TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ - VIỄN THÔNG

--ଓ📖ଌ--

ĐỒ ÁN III

Đề tài: Thực thi và kiểm thử bảo mật hệ thống IoT

Giảng viên hướng dẫn: PGS.TS Đỗ Trọng Tuấn

Sinh viên thực hiện:

Vũ Đức Thao 20163668

Hà Nội, 10-2020

MỤC LỤC

DANH MỤC HÌNH ẢNH

LỜI NÓI ĐẦU

Trong thời gian qua, khái niệm "cách mạng công nghiệp lần thứ tư" ngày càng trở nên phổ biến, cùng với đó là các xu hướng được đự đoán sẽ là kim chỉ nam cho cuộc cách mạng này. Internet of Things (hay IoT) là một xu hướng sẽ phát triển mạnh mẽ trong cuộc cách mạng 4.0, nó mở ra một kịch bản của thế giới, khi mà mỗi đồ vật, con người được cung cấp một định danh của riêng mình, và tất cả có khả năng truyền tải, trao đổi thông tin, dữ liệu qua một trạng thái duy nhất. Các thiết bị trong nhà cũng sẽ được kết nối vào mạng duy nhất đó và được điều khiển bởi con người ngay cả khi ở khoảng cách rất xa, chỉ cần truy cập vào mạng duy nhất trên là người sử dụng có thể làm được điều đó. Nhưng cùng với đó là hàng loạt vấn đề về quyền riêng tư người dung, vấn đề bảo mật dữ liệu trong hệ thống IoT. Thấy được tầm quan trọng của vấn đề này nên nhóm em đã thực hiện đề tài "**Thực thi và kiểm thử bảo mật hệ thống IoT"** nhằm tìm hiểu và đưa ra các giải pháp giúp giải quyết vấn đề này.

Đề tài này có sử dụng hệ thống IoT xây dựng theo giao thức MQTT, phương thức bảo mật chính là hệ mật AES (Advanced Encryption Standard), dữ liệu được hiển thị trên trang web viết bằng VueJs.

Với sự hướng dẫn tận tình của PGS.TS Đỗ Trọng Tuấn qua môn học Đồ Án III nhóm em đã hoàn thành đề tài này. Tuy đã cố gắng tìm hiểu thực hiện song vẫn không tránh khỏi những thiếu sót trong kinh nghiệm thực tế. Nhóm em rất mong nhận được sự thông cảm và góp ý từ Thầy.

Em xin chân thành cảm ơn!

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

1.1 Giao Thức MQTT

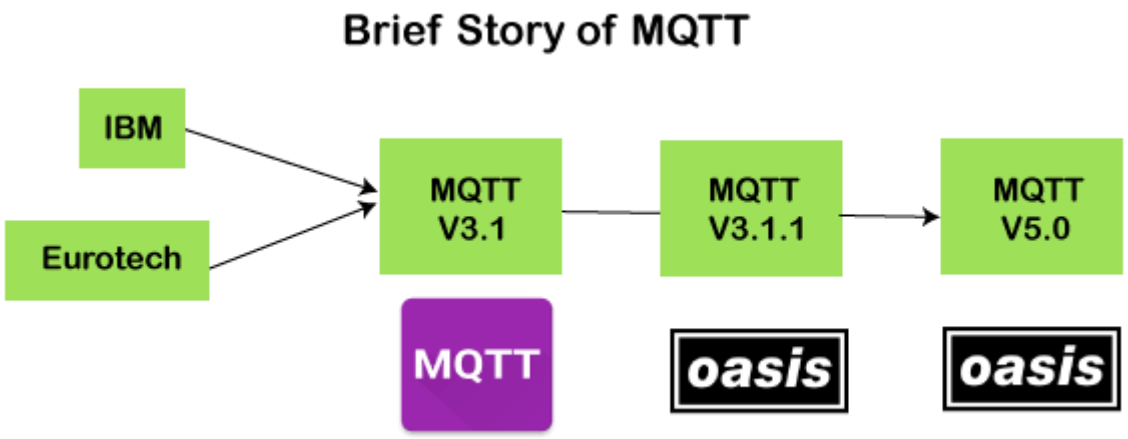
1.1.1 MQTT là gì?

Định nghĩa

- MQTT (Message Queuing Telemetry Transport) là một giao thức truyền thông điệp (message) theo mô hình publish/subscribe (cung cấp/ thuê bao), được sử dụng trong cho các hệ thống IoT với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định.
- MQTT là lựa chọn lý tưởng trong các môi trường như:
    - Những nơi mà giá mạng viễn thông đắt đỏ hoặc băng thông thấp hay thiếu tin cậy.
    - Khi chạy trên thiết bị nhúng bị giới hạn về tài nguyên tốc độ và bộ nhớ.
    - Bởi vì giao thức này sử dụng băng thông thấp trong môi trường có độ trễ cao nên nó là một giao thức lý tưởng cho các ứng dụng M2M (Machine to Machine).

Lịch sử hình thành

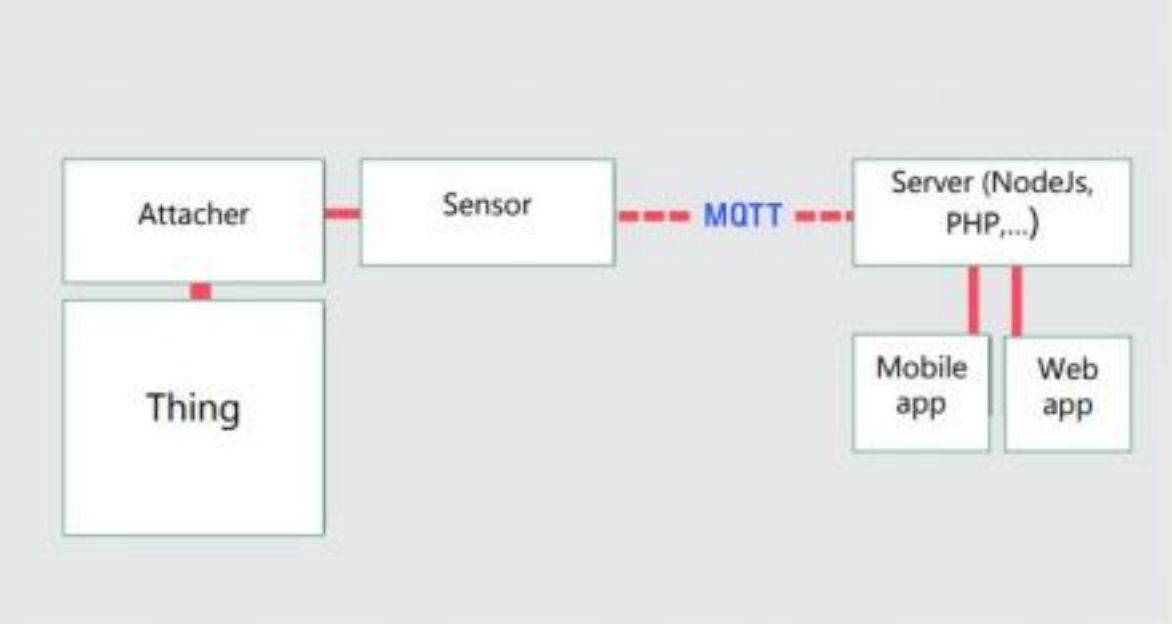
- MQTT được phát minh bởi Andy Stanford – Clark (IBM) và Arlen Nipper (EUROTECH) cuối năm 1999 khi mà nhiệm vụ của họ là tạo ra một giao thức sao cho sự hao phí năng lượng và băng thông là thấp nhất để kết nối đến đường ông dẫn dầu thông qua sự kết nối của vệ tinh.
- Năm 2011, IBM và Eurotech đã trao lại MQTT cho một dự án của Eclipse có tên là Paho
- Năm 2013, MQTT đã được đệ trình lên OASIS (Organization for the Advancement of Structured Information Standards) để chuẩn hóa.



Hình 1.1 Lịch sử ra đời

Vị trí của MQTT trong mô hình IoT

Một số ưu điểm nổi bật của MQTT như: băng thông thấp, độ tin cậy cao và có thể sử dụng ngay cả khi hệ thống mạng không ổn định, tốn rất ít byte cho việc kết nối với server và connection có thể giữ trạng thái open xuyên suốt, có thể kết nối nhiều thiết bị (MQTT client) thông qua một MQTT server (broker).



Hình 1.2 Vị trí MQTT trong hệ thống IOT

Tính năng, đặc điểm nổi bật

- Dạng truyền thông điệp theo mô hình Pub/Sub cung cấp việc truyền tin phân tán một chiều, tách biệt với phần ứng dụng.

- Việc truyền thông điệp là ngay lập tức, không quan tâm đến nội dung được truyền.
- Sử dụng TCP/IP là giao thức nền.
- Tồn tại ba mức độ tin cậy cho việc truyền dữ liệu (QoS: Quality of Service)
    - QoS 0: Broker/client sẽ gửi dữ liệu đúng một lần, quá trình gửi được xác nhận bởi vì giao thức TCP/IP.
    - QoS 1: Broker/client sẽ gửi dữ liệu với ít nhất một lần xác nhận từ đầu kia, nghĩa là có thể có nhiều hơn 1 lần xác nhận đã nhận được dữ liệu.
    - QoS 2: Broker/client đảm bảo khi gửi dữ liệu thì phía nhận chỉ nhận được đúng một lần, quá trình này phải trải qua 4 bước bắt tay.
- Phần bao bọc dữ liệu truyền nhỏ và được giảm đến mức tối thiểu để giảm tải cho đường truyền.

