

BẢNG PHÂN CÔNG CÔNG VIỆC VÀ KẾT QUẢ THỰC HIỆN ĐỀ TÀI CỦA TỪNG THÀNH VIÊN

STT	Họ và tên	MSSV	Nhiệm vụ	Kết quả	Chữ ký
1.	Nguyễn Thị Mai Thy	1713462	Tìm hiểu về giao thức MQTT, xây dựng giao diện trên PC dùng ngôn ngữ Python, thực hiện vẽ Schematic và PCB Layout và thi công mạch	Hoàn thành	
2.	Trần Ngọc Trâm	1711389	Viết code cho ESP – 01, các cảm biến và STM32F103C8T6, tìm hiểu về Thingsboard và Gateway, gửi dữ liệu lên Thingsboard và xây dựng Dashboard	Hoàn thành	

MỤC LỤC

PHẦN MỞ ĐẦU	4
I. Giới thiệu chung	5
1. Giao thức MQTT	5
1.2. Một số khái niệm:.....	5
1.3. MQTT Broker và Client:	9
1.4. Kết nối giữa các thiết bị MQTT:	10
1.5. Quá trình Publish và Subscribe:.....	11
1.6. Bảo mật trong giao thức MQTT:.....	12
2. Thingsboard:	14
3. Thingsboard Gateway:	14
4. Các phần mềm hỗ trợ:	16
I. Đặc tả hệ thống:	16
1. Sơ đồ khối hệ thống:.....	17
2. Yêu cầu về sản phẩm:	17
2.1. Tên đề tài:	17
2.2. Mục đích:	17
2.3. Ngõ vào/ngõ ra:.....	18
2.4. Cách sử dụng:	18
2.5. Nguyên lý hoạt động và chức năng của hệ thống:	18
HIỆN THỰC HỆ THỐNG	19
I. Phần cứng	19
1. Lựa chọn phần cứng:	19
1.1. Vi điều khiển STM32F103C8T6:	19
1.2. Cảm biến nhiệt độ, độ ẩm DHT11:	22
1.3. Cảm biến độ ẩm đất FC – 28:.....	23
1.4. Cảm biến cường độ ánh sáng MAX44009:.....	25
1.5. Module Wifi ESP – 01:.....	25
1.6. Mạch nguồn:	27
2. Sơ đồ khối, schematic và layout:	28
II. Phần mềm	30
1. Cấu hình cho STM32F103 dùng CubeMX:.....	30

2. Đọc dữ liệu cảm biến và gửi lên Thingsboard thông qua giao thức MQTT:	32
2.1. Đọc và xử lý dữ liệu từ cảm biến nhiệt độ và độ ẩm môi trường DHT11	32
2.2. Đọc và xử lý dữ liệu từ cảm biến độ ẩm đất FC – 28	34
2.3. Đọc và xử lý dữ liệu từ cảm biến cường độ ánh sáng MAX44009:	35
2.4. Lập trình cho module Wifi:	36
2.5. Gửi dữ liệu từ STM32 sang module ESP -01:	38
2.6. Cấu hình cho Thingsboard và tạo Dashboard:	39
2.7. Cấu hình Gateway – bên trung gian đóng vai trò kết nối MQTT Broker và Thingsboard	43
3. Giao diện Python:	47
III. Sản phẩm hoàn thiện:	50
ĐÁNH GIÁ HOẠT ĐỘNG CỦA HỆ THỐNG	51
PHẦN KẾT	52
PHỤ LỤC	53
I. Code đọc dữ liệu các cảm biến, đưa về STM32F103C8T6 xử lý:	53
II. Code cho module wifi	56
III. Code giao diện Python	59

PHẦN MỞ ĐẦU

Hệ thống Internet of Things ngày càng khẳng định được tầm quan trọng của mình trong thời kỳ đất nước đang dần chuyển mình thành một nước công nghiệp hiện đại. Mạng lưới chia sẻ dữ liệu khổng lồ này ngày nay không còn là một xu hướng mà là một phần không thể thiếu kết nối con người chúng ta với các thiết bị, các công trình thông minh xung quanh.

Trong quá trình tiếp cận và tìm hiểu về hệ thống Internet of Things cũng như dòng vi điều khiển ARM cortex M3/4, nhóm mong muốn thực hiện một hệ thống kết hợp cả hai yếu tố trên để có cơ hội tìm hiểu sâu hơn. Nhận thấy sự thông dụng của việc sử dụng các cảm biến thu thập dữ liệu từ môi trường để phân tích về tình hình thời tiết và những ứng dụng khác, nhóm đã lên ý tưởng thực hiện một hệ thống thu thập dữ liệu từ môi trường và theo dõi thông qua máy tính cá nhân hoặc điện thoại với sự hỗ trợ của mạng Internet.

Để hỗ trợ cho việc truyền dữ liệu trong hệ thống Internet of Things của mình, chúng ta cần một giao thức nào đó hỗ trợ. Nhóm nhận thấy MQTT là sự lựa chọn thích hợp cho những hệ thống vừa và nhỏ như đồ án này. Bên cạnh đó, việc theo dõi những dữ liệu thu được sẽ trở nên thuận tiện hơn vì dữ liệu sẽ được lưu trữ và biểu diễn dưới dạng biểu đồ thông qua Thingsboard Server. Đồng thời, trên máy tính cá nhân của mình, người dùng còn có thể theo dõi dữ liệu tại một thời điểm nhất định thông qua giao diện xây dựng bằng ngôn ngữ Python.

Đồ án này gồm những phần chính là Tổng quan về hệ thống, Hiện thực hệ thống và Đánh giá hệ thống. Tổng quan về hệ thống sẽ nêu những gì nhóm đã tìm hiểu được về giao thức MQTT, Thingsboard Server và những thành phần cấu thành phần cứng của hệ thống. Phần tiếp theo - Hiện thực hệ thống sẽ ghi nhận lại quá trình nhóm hiện thực phần cứng, phần mềm để cho ra kết quả cuối cùng. Ở phần sau cùng – Đánh giá hệ thống là điểm lại những gì nhóm đã tìm hiểu và làm được qua đồ án này cũng như những hạn chế mà nhóm còn chưa giải quyết được.

Trong quá trình thực hiện đồ án, nhóm đã nhận được sự chỉ dạy tận tình từ Thầy Bùi Quốc Bảo. Nhóm kính gửi đến Thầy lời cảm ơn chân thành nhất và chúc Thầy thật nhiều sức khoẻ để tiếp tục hướng dẫn nhiều thế hệ sinh viên chúng em.

TỔNG QUAN VỀ HỆ THỐNG

I. Giới thiệu chung

1. Giao thức MQTT

1.1. Giao thức MQTT là gì

MQTT được phát minh vào năm 1999 bởi Andy Stanford-Clark (IBM) và Arlen Nipper (Eurotech). Ban đầu, họ thiết kế MQTT để dùng trong hệ thống điều khiển giám sát và thu thập dữ liệu (SCADA) cho hệ thống dẫn dầu thông qua vệ tinh. Họ đề ra nhiều yêu cầu trong lúc thực hiện dự án này:

- Dễ dàng sử dụng
- Chất lượng đường truyền ổn định
- Gói tin nhẹ và sử dụng băng thông một cách hiệu quả nhất
- Bảo mật dữ liệu

Để tìm hiểu những nhà phát minh đã giải quyết những yêu cầu đó như thế nào, chúng ta sẽ đi sâu hơn ở phần sau khi phân tích các thành phần của MQTT. Đến nay, những nhà phát triển vẫn không ngừng nỗ lực để đáp ứng các yêu cầu đó một cách tốt hơn, bằng chứng là việc tung ra phiên bản mới nhất là MQTT v5.0 vào tháng 3/2019. Ban đầu MQTT được biết đến là viết tắt của MQ Telemetry Transport nhưng qua thời gian phát triển và sử dụng, người ta định nghĩa MQTT là Message Queue Telemetry Transport.

MQTT được sử dụng phổ biến trong hệ thống IoT và được thiết kế cho mạng lưới TCP/IP. MQTT là một giao thức truyền tin nhắn (message) theo mô hình publish/subscribe. Giao thức này có gói tin nhẹ (1 tin nhắn là mảng kích thước 1 byte), code đơn giản, tiêu tốn ít băng thông và dễ dàng sử dụng, thích hợp khi đường truyền bị giới hạn. Vì những đặc điểm vừa nêu, giao thức MQTT rất phù hợp để sử dụng trong nhiều trường hợp như trong giao tiếp Machine to Machine (M2M) và đặc biệt là trong hệ thống Internet of Things. Phiên bản MQTT được sử dụng phổ biến hiện nay là MQTT v3.1.1.

1.2. Một số khái niệm:

- **Topic:**

Topic là chủ đề mà Broker tạo ra để các Client gửi dữ liệu vào đó. Topic giống như một thư mục để chứa dữ liệu nên nó có dạng đường dẫn abc/xyz/...

- **Publish:**

Là gói tin mà Client gửi lên Broker. Khi Client publish dữ liệu vào topic, các Client sẽ nhận được dữ liệu nếu đã đăng ký topic đó. Gói Publish có thể là gói gửi dữ liệu gửi vào topic hoặc gói cài đặt Retain Message hay LWT Message do cờ Retain và cờ LWT quy định.

- **Subscribe:**

Là gói đăng kí nhận dữ liệu từ topic đã đăng ký trước đó (Broker sẽ tạo một topic mới nếu không tìm được topic đăng ký).

- **Quality of Service (QoS):**

QoS là chất lượng đường truyền. Nó gồm có 3 mức 0,1 và 2. Mức QoS thể hiện sự đảm bảo về việc truyền tin nhắn từ phía gửi đến phía nhận.

- QoS 0:

Đây là mức thấp nhất của QoS. Tin nhắn sẽ được gửi đi mà phía nhận sẽ không phản hồi là đã nhận được tin nhắn hay chưa và phía gửi cũng không ghi lại tin nhắn trong bộ đệm của mình. QoS 0 thường được gọi là “fire and forget” và được sử dụng mặc định.

QoS 0 được sử dụng khi đường truyền giữa phía gửi và phía nhận ổn định và người dùng không tâm đến việc một số tin nhắn có thể bị mất trong quá trình truyền đi. Ngoài ra, đối với QoS 0 thì tin nhắn không được xếp vào hàng đợi queue.

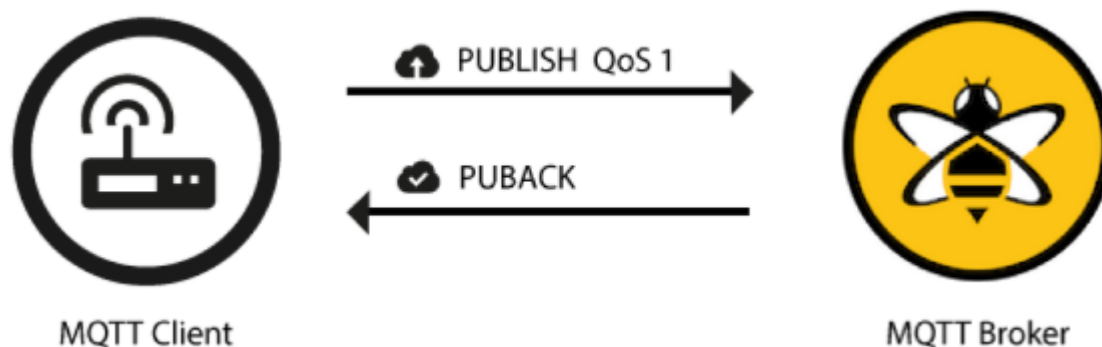


Hình 1. QoS 0

- QoS 1:

QoS 1 đảm bảo rằng tin nhắn sẽ được gửi tới phía nhận ít nhất 1 lần. Phía gửi sẽ lưu nội dung tin nhắn cho tới khi nó nhận được gói PUBACK – xác nhận đã nhận được tin nhắn từ phía thu. Một tin nhắn có thể được gửi nhiều lần.

QoS 1 được sử dụng khi phía nhận muốn nhận được tất cả các tin nhắn và không quan tâm đến việc có thể nhận được nhiều tin nhắn trùng nhau. QoS 1 được sử dụng thường xuyên hơn so với 2 mức còn lại vì độ an toàn và tốc độ truyền nhanh.



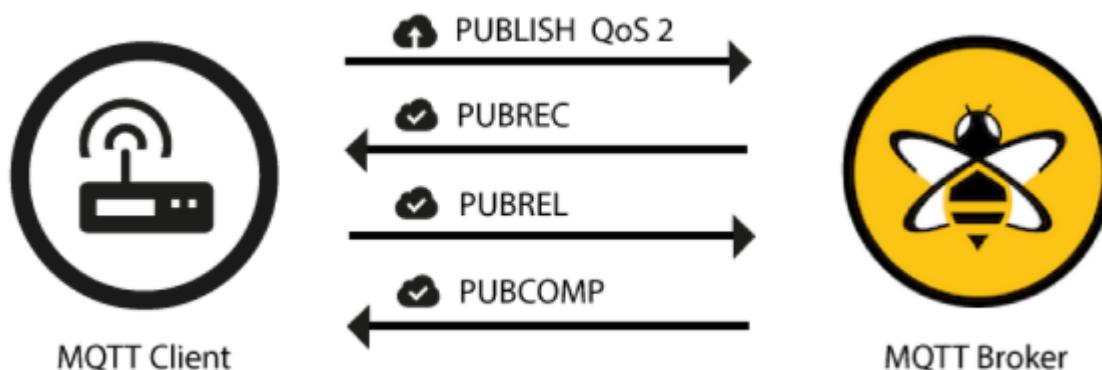
Hình 2. QoS 1

- QoS 2:

QoS 2 đảm bảo rằng tin nhắn sẽ được gửi đến phía nhận duy nhất một lần. Đây là mức an toàn nhất và cũng mất nhiều thời gian nhất trong 3 mức. Việc truyền 1 tin nhắn thành công được đảm bảo bởi ít nhất 2 gói request và 2 gói response.

QoS 2 được cân nhắc sử dụng khi hệ thống muốn nhận được tin nhắn chính xác một lần nếu không sẽ ảnh hưởng đến hoạt động của hệ thống. Tuy nhiên, việc truyền tin nhắn của QoS 2 sẽ mất khá nhiều thời gian để hoàn thành.

- Hàng đợi tin nhắn của QoS 1 và QoS 2:



Hình 3. QoS 2

Hàng đợi tin nhắn được sử dụng để lưu trữ tin nhắn khi một Client bị mất kết nối một cách đột ngột. Nó chỉ được kích hoạt khi người dùng cài đặt Clean session là False.

- **ClientID:**

Mỗi Client khi kết nối với Broker thì sẽ được nhận dạng bằng 1 ClientID. Broker sẽ dùng ID này để nhận dạng Client và xác định trạng thái hiện tại của nó. Vì thế phải có 1 ID duy nhất cho mỗi Client. Đối với MQTT 3.1.1, khi cài đặt chúng ta có thể để trống phần ClientID và broker sẽ tự tạo cho nó một ID ngẫu nhiên. Lúc này, Clean session phải được đặt là True nếu không kết nối sẽ bị từ chối.

- **Clean session:**

Cờ Clean session nhận 2 giá trị True – 1 hoặc False – 0. Nếu Client muốn sử dụng persistent session thì nó sẽ báo cho Broker bằng cách đặt cờ Clean session là False. Lưu ý rằng để sử dụng persistent session thì QoS phải ở mức 1 hoặc 2. Persistent session là nơi mà Broker sẽ lưu trữ những yêu cầu subscribe vào một chủ đề nào đó của Client và lưu trữ lại những tin nhắn gửi đi không thành công. Nếu như Clean session là True thì Broker sẽ không lưu lại bất kỳ thông tin nào về Client và xóa hết những gì persistent session đã lưu trước đó.

- **Username/Password:**

MQTT sẽ dùng username và password để xác thực và ủy quyền cho Client. Tuy nhiên, hai yếu tố này không được mã hoá mà ở dạng chữ viết con người có thể đọc được. Vì thế, khi sử dụng MQTT trong việc truyền dữ liệu, người ta khuyến khích sử dụng cả password và username để nâng cao tính bảo mật.

- **Last Will and Testament (LWT):**

Trong MQTT, LWT được Broker dùng để thông báo với các Client khác khi có một Client nào đó bị ngắt kết nối một cách không rõ ràng.

Broker sẽ gửi LWT trong các trường hợp sau:

- Broker phát hiện ra một thiết bị ngõ vào/ ngõ ra nào đó đang bị lỗi hoặc là kết nối bị lỗi.
- Client không phản hồi trong thời gian Keep alive
- Broker ngừng việc kết nối khi có một lỗi nào đó do giao thức gây ra
- Client không gửi gói DISCONNECT trước khi nó ngừng kết nối.

- **Retain message:**

Retain message là một tin nhắn bình thường với cờ Retain được đặt giá trị True. Nó giúp cho những Client mới (subscribe và kết nối thành công) nhận được tin nhắn ngay khi kết nối mà không cần đợi publisher gửi tin nhắn tiếp theo.

- **Keep Alive:**

Keep Alive là khoảng thời gian được tính bằng giây có ý nghĩa là trong thời gian này kết nối giữa Broker và Client đang mở, chờ việc trao đổi dữ liệu giữa các Client.

- **Cấu tạo của một MQTT message:**

Một tin nhắn trong giao thức MQTT sẽ gồm những thành phần cơ bản sau:

- Chủ đề (Topic)
- Cờ Retain
- Qos
- Payload

1.3. MQTT Broker và Client:

Kiến trúc mức cao của MQTT gồm 2 phần chính là Broker và Client

a. Broker:

Broker Đóng vai trò trung gian trong kết nối giữa các publisher và các subscriber. Tùy vào việc cài đặt, một broker có thể xử lý hàng nghìn Client kết nối tại cùng một thời điểm. Broker có nhiệm vụ thu thập tất cả các tin nhắn, lọc tin nhắn, phân tích xem client nào đã subscribe tin nhắn này và gửi đến đúng địa chỉ nhận. Ngoài ra, broker còn có khả năng thực hiện một số chức năng khác như bảo mật tin nhắn, lưu trữ tin nhắn, cũng như xác thực và ủy quyền cho các client. Vì thế broker cần phải có tính ổn định cao, ổn định và dễ giám sát. Chúng ta có hai lựa chọn là cài đặt broker local hoặc sử dụng broker online.

