ADC_DELAY_US(3);
            WriteUartTransmit(UART_PORT_2,0x23,g_Light);
            WriteUartTransmit(UART_PORT_2,0x21,g_Doamdat);
            StartLongTimer(3, ReadADCProcess, NULL);
    }
```

- Code chương trình gửi dữ liệu UART:

```
VOID WriteUartTransmit(BYTE nPort, BYTE Command, BYTE Data)
{
   m_stUartBuffer.nCommand_ = Command;//mã lệnh
   m_stUartBuffer.nData_ = Data;//dữ liệu

   USART_SendData(UART_PORT, SERIAL_START_BYTE); Gửi Start byte
   while (USART_GetFlagStatus(UART_PORT, USART_FLAG_TXE) == RESET);

   USART_SendData(UART_PORT, m_stUartBuffer.nCommand_); Gửi command
   while (USART_GetFlagStatus(UART_PORT, USART_FLAG_TXE) == RESET);

   USART_SendData(UART_PORT,  m_stUartBuffer.nData_); Gửi Data
   while (USART_GetFlagStatus(UART_PORT, USART_FLAG_TXE) == RESET);
```

```
   USART_SendData(UART_PORT, SERIAL_STOP_BYTE); Gửi Stop Byte
   while (USART_GetFlagStatus(UART_PORT, USART_FLAG_TXE) == RESET);
}
```

- Code chương trình điều chỉnh độ sáng đèn bằng mạch dimmer

```
INTERNAL TIM_TimeBaseInitTypeDef  g_stTIMTimeBaseStructure;
INTERNAL TIM_OCInitTypeDef  g_stTIMOCInitStructure;
INTERNAL VOID GPIOPWMConfiguration();
VOID InitPWM()
{
   GPIOPWMConfiguration();
   INTERNAL TIM_BDTRInitTypeDef TIM_BDTRInitStructure;
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM1 | RCC_APB2Periph_AFIO, ENABLE);
         /* TIM1 Configuration ---------------------------------------------------
         TIMxCLK = 24 MHz, Prescaler = 5, TIMx counter clock = 4.8 MHz
         TIMx frequency = TIMxCLK/(TIMx_Period + 1) = 10 000 Hz
         ----------------------------------------------------------------------- */
   g_stTIMTimeBaseStructure.TIM_Prescaler = 2 - 1;
   g_stTIMTimeBaseStructure.TIM_ClockDivision = 0;
   g_stTIMTimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
   g_stTIMTimeBaseStructure.TIM_Period = 480 - 1;
   g_stTIMTimeBaseStructure.TIM_ClockDivision = 0;
   g_stTIMTimeBaseStructure.TIM_RepetitionCounter = 0;
   TIM_TimeBaseInit(TIM1, &g_stTIMTimeBaseStructure);

   g_stTIMOCInitStructure.TIM_OCMode = TIM_OCMode_PWM2; //ok
   g_stTIMOCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
   g_stTIMOCInitStructure.TIM_OCNPolarity = TIM_OCNPolarity_Low;
   g_stTIMOCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
   g_stTIMOCInitStructure.TIM_OutputNState = TIM_OCNIdleState_Reset;
   g_stTIMOCInitStructure.TIM_OCIdleState = TIM_OCIdleState_Set;
   g_stTIMOCInitStructure.TIM_OCNIdleState = TIM_OCIdleState_Set;
   g_stTIMOCInitStructure.TIM_Pulse = 0;

   TIM_OC1Init(TIM1, &g_stTIMOCInitStructure);
   TIM_OC1PreloadConfig(TIM1, TIM_OCPreload_Enable);
   TIM_BDTRInitStructure.TIM_OSSIState = TIM_OSSIState_Disable; //The Off-State selection used in Run
mode.
   TIM_BDTRInitStructure.TIM_OSSRState = TIM_OSSIState_Disable;
   TIM_BDTRInitStructure.TIM_LOCKLevel = TIM_LOCKLevel_OFF;
   TIM_BDTRInitStructure.TIM_Break = TIM_Break_Disable;// Break input
   TIM_BDTRInitStructure.TIM_BreakPolarity = TIM_BreakPolarity_High; //
   TIM_BDTRInitStructure.TIM_AutomaticOutput = TIM_AutomaticOutput_Enable; // TIM Automatic Output
feature is enabled
   TIM_BDTRConfig(TIM1, &TIM_BDTRInitStructure);
   TIM_ARRPreloadConfig(TIM1, ENABLE);
   TIM_Cmd(TIM1, ENABLE);
   TIM_CtrlPWMOutputs(TIM1, ENABLE);
   SetDutyCycle(PWM_CHANNEL_1, 65); //50%
}
// InPut %
VOID SetDutyCycle(BYTE nChannel, WORD nDutyCycle)
{
   WORD nPWM_CCR;
   nPWM_CCR =  nDutyCycle*480/100;
         switch (nChannel)
```