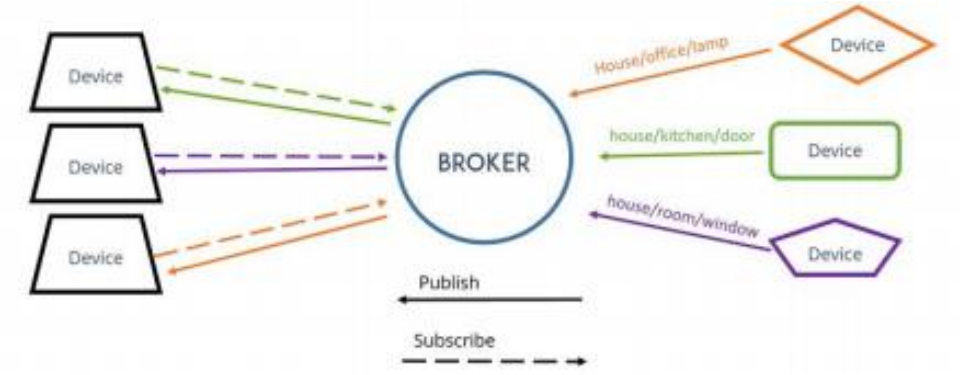
1.1.2 Mô hình Pub/Sub

Thành phần

- Client
    - Publisher – Nơi gửi thông điệp
    - Subscriber – Nơi nhận thông điệp
- Broker – Máy chủ môi giới

Trong đó Broker được coi như trung tâm, nó là điểm giao của tất cả các kết nối đến từ Client (Publisher/Subscriber). Nhiệm vụ chính của Broker là nhận thông điệp (message) từ Publisher, xếp vào hàng đợi rồi chuyển đến địa điểm cụ thể. Nhiệm vụ phụ của Broker là nó có thể đảm nhiệm một vài tính năng liên quan tới quá trình truyền thông như: bảo mật message, lưu trữ message, logs, …

Client thì được chia thành hai nhóm là Publisher và Subscriber. Client chỉ làm ít nhất một trong hai việc là publish các thông điệp (message) lên một chiều/nhiều topic cụ thể hoặc subscribe một/nhiều topic nào đó để nhận message từ topic này.



Hình 1.3 Mô hình Pub/Sub

MQTT Clients tương thích với hầu hết các nền tảng hệ điều hành hiện có: MAC OS, Windows, Linux, Android, IOS, …

Ưu điểm

- Kết nối riêng rẽ, độc lập
- Khả năng mở rộng.
- Thời gian tách biệt
- Đồng bộ riêng rẽ.

Nhược điểm

- Máy chủ môi giới (Broker) không cần thông báo về trạng thái gửi thông điệp. Do đó không có cách nào để phát hiện xem thông điệp đã gửi đúng hay chưa.
- Publisher không hề biết gì về trạng thái của subscribe và ngược lại.
- Những kẻ xấu có thể gửi thông điệp xấu, và các Subscriber sẽ truy cập vào những thứ mà họ không nên nhận.

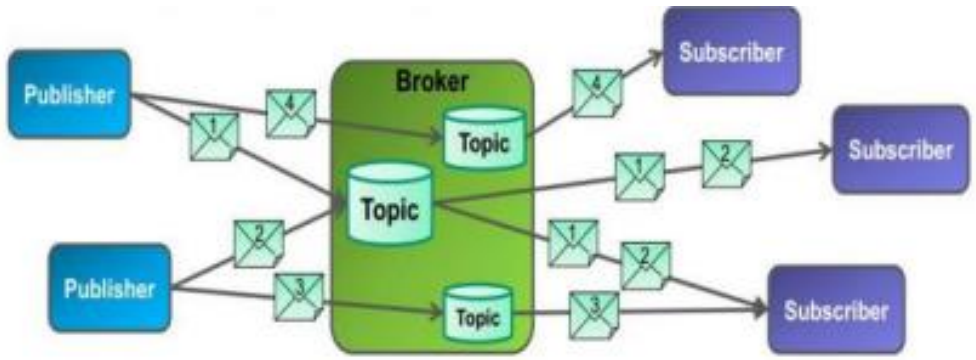
1.1.3 Cơ chế hoạt động của MQTT theo mô hình Pub/Sub

Tính chất và những đặc điểm riêng

- Tính chất:
    - Không gian tách biệt (Space decoupling)
    - Thời gian tách biệt (Time decoupling)
    - Sự đồng bộ riêng rẽ (Synchronization decoupling)
- Đặc điểm riêng:
    - MQTT sử dụng cơ chế lọc thông điệp dựa vào tiêu đề (subject-based)

- MQTT có một tầng gọi là chất lượng dịch vụ (Quality of Services – QoS). Nó giúp cho dễ dàng nhận biết được là message có được truyền thành công hay không

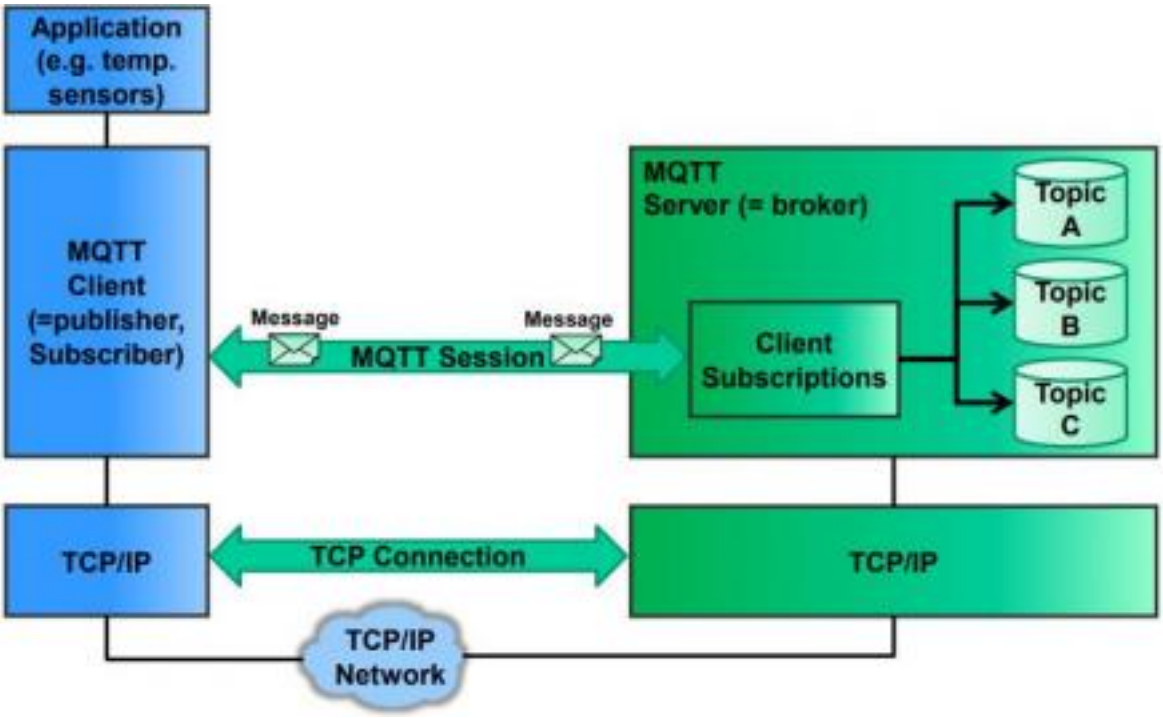
Cơ chế tổng quan



Hình 1.4 Cơ chế tổng quan MQTT

- MQTT hoạt động theo cơ chế client/server, nơi mà mỗi cảm biến là một khác hàng (client) và kết nối đến một máy chủ, có thể hiểu như một Máy chủ môi giới (broker), thông qua giao thức TCP (Transmission Control Protocol). Broker chịu trách nhiệm điều phối tất cả các thông điệp giữa phía gửi đến đúng phía nhận.
- MQTT là giao thức định hướng bản tin. Mỗi bản tin là một đoạn rời rạc của tín hiệu và broker không thể nhìn thấy. Mỗi bản tin được publish một địa chỉ, có thể hiểu như một kênh (Topic). Client đăng kí vào một vài kênh để nhận/gửi dữ liệu, gọi là subscribe. Client có thể subscribe vào nhiều kênh. Mỗi client sẽ nhận được dữ liệu khi bất kỳ trạm nào khác gửi dữ liệu vào kênh đã đăng ký. Khi một client gửi một bản tin đến một kênh nào đó gọi là publish.

Kiến trúc thành phần



Hình 1.5 Kiến trúc thành phần MQTT

- Thành phần chính của MQTT là Client (Publisher/Subscriber), Server (Broker), Sessions (tạm dịch là Phiên làm việc), Subscriptions và Topics.
- MQTT Client (Publisher/Subscriber): Clients sẽ subscribe một hoặc nhiều topics để gửi và nhận thông điệp từ những topic tương ứng.
- MQTT Server (Broker): Broker nhận những thông tin subscribe (Subscriptions) từ client, nhận thông điệp, chuyển những thông điệp đến các Subscriber tương ứng dựa trên Subscriptions từ client.
- Topic: Có thể coi Topic là một hàng đợi các thông điệp, và có sẵn khuôn mẫu dành cho Subscriber hoặc Publisher. Một cách logic thì các topic cho phép Client trao đổi thông tin với những ngữ nghĩa đã được định nghĩa sẵn. Ví dụ: Dữ liệu cảm biến nhiệt độ của một tòa nhà.
- Session: Một session được định nghĩa là kết nối từ client đến server. Tất cả các giao tiếp giữa client và server đều là 1 phần của session.
- Subscription: Không giống như session, subscription về mặt logic là kết nối từ client đến topic. Khi đã subscribe một topic, Client có thể nhận/gửi thông điệp (message) với topic đó.