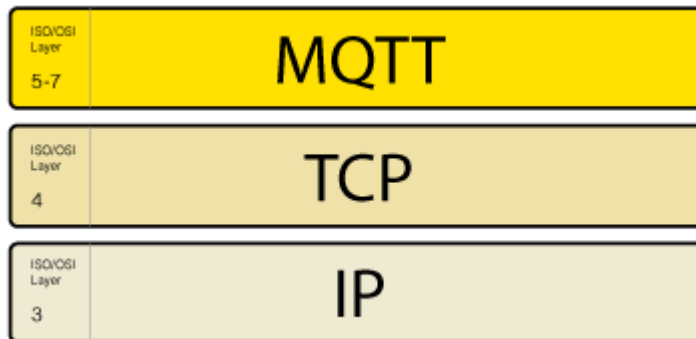
b. Clients:

Bất cứ một thiết bị nào dựa trên TCP/IP, cài đặt thư viện MQTT và kết nối đến broker đều trở thành một MQTT Client. Trong đồ án này, MQTT Client là điện thoại android và máy tính cá nhân. Client được chia làm hai nhóm là publisher và subscriber. Không có một kết nối trực tiếp nào giữa một publisher và một subscriber mà phải thông qua broker và chúng cũng không biết về sự tồn tại của nhau. Một thiết bị MQTT Client có thể vừa là publisher vừa là subscriber. Các publisher sẽ gửi dữ liệu lên broker thông qua một hoặc nhiều tên chủ đề

(topic). Các subscriber sẽ nhận được dữ liệu từ một hoặc nhiều chủ đề cụ thể khi đăng ký (subscribe) chủ đề đó.

1.4. Kết nối giữa các thiết bị MQTT:

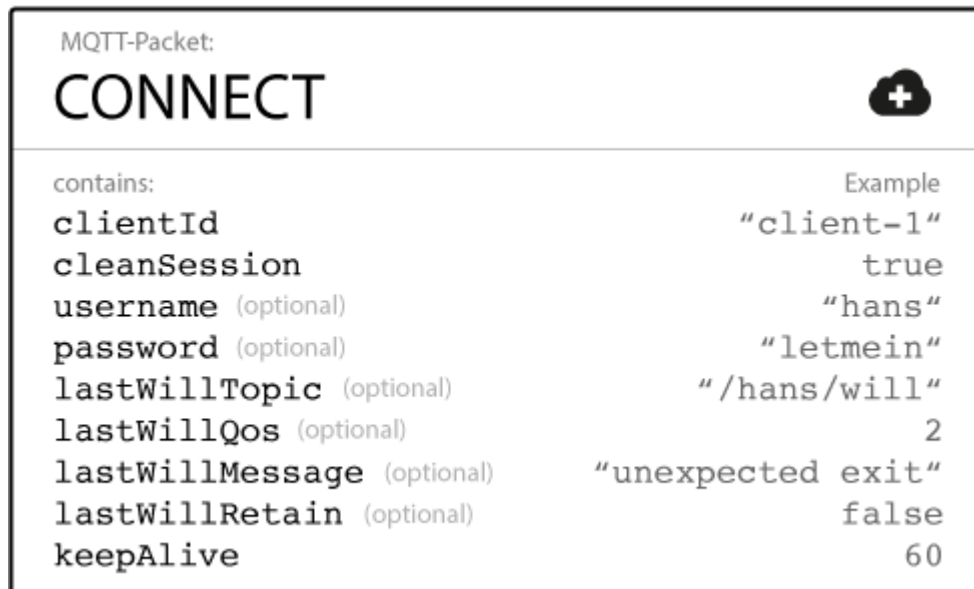
Giao thức MQTT dựa trên TCP/IP nên cả client và broker đều phải có TCP/IP stack



Hình 4. Các lớp giao tiếp

Client sẽ bắt đầu một kết nối bằng cách gửi gói tin CONNECT đến Broker. Nếu gói tin CONNECT không đúng format hoặc mất quá nhiều thời gian giữa thời điểm mở kết nối và gửi gói CONNECT thì kết nối sẽ bị đóng.

Cấu tạo của gói CONNECT:



Hình 5. Cấu tạo gói CONNECT

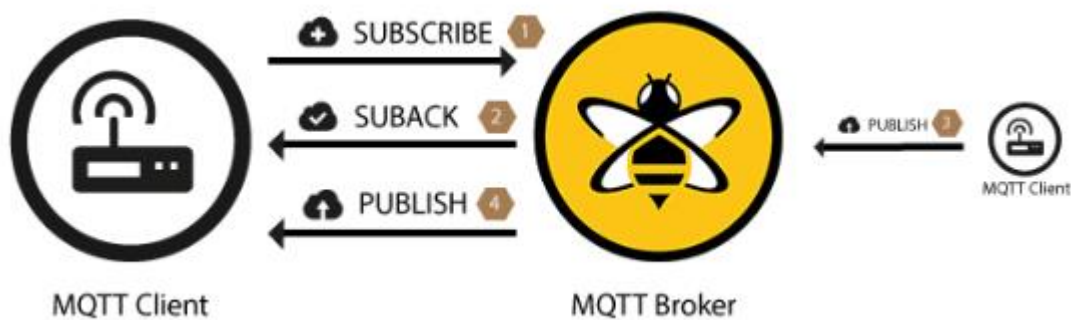
Broker sẽ phản hồi bằng gói tin CONNACK. Gói này bao gồm một Return code – giúp chúng ta biết được kết nối có được thiết lập thành công hay không. Return code nhận 6 giá trị trả về như sau:

- Rc = 0 : kết nối được chấp nhận
- Rc = 1 : kết nối bị từ chối, phiên bản giao thức MQTT giữa Client và Broker

không tương thích

- Rc = 2: kết nối bị từ chối, quá trình xác minh Client không được chấp nhận
- Rc = 3 : kết nối bị từ chối, broker chưa được kích hoạt
- Rc = 4 : kết nối bị từ chối, username hoặc password bị lỗi
- Rc = 5 : kết nối bị từ chối, Client không được Broker uỷ quyền

1.5. Quá trình Publish và Subscribe:



Hình 6. Quá trình Publish và Subscribe

a. Publish:

Client có thể publish tin nhắn ngay khi kết nối với Broker. MQTT sử dụng bộ lọc tin nhắn dựa trên chủ đề. Mỗi tin nhắn sẽ có một payload để chứa dữ liệu muốn truyền đi. Payload có thể có kiểu nhị phân, text hoặc kiểu JSON tùy vào cài đặt phía Publisher. Để bắt đầu truyền dữ liệu thì Publisher sẽ gửi đi một gói PUBLISH. Publisher sẽ không nhận được bất cứ phản hồi nào về việc dữ liệu đã truyền đến Subscriber chưa hay có bao nhiêu Client đã Subscribe vào chủ đề mà nó vừa Publish.

MQTT-Packet:	
PUBLISH	
	
contains:	Example
packetId (always 0 for qos 0)	4314
topicName	"topic/1"
qos	1
retainFlag	false
payload	"temperature:32.5"
dupFlag	false

Hình 7. Cấu tạo gói PUBLISH

b. Subscribe:

Để nhận được dữ liệu từ một Topic, Subscriber phải gửi một gói SUBSCRIBE đến Broker. Cấu tạo của gói này đơn giản hơn so với gói PUBLISH, nó chỉ bao gồm phần để xác thực Client và danh sách các Topic muốn Subscribe.

Yêu cầu Subscribe sẽ được xác nhận từ Broker thông qua việc gửi về Client gói tin SUBACK.

MQTT-Packet:	
SUBACK	
	
contains:	Example
packetId	4313
returnCode 1 (one returnCode for each	2
returnCode 2 topic from SUBSCRIBE,	0
... in the same order)	...

Hình 8. Cấu tạo của gói SUBACK

Sau khi nhận được gói SUBACK, Subscriber sẽ nhận được tất cả các tin nhắn từ Topic mà nó đã Subscribe.

1.6. Bảo mật trong giao thức MQTT:

Đối với những hệ thống nhỏ, bảo mật chưa thực sự là vấn đề đáng kể. Tuy nhiên, đối với hệ các hệ thống IoT lớn, bảo mật là vấn đề đáng quan tâm. Giao thức MQTT được thiết kế gọn nhẹ và dễ sử dụng nên nó chỉ cung cấp một số cơ chế bảo mật đơn giản. Thông thường sẽ có 3 lớp bảo mật được sử dụng bao gồm: lớp mạng (Network Layer), lớp giao vận (Transport Layer) và lớp ứng dụng (Application Layer). Trong đồ án này nhóm thực hiện bảo mật dựa trên lớp ứng dụng.

a. Xác thực với Username và Password: nhà phát triển MQTT khuyến khích sử dụng cả username và password để đạt được hiệu quả bảo mật tốt nhất

User name và password là hai trường nằm trong gói CONNECT mà khi muốn thực hiện kết nối thì Client sẽ gửi gói này đến Broker. Broker sẽ dựa vào những thành phần trong gói CONNECT để tiến hành xác thực Client.

Khi sử dụng xác thực bằng tên username và password, MQTT Broker sẽ đánh giá thông tin Client dựa trên cơ chế xác thực đã được triển khai và gửi trả về gói tin phản hồi gọi là CONNACK, trong đó bao gồm một mã trạng thái gọi là Return code.

User name và password phải được cài đặt từ phía Broker thông qua các bước sau (việc cài đặt này chỉ thực hiện được nếu máy tính chạy Windows 7 và 10 hoặc hệ điều hành Linux):

- Tạo file chứa Username và Password:
- Copy file chứa hai thông tin này vào file chứa Mosquitto và chỉnh sửa file mosquitto.conf để cho phép sử dụng password

Một Client có thể chỉ dùng username mà không cần password nhưng không có trường hợp ngược lại

b. Xác thực với Client Identifier

Mỗi MQTT client đều có một mã nhận dạng duy nhất được gọi là Client identifier. Tương tự như Username/Password, mã nhận dạng này cũng được cung cấp bởi Client khi gửi một gói tin CONNECT đến Broker, được cung cấp trong trường ClientID.

Trong quá trình xác thực, Identifier của Client thường được sử dụng để xác thực bên cạnh tên user name và password. Mặc dù không phải là cách thức bảo mật tốt khi sử dụng trong thực tiễn tuy nhiên đối với các hệ thống khép kín và ít thiết bị thì loại xác thực này cũng đủ để cân nhắc sử dụng.

2. Thingsboard:

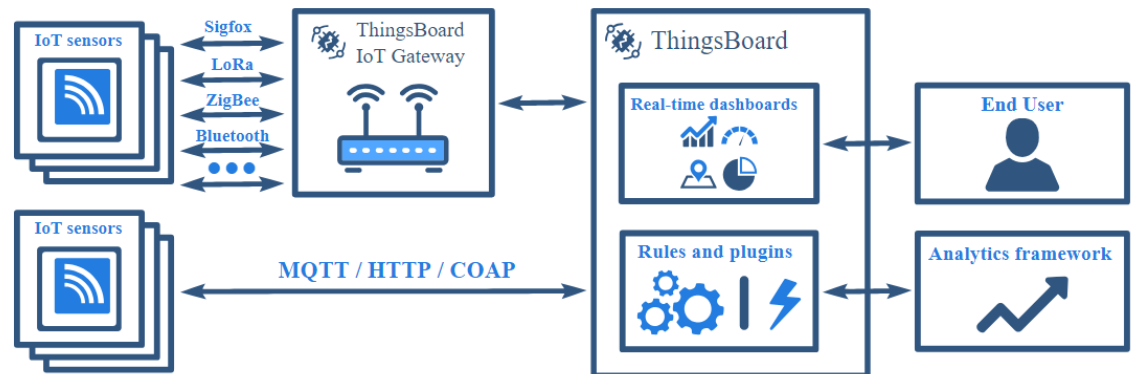
Thingsboard là một nền tảng IoT nguồn mở để thu thập, xử lý, trực quan hóa và quản lý thiết bị dữ liệu. Nó cho phép kết nối thiết bị thông qua các giao thức IoT : MQTT, CoAP và HTTP và hỗ trợ cả triển khai trên nền tảng đám mây.

Các đặc trưng của Thingsboard:

- Thu thập và trực quan hóa dữ liệu thu được từ các thiết bị
- Phân tích tin nhắn đến, kích hoạt các cảnh báo bằng xử lý các sự kiện phức tạp
- Điều khiển thiết bị bằng RPC (Remote Procedure Calls)

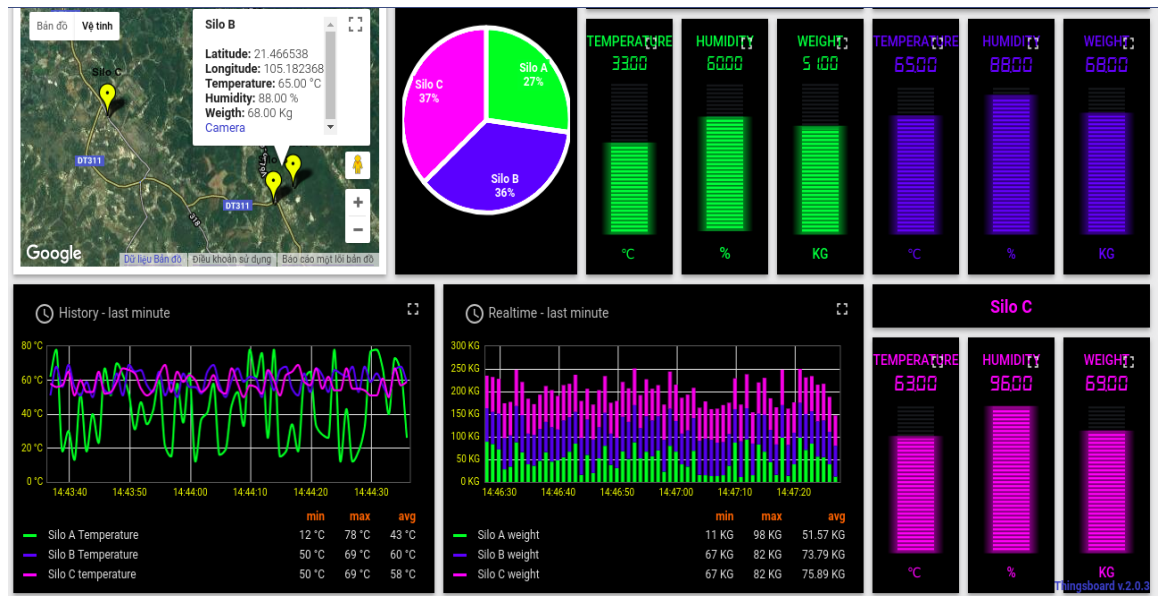
3. Thingsboard Gateway:

Thingsboard IoT gateway là một giải pháp cho phép bạn tích hợp các thiết bị được kết nối với các hệ thống bên thứ ba và bên thứ ba với Thingsboard

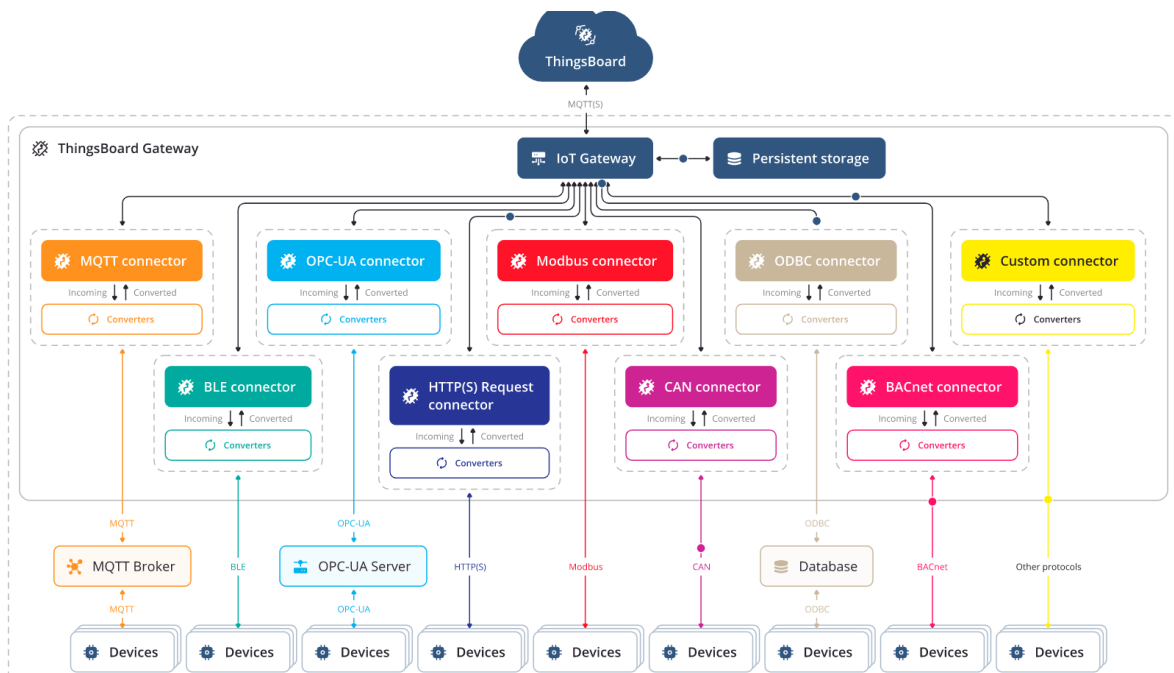


Hình 9. Mô hình truyền dữ liệu tới người dùng sử dụng Thingsboard platform)

Thingsboard giúp trực quan hóa dữ liệu, giúp cho việc theo dõi dữ liệu, và điều khiển thiết bị trở nên dễ dàng hơn.



