

**CÔNG TY TNHH**

**CloudFERMI**

-----o0o-----



**BÁO CÁO**  
**THỰC TẬP TỐT NGHIỆP**  
**HỆ THỐNG GIÁM SÁT NHIỆT ĐỘ, ĐỘ ẨM**  
**QUA MẠNG LORA**

**Nhóm Internship 1**

**Trịnh Vĩ Luân**

**Nguyễn Văn Bình**

**Hoàng Văn Quốc**

**Khương Đức Mạnh**

**TP. HỒ CHÍ MINH, 19 THÁNG 08 NĂM 2018**

## ***LỜI CẢM ƠN***

Xin chân thành gửi lời cảm ơn đến thầy Lê Chí Thông và anh Bùi Hoàng Cương giám đốc công ty TNHH CloudFERMI đã tận tình giúp đỡ em trong suốt thời gian vừa qua. Những lời nhận xét, góp ý chân thành của thầy và anh đã giúp em thấy được khuyết điểm của mình để ngày càng khắc phục tốt hơn.

*Tp. Hồ Chí Minh, ngày 19 tháng 08 năm 2018.*

**Sinh viên**

# MỤC LỤC

I GIỚI THIỆU .....	1
1 Giới thiệu về công ty.....	1
2 Nhiệm vụ được giao thực tập.....	1
3 Thời gian và lịch trình thực tập.....	1
II NỘI DUNG THỰC TẬP .....	2
1. Tổng quan : .....	2
1.1 Đặt vấn đề :.....	2
1.2 Tình hình nghiên cứu trong và ngoài nước: .....	4
1.2.1 Nghiên cứu trong nước : .....	4
1.2.2 Nghiên cứu ngoài nước :.....	4
2. Lý thuyết : .....	5
2.1 Cơ bản về mạng LORA : .....	5
2.1.1 Tổng quan về LORA : .....	5
2.1.2 Giao thức LORAWAN :.....	13
2.2 Nguyên lý truyền và nhận dữ liệu trong mạng LoRa .....	15
2.2.1. Truyền dữ liệu.....	15
2.2.2. Nhận dữ liệu trong mạng LoRa .....	16
2.2.3. Cách kết nối giữa node và Server .....	17
2.3. Giới thiệu ESP32 .....	18
2.3.1 Sơ đồ khối .....	18
2.3.2 Sơ đồ chân.....	19

2.3.3 CPU .....	19
2.3.4 Hỗ trợ 2 giao tiếp không dây .....	20
2.3.5 Hỗ trợ tất cả các loại giao tiếp .....	20
2.3.6 Cảm biến tích hợp trên chip esp32 .....	20
2.3.7 Bảo mật .....	20
2.3.8 Nguồn điện hoạt động .....	20
2.3.9 Đánh giá chung .....	21
2.4 Giao thức MQTT .....	22
2.5 SD Card.....	24
2.6 Chip LAN 8270A.....	26
2.7 Giới thiệu module, IC .....	30
2.7.1 Module Sensor-Node-Lora :.....	30
2.7.2. Module SOM-ESP32-3 : .....	31
2.7.3. Cảm biến nhiệt độ và độ ẩm ( DHT21/AM2301) : .....	32
2.8 Phần mềm : .....	35
2.8.1 Giải thuật các Node .....	35
2.8.2 MQTT Broker.....	37
2.8.3 Arduino ESP32 into an Access Point.....	39
2.8.4 Arduino ESP32 với HTML .....	41
2.9 Lưu đồ giải thuật:.....	45
2.10 Kết quả thực hiện .....	47
3. TỔNG KẾT CÔNG VIỆC THỰC TẬP .....	52
3.1. Kết quả công việc thực tập .....	52

3.1.2. Kinh nghiệm học được sau khi thực tập .....	53
4. TÀI LIỆU THAM KHẢO .....	53
5. PHỤ LỤC.....	54

# I GIỚI THIỆU

## 1 Giới thiệu về công ty

- CloudFERMI là công ty sản xuất thiết bị điện tử , hướng IOT ( kết nối , xử lý thông minh ) , tập trung vào cảm biến và điều khiển . Công ty tạo ra thiết bị điện tử bằng cách thiết kế bo mạch , vẽ bo mạch , hàn linh kiện , lập trình nhúng , lập trình ứng dụng , sản xuất , kiểm soát chất lượng . Lợi thế cốt lõi dựa trên năng lực R&D , năng lực sản xuất và dịch vụ khách hàng .

## 2 Nhiệm vụ được giao thực tập

- Nhiệm vụ : Thiết kế hệ thống giám sát nhiệt độ, độ ẩm sử dụng mạng LORA thông qua 2 bo mạch Easy Lora Node và SOM-ESP32-3 do công ty sản xuất.

**Nội dung 1:** Tìm hiểu về VĐK STM32 , cảm biến DHT21 , mạng LORA, ESP32 , WIFI , LAN và giao thức MQTT .

**Nội dung 2:** Tiến hành đọc nhiệt độ ,độ ẩm từ DHT21 về VĐK STM32 và truyền dữ liệu thông qua LORA SX1278 .

**Nội dung 3:** dùng ESP32 đọc dữ liệu từ mạng LORA và gửi lên WEBSERVER thông qua giao thức MQTT . Giám sát các thông số và điều khiển thiết bị thông qua APP điện thoại

## 3 Thời gian và lịch trình thực tập

- Thời gian thực tập: 2 tháng ( từ ngày 17/6/2018 đến ngày 17/8/2018 )

- Lịch trình thực tập:
  - Tuần 1: Tìm hiểu cơ sở lý thuyết liên quan đến đề tài được giao
  - Tuần 2: Tiến hành đọc dữ liệu từ DHT21 sử dụng STM32 với độ chính xác cao .
  - Tuần 3: Tiến hành truyền dữ liệu sử dụng mạng LORA
  - Tuần 4 : Sử dụng ESP32 đọc dữ liệu từ LORA và xử lý .
  - Tuần 5 : Gửi dữ liệu lên WEBSERVER qua giao thức MQTT sử dụng WIFI và LAN .
  - Tuần 6 : Tạo app trên hệ điều hành Android để giám sát nhiệt độ , độ ẩm và điều khiển các thiết bị .
  - Tuần 7,8 : Kiểm tra hệ thống , khắc phục lỗi và báo cáo.

## II NỘI DUNG THỰC TẬP

### 1. Tổng quan :

#### 1.1 Đặt vấn đề :

- Trong những thập niên gần đây, tình trạng ô nhiễm môi trường đang ở mức đáng báo động, và đặc biệt nghiêm trọng. Sự thải các chất ô nhiễm môi trường không qua xử lý gây nên hậu quả nghiêm trọng đến sức khỏe con người. Một trong những loại khí thải gây tác động xấu tới sức khỏe con người đó chính là Cacbon monôxít (CO). Những nơi có mật độ khí CO lớn là những điểm tắc nghẽn giao thông, các bến xe, nhà để xe hoặc trong những nơi làm việc kín như hầm lò, nhà máy. CO có độc tính cao cực kỳ nguy hiểm với sức khỏe con người, nếu bị hít một lượng lớn sẽ gây thương tổn với cơ thể, nó cản trở khả năng vận chuyển oxy trong máu. Chính vì thế, việc giám sát khí CO rất quan trọng, từ đó tìm ra những giải pháp để giảm thiểu lượng khí thải đó. Việc thu

thập chất lượng môi trường không khí (trong đó có CO) được thu thập thông qua mạng IoT. Vậy IoT là gì?

- IoT (Internet of Thing) là sự kết nối tất cả các thiết bị với nhau, có khả năng trao đổi thông tin, cung cấp dữ liệu với con người mà không cần phải tương tác trực tiếp.
- Con người có thể kết nối tất cả các thiết bị với mạng Internet thông qua mạng nội bộ.
- Trong những năm gần đây, IoT đang phát triển nhanh đến chóng mặt. Theo sự tính toán thông kê có đến 50 triệu thiết bị được kết nối cho tới năm 2020. Con người đang biến tất cả các thiết bị trong đời sống hằng ngày như ô tô, thiết bị sản xuất, dụng cụ trong nhà, đồ mặc ... đều có thể điều khiển, kiểm soát, thu thập dữ liệu chỉ bằng laptop hay điện thoại. Công nghệ IoT giúp cho con người sống tốt hơn, và đối phó với vấn đề lớn nhất đang gặp phải của thế giới đó là biến đổi khí hậu, kiểm soát ô nhiễm, cảnh báo các vấn đề tự nhiên. Tuy nhiên, đòi hỏi về công suất thấp cho các thiết bị IoT không hề đơn giản, các thiết bị hiện nay dùng RFID, Bluetooth hay Wifi đều là những công nghệ với công suất thấp nhưng khoảng cách ngắn. Để đáp ứng được công suất thấp và khoảng cách xa, LoRa là một giải pháp tốt nhất tại thời điểm hiện nay.