1.2 Hệ Mật AES

1.2.1 Giới Thiệu Chung

Tổng quan

- AES (viết tắt của từ tiếng anh: Advanced Encryption Standard, hay Tiêu chuẩn mã hóa nâng cao) là một thuật toán mã hóa khối được chính phủ Hoa Kỳ áp dụng làm tiêu chuẩn mã hóa.
- Thuật toán được xây dựng dựa trên Rijndael Cipher phát triển bởi 2 nhà mật mã học người Bỉ: Joan Daemen và Vincent Rijmen.
- AES làm việc với các khối dữ liệu 128bit và độ dài khóa 128bit, 192bit hoặc 256bit. Các khóa mở rộng sử dụng trong chu trình được tạo ra bởi thủ tục sinh khóa Rijndael.
- Hầu hết các phép toán trong thuật toán AES đều thực hiện trong một trường hữu hạn của các byte. Mỗi khối dữ liệu đầu vào 128bit được chia thành 16byte, có thể xếp thành 4 cột, mỗi cột 4 phần tử hay một ma trận 4x4 của các byte, nó gọi là ma trận trạng thái.
- Tùy thuộc vào độ dài của khóa khi sử dụng 128bit, 192bit hay 256bit mà thuật toán được thực hiện với số lần lặp khác nhau.

Các bước xử lý chính

- Quá trình mở rộng khóa sử dụng thủ tục sinh khóa Rijndael.
- Quá trình mã hóa.

1.2.2 Xây dựng thuật toán

Xây dựng bảng S-box

i, Bảng S-box thuận

- Bảng S-box thuận được sinh ra bằng việc xác định nghịch đảo cho một giá trị nhất định trên $GF(2^8) = \frac{GF(2)[x]}{x^8 + x^4 + x^3 + x + 1}$ (trường hữu hạn Rijindael).

  Giá trị 0 không có nghịch đảo thì được ánh xạ với 0. Những nghịch đảo được chuyển đổi thông qua phép biến đổi affine.
- Công thức tính các giá trị bảng S-box và bảng S- box tương ứng:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1x	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2x	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3x	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4x	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5x	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6x	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7x	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8x	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9x	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
ax	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
bx	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
cx	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
dx	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
ex	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
fx	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Hình 1.6 Bảng S-box thuận

ii, Bảng S-box đảo

- S-box nghịch đảo chỉ đơn giản là S-box chạy ngược. Nó được tính bằng phép biến đổi affine nghịch đảo các giá trị đầu vào. Phép biến đổi affine nghịch đảo được biểu diễn như sau:

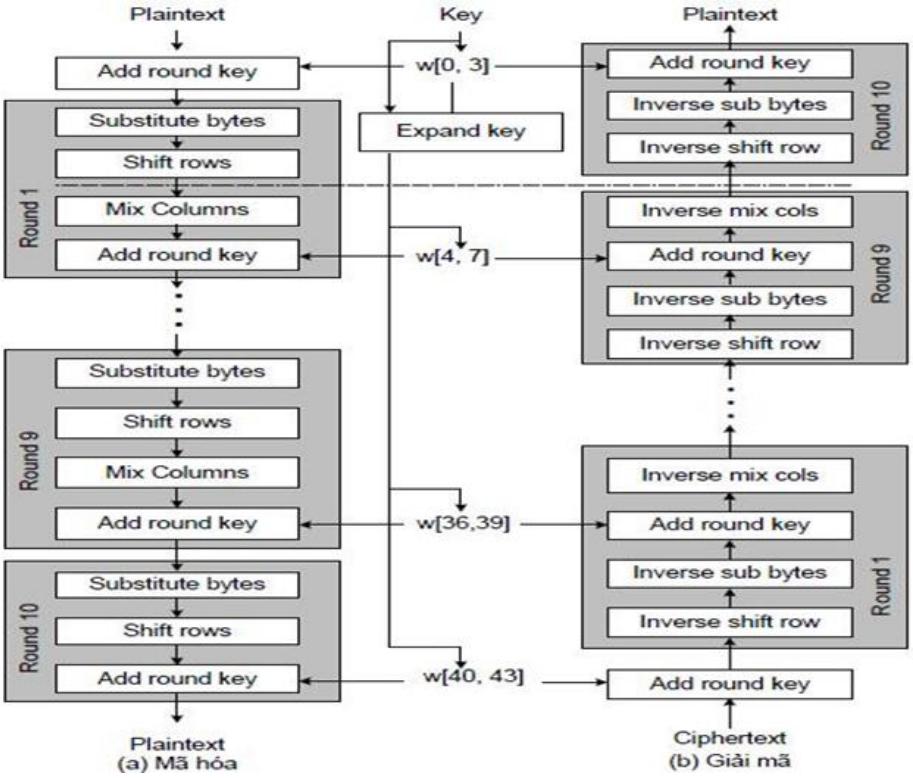
$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1x	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2x	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3x	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4x	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5x	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6x	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7x	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8x	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9x	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
ax	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
bx	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
cx	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
dx	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
ex	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
fx	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Hình 1.7 Bảng S-box Đảo

Quá trình mã hóa

i, Sơ đồ tổng quát



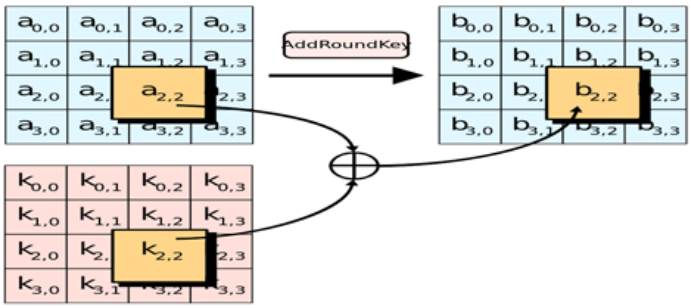
Hình 1.8 Sơ đồ tổng quát AES

ii, Hàm AddRoundKey

- Được áp dụng từ vòng lặp thứ 1 tới vòng lặp Nr
- Trong biến đổi Addroundkey(), một khóa vòng được cộng với state bằng một phép XOR theo từng bit đơn giản.
- Mỗi khóa vòng gồm có 4 từ (128 bit) được lấy từ lịch trình khóa. 4 từ đó được cộng vào mỗi cột của state, sao cho:

$$[S_{0,c}, S_{1,c}, S_{2,c}, S_{3,c}] = [S_{0,c}, S_{1,c}, S_{2,c}, S_{3,c}] \oplus [W(4i + c)]$$

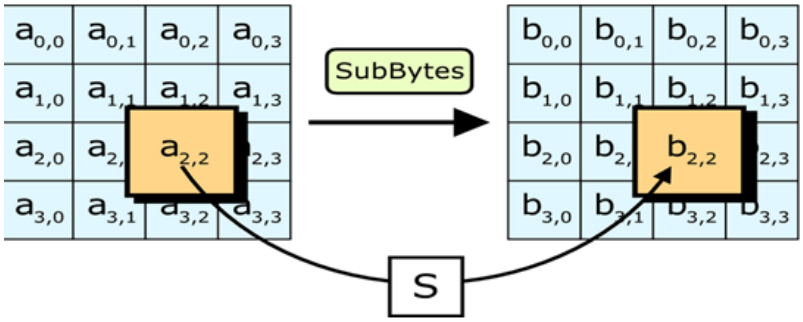
$$0 \le c < 4$$



Hình 1.9 Sơ đồ AddRoundKey

iii, Hàm SubBytes

- Biến đổi SubBytes() thay thế mỗi byte riêng rẽ của state $S_{r,c}$ bằng một giá trị mới $S'_{r,c}$ sử dụng bảng thay thế (S - box) được xây dựng ở trên.



Hình 1.10 Sơ đồ SubBytes

iv, Hàm ShifRows.

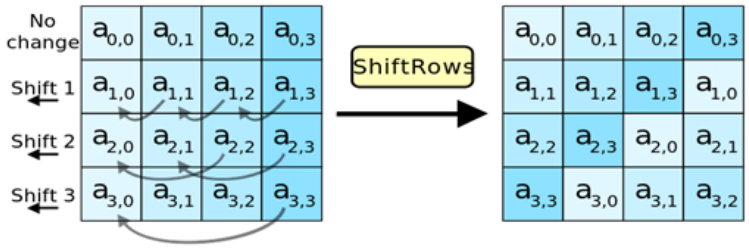
- Trong biến đổi ShiftRows(), các byte trong ba hàng cuối cùng của trạng thái được dịch vòng đi các số byte khác nhau (độ lệch). Cụ thể :

$$S'_{r,c} = S_{r,(c+shift(r,Nb)) \bmod Nb} \quad (Nb=4)$$

- Trong đó giá trị dịch shift (r, Nb) phụ thuộc vào số hàng r như sau:

$$shift(1,4) = 1$$
$$shift(2,4) = 2$$
$$shift(3,4) = 3$$

- Hàng đầu tiên không bị dịch, ba hàng còn lại bị dịch tương ứng:
  - Hàng thứ 1 giữ nguyên.
  - Hàng thứ 2 dịch vòng trái 1 lần.
  - Hàng thứ 3 dịch vòng trái 2 lần.
  - Hàng thứ 4 dịch vòng trái 3 lần.



Hình 1.11 Sơ đồ ShiftRows

v, Hàm MixColumns

- Biến đổi MixColumns() tính toán trên từng cột của state. Các cột được coi như là đa thức trong trường GF(28) và nhân với một đa thức a(x) với:

$$a(x) = 03_{16} \cdot x^3 + 01_{16} \cdot x^2 + 01_{16} \cdot x + 02_{16}$$

- Biến đổi này có thể được trình bày như phép nhân một ma trận, mà mỗi byte được hiểu như là một phần tử trong trường GF(28): $s'(x) = a(x) \otimes s(x)$
- Mô tả bằng ma trận như sau :

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

Quá trình giải mã

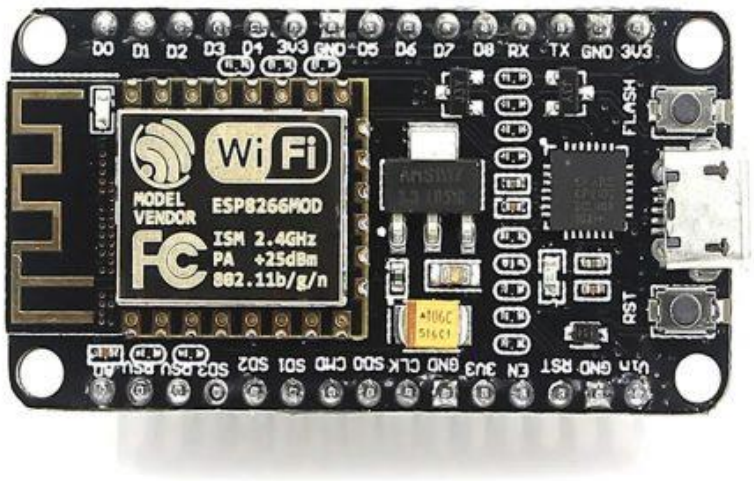
Thuật toán giải mã khá giống với thuật toán mã hóa về mặt cấu trúc nhưng 4 hàm sử dụng là 4 hàm ngược của quá trình mã hóa.