Hình 10. Một ví dụ về Thingsboard Dashboard



Hình 11. Mô hình Gateway

Một vài đặc trưng của Thingsboard Gateway

- MQTT connector : kiểm soát, định cấu hình và thu thập dữ liệu từ các thiết bị IoT được kết nối với MQTT brokers bên ngoài bằng các giao thức hiện có.
- OPC-UA connector: thu thập dữ liệu từ các thiết bị IoT được kết nối với máy chủ OPC-UA
- Modbus connector: thu thập dữ liệu từ các thiết bị IoT được kết nối thông qua giao thức Modbus
- Custom connector: thu thập dữ liệu từ các thiết bị IoT được kết nối bởi các giao thức khác nhau. (Ta có thể tạo trình kết nối của riêng mình cho giao thức yêu cầu)

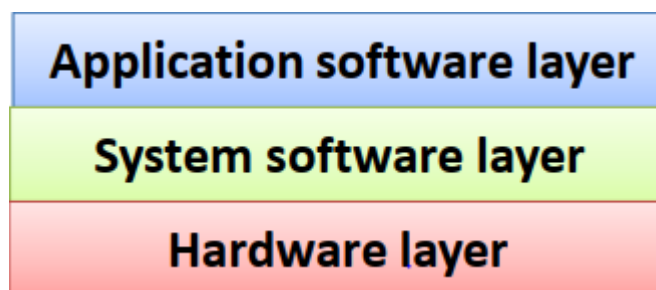
4. Các phần mềm hỗ trợ:

Trong quá trình thực hiện đề án này, nhóm sử dụng một số phần mềm để hỗ trợ cho việc lập trình và hiện thực hệ thống như sau:

- Keil C và STM32CubeMX: cấu hình xung nhịp, cấu hình GPIO và lập trình cho STM32F103C8T6 và các cảm biến
- Arduino IDE: lập trình cho module Wifi Esp – 01
- Mosquitto: MQTT Broker
- Thingsboard: server – nơi mà dữ liệu sẽ được chuyển lên và giám sát qua đồ thị
- Python 3.8 và Visual Studio Code: xây dựng giao diện bằng ngôn ngữ Python với sự hỗ trợ của PyQt5
- Cadence Orcad và Allegro: thiết kế sơ đồ nguyên lý và vẽ PCB

I. Đặc tả hệ thống:

Một hệ thống sẽ cấu tạo từ 3 lớp như sau



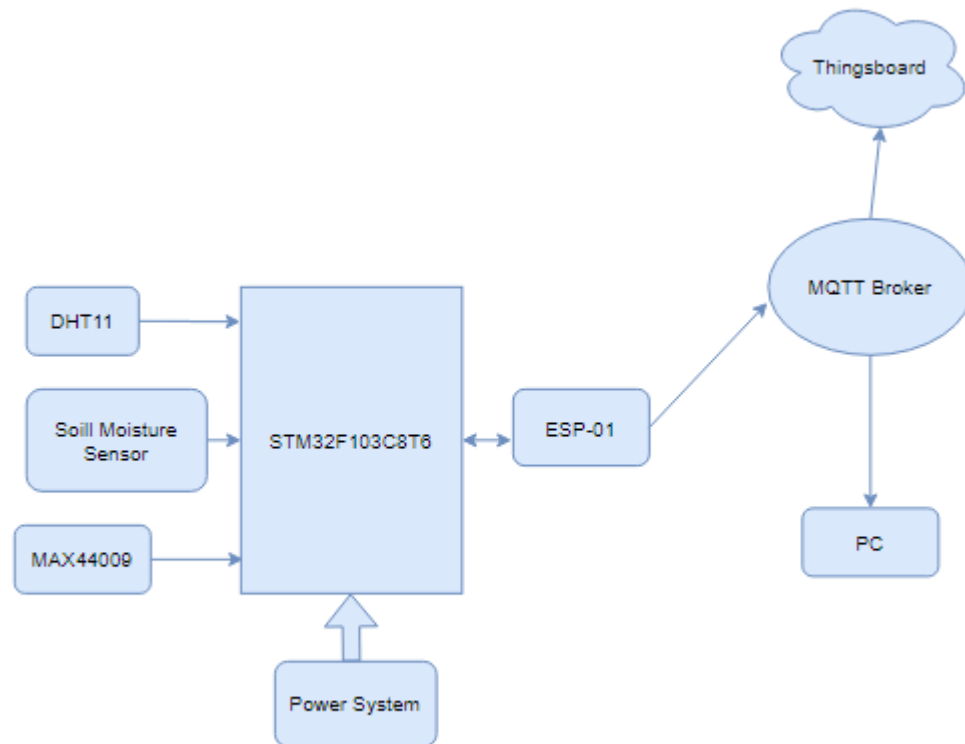
Hình 12. Cấu tạo của một hệ thống nhúng

- Application software layer – lớp phần mềm ứng dụng :

Giao diện Python dùng để hiển thị dữ liệu thu được từ cảm biến trên máy tính cá nhân

- System software layer – lớp phần mềm hệ thống:
 - + Thư viện HAL dùng để lập trình cho STM32F103C8T6
 - + Thư viện PubSubClient lập trình cho ESP-01 gửi dữ liệu lên MQTT Broker thông qua giao thức MQTT
 - + Gói Paho và gói PyQt5 trên Python để xây dựng giao diện Python
- Hardware layer – lớp phần cứng: Kit phát triển, các cảm biến, các thành phần cấu tạo nên mạch nguồn và mạch in

1. Sơ đồ khối hệ thống:



Hình 13. Sơ đồ khối hệ thống

2. Yêu cầu về sản phẩm:

2.1. Tên đề tài:

Hệ thống thu thập dữ liệu từ cảm biến, gửi dữ liệu lên Thingsboard thông qua giao thức MQTT và hiển thị dữ liệu trên máy tính thông qua giao diện Python

2.2. Mục đích:

Thu thập liên tục các dữ liệu về độ ẩm, nhiệt độ môi trường, độ ẩm đất và cường độ ánh sáng gửi lên Thingsboard server để quan sát thông qua đồ thị về sự thay đổi của các thông số trên. Đồng thời, dữ liệu cũng được quan sát từ máy tính cá nhân thông qua giao diện Python. Hệ thống có thể phát triển lên thành một phần trong hệ thống dự báo thời tiết hoặc mô hình vườn thông minh. Thông qua việc theo dõi các thông số nêu trên một cách liên tục giúp người dùng phát hiện những bất thường và để đưa ra những giải pháp kịp thời.

2.3. Ngõ vào/ngõ ra:

- Ngõ vào:

Các cảm biến đo nhiệt độ, độ ẩm môi trường, đo độ ẩm đất, cảm biến đo cường độ ánh sáng và bộ nguồn.

- Ngõ ra:

Module wifi

2.4. Cách sử dụng:

Người dùng cấp nguồn cho hệ thống và theo dõi dữ liệu thông qua hai kênh: Thingsboard hoặc giao diện Python.

2.5. Nguyên lý hoạt động và chức năng của hệ thống:

- STM32F103C8T6 đóng vai trò là trung tâm thu thập và xử lý dữ liệu.
- Module ESP – 01 là trung gian để truyền dữ liệu từ vi điều khiển trung tâm đến MQTT Broker thông qua giao thức MQTT.
- Giao diện Python biến máy tính thành một MQTT Client, lấy dữ liệu trực tiếp từ MQTT Broker
- Để gửi dữ liệu lên Thingsboard, cần có một trung gian để kết nối giữa MQTT Broker và Thingsboard đó là Gateway

HIỆN THỰC HỆ THỐNG

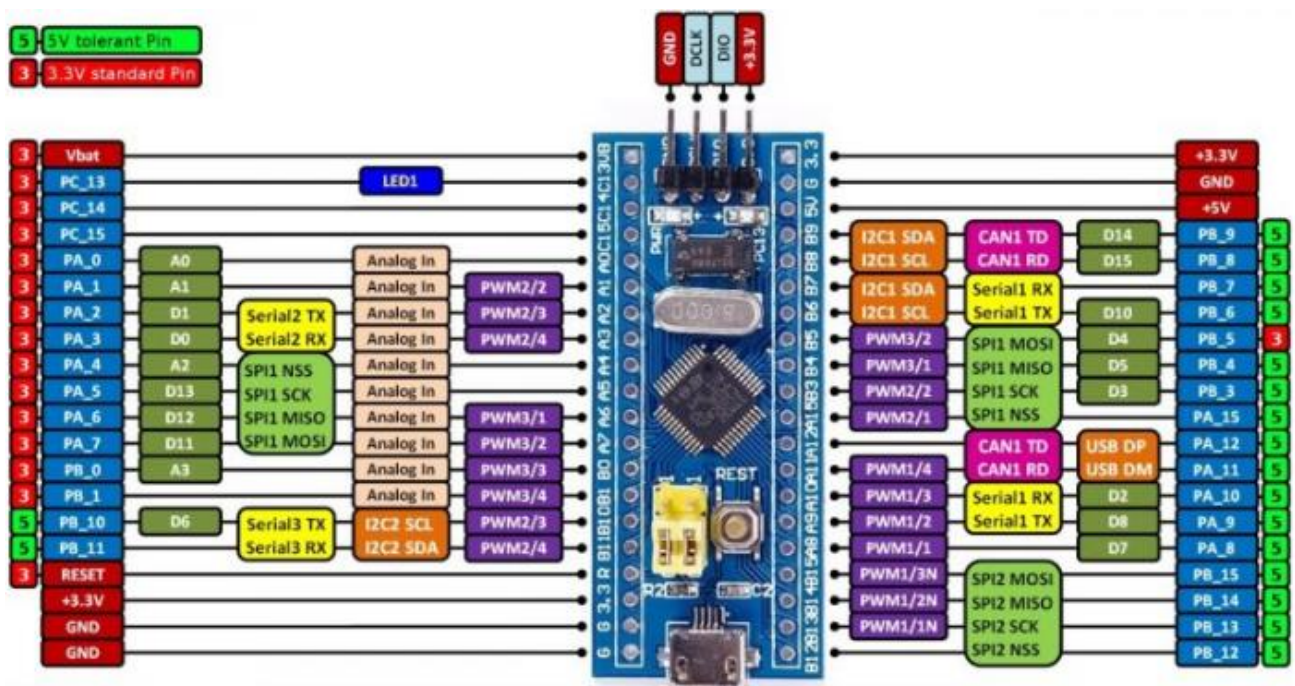
I. Phần cứng

1. Lựa chọn phần cứng:

1.1. Vi điều khiển STM32F103C8T6:

Trong đồ án này, nhóm chọn sử dụng Kit phát triển dựa trên vi điều khiển ARM Cortex M3 là STM32F103C8T6 vì một số lý do sau đây:

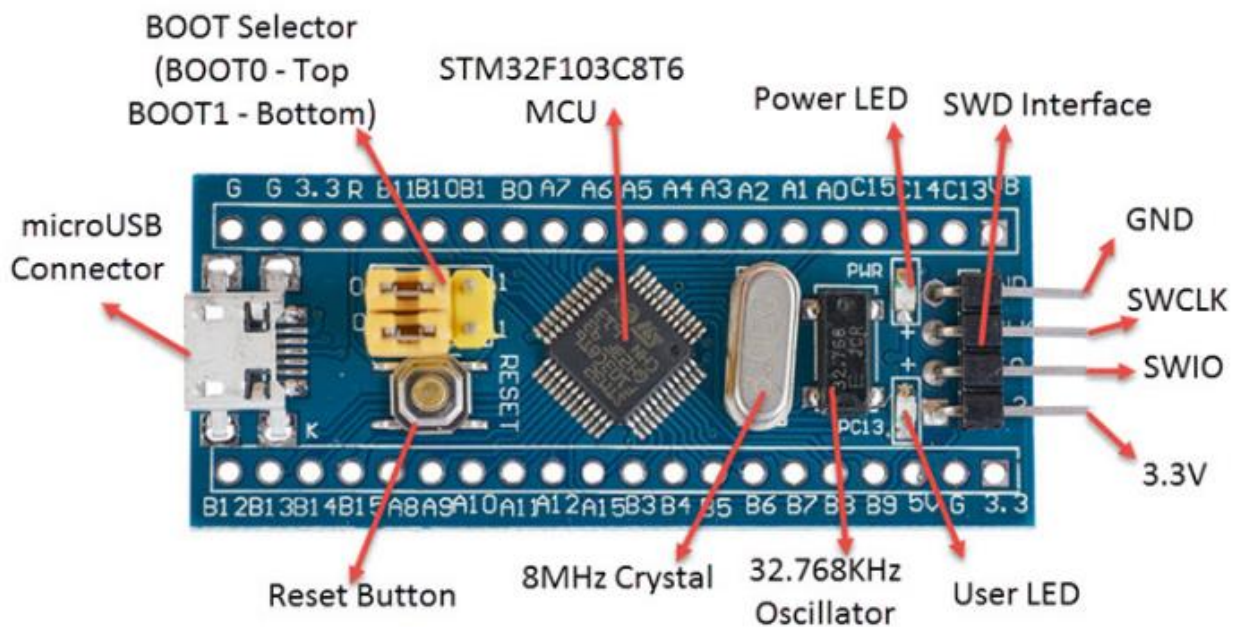
- Giá thành thấp
- Kích thước nhỏ gọn
- Nguồn cấp nhỏ, có thể sử dụng nguồn rời bên ngoài hoặc dùng nguồn từ cổng USB của máy tính. Tuy nhiên không nên cấp hai nguồn này đồng thời để đảm bảo vi điều khiển không bị hỏng.
- Vi điều khiển có đầy đủ các chân chức năng và các giao tiếp mà nhóm cần sử dụng. Bộ nhớ Flash cũng như SRAM vừa đủ dùng, không gây lãng phí.
- Thư viện hỗ trợ dễ dàng sử dụng



Hình 14. Kit phát triển STM32F103C8T6

STM32F103C8T6 thường được gọi là Blue Pill. Việc lập trình cho Blue Pill khá dễ dàng khi ta có thể dùng Arduino IDE hoặc cấu hình cho nó thông qua STM32CubeMX và lập trình trên Keil C.

Trước khi sử dụng kit, ta cần chú ý một số thành phần cơ bản sau đây:



Hình 15. Thành phần cơ bản trên kit

- Vi điều khiển chính STM32F103C8T6 – đây là dòng vi điều khiển 32 bit

- Nút Reset
- Cổng micro USB dùng cho việc cấp nguồn và giao tiếp UART với máy tính
- Giao tiếp SWD dùng để lập trình và gỡ rối thông qua mạch nạp ST – link
- Thạch anh 8MHZ cung cấp xung nhịp chính cho hệ thống
- Thạch anh 32.768KHZ cung cấp xung nhịp cho các ứng dụng RTC

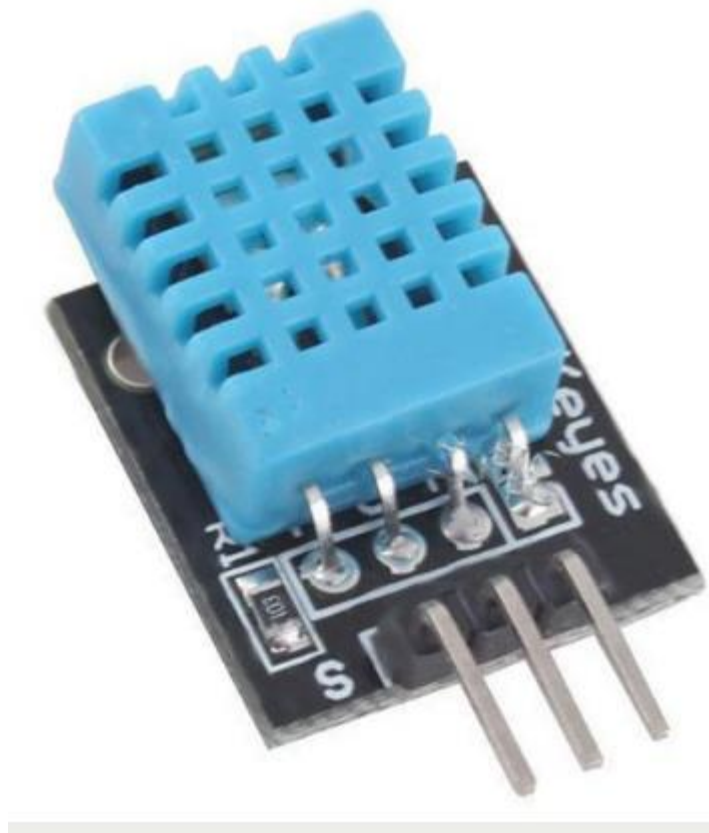
Các thông số quan trọng của Kit:

- Nguồn cấp: có hai lựa chọn
 - Nguồn cấp từ cổng micro USB của máy tính là 5VDC, nó sẽ được chuyển xuống thành 3.3VDC sau khi đi qua IC nguồn để cấp nguồn cho vi điều khiển chính
 - Sử dụng nguồn 3.3VDC bên ngoài
- ⇒ Nhóm chọn sử dụng nguồn 3.3VDC rời bên ngoài vì tính di động của hệ thống và để có được giá trị điện áp ổn định cấp cho các cảm biến
- Với thạch anh chính 8MHZ, kit có thể hoạt động với tần số xung nhịp tối đa lên đến 72MHZ

- Bộ nhớ:
 - Bộ nhớ FLASH 64KB
 - SRAM 20KB
- Các chân GPIO:
 - Kit Blue Pill có ba Port chính
 - + Port A: 16 chân
 - + Port B: 16 chân
 - + Port C: 3 chân
 - Các chân GPIO có nhiều chức năng như: Input, Output, Analog, PWM
 - + 15 chân PWM
 - + 10 kênh Analog với bộ ADC 12 bit
 - Các chân của kit có hỗ trợ ngắt ngoài
- Các chuẩn giao tiếp:
 - 2 cặp chân tương ứng với 2 giao tiếp I2C
 - 3 cặp chân tương ứng với 3 giao tiếp USART
 - 2 giao tiếp SPI, mỗi giao tiếp cần 4 chân

- USB 2.0
- CAN 2.0
- Timer: kit gồm 4 timer
- 3 timer 16 bit
- 1 timer 16 bit hỗ trợ điều chế xung PWM

1.2. Cảm biến nhiệt độ, độ ẩm DHT11:



Hình 16 . Module cảm biến DHT11

Nhóm chọn sử dụng module cảm biến DHT11 vì những lý do sau:

- Không cần sử dụng thêm điện trở hạn dòng vì trên module lắp đặt sẵn
- Giá thành rẻ, kích thước nhỏ gọn
- Hoạt động khá ổn định
- Thời gian phản hồi khá nhanh
- Giao tiếp giữa cảm biến và vi điều khiển đơn giản, chỉ cần một dây để truyền dữ liệu
- Dữ liệu ở dạng digital nên việc xử lý dữ liệu không quá phức tạp

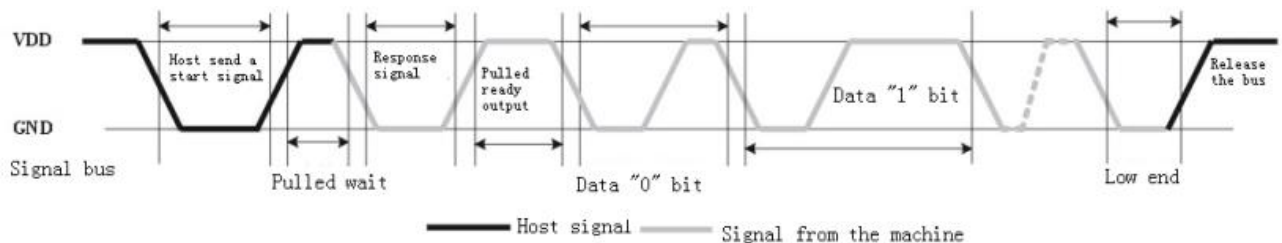
Nguyên lý hoạt động: DHT11 là cảm biến đo nhiệt độ, độ ẩm môi trường cơ bản, với đầu ra tín hiệu số và được sử dụng khá rộng rãi hiện nay. DHT11 sử dụng cảm biến độ ẩm điện dung và điện trở nhiệt để đọc dữ liệu về độ ẩm và nhiệt độ từ môi trường. Khi độ ẩm thay đổi, độ dẫn của chất nền thay đổi hoặc điện trở giữa các điện cực của điện dung này thay đổi. Sự thay đổi điện trở này được đo và xử lý bởi IC khiến cho vi điều khiển luôn sẵn sàng để đọc. Cũng tương tự như vậy, khi nhiệt độ thay đổi, điện trở của nhiệt điện trở cũng thay đổi theo.