- LoRa là công nghệ mạng không dây được phát triển để tạo ra được công suất thấp (low-power), mạng lưới rộng (LPWANs- Low Power Wide Area Networks) dùng cho các ứng dụng Internet of Thing. Công nghệ này hấp dẫn với khoảng cách xa, công suất tiêu thụ thấp và việc truyền dữ liệu an toàn. Ưu điểm của mạng lưới được xây dựng với LORA so với mạng lưới hiện tại là việc phủ sóng lớn. Với khoảng cách xa và công suất thấp, LORA tự tin là ứng cử viên cho công nghệ thông minh trong hạ tầng dân dụng ( chẳng hạn như giám sát sức



khỏe , đo lường thông minh , giám sát môi trường ) ... cũng như các ứng dụng công nghiệp .

## **1.2 Tình hình nghiên cứu trong và ngoài nước:**

### **1.2.1 Nghiên cứu trong nước :**

Qua tìm hiểu về tình hình nghiên cứu trong việc ứng dụng mạng LORA khá thấp, cho thấy việc ứng dụng vào công nghệ này chưa được quan tâm nhiều . Tuy nhiên hiện nay việc ứng dụng mạng LORA đang được xem xét, đầu tư vào các ứng dụng như đọc chỉ số đồng hồ điện nước, quản lý bãi xe , nông nghiệp thông minh , ... Đây là dấu hiệu khả quan trong việc nghiên cứu mạng LORA .

### **1.2.2 Nghiên cứu ngoài nước :**

Hiện nay có nhiều cá nhân, công ty nghiên cứu phát hành sản phẩm LoRa một hay nhiều kênh truyền dựa trên chipset của Semtech. Các công trình nghiên cứu:

- Đề tài “A DIY low-cost LoRa gateway”[1][1] của Giáo sư Phạm Công Đức, trường đại học Pau, Pháp sử dụng chip SX1276 của Semtech với gateway một kênh truyền.
- Sản phẩm EMB-GW1301 của công ty Embit, Italy[2][2] dựa trên chipset SX1301 của Semtech cho phép hoạt động nhiều kênh truyền trong cùng một khoảng thời gian, cung cấp giải pháp cho công nghệ IoT.
- Các sản phẩm cảm biến không dây của công ty nke Wattec, Pháp với các sản phẩm công nghệ LoRa như cảm biến nhiệt độ, độ ẩm, ánh sáng, đo lượng nước,..[3][3] cho hoạt động hiệu quả và thời gian sử dụng pin tốt nhất.
- Module LoRaWan IXM-LPWA-800-16-K9 của Cisco[4][4] cho các ứng dụng cần công suất thấp, diện tích phủ rộng lớn như tracking vật thể,

đo nước hay khí, các tòa nhà thông minh, đèn đường, giám sát môi trường và nông nghiệp thông minh.

## **2. Lý thuyết :**

### **2.1 Cơ bản về mạng LORA :**

#### **2.1.1 Tổng quan về LORA :**

##### **2.1.1.1 Giao thức LORA :**

LoRa là viết tắt của Long Range Radio được nghiên cứu và phát triển bởi Cycleo và sau này được mua lại bởi công ty Semtech năm 2012. Với công nghệ này, chúng ta có thể truyền dữ liệu với khoảng cách lên hàng km mà không cần các mạch khuếch đại công suất; từ đó giúp tiết kiệm năng lượng tiêu thụ khi truyền/nhận dữ liệu. Do đó, LoRa có thể được áp dụng rộng rãi trong các ứng dụng thu thập dữ liệu như sensor network trong đó các sensor node có thể gửi giá trị đo đạc về trung tâm cách xa hàng km và có thể hoạt động với battery trong thời gian dài trước khi cần thay pin.

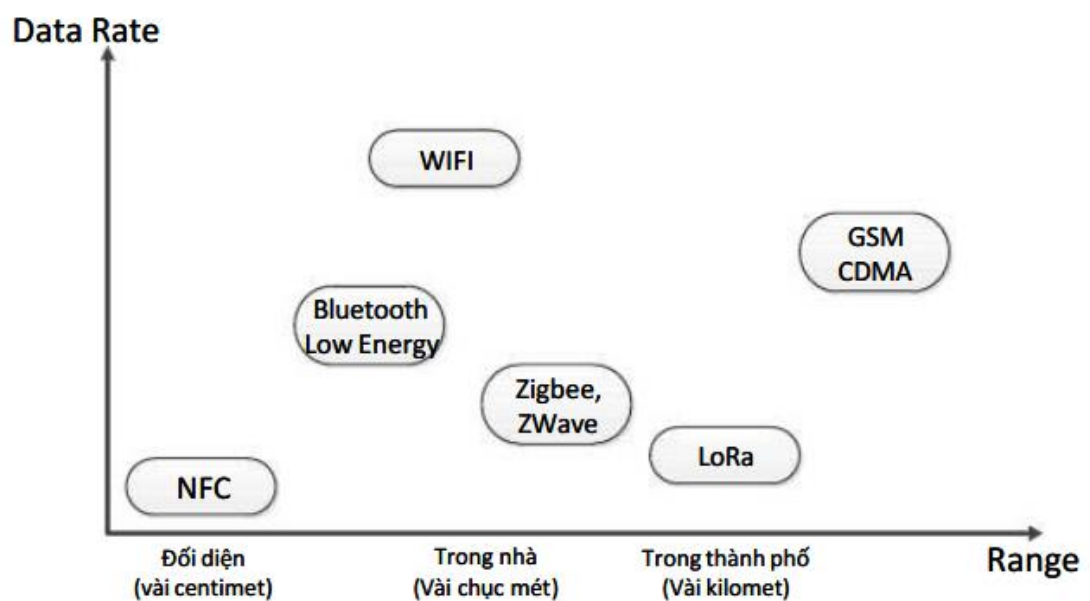
Lớp vật lý LORA được phát triển bởi Semtech, cho khoảng cách xa, công suất thấp và giao tiếp băng thông thấp. Hoạt động trên băng tần ISM 433-,868- hoặc 915-MHZ , phụ thuộc vào từng khu vực triển khai . Khối lượng (gọi là payload) của mỗi đợt truyền tải có thể đạt từ 2-255 byte, tốc độ data có thể đạt tới 50Kbps.

Dưới đây là bảng đặc điểm của công nghệ LORA về khoảng cách, chuẩn giao tiếp, công suất, lớp vật lý .

Đặc điểm kỹ thuật	Định nghĩa
Khoảng cách	2-5 kilomet trong khu vực thành thị và 15 km trong vùng ngoại ô
Băng tần	Băng tần ISM 433MHz, 868MHz và 915 MHz
Chuẩn giao tiếp	IEEE 802.15.4g
Điều chế	Điều chế trải phổ được sử dụng xung dải băng tần FM. Tần số tăng hoặc giảm trên thời gian nhất định được sử dụng để mã hóa dữ liệu khi được gửi.
Công suất	Một mạng Lora quản lý hàng ngàn node
Điện năng	Sử dụng lâu
Lớp vật lí	Quản lý tần số, công suất, điều chế, tín hiệu giữa các node và Gateway