Mã Hóa	Giải Mã
AddRoundKey()	InvAddRoundKey()
SubBytes()	InvSubBytes()
ShiftRows()	InvShiftRows()
MixColumns()	InvMixColumns()

1.3 NodeMCU ESP8266

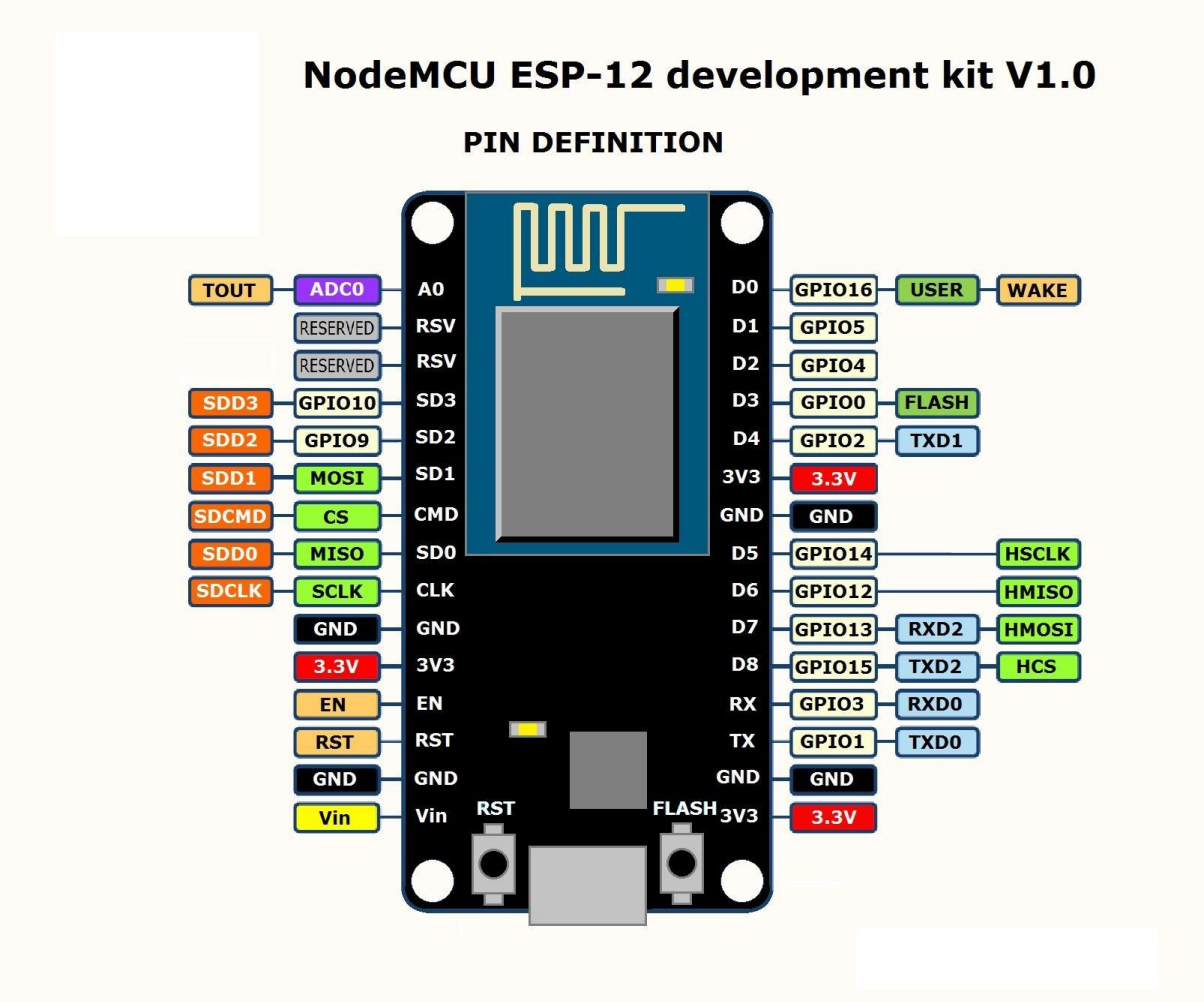
1.3.1 Giới thiệu về ESP8266 nodeMCU

ESP8266 là một mạch vi điều khiển có thể giúp chúng ta điều khiển các thiết bị điện tử.Thêm vào đó nó được tích hợp wi-fi 2.4GHz có thể dùng cho lập trình.



Hình 1.12 NodeMCU ESP8266

1.3.2 Thông số kỹ thuật



Hình 1.13 Sơ đồ chân NodeMCU

- WiFi: 2.4 GHz hỗ trợ chuẩn 802.11 b/g/n

- Điện áp hoạt động: 3.3V

- Điện áp vào: 5V thông qua cổng USB

- Số chân I/O: 11 (tất cả các chân I/O đều có Interrupt/PWM/I2C/One-wire, trừ chân D0)

- Số chân Analog Input: 1 (điện áp vào tối đa 3.3V)

- Bộ nhớ Flash: 4MB

- Giao tiếp: Cable Micro USB (tương đương cáp sạc điện thoại)

- Hỗ trợ bảo mật: WPA/WPA2

- Tích hợp giao thức TCP/IP

- Lập trình trên các ngôn ngữ: C/C++, Micropython,…

1.3.3 Một số ứng dụng khi sử dụng ESP8266

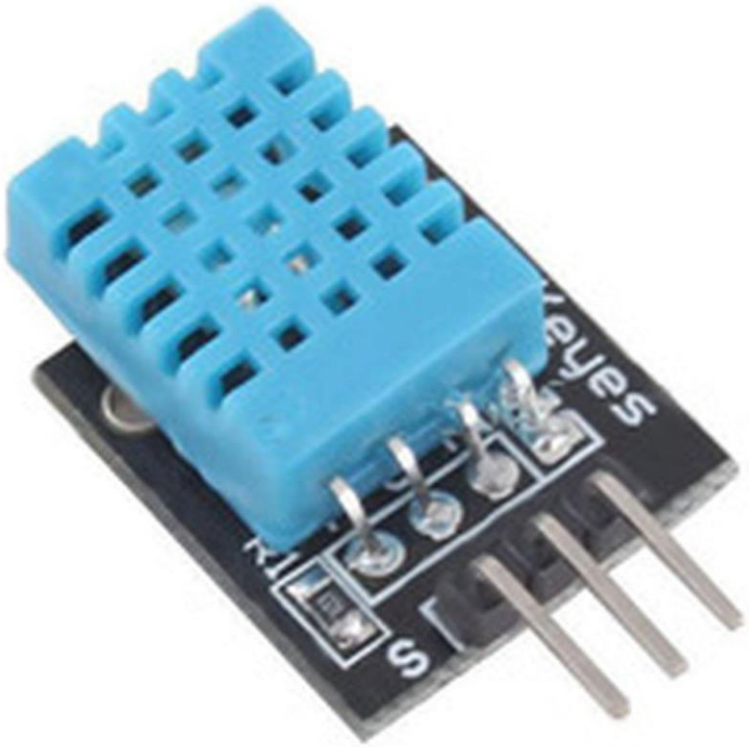
- Điều khiển công tắc bật/tắt Led

- Đọc nhiệt độ trên cảm biến DHT11

- Điều khiển bật/tắt Led bằng giọng nói

1.4 Cảm biến nhiệt độ độ ẩm DHT11

1.4.1 Thông số kỹ thuật



Hình 1.14 DHT11

- Điện áp hoạt động: 3 --> 5V

- Dải nhiệt độ đo: 0 -> 50°C với độ chính xác là ±2°C

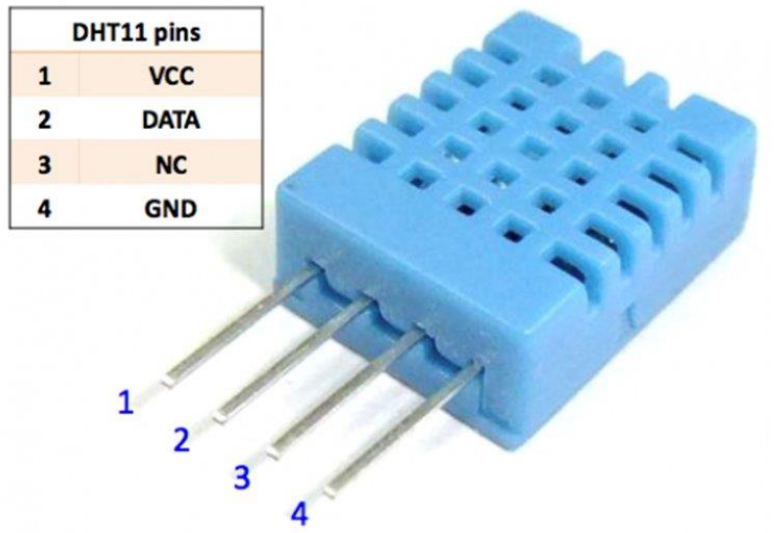
- Dải độ ẩm đo: 20 -> 80% với độ chính xác là 5%

- Kích thước: 15.5mm x 12mm x 5.5mm

- Tần số lấy mẫu: 1Hz , nghĩa là 1 giây DHT11 lấy mẫu một lần.