Những thông số của DHT11:

- Phạm vi đo nhiệt độ: $0 - 50^{\circ}\text{C} \pm 2^{\circ}\text{C}$
- Phạm vi đo độ ẩm: $20\% - 90\% \pm 5\%$
- Tốc độ lấy mẫu tối đa 1HZ
- Nguồn cấp: 3 – 5V
- Dòng ra cực đại 2.5mA

Những lưu ý khi sử dụng module DHT11:

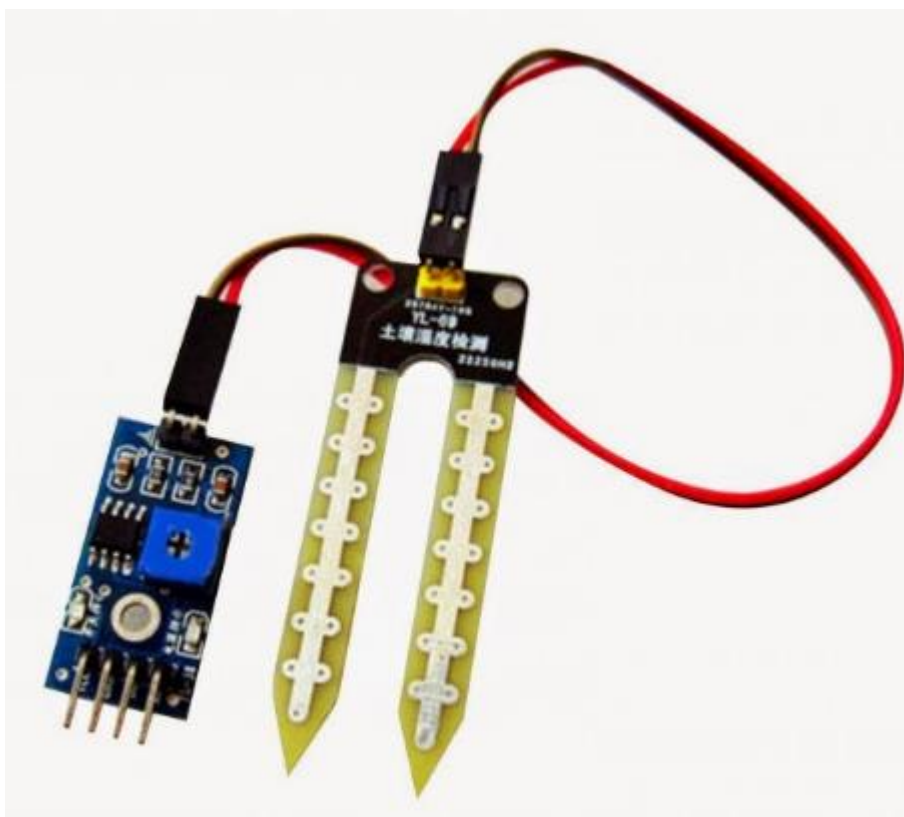
- Vì dùng giao tiếp một dây để giao tiếp giữa vi điều khiển và cảm biến nên ta phải cấu hình chân của của vi điều khiển vừa là input vừa là output. Giải đồ xung sau đây sẽ làm rõ việc cấu hình này.



Hình 17. Giải đồ xung mô tả việc truyền nhận dữ liệu giữa các cảm biến

- Vì thời gian gửi bit 0 và bit 1 là khác nhau nên ta cần sử dụng một timer để định thời
- Để thu được dữ liệu với sai số thấp nhất, nhà sản xuất khuyến khích nên đọc dữ liệu mỗi 2 giây.

1.3. Cảm biến độ ẩm đất FC – 28:



Hình 18. Cảm biến độ ẩm đất FC - 28

Lý do nhóm lựa chọn sử dụng cảm biến độ ẩm đất FC – 28:

- Giá thành rẻ, kích thước nhỏ gọn
- Hai đầu dò giúp dễ dàng cắm vào đất
- Có đầu ra Analog và Digital
- Độ nhạy có thể điều chỉnh bằng biến trở

Những thông số của cảm biến:

- Nguồn cấp: 3.3V - 5V
- Điện áp ngõ ra: 0 – 4.2V
- Dòng điện: 35mA
- Ngõ ra Digital trả về hai giá trị 0 hoặc 1
- Ngõ ra Analog giúp ta đọc chính xác giá trị độ ẩm đất
- Tuổi thọ không cao vì sử dụng trong môi trường ẩm ướt

Nguyên lý hoạt động của cảm biến: cảm biến độ ẩm đất dùng để đo độ ẩm của đất và các môi trường tương tự. Hai đầu dò cho phép dòng điện đi qua đất, lúc này cảm biến sẽ dựa vào điện trở của đất để xác định độ ẩm. Khi có nhiều nước, đất sẽ dẫn điện tốt hơn, nghĩa

là điện trở của đất sẽ thấp hơn. Đất khô dẫn điện kém, nghĩa là khi ít nước thì điện trở của đất sẽ cao hơn. Giá trị ngõ ra của cảm biến sẽ tỉ lệ nghịch với điện trở của đất.

1.4. Cảm biến cường độ ánh sáng MAX44009:



Hình 19. Cảm biến cường độ ánh sáng MAX44009

Cảm biến cường độ ánh sáng MAX44009 Digital Light Sensor được sử dụng để đo cường độ ánh sáng của môi trường xung quanh với độ chính xác và độ ổn định cao, cảm biến trả ra giá trị cường độ ánh sáng Lux qua giao tiếp I2C, thích hợp với các ứng dụng trong nông nghiệp, nhà thông minh, đo cường độ sáng...

Lý do nhóm lựa chọn sử dụng:

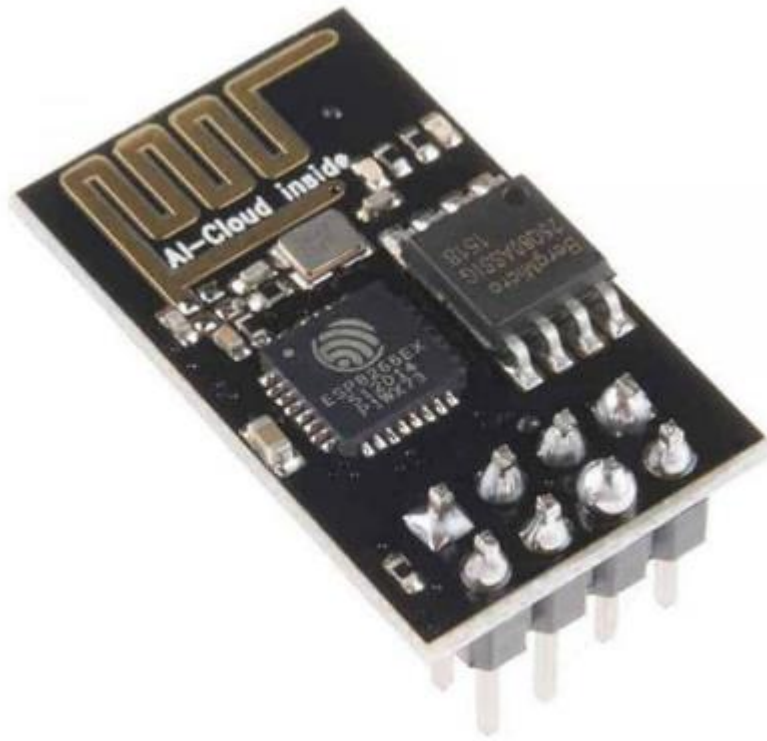
- Giá thành hợp lý
- Kích thước nhỏ gọn
- Tiêu thụ công suất thấp
- Tầm đo rộng

Thông số kỹ thuật:

- IC chính: MAX44009
- Điện áp sử dụng: 1.7~3.6VDC
- Điện áp giao tiếp: TTL 3.3~5VDC
- Chuẩn giao tiếp: I2C
- Cường độ ánh sáng đo được: 0.045~188,000 Lux

1.5. Module Wifi ESP – 01:

Mạch thu phát Wifi ESP8266 UART ESP-01S Ai-Thinker được sản xuất bởi Ai-Thinker sử dụng IC Wifi SoC ESP8266 của hãng Espressif, được sử dụng để kết nối với vi điều khiển thực hiện chức năng truyền nhận dữ liệu qua Wifi



Hình 20. ESP – 01

Lý do lựa chọn sử dụng ESP – 01:

- Module này được sử dụng rộng rãi trong các ứng dụng truyền nhận dữ liệu bằng

Wifi

- Kích thước nhỏ gọn
- Hỗ trợ nhiều chuẩn bảo mật
- Có nhiều chế độ hoạt động

Những thông số của ESP – 01:

- Nguồn cấp: 3.3V
- Dòng điện tối đa: 320mA
- Hỗ trợ chuẩn 802.11 b/g/n.

- Wi-Fi 2.4 GHz, hỗ trợ các chuẩn bảo mật như: OPEN, WEP, WPA_PSK, WPA2_PSK, WPA_WPA2_PSK.

- Hỗ trợ cả 2 giao tiếp TCP và UDP.
- Chuẩn giao tiếp UART với Firmware hỗ trợ bộ tập lệnh AT Command, tốc độ Baudrate mặc định 9600 hoặc 115200.

- Có 3 chế độ hoạt động: Client, Access Point, Both Client and Access Point

Đối với ESP – 01, nhà sản xuất khuyến khích sử dụng nguồn 3.3VDC rời (không dùng ngõ ra 3.3V từ vi điều khiển) để đảm bảo cung cấp được điện áp ổn định giúp module hoạt động một cách bình thường

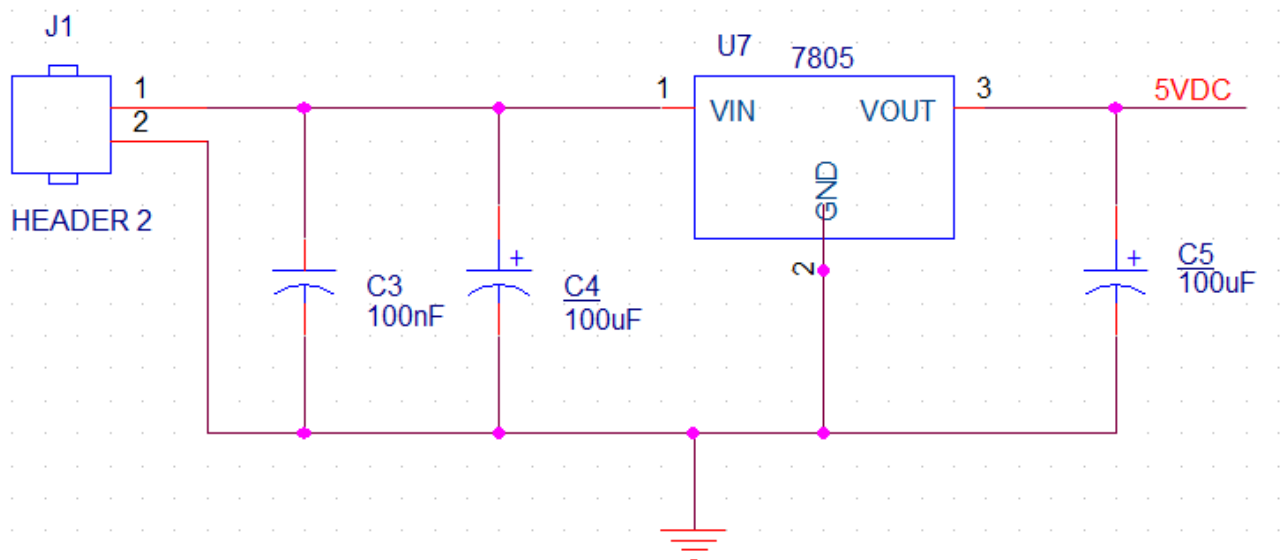
1.6. Mạch nguồn:

- Mạch nguồn 9VDC - 5VDC dùng LM7805:

Mạch nguồn bao gồm:

- Pin 9V : nguồn cấp tại ngõ vào
- LM7805: IC ổn áp với đầu ra 5V
- Tụ điện : 2 tụ điện phân cực 100uF, 1 tụ không phân cực 100nF dùng để lọc

nhiều, giúp điện áp ngõ ra ổn định

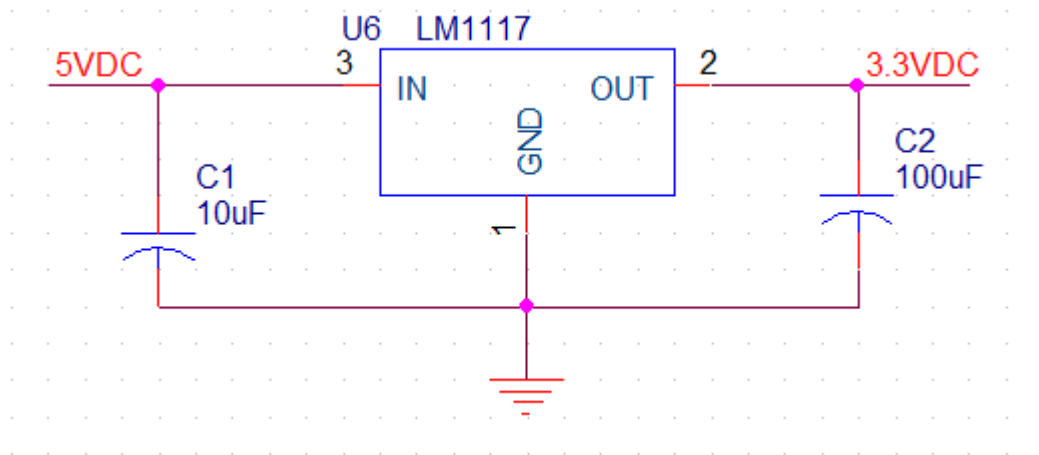


Hình 21. Mạch nguồn 9VDC – 5VDC dùng LM7805

- Mạch nguồn 5VDC – 3.3VDC dùng LM1117:

Mạch nguồn bao gồm:

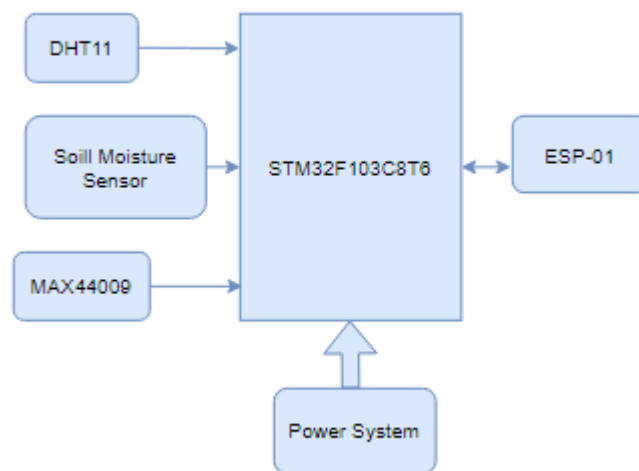
- Ngõ ra của mạch nguồn 9VDC - 5VDC dùng LM7805 là ngõ vào của mạch nguồn này
- LM1117: IC ổn áp với đầu ra 3.3V
- Tụ điện:



Hình 22. Mạch nguồn 5VDC – 3.3VDC dùng LM1117

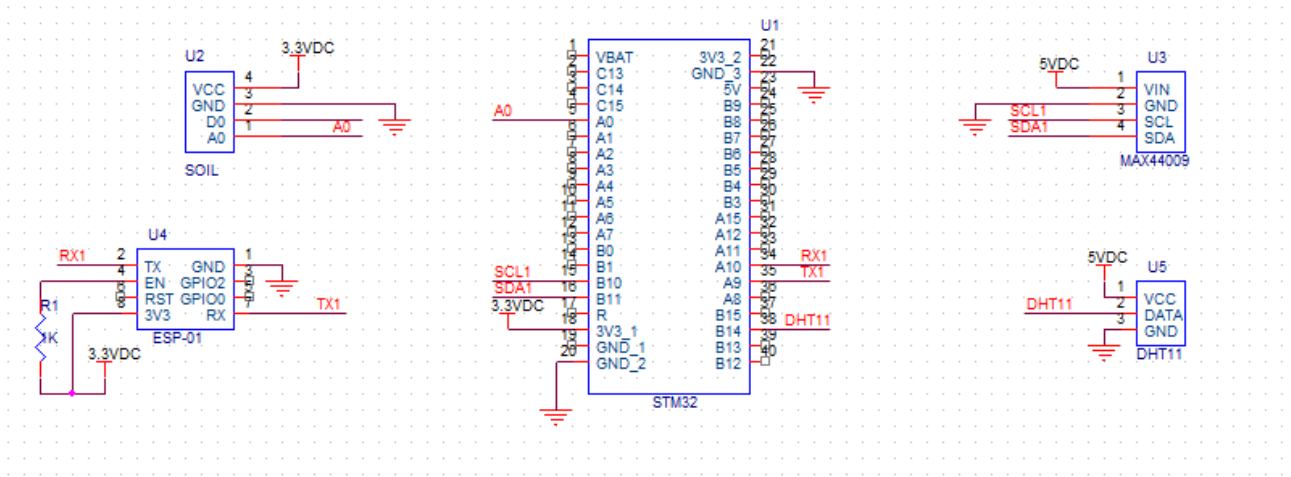
2. Sơ đồ khối, schematic và layout:

- Sơ đồ khối phần cứng:

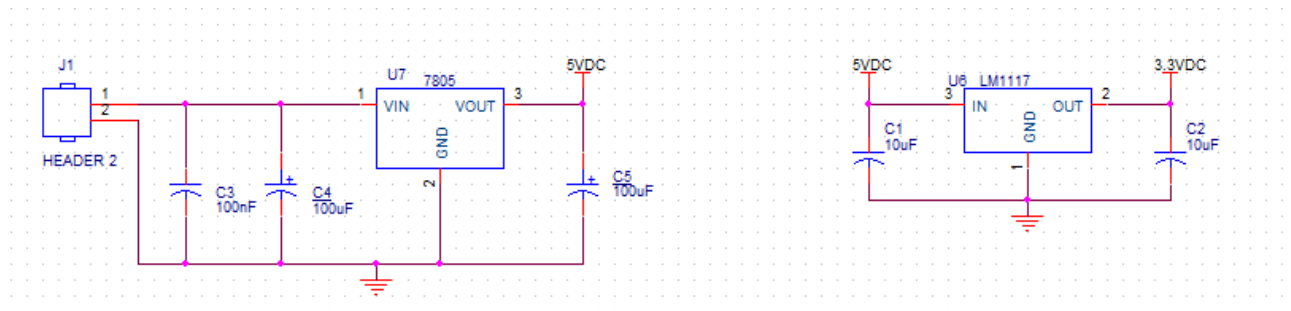


Hình 23. Sơ đồ khối phần cứng

- Schematic: được thực hiện bằng phần mềm Orcad

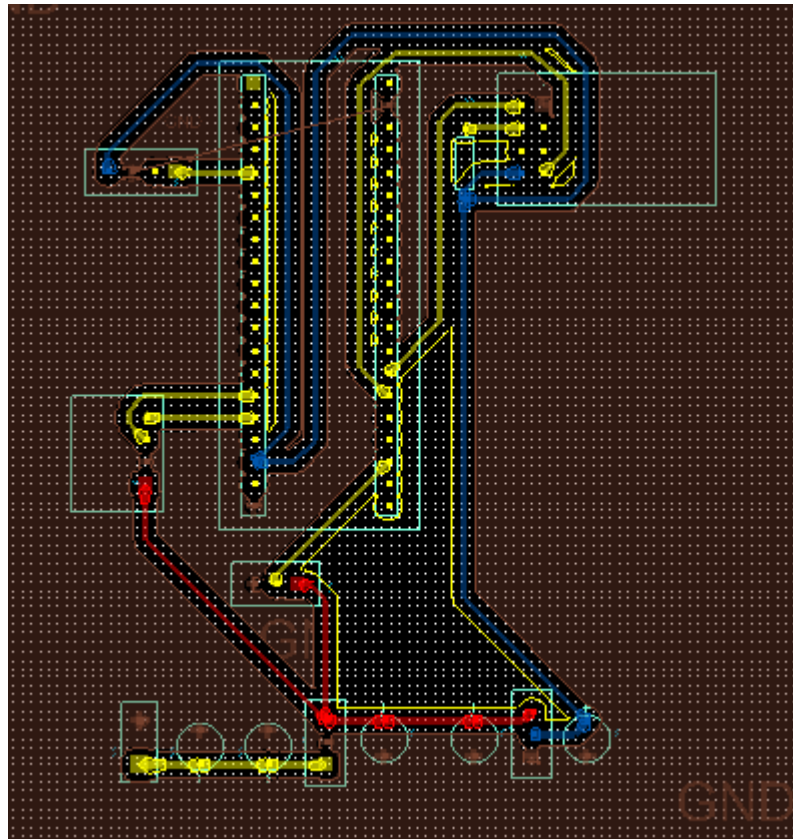


Hình 24. Khối xử lý



Hình 25. Khối nguồn

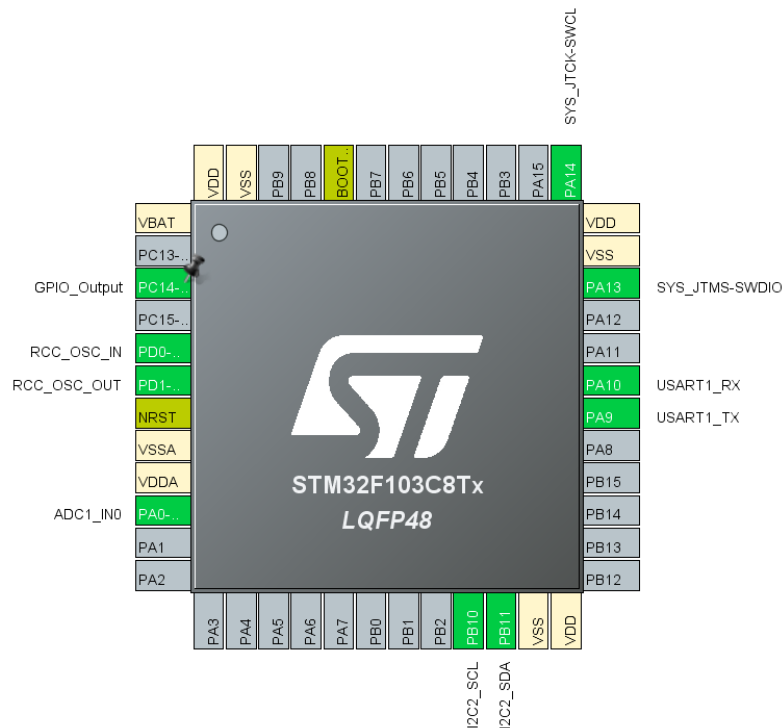
- PCB layout: được thực hiện bằng Allegro



Hình 26. PCB layout

II. Phần mềm

1. Cấu hình cho STM32F103 dùng CubeMX:



Hình 27. Cấu hình trên CubeMX

Cấu hình để đọc ADC

Sử dụng kênh 0 của ADC 1 để đọc, thời gian lấy mẫu là 239.5 chu kỳ máy.

▼ ADC_Regular_ConversionMode	
Enable Regular Conversions	Enable
Number Of Conversion	1
External Trigger Conversion Source	Regular Conversion launched by software
▼ Rank	1
Channel	Channel 0
Sampling Time	239.5 Cycles

Cấu hình để đọc giá trị cảm biến MAX44009

Đọc ở chế độ Standard Mode với tốc độ giao tiếp tối đa là 100 KHz.

▼ Master Features	
I2C Speed Mode	Standard Mode
I2C Clock Speed (Hz)	100000

Cấu hình để sử dụng UART 1

▼ Basic Parameters	
Baud Rate	9600 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1

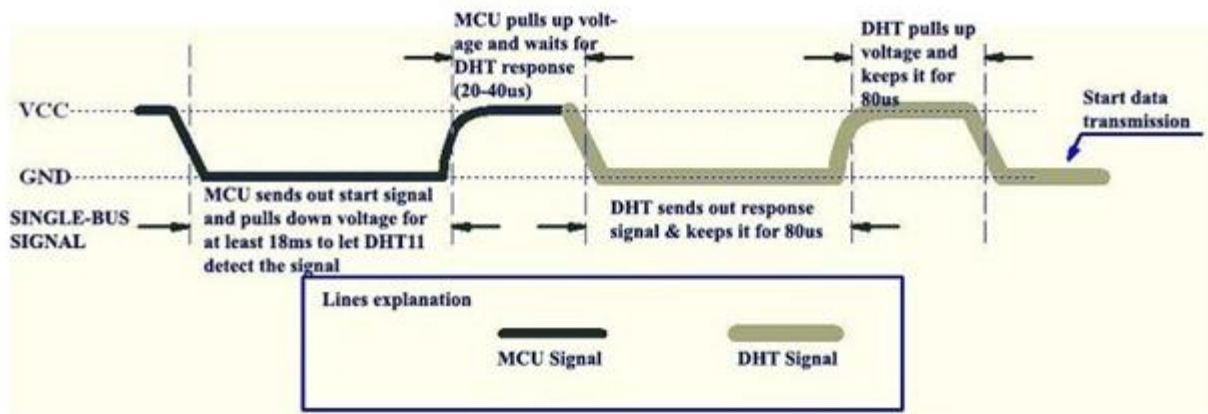
Tốc độ Baud là 9600 Bits/s, truyền theo khung gồm 1 bit Start, 8 bit dữ liệu (không có bit Parity), 1 bit Stop.

2. Đọc dữ liệu cảm biến và gửi lên Thingsboard thông qua giao thức MQTT:

2.1. Đọc và xử lý dữ liệu từ cảm biến nhiệt độ và độ ẩm môi trường DHT11

Để giao tiếp với DHT11 cần các bước như sau:

- **Bước 1:** vi điều khiển gửi tín hiệu thông báo với cảm biến về việc nó muốn nhận dữ liệu (tín hiệu Start) và cảm biến sẽ gửi tín hiệu phản hồi



Hình 28. Giải đồ xung việc gửi tín hiệu request và response giữa vi điều khiển và cảm biến

Ban đầu, chân nối với chân DATA cảm biến của vi điều khiển sẽ được cấu hình là Output. Chân này sẽ được đặt giá trị là 0 trong ít nhất là 18ms để cảm biến nhận thấy được tín hiệu Start

Sau đó, chân này được đặt lại giá trị là 1 và được cấu hình lại thành Input để chờ đọc dữ liệu

Sau khoảng 20us – 40us, DHT11 sẽ cho chân DATA của nó xuống mức thấp. Nếu > 40us mà chân DATA không được kéo xuống thấp nghĩa là không giao tiếp được với DHT11.

Chân DATA sẽ ở mức thấp trong 80us sau đó nó được DHT11 kéo nên cao trong 80us. Bằng việc giám sát chân DATA, vi điều khiển có thể biết được có giao tiếp được với DHT11 không. Nếu tín hiệu đo được DHT11 lên cao, khi đó hoàn thiện quá trình giao tiếp giữa vi điều khiển với DHT11

- **Bước 2:** đọc và xử lý dữ liệu gửi về từ cảm biến

Dữ liệu nhiệt độ, độ ẩm của DHT11 bao gồm 40bit, chia làm 5 byte như sau:

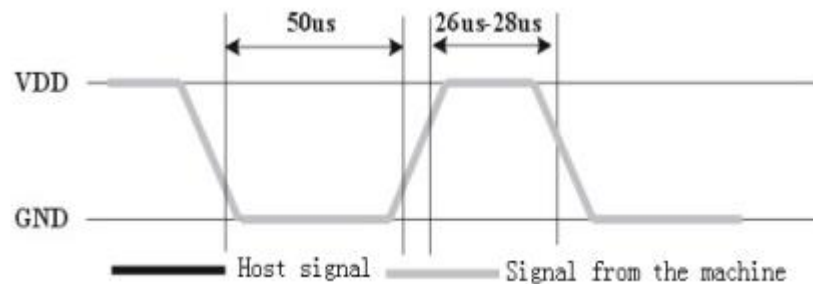
- Byte 1: giá trị phần nguyên của độ ẩm (RH%)
- Byte 2: giá trị phần thập phân của độ ẩm (RH%)

- Byte 3: giá trị phần nguyên của nhiệt độ (°C)
- Byte 4 : giá trị phần thập phân của nhiệt độ (°C)
- Byte 5 : kiểm tra lỗi

Nếu Byte 5 bằng tổng của 4 Byte trước thì giá trị nhiệt độ và độ ẩm đọc về là chính xác

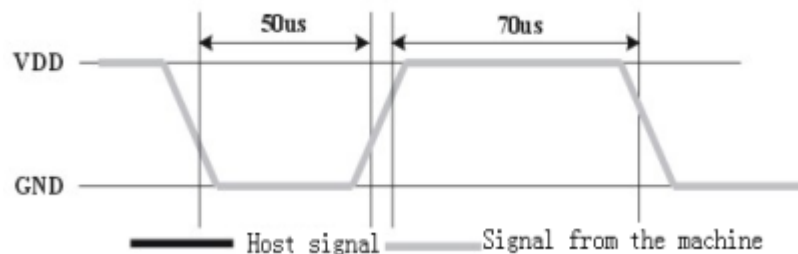
Sau khi giao tiếp được với DHT11, DHT11 sẽ gửi liên tiếp 40 bit 0 hoặc 1 về vi điều khiển.

- Bit 0:



Hình 29. Giải đồ xung của bit 0

- Bit 1:



Hình 30. Giải đồ xung của bit 1

Phần code để đọc nhiệt độ, độ ẩm từ DHT11

Khởi tạo các điều kiện đầu để giao tiếp với DHT11

```
void DHT_start(void) //khởi động dht
{
    Set_Pin_Output(GPIOB, GPIO_PIN_14);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, GPIO_PIN_RESET);
    delay_us(25000);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, GPIO_PIN_SET);
    Set_Pin_Input(GPIOB, GPIO_PIN_14);
}
```

Kiểm tra đã giao tiếp được với DHT11 hay không

```
char DHT_check(void)//check kết nối
{
    uint8_t Response=0;
    delay_us(40);
    if((HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_14))==0)
    {
        delay_us(80);
        if((HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_14))==1)
            {Response =1;}
        else
            {Response =0;}
    }
    while((HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_14))==1);
    return Response;
}
```

Đọc 5 byte của DHT sau khi giao tiếp thành công

```
char DHT_Byte(void)// đọc byte dht
{
    uint8_t i,j;
    for (j=0;j<8;j++)
    {
        while (!(HAL_GPIO_ReadPin (GPIOB,GPIO_PIN_14))); // wait for the pin to go high
        delay_us(50); // wait for 40 us
        if ((HAL_GPIO_ReadPin (GPIOB,GPIO_PIN_14))==0) // if the pin is low
        {
            i&= ~(1<<(7-j)); // write 0
        }
        else i|= (1<<(7-j)); // if the pin is high, write 1
        while ((HAL_GPIO_ReadPin (GPIOB,GPIO_PIN_14))==1); // wait for the pin to go low
    }
    return i;
}

void DHT_ReadByte(void)
{
    for(uint8_t i=0;i<5;i++)
    {
        byte[i]=DHT_Byte();
    }
}
```

Chuyển đổi giá trị từ byte sang số thực

2.2. Đọc và xử lý dữ liệu từ cảm biến độ ẩm đất FC – 28

Việc lập trình cho cảm biến độ ẩm đất gồm các bước sau:

- ADC được dùng có độ phân giải 12 bit.
- Khai báo để sử dụng module ADC.

```
HAL_ADC_Start(&hadc1);
```

- Đọc dữ liệu từ cảm biến về 10 lần liên tục, ta sẽ lấy giá trị trung bình để dữ liệu có độ tin cậy cao hơn.

- Công thức chuyển đổi.

ADC 12bit: $V_{in} = (V_{ref} * ADC) / 4096$ với V_{ref} là điện áp tham chiếu, ADC là giá trị sau chuyển đổi.

$$Value = (V_{in} / V_{ref}) * 100$$

```
uint8_t read_soil(uint8_t times)
{
    uint16_t raw=0;
    for(uint8_t t=1;t<=times;t++)
    {
        raw +=HAL_ADC_GetValue(&hadc1);
        HAL_Delay(10);
    }
    return ((raw*100)/(times*4096));
}
```

2.3. Đọc và xử lý dữ liệu từ cảm biến cường độ ánh sáng MAX44009:

Địa chỉ của MAX44009: 1001 010x hoặc 1001 011x

Lựa chọn địa chỉ để giao tiếp thông qua chân A0. Chân A0 kéo lên VCC để chọn địa chỉ 1001 011x hoặc kéo xuống thấp để chọn địa chỉ 1001 010x.

Các thanh ghi của MAX44009 mà chúng ta thao tác:

LUX READING											
Lux High Byte	E3	E2	E1	E0	M7	M6	M5	M4	0x03	0x00	R
Lux Low Byte	—	—	—	—	M3	M2	M1	M0	0x04	0x00	R

Lần lượt gửi yêu cầu đọc độ rọi sáng và nhận độ rọi sáng đến thanh ghi Lux High Byte (0x03) và thanh ghi Lux Low Byte (0x04) bằng hàm `HAL_I2C_Master_Transmit` và hàm `HAL_I2C_Master_Receive`.

```
void MAX44009_start(void)
{
    uint8_t Tx1=0x03;
    HAL_I2C_Master_Transmit(&hi2c2,0x94,&Tx1,1,500);
    HAL_I2C_Master_Receive(&hi2c2,0x94,&data[0],1,500);
    uint8_t Tx2=0x04;
    HAL_I2C_Master_Transmit(&hi2c2,0x94,&Tx2,1,500);
    HAL_I2C_Master_Receive(&hi2c2,0x94,&data[1],1,500);
}
```

Chuyển đổi giá trị độ rọi sáng đọc được (dạng byte) thành giá trị số thực

```
float MAX44009_read(void)
{
    MAX44009_start();
    HAL_Delay(800);
    uint8_t exponent = (data[0] & 0xF0) >> 4;
    uint8_t mantissa = ((data[0] & 0x0F) << 4) | (data[1] & 0x0F);
    float luminance = pow(2, exponent) * mantissa * 0.045;
    return luminance;
}
```

2.4. Lập trình cho module Wifi:

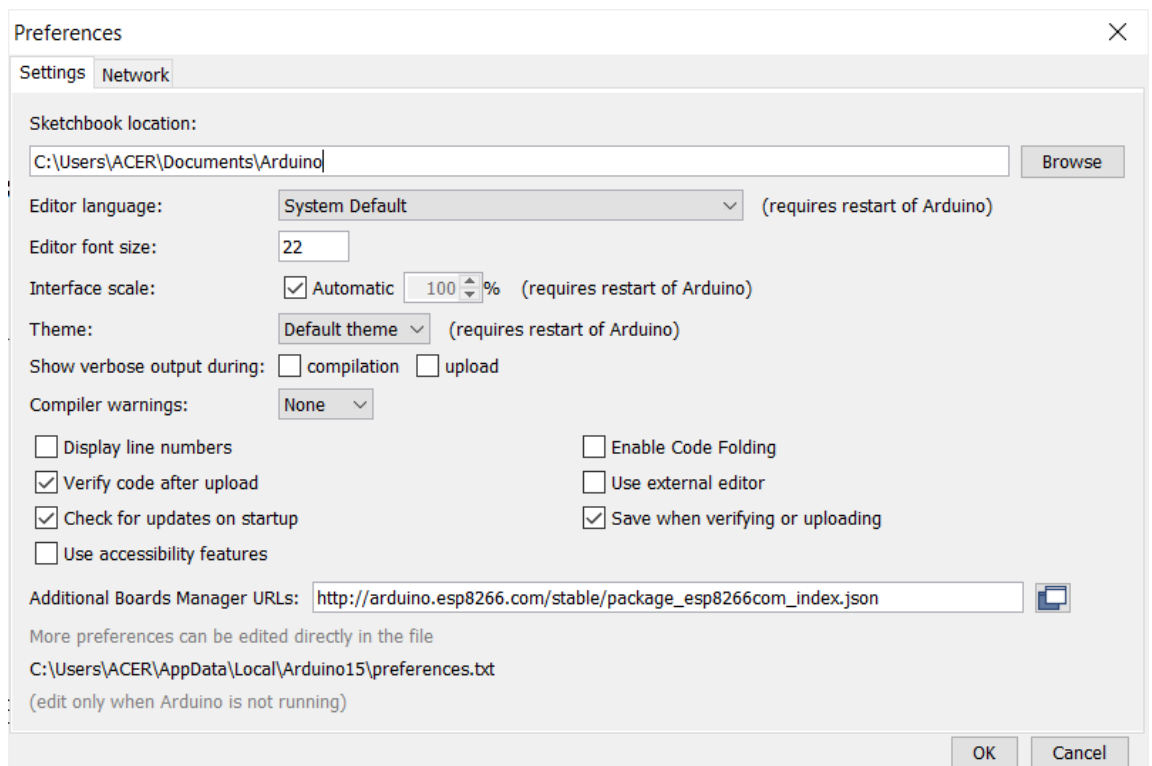
Phần 1: Cấu hình Arduino IDE

Đầu tiên, cài đặt thư viện vào IDE

Vào **File** → **Preferences**, vào textbox Additional Board Manager URLs thêm đường link sau vào.