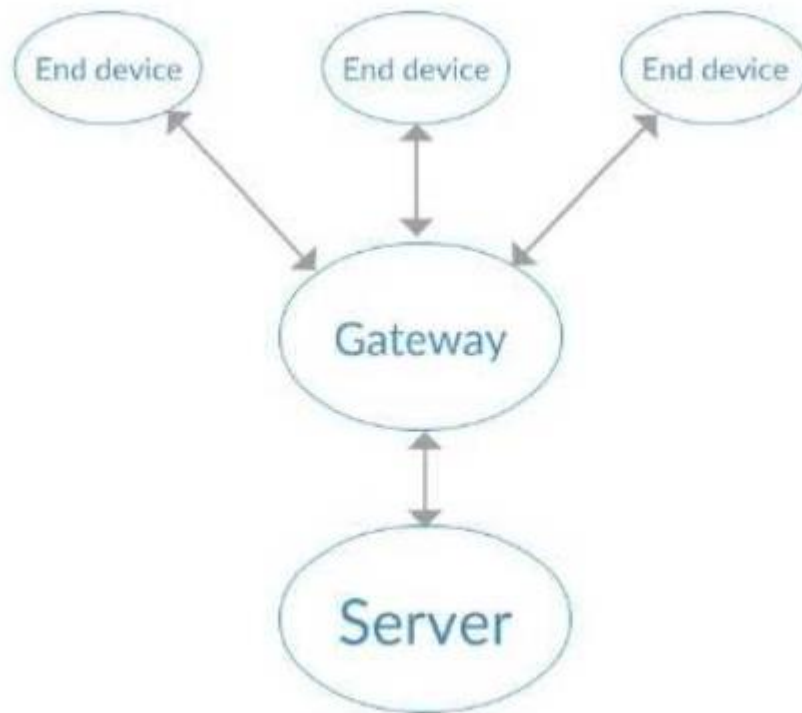
Bảng 2. 1: Đặc điểm kỹ thuật của chuẩn giao tiếp LoRa

Biểu đồ so sánh giữa mạng LORA và các chuẩn giao tiếp thông dụng:



### 2.1.1.2 Kiến trúc mạng LORA :

Một mạng LORA đặc trưng cấu hình mạng sao, bao gồm 3 loại thiết bị khác nhau



Kiến trúc cơ bản của mạng LoRaWAN: các thiết bị cuối giao tiếp với Gateway sử dụng LoRa với LoRaWAN. Gateway chuyển các frame LoRaWAN từ thiết bị tới một Server mạng, sau đó chuyển lên thiết bị có băng thông hơn, đặc trưng là Ethernet hoặc 3G.

Có 3 lớp thiết bị cuối : Lớp A ( cho tất cả ) , lớp B ( cho Beacon ) , lớp C ( cho việc nghe liên tục ) .

### 2.1.1.3 Lớp vật lý LORA :

Điều chế LORA là một công nghệ độc quyền của Semtech. Phần này phân tích, đánh giá (phần độc quyền của LORA) với mục đích hiểu rõ liệu rằng hiệu suất của LORA được quan sát trong thực tế .

#### c.1) Tổng quan về lớp vật lý:

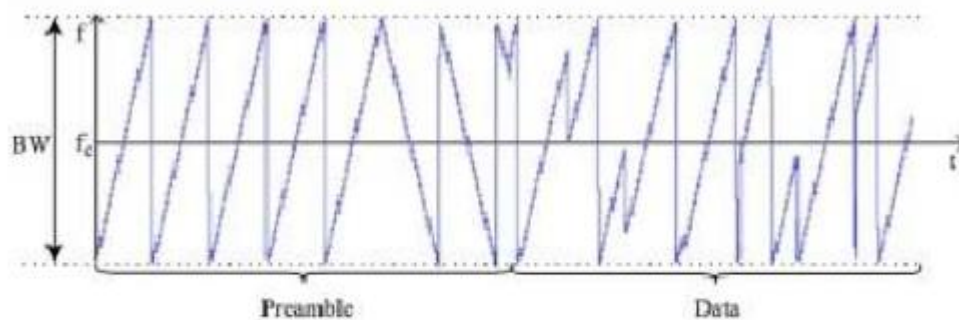
LoRa là điều chế trải phổ theo cường độ và pha (Chirp Spread Spectrum), sử dụng cường độ và pha tần số với sự biến đổi tuyến tính của tần số theo thời gian để mã hóa thông tin. Bởi vì sự tuyến tính của xung trải phổ, độ lệch tần số giữa các thiết bị thu và phát tương ứng với độ lệch thời gian, dễ dàng bị loại bỏ trong giải mã. Điều này làm cho việc miễn nhiễm việc điều chế do ảnh hưởng của hiệu ứng Doppler, tương ứng với độ lệch tần số. Độ lệch tần số giữa thu và phát có thể đạt đến 20% băng thông mà không ảnh hưởng hiệu suất mã hóa. Việc nhận LoRa có thể khóa tần số nhận được, cung cấp độ nhạy lên tới -130 dBm. Khi thời gian sống của kỹ thuật LoRa dài hơn so với công nghệ trải phổ nhảy tần Frequency Hopping Spread Spectrum (FHSS), các lỗi được tạo bởi nhiễu có thể dễ dàng được sửa thông qua việc sửa lỗi tiếp chuyển (Forward Error-correction Codes - FECs). Nhanh hơn điều chế truyền thông chẳng hạn như FSK làm cho LoRa thích hợp với công suất thấp và truyền tin với khoảng cách dài.

#### c.2) Thông số của lớp vật lý:

Một vài thông số có sẵn cho việc tùy biến trong việc điều chế LoRa như : băng thông (BW), hệ số trải phổ (SF) và tỉ lệ mã hóa (CR). LoRa sử dụng một định nghĩa độc đáo cho việc trải phổ như logarit, số lượng Chirp/symbol. Các thông số ảnh hưởng tới tốc độ bit, làm giảm ảnh hưởng của nhiễu, và dễ dàng giải mã. Băng thông là thông số quan trọng nhất trong việc điều chế LoRa. Một symbol LoRa tạo ra 2SF chirs, bao phủ toàn bộ băng thông tần số. Nó bắt đầu với

một chuỗi chirp được tăng lên. Khi tần số của băng thông đạt cực đại, tần số sẽ được bọc kín xung quanh, việc tăng tần số được bắt đầu lại từ tần số nhỏ nhất.

Hình 2.8 đưa ra ví dụ về việc vận chuyển LoRa trong việc thay đổi tần số theo thời gian. Vị trí không liên tục trong chuỗi tần số được mã hóa thông tin được chuyển đi. Có  $2^{SF}$  trong một ký tự, một ký tự có thể mã hóa SF(bits) một cách hiệu quả.



**$F_c$  là tần số trung tâm của kênh truyền, và BW là băng thông**

Trong LoRa, tỉ lệ Chirp phụ thuộc vào băng thông: tỉ lệ chirp bằng với băng thông (một chirp/second/Hz băng thông). Một vài chuỗi trong điều chế: việc tăng hệ số trải phổ sẽ chia khoảng cách tần số một chirp (bằng  $2^{SF}$  chirp trên toàn bộ băng thông) và thời gian sống của một symbol được nhân lên gấp đôi. Tuy nhiên, việc chia hai tỉ lệ bit, khi nhiều bit hơn sẽ được vận chuyển trên mỗi symbol. Hơn thế nữa, tỉ lệ symbol và tỉ lệ bit được cho bởi hệ số trải phổ tỉ lệ thuận với băng thông tần số, khi băng thông gấp đôi sẽ gấp đôi tỉ lệ truyền được. Điều này được đưa ra với phương trình dưới đây, liên hệ giữa thời gian sống của một symbol ( $T_s$ ) với băng thông và hệ số trải phổ.

$$T_s = \frac{2^{SF}}{BW}$$

LoRa chứa mã hóa sửa lỗi. Code rate (CR) bằng  $4/(4+n)$ , với  $n \in \{1,2,3,4\}$ .

Phương trình sau cho phép tính toán tỉ lệ bit ( $R_b$ ).

$$R_b = SF \times \frac{BW}{2^{SF}} \times CR$$

Ví dụ,  $BW = 125 \text{ kHz}$ ,  $SF = 7$ ,  $CR = 4/5$  cho tốc độ bit  $R_b = 5.5 \text{ kbps}$

Các thông số cũng ảnh hưởng tới độ nhạy giải mã. Việc tăng băng thông làm cho độ nhạy máy thu thấp hơn, trong khi đó tăng hệ số trải phổ sẽ tăng độ nhạy máy thu. Giảm tốc độ mã hóa giúp giảm tỉ lệ lỗi packet khi có sự ảnh hưởng của nhiễu. Ví dụ khi packet nhận được với tỉ tốc độ mã hóa là  $4/8$  sẽ tăng khả năng chống nhiễu hơn so với tín hiệu được vận chuyển với tỉ tốc độ mã hóa là  $4/5$ .