```
        {
        case PWM_CHANNEL_1:
                TIM1->CCR1 = nPWM_CCR;
                break;
        case PWM_CHANNEL_2:
                TIM1->CCR2 = nPWM_CCR;
                break;
    case PWM_CHANNEL_3:
    TIM1->CCR3 = nPWM_CCR;
    break;
    case PWM_CHANNEL_4:
    TIM1->CCR4 = nPWM_CCR;
    break;
  default:
    break;
        }
}
INTERNAL VOID GPIOPWMConfiguration()
{
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

- Code chương trình nhận dữ liệu UART:

```
    RESULT ProcessUartEvents()
    {
      if (g_nProcessIndex != g_nBufferIndex)
      {
        g_pReceivePackage[g_nPackageIndex] = g_pReceiveBuffer[g_nProcessIndex];
        if (g_nPackageIndex == STOP_BYTE_INDEX(g_pReceivePackage))
        {
          if (g_pReceivePackage[g_nPackageIndex] == SERIAL_STOP_BYTE)
          {
            if (fnUartCallBack[UART_DATA_EVENT] != NULL)
            {
              g_stUartBuffer.nTypeBuffer = PAYLOAD_TYPE(g_pReceivePackage);;
              g_stUartBuffer.nLength = PAYLOAD_LENGTH(g_pReceivePackage) - 2;
              g_stUartBuffer.nAddr = g_pReceivePackage[PAYLOAD_OFFSET(g_pReceivePackage)];
              g_stUartBuffer.nCommand = g_pReceivePackage[PAYLOAD_OFFSET(g_pReceivePackage) +
    1];
              g_stUartBuffer.pData = &g_pReceivePackage[PAYLOAD_OFFSET(g_pReceivePackage) + 2];

              fnUartCallBack[UART_DATA_EVENT](&g_stUartBuffer);
              BuzzSet(3);
              switch (g_stUartBuffer.nCommand) {
                    case RELAY_2:
                        if(nI2%2) SetIOState(RELAY2_PORT, RELAY2_PIN,0);
                        else SetIOState(RELAY2_PORT, RELAY2_PIN,1);
                        nI2++;
                        if(nI2>1000)  nI2=1;
                        g_stUartBuffer.nCommand=0;
                       break;

                    case RELAY_1:
```

```
                                if(nI1%2) SetIOState(RELAY1_PORT, RELAY1_PIN,0);
                                else SetIOState(RELAY1_PORT, RELAY1_PIN,1);
                                nI1++;
                                if(nI1>1000)    nI1=1;
                                g_stUartBuffer.nCommand=0;
                              break;

                            case DIMMER_LED:
                               nDimmer= (UINT)g_stUartBuffer.pData;
                               if (nDimmer>=100)
                               {nDimmer =100;
                               SetDutyCycle(PWM_CHANNEL_1,nDimmer );
                               }
                               else SetDutyCycle(PWM_CHANNEL_1,nDimmer );
                                break;
                            default:
                              break;
                           }
                    }
                  }
                  g_nPackageIndex = 0;
               }
               else
                 if (((g_nPackageIndex == 0) && (g_pReceiveBuffer[g_nProcessIndex] == SERIAL_START_BYTE))
          || (g_nPackageIndex > 0))
               {
                  g_nPackageIndex++;
                  if (g_nPackageIndex >= UART_BUFFER_SIZE)
                     g_nPackageIndex = 0;
               }
               g_nProcessIndex++;
               if (g_nProcessIndex >= UART_BUFFER_SIZE)
                  g_nProcessIndex = 0;
               return TRUE;
            }
            return FALSE;
         }
```

P2.2. Mã nguồn của module wifi ESP8266

- Code chương trình gửi dữ liệu UART xuống vi điều khiển:

```
//Định nghĩa các gói tin dưới dạng mảng 6 phần tử
unsigned char RelayNumBer_1[6]= {0xAA,0x03,0x01,0x03,0x66,0x55};//ok
unsigned char RelayNumBer_2[6]= {0xAA,0x03,0x01,0x11,0x66,0x55};//ok
unsigned char g_BlinkLedSuccess[6]= {0xAA,0x03,0x01,0x10,0x66,0x55};
unsigned char g_DimmerLed[6]= {0xAA,0x03,0x01,0x09,0x66,0x55}; //Hàm gửi gói tin
void WriteUart( unsigned char* pData, unsigned char nLen)
{
   for (unsigned char nIndex = 0; nIndex < nLen; nIndex++)
   {
         Serial.write(pData[nIndex]);
   }
```