- 4 chân: VCC(cực (+) nguồn), DATA(chân tín hiệu), NC, GND(cực (-) nguồn)

1.4.2 Cách điều khiển

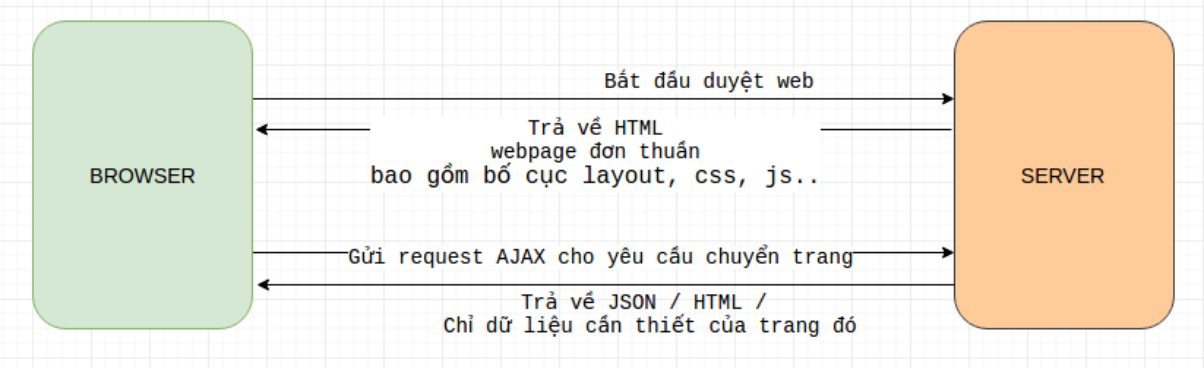


Hình 1.15 Sơ đồ chân DHT11

DHT11 gửi và nhận dữ liệu với một dây tín hiệu DATA, với chuẩn dữ liệu truyền 1 dây này, chúng ta phải đảm bảo sao cho ở chế độ chờ (idle) dây DATA có giá trị ở mức cao, nên trong mạch sử dụng DHT11, dây DATA phải được mắc với một trở kéo bên ngoài(thông thường giá trị là 4.7kΩ).

Dữ liệu truyền về của DHT11 gồm 40bit dữ liệu theo thứ tự: 8 bit biểu thị phần nguyên của độ ẩm + 8 bit biểu thị phần thập phân của độ ẩm + 8 bit biểu thị phần nguyên của nhiệt độ + 8 bit biểu thị phần thập phân của nhiệt độ + 8 bit check sum.

1.5 Single Page Application



Hình 1.16 Single Page Application

Single Page Application (SPA) là một ứng dụng web giúp nâng cao trải nghiệm người dùng bằng cách sử dụng Html 5 và Ajax. Khi truy cập vào một website, toàn bộ resource của web bao gồm các file css, Javascript, master layout hay cấu trúc web page sẽ được load. Ở những lần sau, khi chuyển trang khác, client sẽ gửi những ajax request để lấy về dữ liệu cần thiết (thường là phần nội dung).

Ưu điểm và nhược điểm

- Ưu điểm:
  - Giúp user cảm nhận việc sử dụng website nhanh hơn, thay vì đợi client-server giao tiếp và load lại nguyên page thì giờ chỉ cần đợi browser load lại 1 phần components.
  - Với cách server trả data dạng JSON, phía server sẽ giảm được bớt tài nguyên. Vì vậy chỉ cần tập trung vào các API thay vì xây đựng lại layout kiểu server-side render.
  - Khi server xây dựng các API cho website, chúng ta có thể tận dụng để build các ứng dụng mobile mà không cần phải viết thêm mới.
  - SPA hỗ trợ cho Progressive Web Apps, user có thể sử dụng website mà không cần online (kết nối mạng).

- SPA chia dự án thành 2 đội, backend lo cho việc phát triển các API và đội frontend lo cho việc phát triển UI UX. Chính sự gia tăng SPA trong phát triển website nên vài năm gần đây các tuyển dụng về frontend developer khá nhiều.

- Nhược điểm:
  - Thời lượng load trang ban đầu khá nặng.
  - Ảnh hưởng tới việc chuẩn SEO.
  - Nội dung của SPA không có độ chi tiết cao.

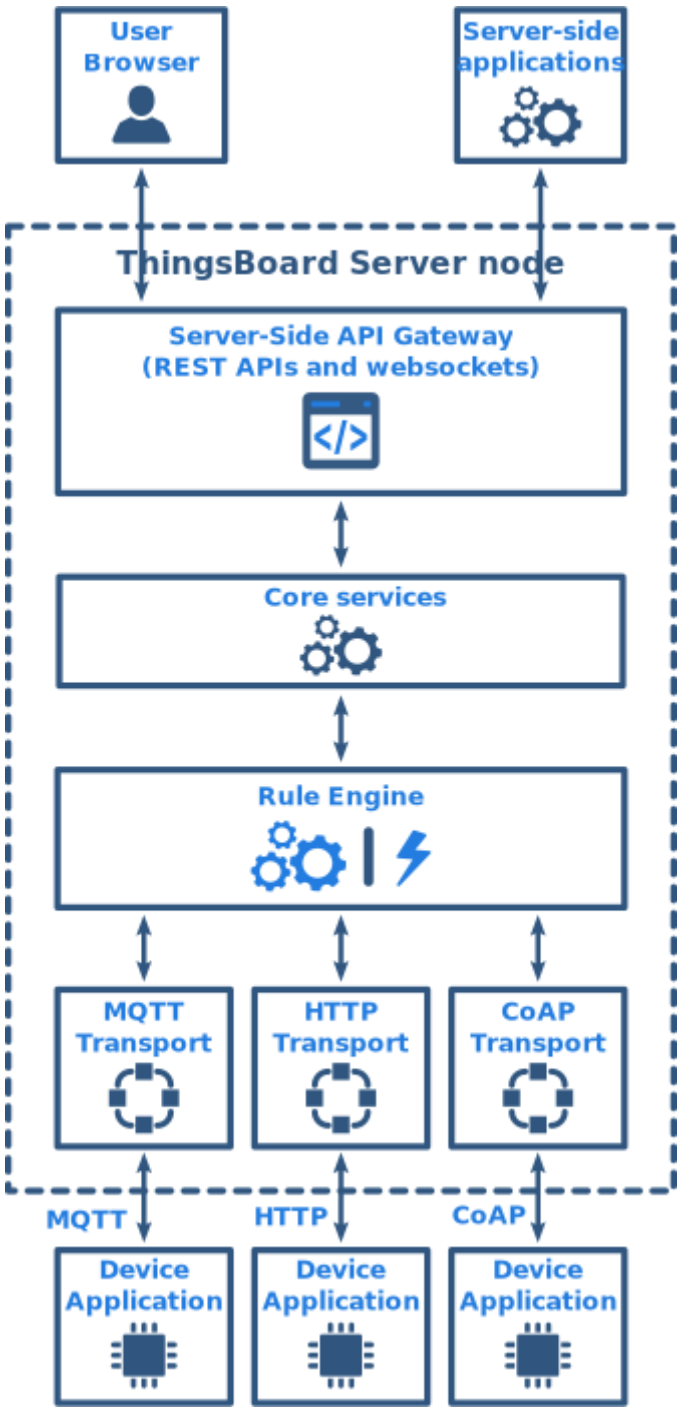
CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1 Yêu cầu hệ thống

- Dữ liệu cảm biến được cập nhật liên tục trên Server Web
- Nguồn ổn định thời gian sử dụng lâu
- Mạch thiết kế gọn nhẹ ưa nhìn
- Trang web hiển thị dễ nhìn

2.2 Sơ đồ khối của hệ thống IoT trên thị trường

Để thuận tiện cho việc thực thi và kiểm thử em sẽ so sánh hệ thống em xây dựng với hệ thống IoT Thingsboard (một platform IoT mã nguồn mở).



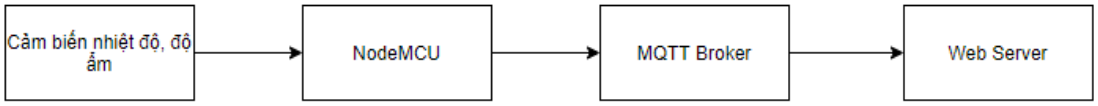
Hình 2.1 Sơ đồ khối thingsboard

- Kết nối thiết bị: Thingsboard hỗ trợ MQTT, CoAP và HTTP cho việc kết nối thiết bị. Có thể hỗ trợ các giao thức khác nhau hoặc tùy chỉnh triển khai hiện có.
- Rule Engine: Thingsboard Rule Engine cho phép xử lý thông báo từ thiết bị và kích hoạt các mô-đun xử lý có thể cấu hình được gọi là Plugin.
- Core services: Thingsboard chứa tập hợp các dịch vụ cốt lõi cho phép quản lý các thực thể sau:

- Thiết bị và thông tin đăng nhập của chúng
- Chuỗi quy tắc và mã quy tắc
- Người thuê và khách hàng
- Các tiện ích và bảng điều khiển
- Quy tắc báo động và sự kiện có thể gọi một tập hợp con nhất định của api này

2.3 Sơ đồ khối và sơ đồ nguyên lý hệ thống

2.3.1 Sơ đồ khối



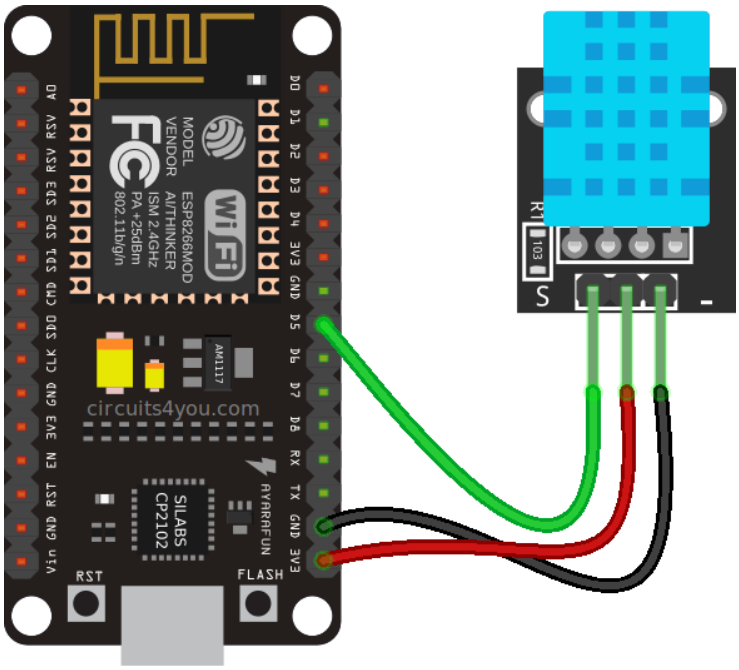
Hình 2.2 Sơ đồ khối

NodeMCU thu thập nhiệt độ, độ ẩm từ các sensors, rồi gửi lên Broker Server qua giao thức MQTT. Server Broker sau đó sẽ truyền lại tất cả các thông tin cho các client (ở đây là WebServer) đang subscribe cùng kênh topic với NodeMCU.