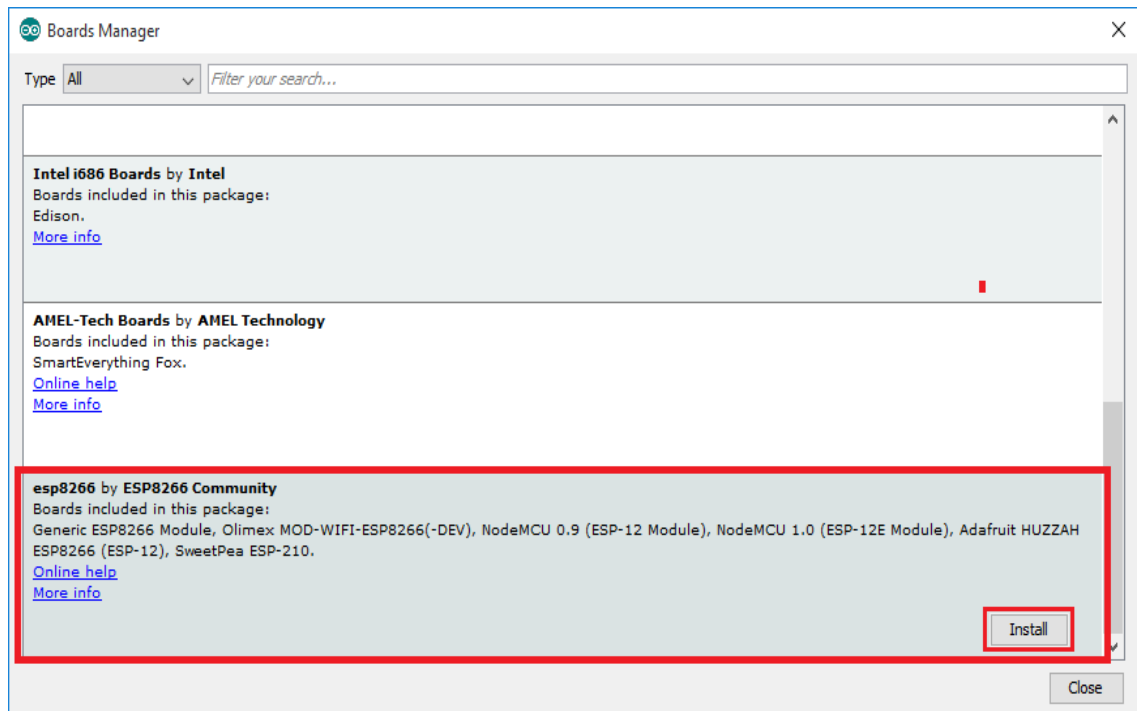
http://arduino.esp8266.com/stable/package_esp8266com_index.json

Sau đó nhấn OK để chấp nhận.



Tiếp theo vào **Tool** → **Board** → **Boards Manager**

Ta nhập vào ô tìm kiếm ESP8266 và chọn ESP8266 by ESP8266 Community, nhấp vào Install. Chờ phần mềm tự động download và cài đặt.

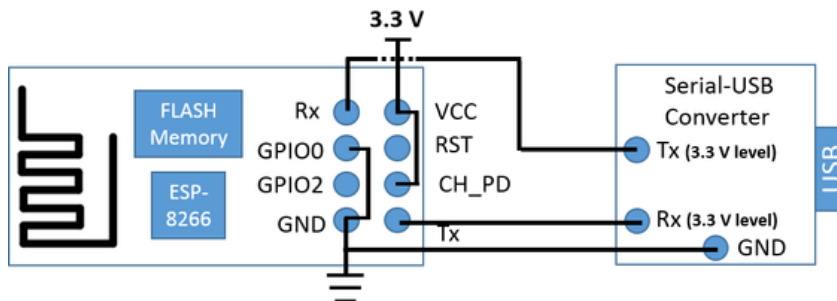


Chọn Board để lập trình cho ESP8266:

Kết nối module USB-to-UART vào máy tính. Vào **Tool**→**Board**→**Generic ESP8266 Module**, chọn cổng COM tương ứng với module USB-to-UART tương ứng.

Phần 2: Cấu hình phần cứng và nạp code

Sơ đồ cắm ESP với USB Uart



Phần chương trình nạp cho ESP-01

Sử dụng thư viện PubSubClient, tạo một client tên là espClient.

```
/*khởi động thư viện Pubsubclient*/
WiFiClient espClient;
PubSubClient client(espClient);
```

Cấu hình kết nối wifi. Đăng ký Mosquitto broker ở port 1883.

```

void setup()
{
    Serial.begin(9600);
    /*Cấu hình để kết nối wifi*/
    setup_wifi();
    /*đăng ký Broker server Mosquitto và TCP port 1883*/
    client.setServer(mqtt_server,1883);
}

```

Kiểm tra dữ liệu đến, chuyển đổi từ byte sang dạng số thực.

```

if (Serial.available() > 0)
{
    Serial.readBytes(dem, 4); //read soil
    soil = Bytes2float(&dem[0]);
    Serial.readBytes(dem, 4); //read temperature
    temp = Bytes2float(&dem[0]);
    Serial.readBytes(dem, 4); //read humidity
    humid = Bytes2float(&dem[0]);
    Serial.readBytes(dem, 4); //read lux
    lux = Bytes2float(&dem[0]);
}

```

Tạo chuỗi json chứa các thông tin (giá trị của cảm biến, tên topic trên broker mà ta muốn publish dữ liệu tới).

```

/*Prepare a JSON payload string*/
String payload = "{";
payload += "\"soil\":\"";
payload += soil;
payload += ",";
payload += "\"temperature\":\"";
payload += temp;
payload += ",";
payload += "\"humidity\":\"";
payload += humid;
payload += ",";
payload += "\"lux\":\"";
payload += lux;
payload += "\"}";
/*Send payload*/
char telemetry[200];
payload.toCharArray(telemetry, 200);
client.publish("Temperature", telemetry);

```

2.5. Gửi dữ liệu từ STM32 sang module ESP -01:

Ta gửi dữ liệu (dạng byte) bằng chuẩn UART. Dùng hàm float2Bytes để chuyển đổi số thực thành byte.

```
void senddata(void)
{
    temp=(float)byte[2]+(float)byte[3]/10;;
    float2Bytes(&a[0],temp);
    HAL_UART_Transmit(&huart1,(uint8_t*)a,4,1000);
    humid=(float)byte[0]+(float)byte[1]/10;
    float2Bytes(&a[0],humid);
    HAL_UART_Transmit(&huart1,(uint8_t*)a,4,1000);
    soil=read_soil(10);
    float2Bytes(&a[0],soil);
    HAL_UART_Transmit(&huart1,(uint8_t*)a,4,1000);
    lux=MAX44009_read();
    float2Bytes(&a[0],lux);
    HAL_UART_Transmit(&huart1,(uint8_t*)a,4,1000);
}
```

2.6. Cấu hình cho Thingsboard và tạo Dashboard:

- Cài đặt và cấu hình Thingsboard:

-Ta có thể sử dụng Thingsboard trên :

+Bản Live Demo của Thingsboard : <https://demo.thingsboard.io/>

+ Cài đặt trực tiếp trên máy tính sử dụng hệ điều hành Window, Ubuntu..

+Cài đặt trên máy tính nhúng: Raspberry

- Sau đây là các bước để cài đặt Thingsboard trên Ubuntu Server:

Bước 1: Cài đặt Java 8

```
sudo apt update
sudo apt install openjdk-8-jdk
```

Bước 2: Cài đặt Thingsboard service

Tải gói Thingsboard từ Github:

wget

<https://github.com/thingsboard/thingsboard/releases/download/v3.0.1/thingsboard-3.0.1.deb>

Cài đặt Thingsboard

```
sudo dpkg -i thingsboard-3.0.1.deb
```

Bước 3: Tạo database cho Thingsboard

Sử dụng SQL để tạo database cho Thingsboard

Các bước cài đặt PostgreSQL

*# install **wget** if not already installed:*

```
sudo apt install -y wget
```

import the repository signing key:

```
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc |  
sudo apt-key add -
```

add repository contents to your system:

```
RELEASE=$(lsb_release -cs)  
echo "deb http://apt.postgresql.org/pub/repos/apt/ ${RELEASE}"-pgdg main  
| sudo tee /etc/apt/sources.list.d/pgdg.list  
# install and launch the postgresql service:
```

```
sudo apt update
```

```
sudo apt -y install postgresql-12
```

```
sudo service postgresql start
```

Sau khi PostgreSQL được tạo xong, chúng ta kết nối với database để tạo database cho Thingsboard

```
psql -U postgres -d postgres -h 127.0.0.1 -W  
CREATE DATABASE thingsboard;  
\q
```

Bước 4: Cấu hình cho Thingsboard

Điều hướng tới file cấu hình cho Thingsboard

```
sudo nano /etc/thingsboard/conf/thingsboard.conf
```

Thêm các dòng sau vào file:

DB Configuration

```
export DATABASE_ENTITIES_TYPE=sql
```

```
export DATABASE_TS_TYPE=sql
```



```

export
SPRING_JPA_DATABASE_PLATFORM=org.hibernate.dialect.PostgreSQLDialect

export SPRING_DRIVER_CLASS_NAME=org.postgresql.Driver

export
SPRING_DATASOURCE_URL=jdbc:postgresql://localhost:5432/thingsboard

export SPRING_DATASOURCE_USERNAME=postgres

export
SPRING_DATASOURCE_PASSWORD=PUT_YOUR_POSTGRESQL_PASSWORD_HERE

export SPRING_DATASOURCE_MAXIMUM_POOL_SIZE=5

# Specify partitioning size for timestamp key-value storage. Allowed values: DAYS, MONTHS, YEARS, INDEFINITE.

export SQL_POSTGRES_TS_KV_PARTITIONING=MONTHS

# Update ThingsBoard memory usage and restrict it to 256MB in /etc/thingsboard/conf/thingsboard.conf

export JAVA_OPTS="$JAVA_OPTS -Xms256M -Xmx256M"

```

Thay “PUT_YOUR_POSTGRESQL_PASSWORD_HERE” bằng password đã đặt khi cài đặt POSTGRESQL

Bước 5:

Sau khi cài đặt dịch vụ Thingsboard và tạo database cho Thingsboard.

Ta chạy script:

```

# --loadDemo option will load demo data: users, devices, assets, rules, widgets.
sudo /usr/share/thingsboard/bin/install/install.sh --loadDemo

```

Sau đó thực thi câu lệnh sau để bắt đầu chạy Thingsboard

```

sudo service thingsboard start

```

Sau khi cài đặt thành công ta mở giao diện web bằng đường link sau:

```

http://localhost:8080/

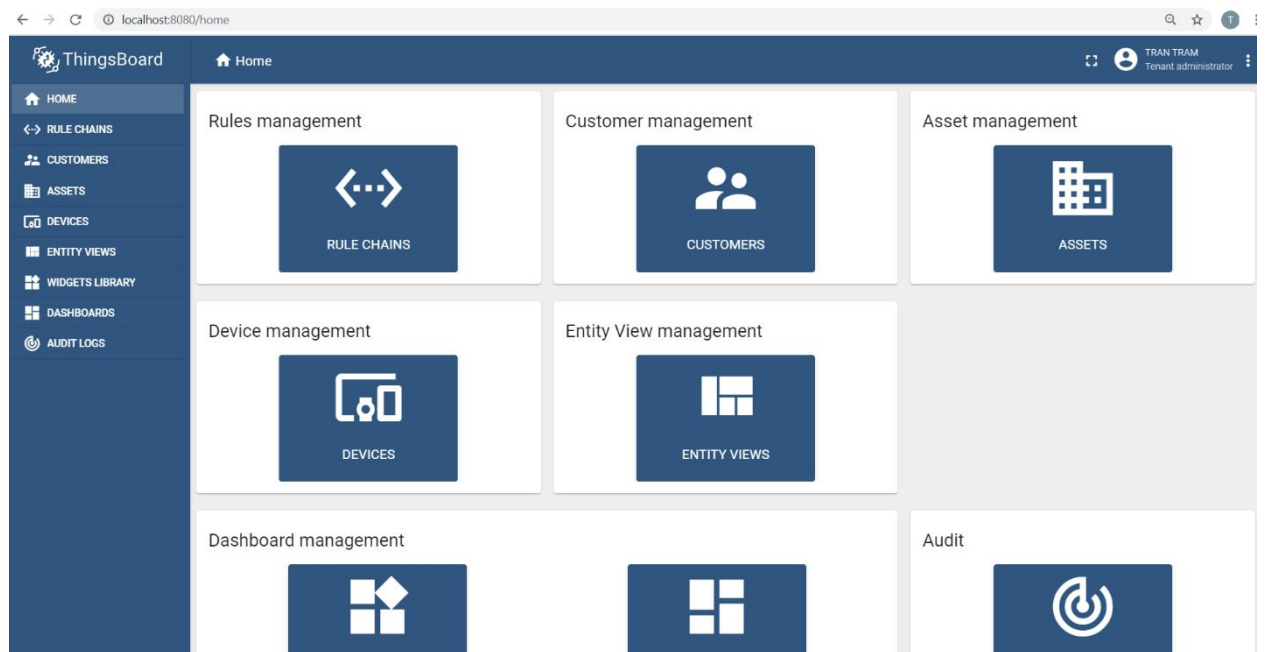
```

Đăng nhập vào Thingsboard bằng tài khoản như sau:

User: tenant@thingsboard.org

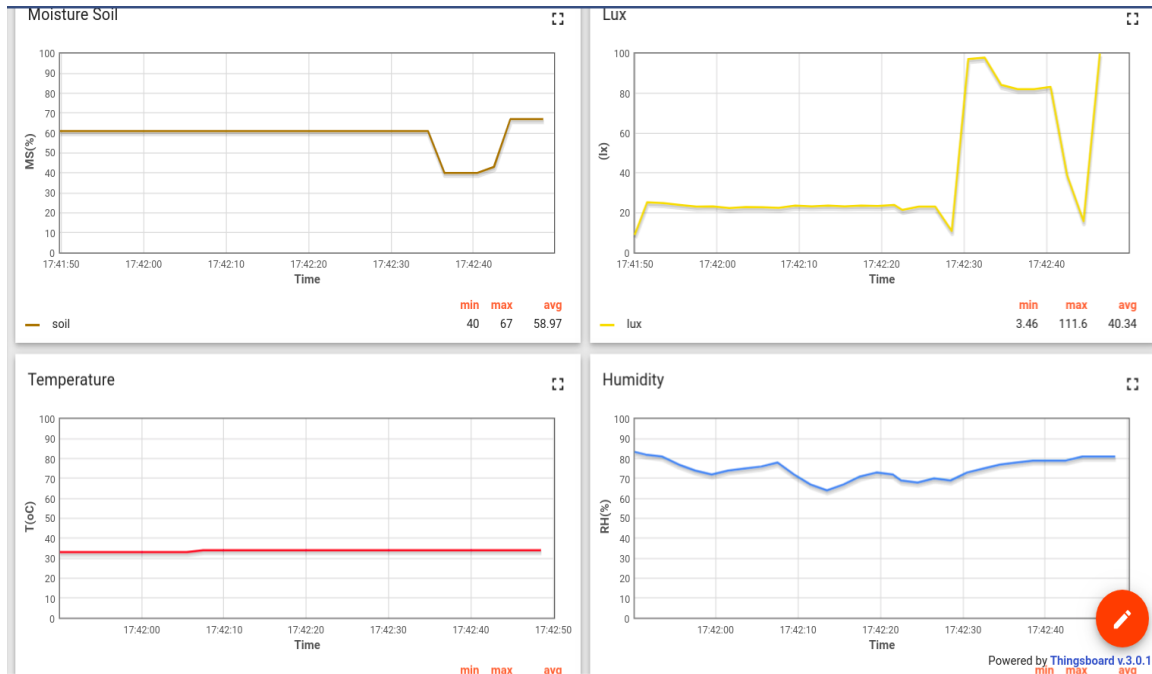
Password:tenant

A login form on a dark blue background. It features two input fields: 'Username (email) *' with an envelope icon and 'Password' with a lock icon. Below the email field is a red error message 'Invalid email format.'. To the right of the password field is a link 'FORGOT PASSWORD?'. Below these fields are three buttons: an orange 'LOGIN' button, a light purple 'CREATE AN ACCOUNT' button, and a red 'LOGIN WITH GOOGLE' button with the Google 'G' logo. Between the 'LOGIN' and 'CREATE AN ACCOUNT' buttons is the text 'Do not have an account?'. Below the 'CREATE AN ACCOUNT' button is the text 'OR'.

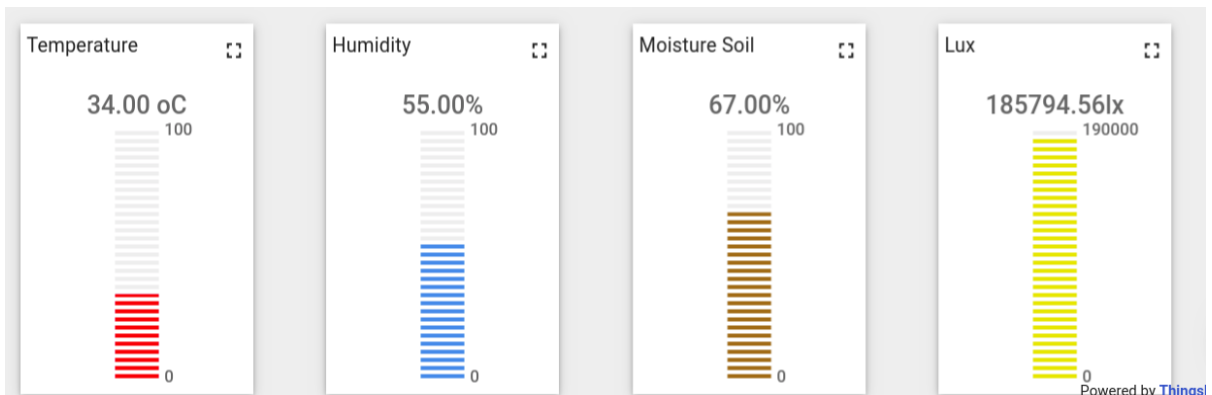


Hình 31. Giao diện Thingsboard

- Tạo Dashboard:



Hình 32. Dashboard biểu diễn đồ thị dạng đường



Hình 33. Dashboard biểu thị đồ thị dạng cột

2.7. Cấu hình Gateway – bên trung gian đóng vai trò kết nối MQTT Broker và Thingsboard

Ở project này: Chúng em sử dụng Thingsboard gateway để lấy dữ liệu từ Mosquitto broker và hiển thị dữ liệu đã lấy lên dashboard trên Thingsboard Server.