- Code chương trình nhận dữ liệu UART từ vi điều khiển:

```
if (Serial.available() > 0) {
   Startbyte=Serial.read();
```

```
            if (Startbyte == 0xAA)
{
                                Commandbyte=Serial.read();
                                Databyte=Serial.read();
                                Stopbyte=Serial.read();
            if (Stopbyte == 0x55)
            {
            if (Commandbyte== 0x20)
              {
                        Nhietdo = Databyte;
                        int x1 = int (Nhietdo);
                        val1 = "{\"status\":" + String(x1) + "}";
                        client.publish(sTopic[9] + "/status", val1);
              }
            if (Commandbyte== 0x21)
              {
                        Khoangcach = Databyte;
                        int x2 = int (Khoangcach);
                        val2 = "{\"status\":" + String(x2) + "}";
                        client.publish(sTopic[10] + "/status", val2);
              }
            if (Commandbyte== 0x22)
              {
                        pH = Databyte;
                        int x3 = int (pH);
                        val3 = "{\"status\":" + String(x3) + "}";
                        client.publish(sTopic[11] + "/status", val3);
              }
            }
}
            Startbyte =0x00;
            Commandbyte =0x00;
            Databyte =0x00;
            Stopbyte =0x00;
    }
//////Trong hàm loop() của Arduino
if (client.connected()) {             //Khi module wifi đã kết nối với Broker
      newtime = millis();
      if (newtime - oldtime > 7000) { // 7 sec
        //Serial.print("Time: ");
      //  Serial.println(val);
        int x = analogRead(pin[4]);
        val = "{\"status\":" + String(x) + "}";
        client.publish(sTopic[4] + "/status", val
        client.publish(sTopic[9] + "/status", val1);
        client.publish(sTopic[10] + "/status", val2);
        client.publish(sTopic[11] + "/status", val3);
        oldtime = newtime;
      }
      client.loop();
    }
```

P2.3. Mã nguồn liên quan đến ứng dụng IoT Manager

- Config số lượng Widgets, khai báo mảng các thông số của một đối tượng:

```
const int nWidgets = 12;
String stat       [nWidgets];
```

```
String sTopic      [nWidgets];
String color       [nWidgets];
String style       [nWidgets];
String badge       [nWidgets];
String widget      [nWidgets];
String descr       [nWidgets];
String page        [nWidgets];
String pageId      [nWidgets];
String thing_config[nWidgets];
String id          [nWidgets];
int    pin         [nWidgets];
int    defaultVal  [nWidgets];
bool   inverted    [nWidgets];
```

- Khởi tạo các thông số ban đầu của đối tượng: Dưới đây em chỉ xét 2 đối tượng

```
void initVar()
 {
  id    [1] = "1";
  page  [1] = "Kitchen";
  pageId[1] = 1;
  descr [1] = "RelayNumber1";
  widget[1] = "toggle";// Đối tượng là một nút đổi trạng thái
  defaultVal[1] = 1;
  inverted[1] = true;
  sTopic[1]   = prefix + "/" + deviceID + "/RelayNumber1";
  color[1]   = "\"color\":\"blue\"";

  id    [5] = "5";
  page  [5] = "Kitchen";
  pageId[5] = 1;
  descr [5] = "Dimmer";// Đối tượng là một thanh trượt thay đổi giá trị
  widget[5] = "range";
  defaultVal[5] = 1023;                    // defaultVal status 0%, inverted
  inverted[5] = true;
  sTopic[5]   = prefix + "/" + deviceID + "/dim-light";
  style[5]   = "\"style\":\"range-calm\"";
  badge[5]   = "\"badge\":\"badge-assertive\"";
……
}
```

- Khai báo chuỗi thing_config[] để phục vụ cho việc gửi dữ liệu khởi tạo lên MQTT Broker:

```
thing_config[1] = "{\"id\":\"" + id[1] + "\",\"page\":\"" + page[1] + "\",\"descr\":\"" + descr[1] + "\",\"widget\":\"" + widget[1] + "\",\"topic\":\"" + sTopic[1] + "\"," + color[1] + "}"; // Chuỗi cấu hình cho nút nhấn thay đổi trạng thái
thing_config[5] = "{\"id\":\"" + id[5] + "\",\"page\":\"" + page[5] + "\",\"descr\":\"" + descr[5] + "\",\"widget\":\"" + widget[5] + "\",\"topic\":\"" + sTopic[5] + "\"," + style[5] + "," + badge[5] + ",\"leftIcon\":\"ion-ios-rainy-outline\",\"rightIcon\":\"ion-ios-rainy\"}"; // Dimmer
```