2.3.2 Nguyên lý hoạt động

Khối Phần Cứng

i, Khái niệm



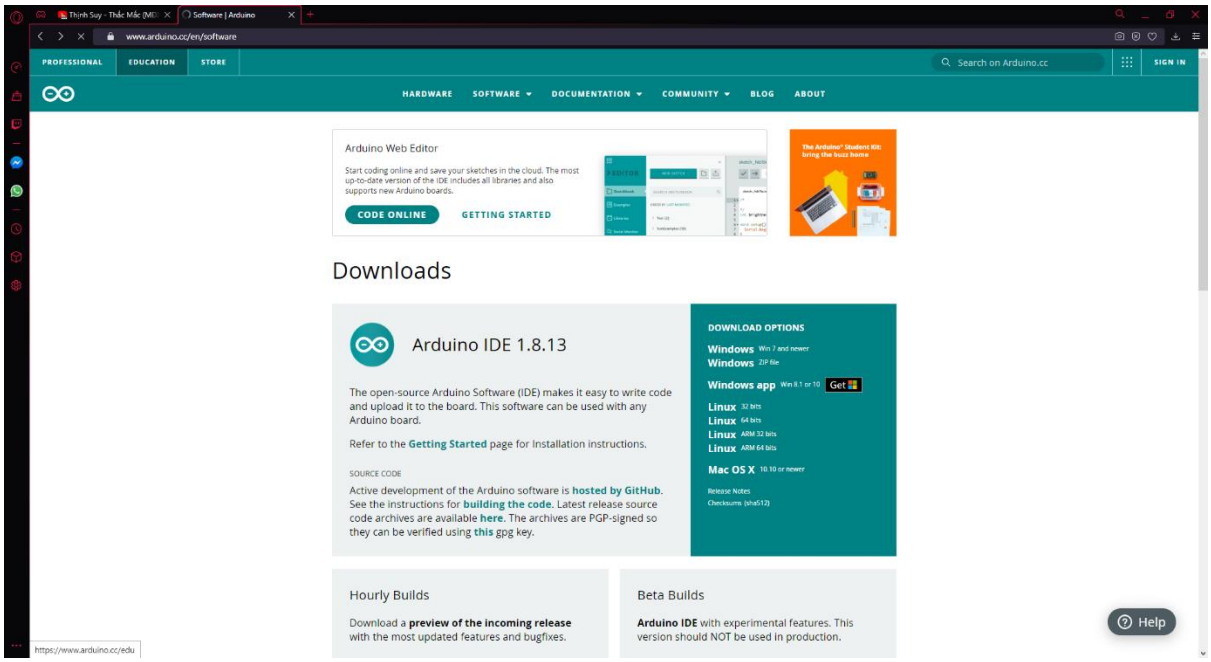
Hình 2.3 Sơ đồ phần cứng

NodeMCU	DHT11
GND	GND
3.3V	VCC
D0-D8	DATA PIN

Khối phần cứng có chức năng nhận dữ liệu từ sensor, sau đó gửi dữ liệu được mã hóa AES lên Broker qua MQTT.

ii, Cài đặt những phần mềm và thư viện cần thiết

Để lập trình cho NodeMCU ESP8266, em dùng ArduinoIDE, có thể tải ở trên trang chủ Arduino: https://www.arduino.cc/en/software

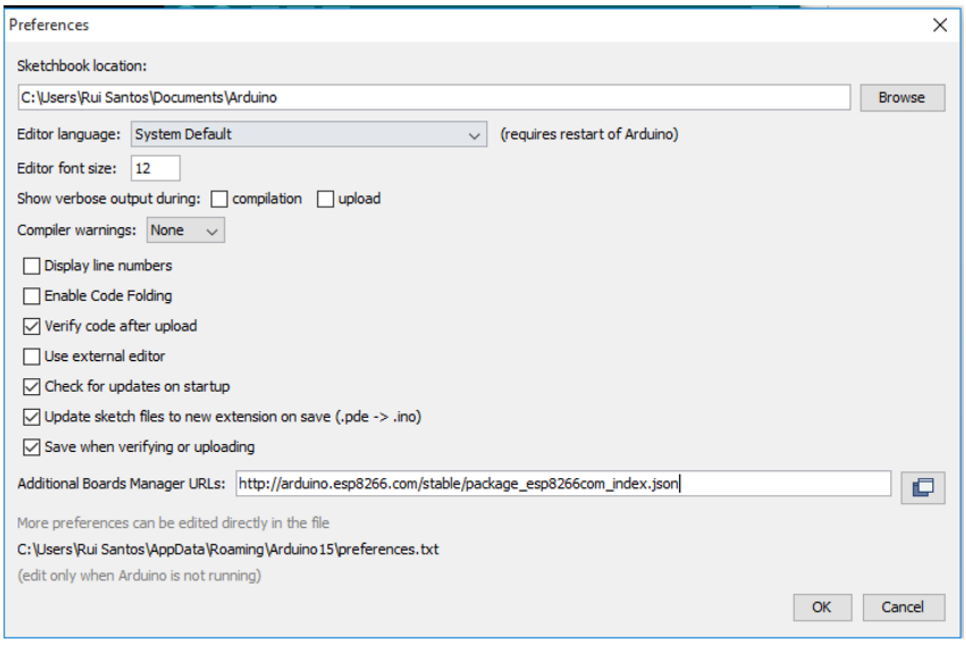


Hình 2.4 Trang chủ Arduino

Để tiến hành cài đặt thư viện và chức năng nạp code cho IDE em làm như sau:

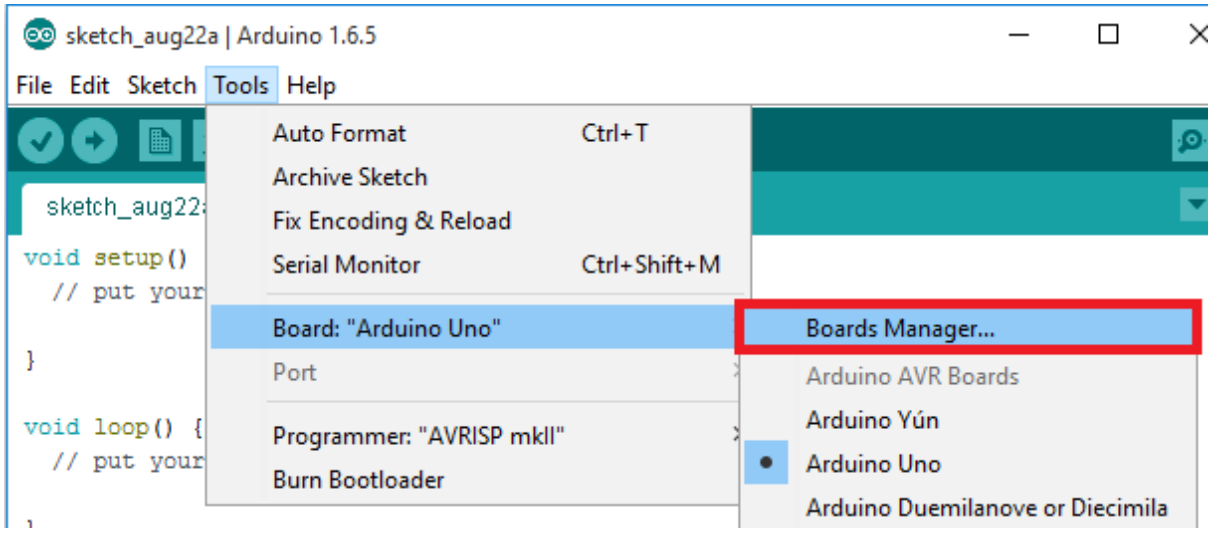
Vào File→ Preferences, vào textbox Additional Board Manager URLs thêm đường link sau vào: http://arduino.esp8266.com/stable/package_esp8266com_index.json

Click **OK** để chấp nhận.