Bảng 2.4 lấy từ datasheet SX1276 :

SF \ BW	7	8	9	10	11	12
125 kHz	-123	-126	-129	-132	-133	-136
250 kHz	-120	-123	-125	-128	-130	-133
500 kHz	-116	-119	-122	-125	-128	-130

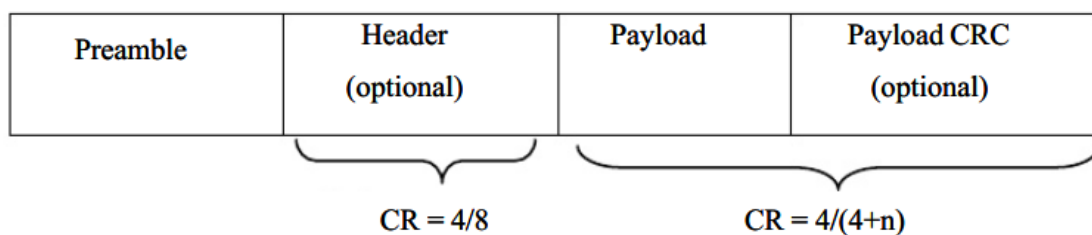
### c.3) Định dạng frame vật lý

Mặc dù điều chế LoRa có thể truyền frame bất kỳ, định dạng frame được xác định và tiến hành theo thu và phát của Semtech. Băng thông và hệ số trải phổ là hằng số cho một frame. Một frame được bắt đầu với một preamble. Preamble bắt đầu với một chuỗi hằng số bắt đầu gọi là upchirps được trải trên toàn băng thông tần số. Hai upchirps cuối cùng được giải mã ký tự đồng bộ (gọi là Sync word). Sync word có giá trị 1 byte, thường để phân biệt mạng LoRa khi sử dụng cùng một băng thông tần số. Một thiết bị được cấu hình với một sync word dừng việc lắng nghe việc vận chuyển nếu sync word không phù hợp với cấu

hình. Sync word được đưa ra với 2.25 downchirps, có độ dài 2.25 symbols.

Sau preamble, có một header tùy chọn. Khi được bật header, header này được gửi đi với tốc độ mã hóa là 4/8. Điều này chỉ ra rằng khối lượng của payload (theo bytes), tỉ lệ mã hóa được sử dụng cho cuối việc vận chuyển và liệu rằng có hay không 16 bit CRC cho payload có được hiện diện cuối frame hay không.

Kích thước của payload được lưu trữ một byte, giới hạn kích thước của payload tới 255 ký tự. Header được tùy chọn để cho phép được tắt trong tình huống liệu rằng có cần thiết hay không, ví dụ khi độ dài payload, tỉ lệ mã hóa và CRC.



### Cấu trúc frame của LORA

Sự liên hệ giữa spreading factor (SF), coding rate (CR) và băng thông tín hiệu (BW), thời gian gửi dữ liệu trong không gian của packet LoRa có thể được tính toán như dưới đây.

$$T_s = \frac{1}{R_s}$$

Thời hạn của packet LoRa là bằng tổng của chuỗi preamble và packet được gửi đi. Độ dài của preamble được tính toán như sau:

$$T_{preamble} = (n_{preamble} + 4.25)T_{sym}$$



Trong đó  $n_{\text{preamble}}$  là độ dài preamble của chương trình, được lấy từ thanh ghi RegPreambleMsb và RegPreambleLsb. Thời gian của payload phụ thuộc vào chế độ header được bật. Công thức dưới đây đưa ra số ký tự payload

$$n_{\text{payload}} = 8 + \max \left( \text{ceil} \left[ \frac{(8PL - 4SF + 28 + 16CRC - 20IH)}{4(SF - 2DE)} \right] (CR + 4), 0 \right)$$

PL : là số byte của payload ( 1 tới 255 )

SF : là spreading factor ( 6 tới 12 )

IH=0 khi header được bật , IH=1 không có header hiện diện .

DE =1 khi *LowdataRateOptimize* =1 .

CR là coding rate (1 tương ứng với 4/5 , 4 là 4/8 ) .

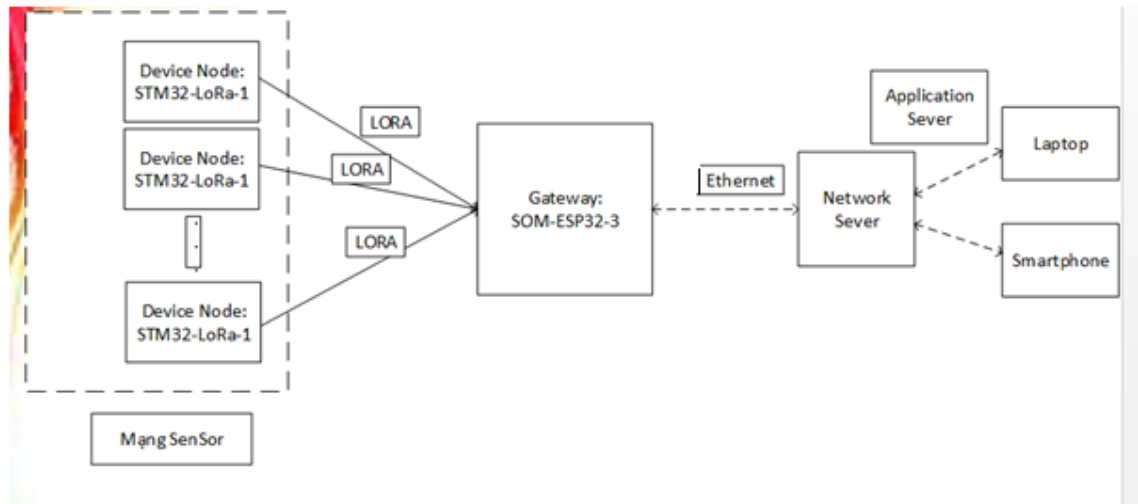
Thời gian của payload :

$$T_{\text{payload}} = n_{\text{payload}} \times T_s$$

Thời gian trong không khí bằng tổng thời gian của preamble và payload :

$$T_{\text{packet}} = T_{\text{preamble}} + T_{\text{payload}}$$

## 2.1.2 Giao thức LORAWAN :



Hệ thống LORA chứa 3 phần chính :

- Thiết bị cuối ( End device ) : các cảm biến , cơ cấu chấp hành được kết nối thông qua giao tiếp LORA tới một hoặc nhiều Gateway .
- Lora Gateway : bộ tập trung dùng làm cầu nối từ các thiết bị cuối tới Server , nó là phần tử trung tâm của kiến trúc mạng .
- Lora Netserver : server của mạng điều khiển toàn bộ hệ thống ( quản lý tài nguyên , kiểm soát truy cập , bảo mật ...)

Điểm khác biệt của các mạng Lora là việc hình dung 3 lớp của thiết bị cuối , lớp A ( cho tất cả ) , lớp B ( cho Beacon) và lớp C ( cho việc nghe liên tục ) .