Các bước cài đặt gateway

Bước 1: Cài đặt Gateway

Cài đặt Java 8 (Bước cài đặt thingsboard đã làm)

Tải package:

```
resources/tb-gateway-ubuntu-download.sh
```

```
wget https://github.com/thingsboard/thingsboard-gateway/releases/download/v1.4.0.1/tb-gateway-1.4.0.deb
```

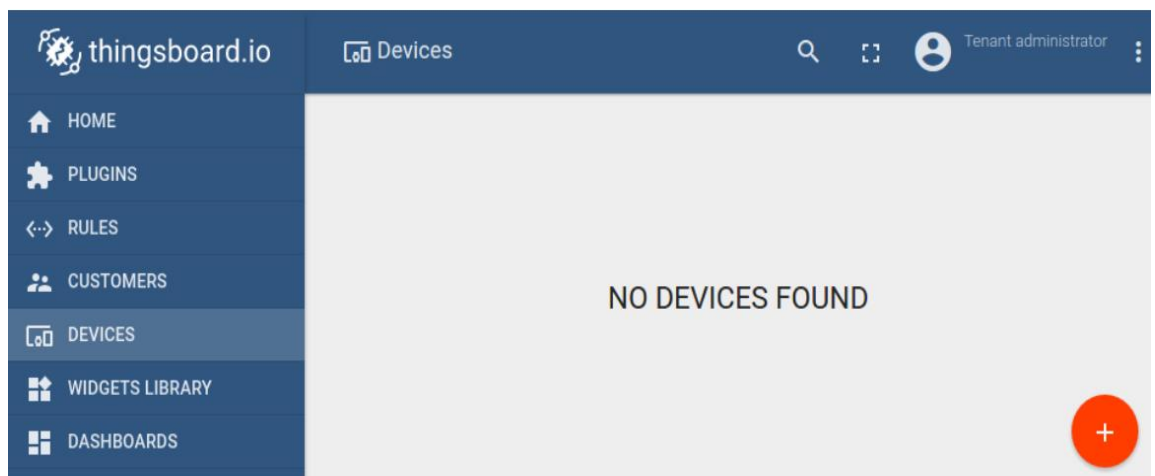
Cài đặt

```
resources/tb-gateway-ubuntu-installation.sh
```

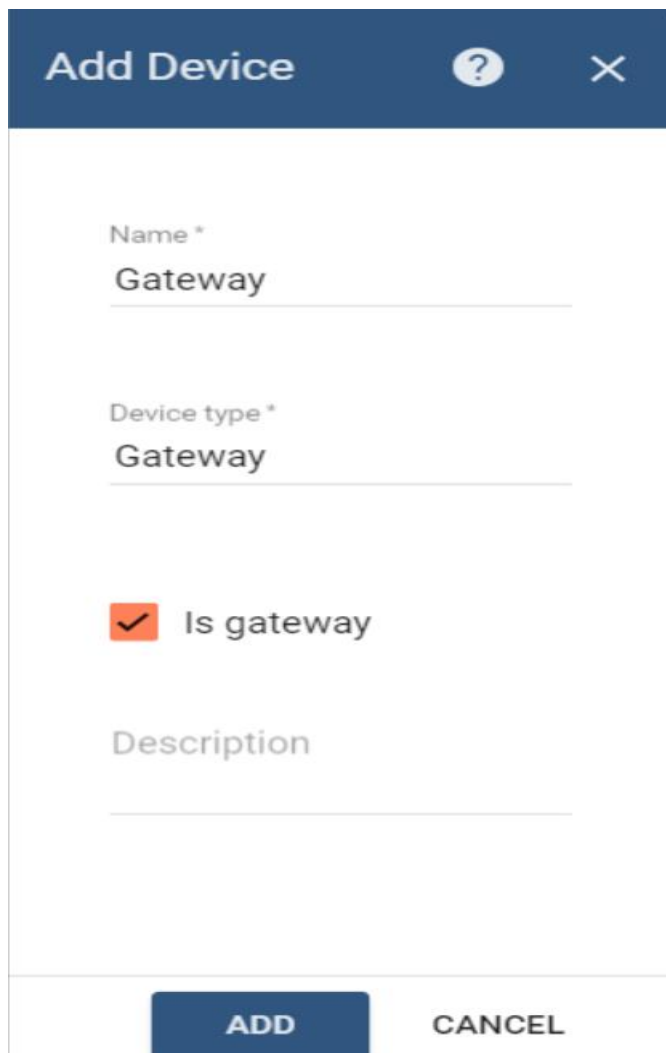
```
sudo dpkg -i tb-gateway-1.4.0.deb
```

Bước 2: Khởi tạo 1 gateway trên Thingsboard

Open Devices and click on big red “+” button in the bottom right corner.

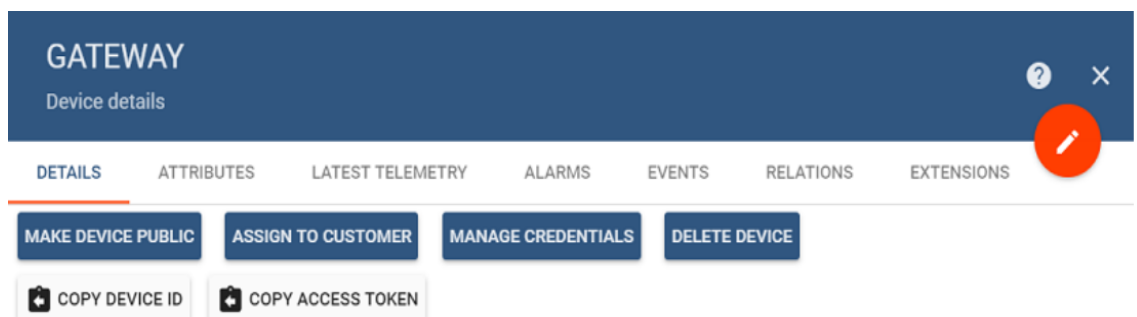


Điền các mục như sau



The image shows a modal window titled "Add Device" with a question mark icon and a close button. It contains four input fields: "Name *" with the value "Gateway", "Device type *" with the value "Gateway", "Is gateway" which is checked with an orange checkbox, and "Description" which is empty. At the bottom are two buttons: "ADD" and "CANCEL".

Mở gateway vừa tạo và click on “Copy Access Token” button. Chúng ta sẽ sử dụng mã TOKEN cho việc cấu hình gateway sau này



The image shows the "GATEWAY" device details page. The header is "GATEWAY" with "Device details" below it. There are tabs for "DETAILS", "ATTRIBUTES", "LATEST TELEMETRY", "ALARMS", "EVENTS", "RELATIONS", and "EXTENSIONS". Below the tabs are four buttons: "MAKE DEVICE PUBLIC", "ASSIGN TO CUSTOMER", "MANAGE CREDENTIALS", and "DELETE DEVICE". At the bottom are two buttons: "COPY DEVICE ID" and "COPY ACCESS TOKEN". A red circular icon with a pencil is visible on the right side of the page.

Bước 3: Cấu hình cho gateway

Điều hướng tới file `tb-gateway.yml` theo đường dẫn `/etc/tb-gateway/conf`

Thay đổi gateway.connection.host thành tên Thingsboard server, gateway.connection.port thành tên port đang lắng nghe thingsboard và gateway.connection.security.accessToken thành mã truy cập TOKEN đã lưu.

```
gateways:
  tenants:
    -
      label: "Tenant"
      reporting:
        interval: 60000
      persistence:
        type: file
        path: storage
        bufferSize: 1000
      connection:
        host: "${GATEWAY_HOST:YOUR_HOST}"
        port: 1883
        retryInterval: 3000
        maxInFlight: 1000
      security:
        accessToken: "${GATEWAY_ACCESS_TOKEN:YOUR_TOKEN}"
      remoteConfiguration: true

server:
  address: "0.0.0.0"
  port: "9090"

updates:
  enabled: "${UPDATES_ENABLED:true}"
```

Bước 4: Kết nối với MQTT broker

Điều hướng tới file mqtt-config.json theo đường dẫn /etc/tb-gateway/conf

Chỉnh sửa một số thuộc tính trong file

```
{
  "brokers": [
    {
      "host": "localhost",
      "port": 1883,
      "ssl": false,
      "retryInterval": 3000
      ...
    }
  ]
}
```

Host: tên máy chủ mqtt broker

Port: port lắng nghe mqtt broker

```

{
  ...
  "mapping": [
    {
      "topicFilter": "sensors",
      "converter": {
        "type": "json",
        "filterExpression": "",
        "deviceNameJsonExpression": "${$.serialNumber}",
        "timeout": 60000,
        "attributes": [
          {
            "type": "string",
            "key": "model",
            "value": "${$.model}"
          }
        ],
        "timeseries": [
          {
            "type": "double",
            "key": "temperature",
            "value": "${$.temperature}"
          }
        ]
      }
    }
  ],
  ...
}

```

topicFilter: tên device được tạo ra khi gateway nhận được dữ liệu

type: kiểu dữ liệu nhận của gateway

key: tên của dữ liệu nhận được

value: giá trị của dữ liệu

Sau khi chỉnh sửa 2 file trên, ta hoàn thành việc cấu hình cho gateway để nhận dữ liệu từ Mosquitto broker và đưa dữ liệu nhận được lên Thingsboard server.

Bước 5: Khởi động gateway

```

sudo service tb-gateway start

```

(Gateway nhận được dữ liệu từ Mosquitto broker và gửi dữ liệu đó lên Thingsboard server. Gateway tự tạo ra device tương ứng với tên “topicFilter” đã cấu hình trong file mqtt-config.json)

3. Giao diện Python:

Giao diện Python biến máy tính cá nhân thành một MQTT Client mà cụ thể là giúp máy tính thực hiện chức năng của một Subscriber.

Giao diện được xây dựng dựa trên ngôn ngữ Python với sự hỗ trợ của gói Paho, PyQt5 và một số gói cần thiết khác dùng để hiển thị dữ liệu thu được từ các cảm biến trên máy tính cá nhân.

Gói PyQt5 giúp xây dựng một giao diện đơn giản một cách dễ dàng và trực quan. Với sự hỗ trợ của gói PyQt5, chúng ta có hai lựa chọn để tạo ra một giao diện với những thành phần theo ý muốn:

- Kéo thả các khối thông qua phần mềm Qt Creator (Qt Creator là một file với định dạng Application mà được hỗ trợ sẵn khi ta cài đặt Python 3.8 về máy. Muốn sử dụng nó ta chỉ cần vào thư mục nơi cài đặt Python, tìm và mở file Qt Creator). Qt Creator cung cấp đầy đủ các thành phần mà ta có thể tìm thấy trên bất cứ một phần mềm máy tính nào như các tab, thanh công cụ, hộp thoại,...

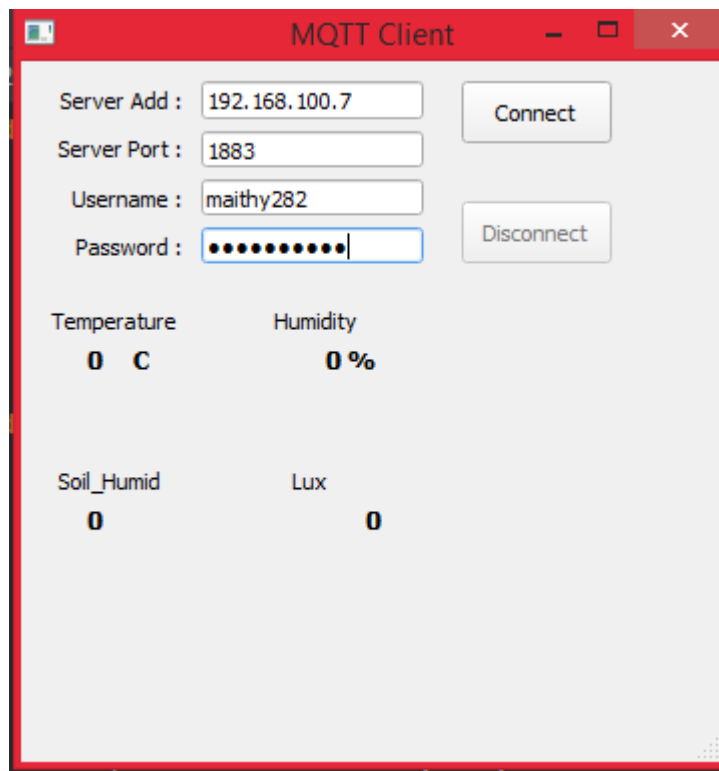
⇒ Cách làm này dễ dàng thực hiện, thích hợp khi xây dựng những giao diện phức tạp, thực hiện nhiều chức năng và không cần dùng đến quá nhiều code Python.

- Viết chương trình bằng ngôn ngữ Python. Sau khi import gói PyQt5, ta sẽ được hỗ trợ các lệnh để tạo từng thành phần của giao diện như tạo nút nhấn, nhãn, hộp thoại và tọa độ mà ta muốn đặt những thành phần đó.

⇒ Nhóm chọn thực hiện giao diện bằng cách này vì nó sẽ giúp ta hiểu được cụ thể một thành phần trên giao diện được tạo ra như thế nào, cần những bước cấu hình nào. Hơn nữa, giao diện gồm hai nhiệm vụ là thiết lập kết nối với Broker và hiển thị. Việc sử dụng code Python giúp ta dễ dàng quản lý và kết hợp hai nhiệm vụ này lại với nhau.

Gói Paho sẽ hỗ trợ trong khâu thiết lập kết nối với Broker. Gói gồm đầy đủ các lệnh cho thiết lập Client, kết nối với Broker và đọc dữ liệu từ Broker.

Giao diện Python được xây dựng đơn giản như một cửa sổ, cửa sổ này sẽ xuất hiện ngay sau khi code Python được biên dịch không lỗi và chạy.



Hình 34. giao diện Python

Giao diện gồm bốn khung để người dùng có thể nhập thông tin vào bao gồm:

- Server Add: địa chỉ của MQTT Broker. Trong đồ án này, nhóm lựa chọn phương án cài đặt Broker Mosquitto trên máy tính cá nhân nên tại khung này sẽ nhập địa chỉ IPv4
- Server Port: sử dụng port mặc định của giao thức MQTT là 1883
- User name và Password: do người dùng cài đặt nhằm tăng thêm tính bảo mật dữ liệu của hệ thống. Việc cài đặt User name và Password ta phải cấu hình từ phía Broker để xin phép sử dụng hai thông số này trong việc kết nối giữa Client và Broker.

Giao diện có hai nút nhấn Connect và Disconnect. Người dùng sẽ nhấn nút Connect để kết nối với MQTT Broker, nếu kết nối thành công thì ngay khi có dữ liệu được gửi lên Broker, chúng cũng đồng thời được hiển thị lên giao diện. Trong một thời điểm chỉ có một nút được cho phép nhấn. Ngay khi chương trình được khởi chạy thì chỉ có nút Connect được phép nhấn (tô đậm) còn nút Disconnect sẽ bị cấm nhấn (bị làm nhạt đi). Sau khi kết nối được thiết lập thành công thì nút Connect sẽ bị cấm.

Dữ liệu từ cảm biến nào sẽ được hiển thị tại nhãn tương ứng. Có bốn nhãn tương ứng với dữ liệu về nhiệt độ (Temperature), độ ẩm (Humidity), độ ẩm đất (Soil_Humid) và cường độ ánh sáng (Lux). Ban đầu chưa có dữ liệu thì giá trị của các nhãn này được đặt là 0.

III. Sản phẩm hoàn thiện:



Hình 35. Sản phẩm hoàn thiện

ĐÁNH GIÁ HOẠT ĐỘNG CỦA HỆ THỐNG

Hệ thống đáp ứng được cơ bản yêu cầu thu thập dữ liệu từ cảm biến và gửi đến người dùng (MqttGUI, Thingsboard Plattform) thông qua giao thức MQTT.

Tuy nhiên, hệ thống vẫn còn một vài chỗ chưa đạt được

- Giao diện Python chỉ đọc được dữ liệu tại một thời điểm nhất định
- Lưu giá trị đọc được từ cảm biến trong khoảng thời gian do người dùng cài đặt (tuần, tháng...)
- Hạn chế về bảo mật
- Thời gian đáp ứng còn chậm

PHẦN KẾT

Chúng em xin cảm ơn quý Thầy/Cô đã đọc đề án này. Do những hạn chế về kiến thức nên đề án này của chúng em không thể tránh khỏi những thiếu sót, chúng em mong nhận được sự góp ý từ phía Thầy/Cô.

Trong tương lai, nhóm chúng em sẽ cố gắng khắc phục những hạn chế hiện tại và phát triển hệ thống của mình tối ưu hơn và có nhiều tính năng thiết thực hơn nữa. Trong quá trình tìm hiểu, chúng em nhận thấy hệ thống của nhóm có thể trở thành một phần của hệ thống dự báo thời tiết thông minh sử dụng trong mô hình nhà thông minh hoặc vườn thực vật thông minh. Để hệ thống hoàn thiện và được sử dụng trong những trường hợp thực tế nói trên hệ thống của nhóm cần dùng nhiều loại cảm biến hơn, tăng thời gian đáp ứng và thiết kế lại bộ nguồn thích hợp cho sử dụng ở nhiều môi trường khác nhau.