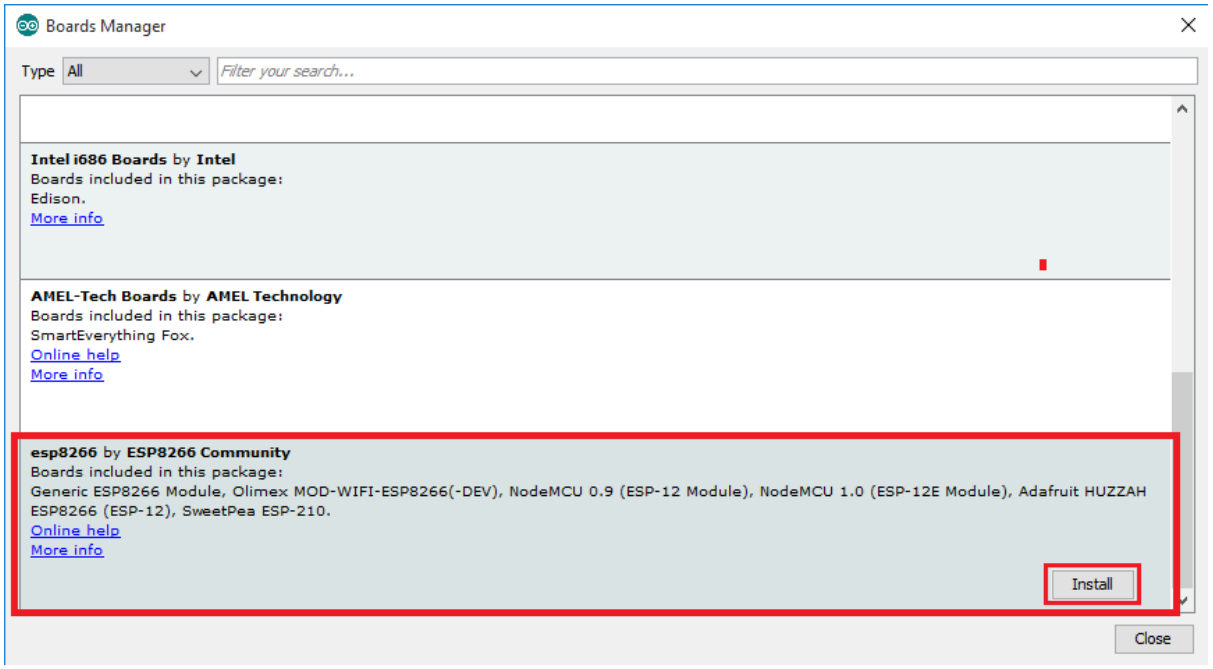
Hình 2.5 Cài đặt board NodeMCU

Tiếp theo vào Tool→Board→Boards Manager



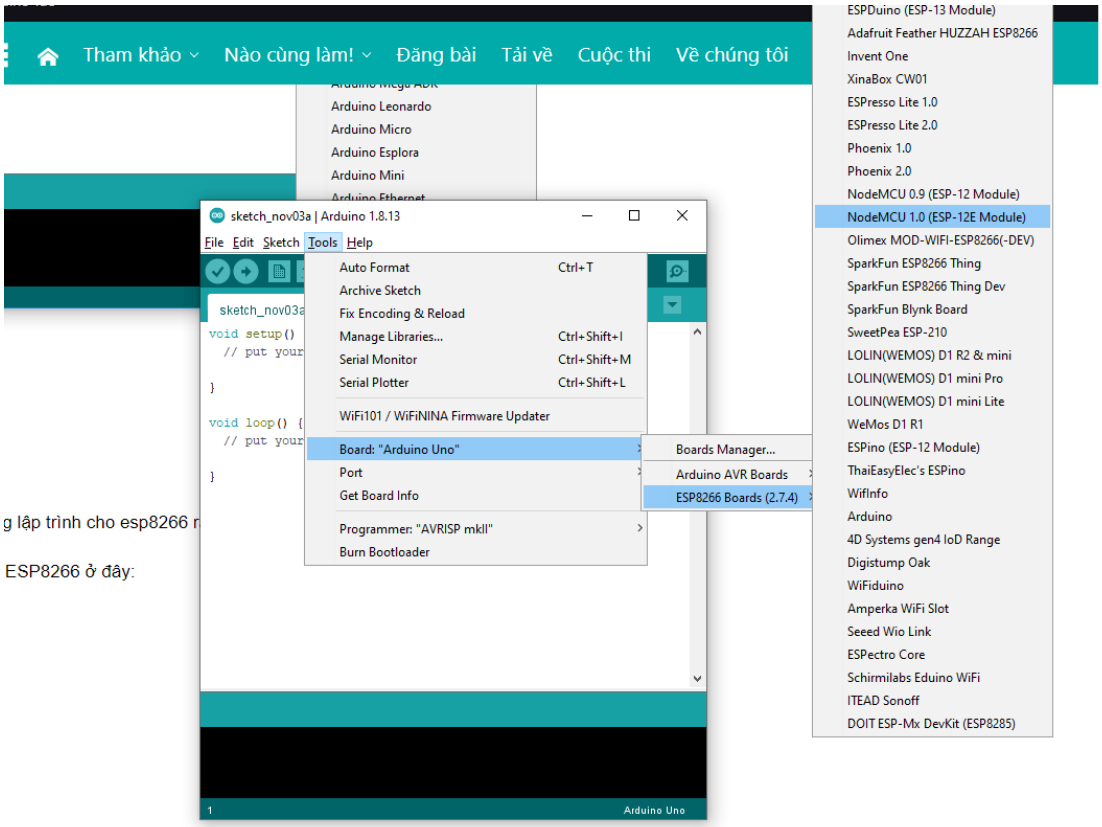
Hình 2.6 Cài đặt board NodeMCU

Đợi một lát để chương trình tìm kiếm. Ta kéo xuống và click vào ESP8266 by ESP8266 Community, click vào Install. Chờ phần mềm tự động download và cài đặt.



Hình 2.7 Cài đặt board NodeMCU

Kết nối NodeMCU vào máy tính. Vào Tool→Board→NodeMCU 1.0, chọn cổng COM tương ứng với module tương ứng.



Hình 2.8 Cài đặt board NodeMCU

Sau đó em cài các thư viện cần thiết để chạy được code, các thư viện dưới đây đều được có chính thức trên mục add thư viện trong ArduinoIDE. Em xin để link dưới đây:

- PubSubClient
- Adafruit Unified Sensor
- DHT sensor library
- AESlib

Khối Server Broker

i, Khái niệm

Server Broker em sử dụng là Mosquitto, một broker mã nguồn mở cho phép thiết bị truyền nhận dữ liệu theo giao thức MQTT version 5.0, 3.1.1 và 3.1.

- Ưu điểm:

- Ưu điểm nổi bật của Mosquitto là tốc độ truyền nhận và xử lí dữ liệu nhanh, độ ổn định cao, được sử dụng rộng rãi và phù hợp với những ứng dụng embedded.
- Mosquitto rất nhẹ và phù hợp để sử dụng trên tất cả các thiết bị.
- Ngoài ra, Mosquitto cũng được hỗ trợ các giao thức TLS/SSL (các giao thức nhằm xác thực server và client, mã hóa các message để bảo mật dữ liệu).

- Nhược điểm:
  - Một số nhược điểm của mosquitto là khó thiết kế khi làm những ứng dụng lớn và ít phương thức xác thực thiết bị nên khả năng bảo mật vẫn chưa tối ưu.

ii, Cài đặt Mosquitto Broker

Để cài đặt Mosquitto Broker trên Ubuntu, cần thực hiện như sau:

```
sudo apt-get update
sudo apt-get install mosquitto
```

Để kiểm tra xem server broker đã hoạt động chưa:

```
sudo systemctl status mosquitto
```

```
thao@thao-HP-ZBook-15:~$ sudo systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT Broker
     Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2020-11-03 16:47:51 +07; 6h left
       Docs: man:mosquitto.conf(5)
             man:mosquitto(8)
    Process: 1136 ExecStartPre=/bin/mkdir -m 740 -p /var/log/mosquitto (code=exited, status=0/SUCCESS)
    Process: 1140 ExecStartPre=/bin/chown mosquitto: /var/log/mosquitto (code=exited, status=0/SUCCESS)
   Main PID: 1141 (mosquitto)
      Tasks: 1 (limit: 9250)
     Memory: 3.3M
     CGroup: /system.slice/mosquitto.service
             └─1141 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

Thg 11 03 16:47:50 thao-HP-ZBook-15 systemd[1]: Starting Mosquitto MQTT Broker...
Thg 11 03 16:47:50 thao-HP-ZBook-15 mosquitto[1141]: 1604396870: Loading config file /etc/mosquitto/conf.d/default.conf
Thg 11 03 16:47:51 thao-HP-ZBook-15 systemd[1]: Started Mosquitto MQTT Broker.
thao@thao-HP-ZBook-15:~$
```

Hình 2.9 Server broker đang hoạt động

Cấu hình mosquitto để có thể gửi data lên Web Server qua Websocket (cổng 9001) trong file /etc/mosquitto/mosquitto.conf:

```
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log
port 1883
listener 9001
protocol websockets
include_dir /etc/mosquitto/conf.d
~
~
```

Hình 2.10 Cấu hình WebSocket cho server

Thiết lập xác thực client cho Server Broker:

sudo mosquitto_passwd -c /etc/mosquitto/passwd <user_name>

Sau khi nhập password 2 lần, em đã cấu hình thành công xác thực user_name, password cho Broker.