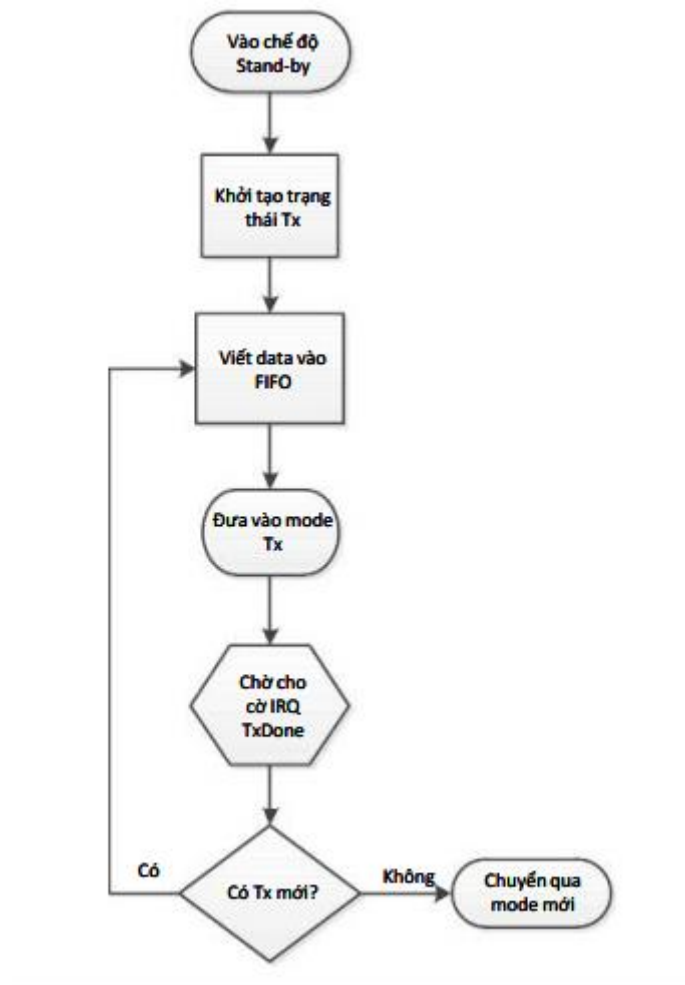
- Lớp A được định nghĩa là chế độ ban đầu của mạng Lora và phải được hỗ trợ bởi các thiết bị Lora . Trong mạng Lora , việc gửi đi luôn được bắt đầu bởi các thiết bị cuối . Sau mỗi lần gửi dữ liệu tới Gateway , thiết bị cuối sẽ mở 2 cửa sổ liên tiếp nhận , chờ lệnh bất kì hay gói dữ liệu được trả về bởi Server .

- Lớp B được giới thiệu dùng để uplink và downlink, được đồng bộ hóa với Server bằng cách gửi gói dữ liệu broadcast bởi Gateway lớp B, và có thể nhận dữ liệu hoặc gói lệnh trong những thời gian riêng biệt, cho dù . Lớp B được sử dụng cho các thiết bị cuối cần nhận lệnh từ sự điều khiển từ xa chẳng hạn như chấp hành, bật tắt hay cần cung cấp dữ liệu cho người sử dụng .
- Cuối cùng, Lớp C được định nghĩa thiết bị cuối không cần quá coi trọng về năng lượng, nó có thể duy trì cửa sổ tiếp nhận luôn mở .

## 2.2 Nguyên lý truyền và nhận dữ liệu trong mạng LoRa

Giao tiếp LoRa kết hợp của 3 loại giao tiếp số, thanh ghi tĩnh, thanh ghi trạng thái và data buffer FIFO. Tất cả được kết nối thông qua giao tiếp SPI .

### 2.2.1. Truyền dữ liệu



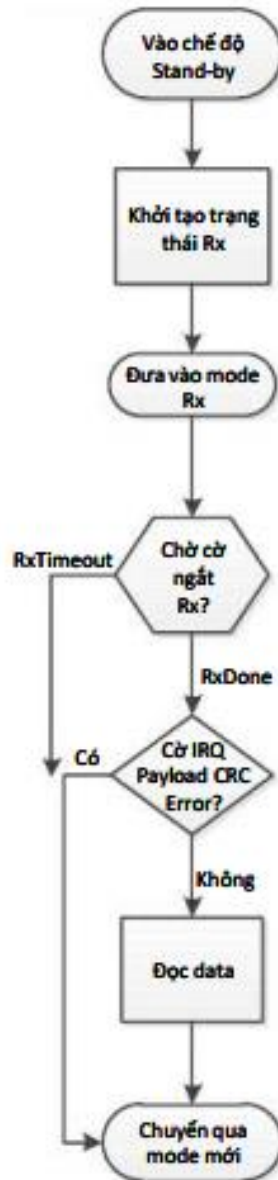
Các thanh ghi tĩnh có thể được truy cập chế độ Sleep , Standby hoặc FSTK .

FIFO Lora chỉ có thể được truyền trong chế độ Standby .

Việc vận chuyển dữ liệu được bắt đầu bằng gửi yêu cầu chế độ TX .

Chờ cho cờ ngắt TxDone ,sau đó chuyển sang chế độ Standby .

### 2.2.2. Nhận dữ liệu trong mạng LoRa



Việc nhận dữ liệu trong mạng LoRa cờ CRC đóng vai trò quan trọng trong việc xác định dữ liệu nhận được có đúng hay không, giúp khắc phục tình trạng chồng lấn hay xung đột dữ liệu giữa hai hay nhiều thiết bị cùng lúc.

### 2.2.3. Cách kết nối giữa node và Server

Cấu trúc gửi data tới Server

Cấu trúc gói packet (11 bytes) được ghi vào FIFO và gửi tới Server có cấu trúc như sau:

<b>Địa chỉ node</b> (1 byte)	<b>Nhiệt độ</b> (2 bytes)	<b>Độ ẩm</b> (2 bytes)
---------------------------------	------------------------------	---------------------------

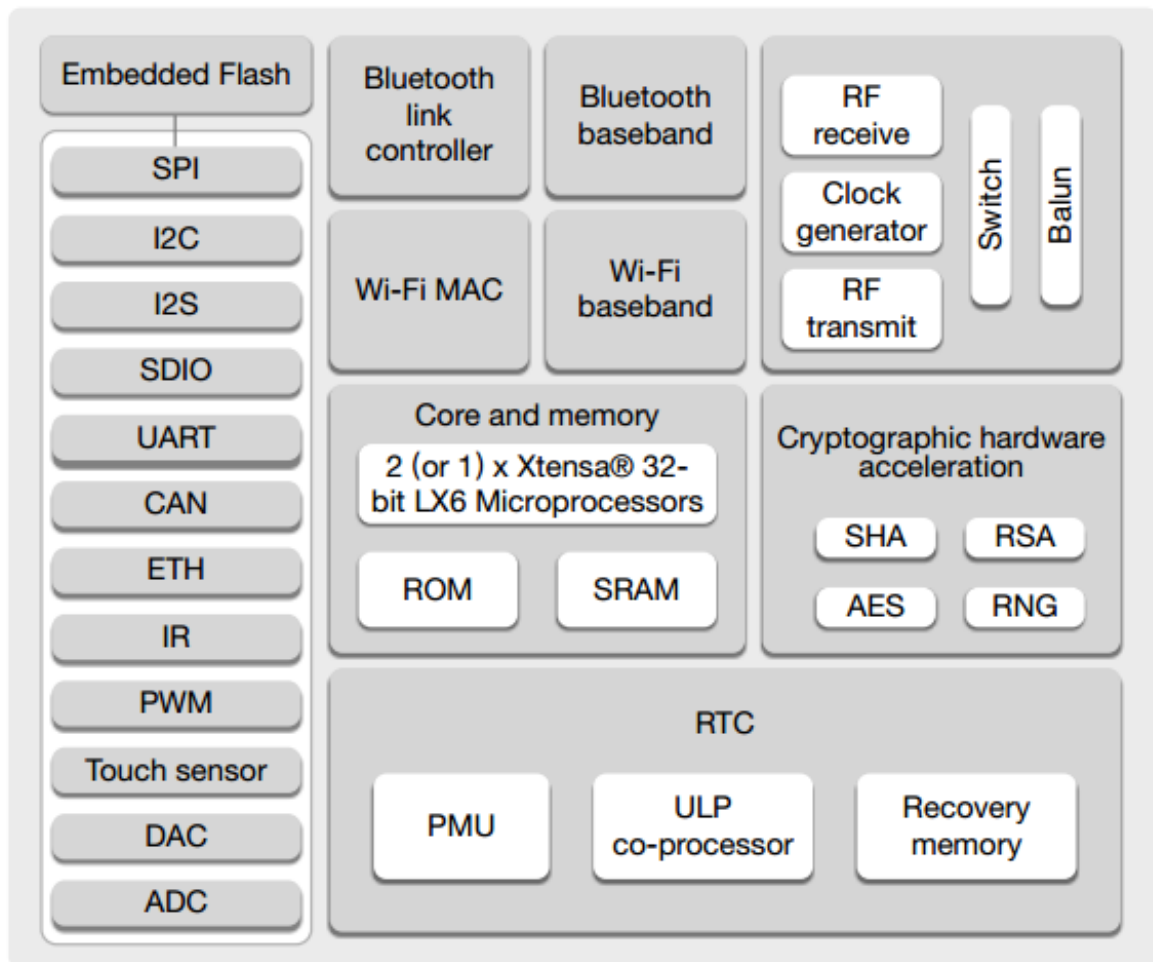
Địa chỉ node chiếm 1 byte dữ liệu

Nhiệt độ (chiếm 2 byte dữ liệu): 1 byte cho nhiệt độ nguyên, 1 byte cho nhiệt độ phân thập phân .