- Gửi dữ liệu lên MQTT Brocker bằng lệnh publish:

```
void pubConfig() {
  bool success;
```

```
  for (int i = 0; i < nWidgets; i = i + 1) {
    success = client.publish(MQTT::Publish(prefix + "/" + deviceID + "/config", thing_config[i]).set_qos(1));
    if (success) {
      Serial.println("Publish config: Success (" + thing_config[i] + ")");
    } else {
      Serial.println("Publish config FAIL! ("    + thing_config[i] + ")");
    }
    delay(50);
  }
  if (success) {
   Serial.println("Publish config: Success");
  } else {
   Serial.println("Publish config: FAIL");
  }
  for (int i = 0; i < nWidgets; i = i + 1) {
    pubStatus(sTopic[i], stat[i]);
    delay(50);
  }
}
```

- Hàm xử lý Callback: Đọc dữ liệu nhận được và gửi lại dữ liệu

```
//////////Hàm Callback
void callback(const MQTT::Publish& sub) {
  if (sub.topic() == sTopic[1] + "/control") {
    if (sub.payload_string() == "0") {
      newValue = 1; // inverted
      stat[1] = stat0;
    } else {
      newValue = 0;
      stat[1] = stat1;
    }
    digitalWrite(pin[1], newValue);
    pubStatus(sTopic[1], stat[1]);
    WriteUart( RelayNumBer_1,6);
  }
  else if (sub.topic() == sTopic[3] + "/control") {
    if (sub.payload_string() == "0") {
      newValue = 1; // inverted
      stat[3] = stat0;
    } else {
      newValue = 0;
      stat[3] = stat1;
    }
    digitalWrite(pin[3], newValue);
    pubStatus(sTopic[3], stat[3]);
    WriteUart( RelayNumBer_3,6);
  } else if (sub.topic() == sTopic[0] + "/control") {
    if (sub.payload_string() == "0") {
      newValue = 1; // inverted
      stat[0] = stat0;
    } else {
      newValue = 0; // inverted
      stat[0] = stat1;
    }
    digitalWrite(pin[0], newValue);
    pubStatus(sTopic[0], stat[0]);
  }
```

```
    else if (sub.topic() == sTopic[5] + "/control") {
    String x = sub.payload_string();
    analogWrite(pin[5], 1023 - x.toInt());
    stat[5] = setStatus(x);
    pubStatus(sTopic[5], stat[5]);
  }
  else if (sub.topic() == sTopic[4] + "/control") {
    // ADC : nothing, display only
  }
  else if (sub.topic() == sTopic[9] + "/control") {
    // Nhiet do : nothing, display only
  }
  else if (sub.topic() == sTopic[10] + "/control") {
    // Khoang cach: nothing, display only
  }
  else if (sub.topic() == sTopic[11] + "/control") {
    // pH: nothing, display only
  }
  else if (sub.topic() == sTopic[2] + "/control") {
    if (sub.payload_string() == "0") {
      newValue = 1; // inverted
      stat[2] = stat0;
    } else {
      newValue = 0; // inverted
      stat[2] = stat1;
    }
    digitalWrite(pin[2], newValue);
    pubStatus(sTopic[2], stat[2]);
    WriteUart( RelayNumBer_2,6);

  } else if (sub.topic() == sTopic[6] + "/control")
 {
    String x = sub.payload_string();
    analogWrite(pin[6], x.toInt());
    stat[6] = setStatus(x);
    pubStatus(sTopic[6], stat[6]);
  }
else if (sub.topic() == sTopic[7] + "/control")
  {
    String x = sub.payload_string();
    analogWrite(pin[7], x.toInt());
    stat[7] = setStatus(x);
    pubStatus(sTopic[7], stat[7]);

    }
else if (sub.topic() == sTopic[8] + "/control")
 {
    String x = sub.payload_string();
    analogWrite(pin[8], x.toInt());
    stat[8] = setStatus(x);
    pubStatus(sTopic[8], stat[8]);
    ////////////////////////////////////Dimmer
    SetDimmer =(int)x.toInt()*100/1023;
    unsigned char g_DimmerLed[6]= {0xAA,0x03,0x01,0x09,SetDimmer,0x55};
    WriteUart( g_DimmerLed, 6);
  }
 else if (sub.topic() == prefix) {
    if (sub.payload_string() == "HELLO") {
```

```
      pubConfig();
    }
  }
}
```