Khối Web Server

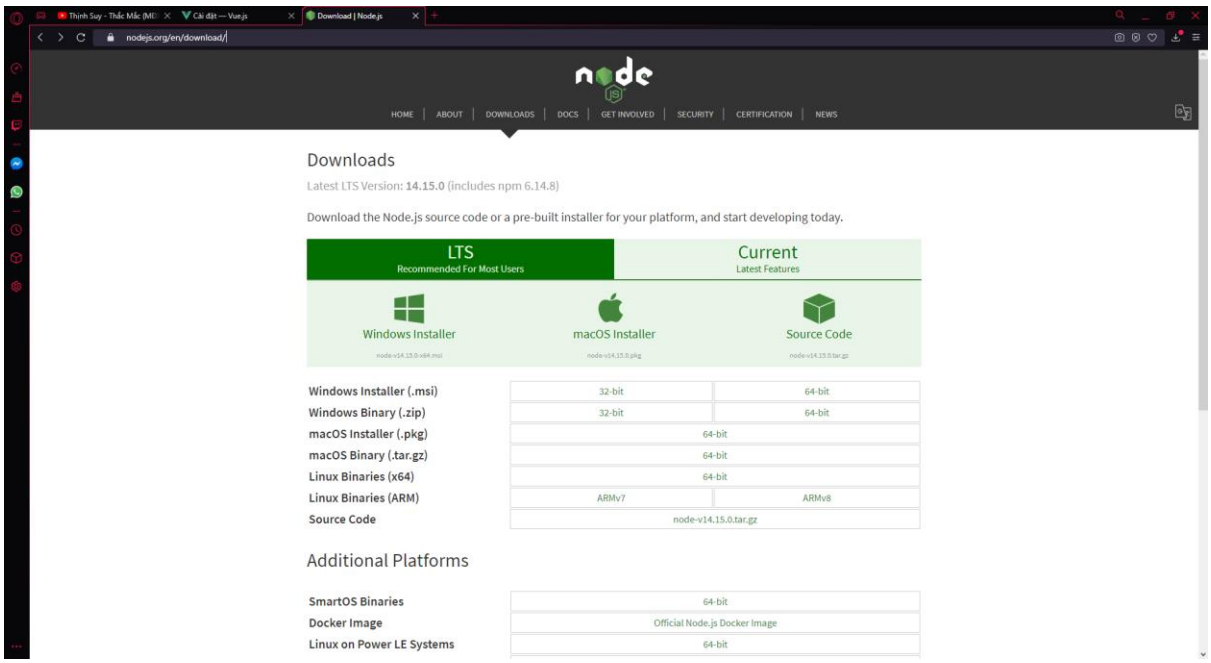
i, Khái niệm

Để lập trình khối Web Server, em dùng framwork VueJs (là một framework linh động dùng để xây dựng giao diện UI.

- Ưu điểm:
    - Tài liệu đơn giản, dễ học, API gọn nhẹ
    - Tăng tốc quá trình học, quá trình xây dựng ứng dụng
    - Thích hợp tạo các Single-page Application
    - Đơn giản, lõi nhỏ gọn tối đa với với khả năng tương thích cao có thể xây dựng ứng dụng ở mọi quy mô.

ii, Cài đặt

Trước khi cài framwork Vuejs, ta cần cài NodeJs (một nền tảng được xây dựng trên V8 JavaScript Engine – trình thông dịch thực thi mã JavaScript, giúp xây dựng các ứng dụng web một cách đơn giản và dễ dàng mở rộng). Download NodeJs từ trang chủ: https://nodejs.org/en/download/.



Hình 2.11 Trang chủ Nodejs

Sau đó em cài đặt VueJS qua npm:

```
npm install vue

npm install --global vue-cli
```

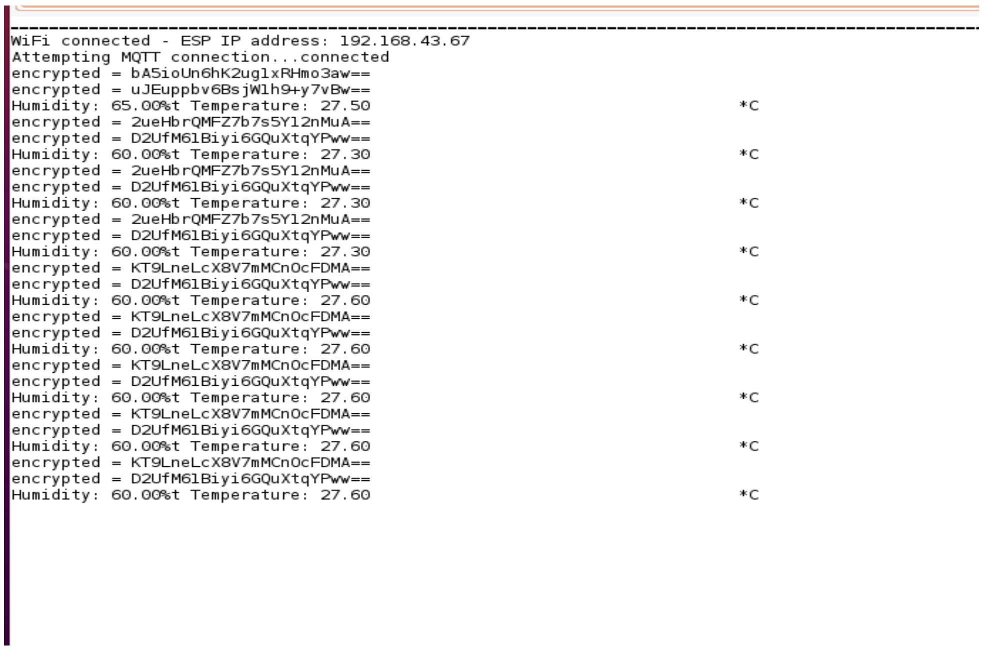
Tạo và chạy project bằng câu lệnh sau:

```
vue init webpack my-project

cd my-project

npm run dev
```

2.4 Thực thi và kiểm thử hệ thống

2.4.1 Phần Cứng

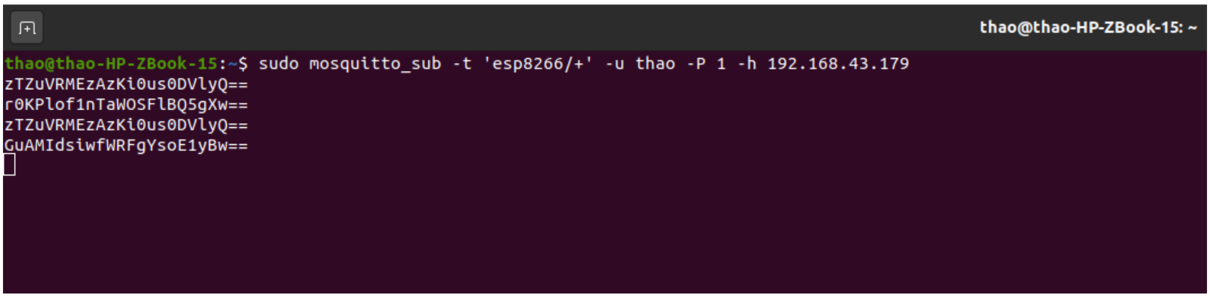
Hình 2.12 Sản phẩm thu được



Hình 2.13 Kết quả thu được từ phần ứng

Kết quả đang xem được lấy từ serial monitor trên ArudinoIDE khi kết nối sản phầm với máy tính qua USB. Đoạn encrypted có 2 dòng lần lượt mô tả 2 giá trị humidity và temperature được mã hóa AES trên đường truyền và gửi sang cho Broker.

2.4.2 Server Broker



Hình 2.14 Kết quả thu được trên server Broker

Kết quả trả về khi subscribe cùng kênh với sensor. Trong đó:

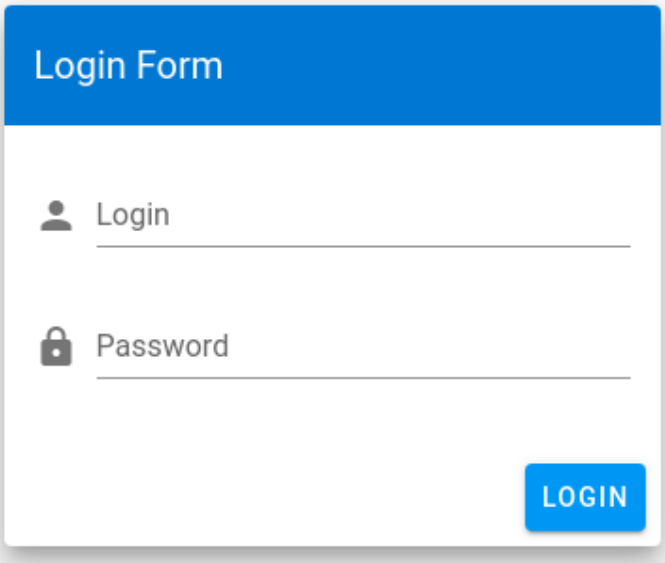
-t : topic của kênh truyền

-u, -P: username và password do em đã thiết lập xác thực cho Broker.

-h: Địa chỉ IP của Server Broker

Các cặp giá trị bên dưới là giá trị nhiệt độ, độ ẩm được mã hóa.

2.4.3 Web Server



Hình 2.15 Màn hình đăng nhập

Để xem được dữ liệu trên web, người dùng cần đăng nhập vào hệ thống, tăng độ bảo mật cho hệ thống IoT.

Hình 2.16 Màn hình hiển thị nhiệt độ độ ẩm

Đây là giao diện Web khi người dùng đăng nhập, hiển thị rõ ràng, dễ nhìn các thông số nhiệt độ, độ ẩm thu thập được sau khi đã mã hóa.

KẾT LUẬN

Sau một thời gian thực hiện đề tài, em đã hoàn thành được một Hệ thống IoT cơ bản và có độ bảo mật tương đối cao.

Sau khi hoàn thành được đề tài này, em đã hiểu thêm về hệ thống IoT. Chúng em có thêm các kỹ năng về lập trình phía phần cứng, phần mềm và biết cách ghép nối lại thành một hệ thống hoàn chỉnh. Nhờ vậy, kỹ năng tự học và vận dụng các công nghệ mới của chúng em được nâng cao hơn.

Bên cạnh đó, do còn thiếu kinh nghiệm thực tế, em vẫn chưa hoàn thiện được giao diện đẹp, và thu hút người dùng. Chúng em sẽ tiếp tục nghiên cứu và phát triển hệ thống thêm nhiều chức năng hơn không chỉ hiển thị nhiệt độ mà còn có thể điều khiển được các thiết bị từ xa.

Em xin chân thành cảm ơn Thầy đã tận tình giúp đỡ và hướng dẫn để em có thể hoàn thành đề tài.

TÀI LIỆU THAM KHẢO

1. https://viblo.asia/newest
2. https://vuejs.org
3. https://mosquitto.org
4. https://vuetifyjs.com/en/
5. https://medium.com/@eranda/setting-up-authentication-on-mosquitto-mqtt-broker-de5df2e29afc
6. http://arduino.vn