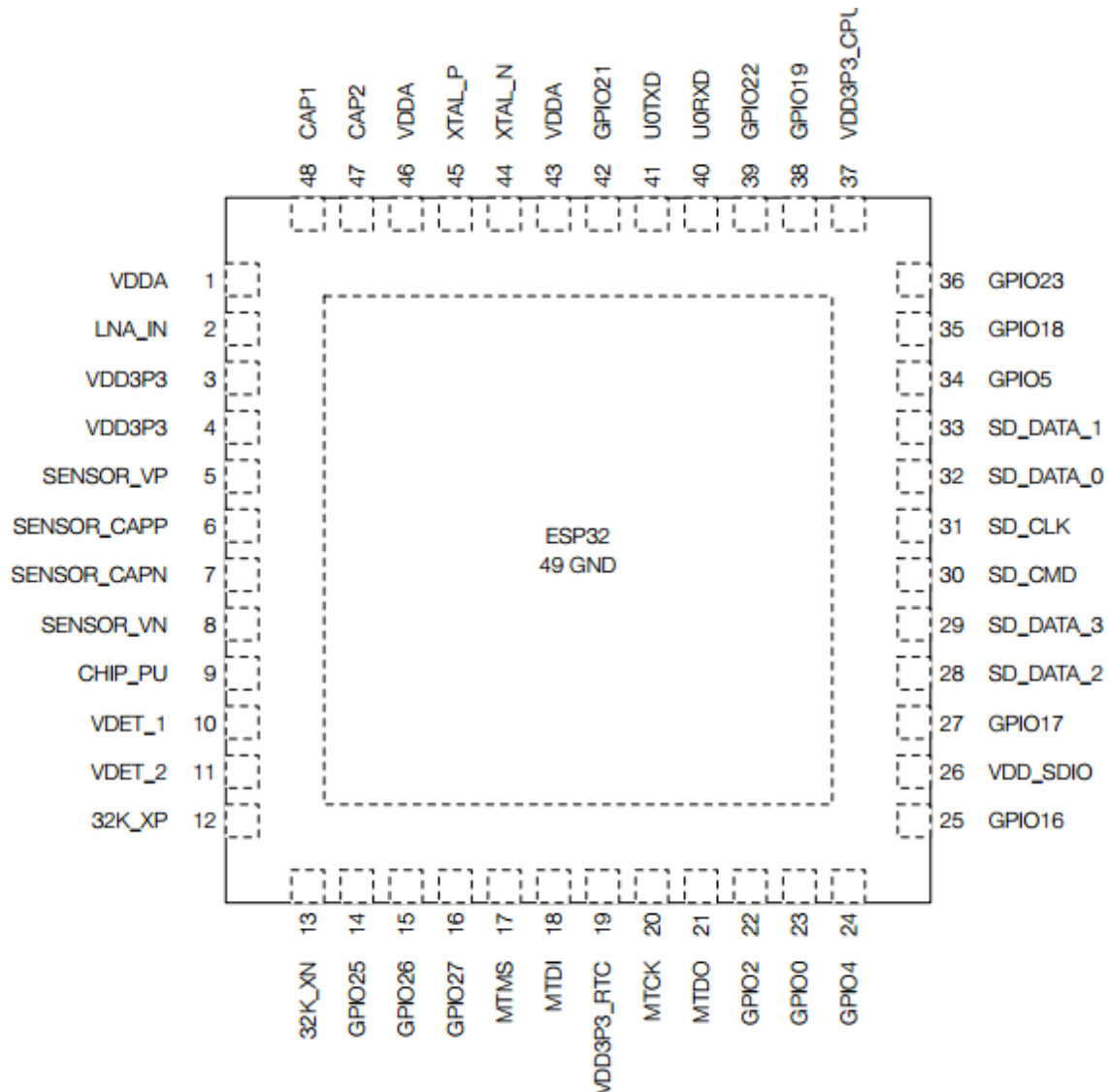
Độ ẩm (chiếm 2 byte dữ liệu): 1 byte cho độ ẩm nguyên, 1 byte cho độ ẩm phân thập phân.

## 2.3. Giới thiệu ESP32

### 2.3.1 Sơ đồ khối



### 2.3.2 Sơ đồ chân



### 2.3.3 CPU

- CPU: Xtensa Dual-Core LX6 microprocessor.
- Chạy hệ 32 bit
- Tốc độ xử lý 160MHz up to 240 MHz
- Tốc độ xung nhịp đọc flash chip 40mhz --> 80mhz (tùy chỉnh khi lập trình)
- RAM: 520 KByte SRAM
- 520 KB SRAM liên chip –(trong đó 8 KB RAM RTC tốc độ cao – 8 KB RAM RTC tốc độ thấp (dùng ở chế độ DeepSleep).



### 2.3.4 Hỗ trợ 2 giao tiếp không dây

- Wi-Fi: 802.11 b/g/n/e/i
- Bluetooth: v4.2 BR/EDR and BLE

### 2.3.5 Hỗ trợ tất cả các loại giao tiếp

- 8-bit DACs( digital to analog) 2 cổng
- Analog(ADC) 12-bit 16 cổng.
- I<sup>2</sup>C – 2 cổng
- UART – 3 cổng
- SPI – 3 cổng (1 cổng cho chip FLASH )
- I<sup>2</sup>S – 2 cổng
- SD card /SDIO/MMC host
- Slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- CAN bus 2.0
- IR (TX/RX)
- Băm xung PWM (tất cả các chân )
- Ultra low power analog pre-amplifier'

### 2.3.6 Cảm biến tích hợp trên chip esp32

- 1 cảm biến Hall (cảm biến từ trường)
- 1 cảm biến đo nhiệt độ

Cảm biến chạm (điện dung) với 10 đầu vào khác nhau

### 2.3.7 Bảo mật

- IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI
- Secure boot
- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG)