PHỤ LỤC

I. Code đọc dữ liệu các cảm biến, đưa về STM32F103C8T6 xử lý:

```
/* -----Soil Functions----- */
uint8_t read_soil(uint8_t times)
{
    uint16_t raw=0;
    for(uint8_t t=1;t<=times;t++)
    {
        raw +=HAL_ADC_GetValue(&hadc1);
        HAL_Delay(10);
    }
    return ((raw*100)/(times*4096));
}

/* -----MAX44009 Functions----- */
void MAX44009_start(void)
{
    uint8_t Tx1=0x03;
    HAL_I2C_Master_Transmit(&hi2c1,0x94,&Tx1,1,500);
    HAL_I2C_Master_Receive(&hi2c1,0x94,&data[0],1,500);
    uint8_t Tx2=0x04;
    HAL_I2C_Master_Transmit(&hi2c1,0x94,&Tx2,1,500);
    HAL_I2C_Master_Receive(&hi2c1,0x94,&data[1],1,500);
}

float MAX44009_read(void)
{
    MAX44009_start();
    HAL_Delay(800);
    uint8_t exponent = (data[0] & 0xF0) >> 4;
    uint8_t mantissa = ((data[0] & 0x0F) << 4) | (data[1] & 0x0F);
    float luminance = pow(2, exponent) * mantissa * 0.045;
```

```

    return luminance;
}

/*-----DHT Functions-----*/

void DHT_start(void)//khởi động dht
{
    Set_Pin_Output(GPIOB,GPIO_PIN_14);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_14,GPIO_PIN_RESET);
    delay_us(25000);
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_14,GPIO_PIN_SET);
    Set_Pin_Input(GPIOB,GPIO_PIN_14);
}

/* -----Kiểm tra giao tiếp-----*/

char DHT_check(void)
{
    uint8_t Response=0;
    delay_us(40);
    if((HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_14))==0)
    {
        delay_us(80);
        if((HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_14))==1)
            {Response =1;}
        else
            {Response =0;}
    }
    while((HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_14))==1);
    return Response;
}

/* -----Đọc từng byte dữ liệu-----*/

char DHT_Byte(void)
{
    uint8_t i,j;
    for (j=0;j<8;j++)

```

```

{
while (!(HAL_GPIO_ReadPin (GPIOB,GPIO_PIN_14)))/wait pin high
    delay_us(50);
    if ((HAL_GPIO_ReadPin (GPIOB,GPIO_PIN_14))==0)/ pin low?
        {
            i&= ~(1<<(7-j));// write 0
        }
        else i|= (1<<(7-j));//write 1 if pin high
while ((HAL_GPIO_ReadPin (GPIOB,GPIO_PIN_14))==1)/wait for pin low
}
return i;
}
/* -----Đọc 5 bytes dữ liệu----- */

```

```

void DHT_ReadByte(void)
{
    for(uint8_t i=0;i<5;i++)
    {
        byte[i]=DHT_Byte();
    }
}

void DHT11(void)
{
    DHT_start();
    Response=DHT_check();
    DHT_ReadByte();
}

/* -----Chuyển số thực thành byte----- */
void float2Bytes(char* bytes_array,float val)
{
    union
    {
        {

```

```

    float float_variable;
    char temp_array[4];
}u;
u.float_variable = val;
memcpy(bytes_array,u.temp_array,4);
}
/* -----Send data to ESP-01----- */
void senddata(void)
{
    temp=(float)byte[2]+(float)byte[3]/10;;
    float2Bytes(&a[0],temp);
    HAL_UART_Transmit(&huart1,(uint8_t*)a,4,1000);
    humid=(float)byte[0]+(float)byte[1]/10;
    float2Bytes(&a[0],humid);
    HAL_UART_Transmit(&huart1,(uint8_t*)a,4,1000);
    soil=read_soil(10);
    float2Bytes(&a[0],soil);
    HAL_UART_Transmit(&huart1,(uint8_t*)a,4,1000);
    lux=MAX44009_read();
    float2Bytes(&a[0],lux);
    HAL_UART_Transmit(&huart1,(uint8_t*)a,4,1000);

}

```

II. Code cho module wifi

```

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
const char* ssid = "wifi-name";
const char* password = "password";
char *mqtt_server = "IP máy cài MQTT Broker";
char dem[4];
float lux;
unsigned int soil,temp,humid;

```



```

/*khởi động thư viện Pubsubclient*/
WiFiClient espClient;
PubSubClient client(espClient);

void setup()
{
  Serial.begin(9600);
  setup_wifi();// cấu hình module để kết nối wifi.
  client.setServer(mqtt_server,1883);// đăng ký Broker server Mosquitto và TCP port 1883
}
void loop()
{
  if (!client.connected())
  {
    reconnect();
  }
  client.loop();
  /* -----Đọc dữ liệu nếu có-----*/
  if(Serial.available()>0)
  {
    Serial.readBytes(dem,4);//read soil
    soil=Bytes2float(&dem[0]);
    Serial.readBytes(dem,4);//read temperature
    temp=Bytes2float(&dem[0]);
    Serial.readBytes(dem,4);//read humidity
    humid=Bytes2float(&dem[0]);
    Serial.readBytes(dem,4);//read lux
    lux=Bytes2float(&dem[0]);
    /* ----- Prepare a JSON payload string -----*/
    String payload = "{";
    payload += "\"soil\":";

```

```

    payload += soil;
    payload += ",";
    payload += "\"temperature\":";
    payload += temp;
    payload += ",";
    payload += "\"humidity\":";
    payload += humid;
    payload += ",";
    payload += "\"lux\":";
    payload += lux;
    payload += "}";

    /* ----- Send payload ----- */
    char telemetry[200];
    payload.toCharArray(telemetry,200);
    client.publish("Temperature",telemetry);
}
}

/* ----- Kết nối wifi ----- */
void setup_wifi()
{
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println("WiFi connected");
}

/* ----- Kết nối lại ----- */
void reconnect()
{
    while (!client.connected())

```

```

{
    if(client.connect("ESP8266Client"))
    {
        Serial.println("Connected");
    }
    else
    {
        Serial.println("Notconnected");
        delay(3000);
    }
}

/* ----- Chuyển dữ liệu từ byte sang float ----- */
float Bytes2float(char* bytes_array)
{
    union
    {
        float float_variable;
        char temp_array[4];
    }u;
    memcpy(u.temp_array,bytes_array,4);
    return u.float_variable;
}

```

III. Code giao diện Python

```

import os
import codecs
import pickle
import json
import paho.mqtt.client as mqtt
from PyQt5 import QtCore, QtGui, QtWidgets

global server_info

```

```

filename = 'settings.txt'

def save_file():
    with open(filename, "wb") as myFile:
        pickle.dump(server_info, myFile)

if os.path.exists(filename):
    # Read Dictionary from this file
    with open(filename, "rb") as myFile:
        server_info = pickle.load(myFile)
else:
    # Create Dictionary Using Default parameters
    server_info = {"Server_Address": "192.168.100.7",\
                  "Server_Port": "1883",\
                  "Username": "maithy282",\
                  "Password": "maithy2802"}
    save_file()

# Callback Function on Connection with MQTT Server
def on_connect(client, userdata, flags, rc):
    print("Connected with Code :" + str(rc))
    if rc == 0:
        # Subscribe Topic from here
        #client.subscribe("test")
        #client.publish("test", THY)
        client.subscribe("sensor/lux")
        client.subscribe("sensor/humid")
        client.subscribe("sensor/soil")

```

```

    client.subscribe("sensor/temp")
    # Enable Disconnect Button and Enable Others
    ui.connect_btn.setDisabled(True)
    # ui.server_add.setEnabled(False) Don't use this
    ui.server_add.setDisabled(True)
    ui.server_port.setDisabled(True)
    ui.username.setDisabled(True)
    ui.password.setDisabled(True)
    ui.disconnect_btn.setEnabled(True)
    ui.statusBar.setStatusTip("Connected")

# Callback Function on Receiving the Subscribed Topic/Message
def on_message(client, userdata, message):
    if message.topic == "sensor/lux":
        lux = json.loads(message.payload.decode("utf-8"))
        lux = message.payload.decode("utf-8")
        ui.lux.setText(lux)
    if message.topic == "sensor/humid":
        humidity = (message.payload.decode("utf-8"))
        ui.humidity.setText(humidity)
    if message.topic == "sensor/soil":
        soil_humid = str(message.payload.decode("utf-8"))
        ui.soil_humid.setText(soil_humid)
    if message.topic == "sensor/temp":
        temp = str(message.payload.decode("utf-8"))
        ui.temp.setText(temp)
def save_server_add():
    global server_info
    server_info["Server_Address"] = ui.server_add.text()
    save_file()

```

```
def save_server_port():  
    server_info["Server_Port"] = ui.server_port.text()  
    save_file()
```

```
def save_username():  
    server_info["Username"] = ui.username.text()  
    save_file()
```

```
def save_password():  
    #server_info["Password"] = ui.password.text()  
    save_file()
```

```
client = mqtt.Client()  
client.on_connect = on_connect  
client.on_message = on_message
```

```
def connect_with_server():  
    global server_info  
    client.username_pw_set(server_info["Username"], server_info["Password"])  
    client.connect(server_info["Server_Address"], int(server_info["Server_Port"]), 60)  
    client.loop_start()
```

```
def disconnect_with_server():  
    client.loop_stop()  
    client.disconnect()  
    # Enable Connect Button and Disable Others
```

```

ui.connect_btn.setEnabled(True)
ui.server_add.setEnabled(True)
ui.server_port.setEnabled(True)
ui.username.setEnabled(True)
ui.password.setEnabled(True)
ui.disconnect_btn.setDisabled(True)
ui.statusBar.setStatusTip("Not Connected")
print("Disconnected")

```

```

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(350, 350)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.connect_btn = QtWidgets.QPushButton(self.centralwidget)
        self.connect_btn.setGeometry(QtCore.QRect(220, 10, 75, 31))
        self.connect_btn.setObjectName("connect_btn")
        self.disconnect_btn = QtWidgets.QPushButton(self.centralwidget)
        self.disconnect_btn.setEnabled(False)
        self.disconnect_btn.setGeometry(QtCore.QRect(220, 70, 75, 31))
        self.disconnect_btn.setObjectName("disconnect_btn")
        self.temp_lbl = QtWidgets.QLabel(self.centralwidget)
        self.temp_lbl.setGeometry(QtCore.QRect(10, 120, 75, 18))
        self.temp_lbl.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.temp_lbl.setAlignment(QtCore.Qt.AlignCenter)
        self.temp_lbl.setObjectName("temp_lbl")
        self.humid_lbl = QtWidgets.QLabel(self.centralwidget)
        self.humid_lbl.setGeometry(QtCore.QRect(110, 120, 75, 18))
        self.humid_lbl.setLayoutDirection(QtCore.Qt.LeftToRight)

```

```

self.humid_lbl.setAlignment(QtCore.Qt.AlignCenter)
self.humid_lbl.setObjectName("humid_lbl")
#them vao nut soil
self.soil_lbl = QtWidgets.QLabel(self.centralwidget)
self.soil_lbl.setGeometry(QtCore.QRect(10, 200, 69, 18))
self.soil_lbl.setLayoutDirection(QtCore.Qt.LeftToRight)
self.soil_lbl.setAlignment(QtCore.Qt.AlignCenter)
self.soil_lbl.setObjectName("soil_lbl")
# them vao nut lux
self.lux_lbl = QtWidgets.QLabel(self.centralwidget)
self.lux_lbl.setGeometry(QtCore.QRect(110, 200, 69, 18))
self.lux_lbl.setLayoutDirection(QtCore.Qt.LeftToRight)
self.lux_lbl.setAlignment(QtCore.Qt.AlignCenter)
self.lux_lbl.setObjectName("lux_lbl")

self.temp = QtWidgets.QLabel(self.centralwidget)
self.temp.setGeometry(QtCore.QRect(10, 140, 31, 18))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.temp.setFont(font)
self.temp.setLayoutDirection(QtCore.Qt.LeftToRight)
self.temp.setAlignment(QtCore.Qt.AlignRight / QtCore.Qt.AlignTrailing /
QtCore.Qt.AlignVCenter)
self.temp.setObjectName("temp")
self.c_lbl = QtWidgets.QLabel(self.centralwidget)
self.c_lbl.setGeometry(QtCore.QRect(50, 140, 21, 18))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)

```



```

self.c_lbl.setFont(font)
self.c_lbl.setLayoutDirection(QtCore.Qt.LeftToRight)
self.c_lbl.setAlignment(QtCore.Qt.AlignCenter)
self.c_lbl.setObjectName("c_lbl")
# vitri dat gia tri cua humidity
self.humidity = QtWidgets.QLabel(self.centralwidget)
self.humidity.setGeometry(QtCore.QRect(110, 140, 50, 18))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.humidity.setFont(font)
self.humidity.setLayoutDirection(QtCore.Qt.LeftToRight)
self.humidity.setAlignment(QtCore.Qt.AlignRight    /    QtCore.Qt.AlignTrailing    /
QtCore.Qt.AlignVCenter)
self.humidity.setObjectName("humidity")
#vitri dat gia tri cua soil_humid
self.soil_humid = QtWidgets.QLabel(self.centralwidget)
self.soil_humid.setGeometry(QtCore.QRect(10, 220, 31, 18))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(True)
font.setWeight(75)
self.soil_humid.setFont(font)
self.soil_humid.setLayoutDirection(QtCore.Qt.LeftToRight)
self.soil_humid.setAlignment(QtCore.Qt.AlignRight    /    QtCore.Qt.AlignTrailing    /
QtCore.Qt.AlignVCenter)
self.soil_humid.setObjectName("soil_humid")
# vitri dat gia tri cua lux
self.lux = QtWidgets.QLabel(self.centralwidget)
self.lux.setGeometry(QtCore.QRect(110, 220, 70, 18))
font = QtGui.QFont()

```

```

        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.lux.setFont(font)
        self.lux.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.lux.setAlignment(QtCore.Qt.AlignRight | QtCore.Qt.AlignTrailing |
QtCore.Qt.AlignVCenter)
        self.lux.setObjectName("soil_humid")
        #vitri dat nhan percent
        self.percent_lbl = QtWidgets.QLabel(self.centralwidget)
        self.percent_lbl.setGeometry(QtCore.QRect(160, 140, 21, 18))
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.percent_lbl.setFont(font)
        self.percent_lbl.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.percent_lbl.setAlignment(QtCore.Qt.AlignCenter)
        self.percent_lbl.setObjectName("percent_lbl")
        self.widget = QtWidgets.QWidget(self.centralwidget)
        self.widget.setGeometry(QtCore.QRect(10, 10, 71, 91))
        self.widget.setObjectName("widget")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.widget)
        self.verticalLayout.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout.setObjectName("verticalLayout")
        self.server_add_lbl = QtWidgets.QLabel(self.widget)
        self.server_add_lbl.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.server_add_lbl.setAlignment(QtCore.Qt.AlignRight | QtCore.Qt.AlignTrailing |
QtCore.Qt.AlignVCenter)
        self.server_add_lbl.setObjectName("server_add_lbl")
        self.verticalLayout.addWidget(self.server_add_lbl)
        self.server_port_lbl = QtWidgets.QLabel(self.widget)

```

```

self.server_port_lbl.setLayoutDirection(QtCore.Qt.LeftToRight)
self.server_port_lbl.setAlignment(QtCore.Qt.AlignRight | QtCore.Qt.AlignTrailing |
QtCore.Qt.AlignVCenter)
self.server_port_lbl.setObjectName("server_port_lbl")
self.verticalLayout.addWidget(self.server_port_lbl)
self.username_lbl = QtWidgets.QLabel(self.widget)
self.username_lbl.setLayoutDirection(QtCore.Qt.LeftToRight)
self.username_lbl.setAlignment(QtCore.Qt.AlignRight | QtCore.Qt.AlignTrailing |
QtCore.Qt.AlignVCenter)
self.username_lbl.setObjectName("username_lbl")
self.verticalLayout.addWidget(self.username_lbl)
self.password_lbl = QtWidgets.QLabel(self.widget)
self.password_lbl.setLayoutDirection(QtCore.Qt.LeftToRight)
self.password_lbl.setAlignment(QtCore.Qt.AlignRight | QtCore.Qt.AlignTrailing |
QtCore.Qt.AlignVCenter)
self.password_lbl.setObjectName("password_lbl")
self.verticalLayout.addWidget(self.password_lbl)
self.widget1 = QtWidgets.QWidget(self.centralwidget)
self.widget1.setGeometry(QtCore.QRect(90, 10, 111, 91))
self.widget1.setObjectName("widget1")
self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.widget1)
self.verticalLayout_2.setContentsMargins(0, 0, 0, 0)
self.verticalLayout_2.setObjectName("verticalLayout_2")
self.server_add = QtWidgets.QLineEdit(self.widget1)
self.server_add.setMaxLength(30)
self.server_add.setObjectName("server_add")
self.verticalLayout_2.addWidget(self.server_add)
self.server_port = QtWidgets.QLineEdit(self.widget1)
self.server_port.setMaxLength(30)
self.server_port.setObjectName("server_port")
self.verticalLayout_2.addWidget(self.server_port)
self.username = QtWidgets.QLineEdit(self.widget1)

```

```

self.username.setMaxLength(30)
self.username.setObjectName("username")
self.verticalLayout_2.addWidget(self.username)
self.password = QtWidgets.QLineEdit(self.widget1)
self.password.setMaxLength(30)
self.password.setEchoMode(QtWidgets.QLineEdit.Password)
self.password.setObjectName("password")
self.verticalLayout_2.addWidget(self.password)
MainWindow.setCentralWidget(self.centralwidget)
self.statusBar = QtWidgets.QStatusBar(MainWindow)
self.statusBar.setObjectName("statusBar")
MainWindow.setStatusBar(self.statusBar)

```

```

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)

```

```

def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "MQTT Client"))
    self.connect_btn.setText(_translate("MainWindow", "Connect"))
    self.disconnect_btn.setText(_translate("MainWindow", "Disconnect"))
    self.temp_lbl.setText(_translate("MainWindow", "Temperature"))
    self.humid_lbl.setText(_translate("MainWindow", "Humidity"))
    self.soil_lbl.setText(_translate("MainWindow", "Soil_Humid"))
    self.lux_lbl.setText(_translate("MainWindow", "Lux"))
    self.temp.setText(_translate("MainWindow", "0"))
    self.c_lbl.setText(_translate("MainWindow", "C"))
    self.humidity.setText(_translate("MainWindow", "0"))
    self.percent_lbl.setText(_translate("MainWindow", "%"))
    self.soil_humid.setText(_translate("MainWindow", "0"))
    self.lux.setText(_translate("MainWindow", "0"))
    self.server_add_lbl.setText(_translate("MainWindow", "Server Add :"))

```

```

self.server_port_lbl.setText(_translate("MainWindow", "Server Port :"))
self.username_lbl.setText(_translate("MainWindow", "Username :"))
self.password_lbl.setText(_translate("MainWindow", "Password :"))
# Main Program Starts from Here
self.statusBar.setStatusTip("Not Connected")
# Update Server Information
global server_info
self.server_add.setText(server_info["Server_Address"])
self.server_port.setText(server_info["Server_Port"])
self.username.setText(server_info["Username"])
self.password.setText(server_info["Password"])
# Button Press Events
self.connect_btn.clicked.connect(connect_with_server)
self.disconnect_btn.clicked.connect(disconnect_with_server)
self.server_add.editingFinished.connect(save_server_add)
self.server_port.editingFinished.connect(save_server_port)
self.username.editingFinished.connect(save_username)
self.password.editingFinished.connect(save_password)

if __name__ == "__main__":
    import sys

    app = QtWidgets.QApplication(sys.argv)
    app.setStyle('Fusion')
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())

```