### 2.3.8 Nguồn điện hoạt động

- Nhiệt độ hoạt động -40 + 85C
- Điện áp hoạt động: 2.2-3.6V

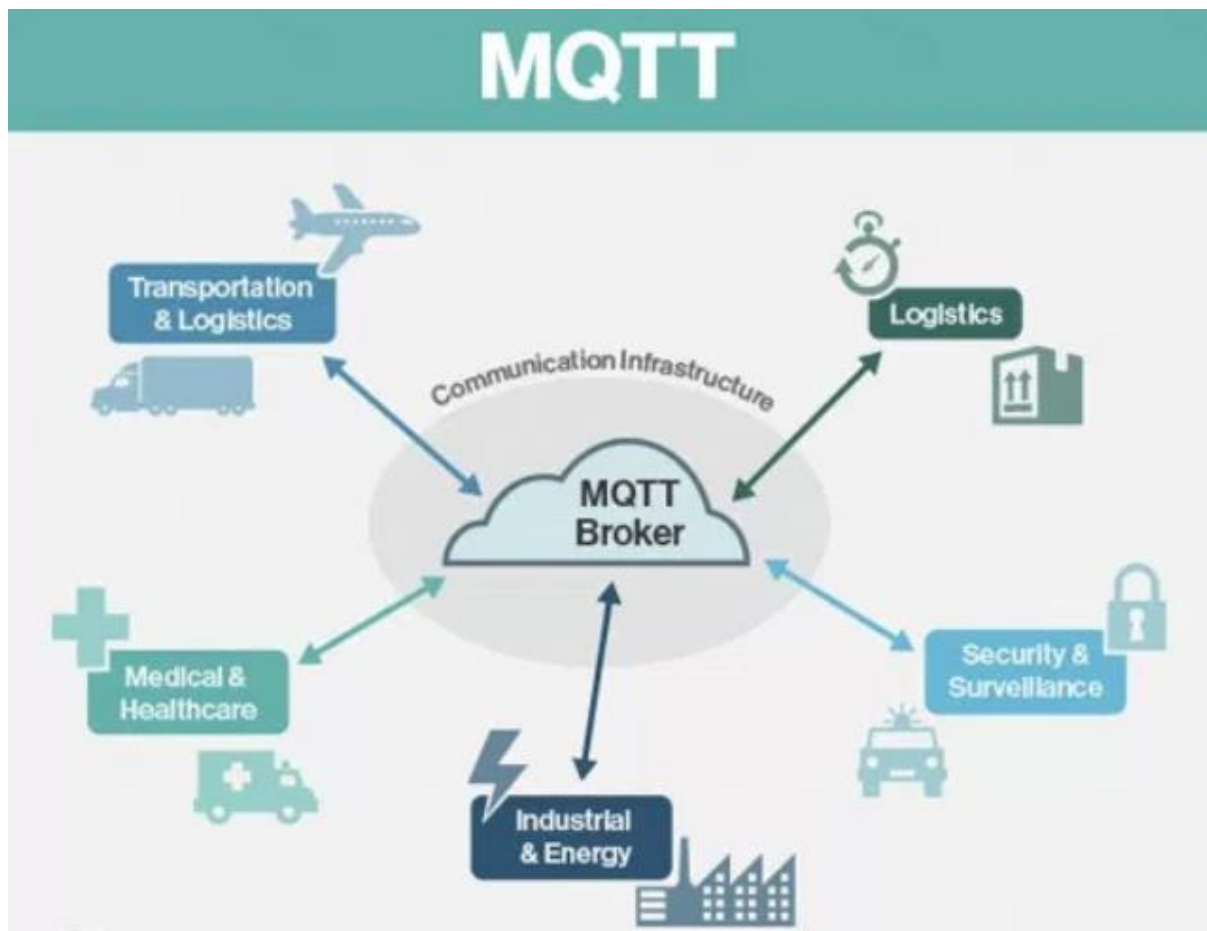
- Số cổng GPIOs : 34

### **2.3.9 Đánh giá chung**

ESP32 xứng đáng với sự mong mỏi ở các cải tiến trên esp8266.

Với esp8266, cùng với wifi , esp32 hỗ trợ thêm truyền nhận Bluetooth, RAM nhiều hơn, Tốc độ xử lý nhanh hơn, số chân GPIO nhiều hơn, nhiều cổng giao tiếp hơn, nhiều chân PWM hơn, nhiều chân ADC hơn, tích hợp cả 3 loại cảm biến (nhiệt độ, hall, touch sensor)... Tất cả ưu điểm đó cũng đủ khiến fan của ESP yêu ngay từ cái nhìn đầu tiên

## 2.4 Giao thức MQTT



- MQTT là một giao thức gửi dạng publish/subscribe sử dụng cho các thiết bị IoT với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định
- MQTT Broker nhận thông tin từ các thiết bị khác nhau, chuyển các thông tin này tới từng thiết bị (client)
- Client này có thể là điện thoại hoặc laptop
- client chỉ giao tiếp với broker qua publish – subscribe
- **Client:** là thiết bị IoT muốn gửi/nhận dữ liệu trong network

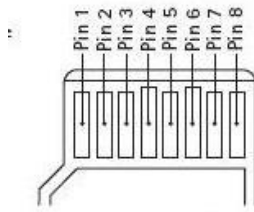
- **Broker:** là thiết bị trung gian nhận dữ liệu từ các client muốn gửi và gửi dữ liệu đó tới các client muốn nhận
- **Publish:** gửi dữ liệu từ 1 thiết bị client đến Broker.
- **Subscribe:** đăng ký nhận dữ liệu từ Broker của 1 thiết bị Client.. Broker nhận dữ liệu từ client, nó sẽ gửi dữ liệu này tới thiết bị client đã đăng ký nhận
- **Topic:** loại dữ liệu mà các thiết bị Client gửi/nhận thông qua MQTT.

Ví dụ : “example/topic1”,

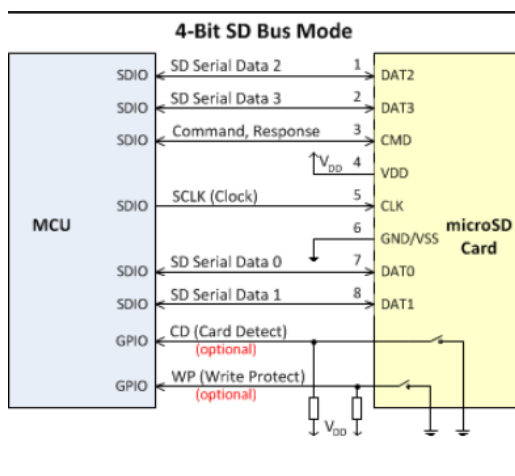
“esp/sensor/nhiệt-do”,

## 2.5 SD Card

TRANSFLASH



SD MODE (chế độ SD 4 bit (4 bit cùng một lúc))



SD MODE

Pin No.	Name	Type	Description
1	DAT2	I/O/PP	Data Line (bit 2)
2	CD/DAT3	I/O/PP	Card Detect/Data line (Bit 3)
3	CMD	PP	Command Response
4	VDD	S	Supply Voltage
5	CLK	I	Clock
6	VSS	S	Supply Voltage Ground
7	DAT0	I/O/PP	Data Line (bit 0)
8	DAT1 <sup>2*</sup>	I/O/PP	Data Line (bit 1)

Giao thức chế độ bus SD:

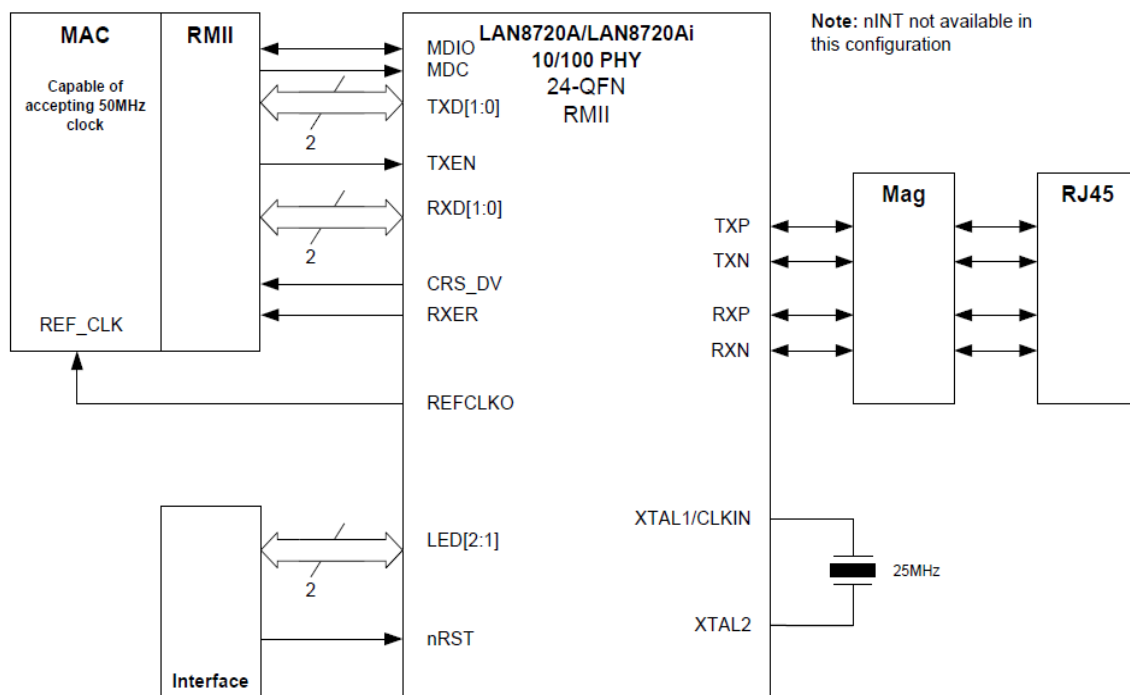
- Bus SD cho phép thiết lập cấu hình động của số đường dữ liệu từ tín hiệu dữ liệu hai chiều từ 1-4. Sau khi bật nguồn lên theo mặc định, thẻ microSD sẽ chỉ sử dụng DAT0. Sau quá trình khởi tạo, thiết bị chủ có thể thay đổi độ rộng của bus.
- Có sẵn nhiều kết nối thẻ microSD dành cho thiết bị chủ. Các kết nối tín hiệu Vdd, Vss and CLK phổ biến sẵn có trong nhiều kết nối. Tuy nhiên đường Lệnh, Phản hồi và Dữ liệu (DAT0-DAT3) sẽ được phân chia cho mỗi thẻ từ thiết bị chủ.
- Tính năng này cho phép cân bằng dễ dàng giữa chi phí phần cứng và hiệu năng hệ thống. Liên lạc qua bus microSD dựa trên lệnh và dòng bit dữ liệu được một bit khởi động tạo và bit dừng kết thúc.

- **Lệnh:** Lệnh được truyền tuần tự trên đường CMD. Lệnh là một mã để bắt đầu một thao tác từ thiết bị chủ đến thẻ. Lệnh được gửi đến một thẻ đơn có địa chỉ (lệnh có địa chỉ) hoặc đến tất cả thẻ được kết nối (lệnh phát sóng).
- **Phản hồi:** Phản hồi được truyền tuần tự trên đường CMD. Phản hồi là một mã để trả lời một lệnh đã nhận được trước đó. Phản hồi được gửi từ một thẻ đơn có địa chỉ hoặc từ tất cả các thẻ được kết nối.

### KẾT NỐI CHÂN GIỮA ESP32 VỚI SD CARD

PIN	SD MODE			ESP32-WROOM32	
	Name	I/O <sup>1</sup>	Mô tả	Pin	Name
1	DAT2	I/O/PP	Data line[bit2]	17	SD2
2	CD/DAT3	I/O/PP	Card Detect/Data line[bit3]	18	SD3
3	CMD	PP	Command/Response	19	CMD
4	V <sub>dd</sub>	S	Nguồn cấp	3.3V	
5	CLK	I	Clock	20	CLK
6	V <sub>ss</sub>	S	GND	GND	
7	DAT0	I/O/PP	Data line[bit0]	21	SD0
8	DAT1	I/O/PP	Data line[bit1]	22	SD1

## 2.6 Chip LAN 8270A



### MAC Interface

Giao diện RMII có các đặc điểm sau:

- Nó có khả năng hỗ trợ tốc độ dữ liệu 10Mbps và 100Mbps
- Một tham chiếu đồng hồ đơn được sử dụng cho cả truyền và nhận
- Nó cung cấp đường dẫn truyền và nhận dữ liệu rộng 2 bit (di-bit) độc lập
- Nó sử dụng các mức tín hiệu LVCMOS, tương thích với các quy trình ASIC CMOS kỹ thuật số thông thường

RMII bao gồm các tín hiệu giao diện sau (1 tùy chọn):

- truyền dữ liệu - TXD [1: 0]
- truyền phát sáng - TXEN

- nhận dữ liệu - RXD [1: 0]
- nhận lỗi - RXER (Tùy chọn)
- Sense cảm giác sóng mang - CRS\_DV
- Tham khảo Đồng hồ - (Tham chiếu RMII thường xác định tín hiệu này là REF\_CLK)

#### Cấu hình địa chỉ vật lý (PHY\_ADDR)

- Địa chỉ SMI của thiết bị có thể được định cấu hình bằng cách sử dụng cấu hình phần cứng cho giá trị 0 hoặc 1. **Người dùng có thể định cấu hình địa chỉ PHY bằng Cấu hình phần mềm nếu cần có địa chỉ lớn hơn 1.** Địa chỉ PHY có thể được viết (sau khi truyền thông SMI tại một số địa chỉ được thiết lập) bằng cách sử dụng các bit PHYAD của thanh ghi chế độ đặc biệt. Dây đeo cấu hình phần cứng PHYAD0 được ghép với chân RXER



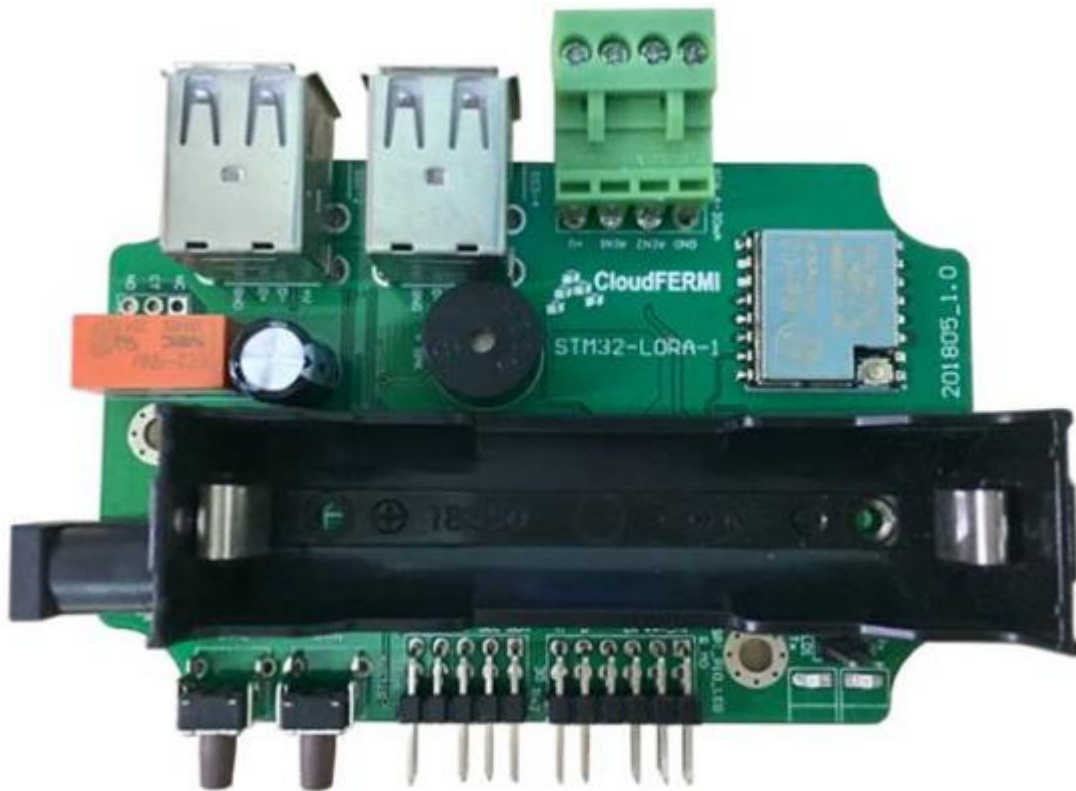
PIN	LAN8720A				ESP32-WROOM32	
	Name	SYMBOL	BUFF TYPE	DESCRIPTION	PIN	NAME
17	Transmit Data 0	TXD0	VIS	MAC truyền dữ liệu đến bộ thu phát bằng tín hiệu này	31	EMAC_TXD0
18	Transmit Data 1	TXD1	VIS	MAC truyền dữ liệu đến bộ thu phát bằng tín hiệu này	36	EMAC_TXD1
16	Transmit Enable	TXEN	VIS(PD)	Cho biết dữ liệu truyền hợp lệ có trên TXD [1: 0].	33	EMAC_TX_EN
8	Receive Data 0  PHY Operating Mode 0 Configuration Strap	RXD0  MODE0	VO8  VIS (PU)	Bit0 được gửi bởi bộ thu phát trên đường dẫn nhận.  Kết hợp với MODE1 và MODE2, được cấu hình mặc định ở PHY	10	EMAC_RXD0
7	Receive Data 1  PHY Operating Mode 1 Configuration Strap	RXD1  MODE1	VO8  VIS (PU)	Bit1 được gửi bởi bộ thu phát trên đường dẫn nhận.  Kết hợp với MODE0 và MODE2, được cấu hình mặc định ở PHY	11	EMAC_RXD1
10	Receive error  PHY Address 0 Configuration Strap	RXER	VO8  VIS (PD)	Tín hiệu này được xác nhận để chỉ ra rằng một lỗi đã được phát hiện ở đầu đó trong khung hiện đang được chuyển từ bộ thu phát.  cấu hình này đặt địa chỉ SMI của trình thu phát.	16	EMAC_RX_ER
11	Carrier Sense / Receive Data Valid  PHY Operating Mode 2 Configuration Strap	CRS_DV  MODEL 2	VO8  VIS (PD)	Tín hiệu này được xác nhận để cho biết môi trường nhận là không nhân rồi.  Kết hợp với MODE0 và MODE1, được cấu hình mặc định ở PHY	12	EMAC_RX_DV
	SMI Data Input/Output	MDIO	VIS/VOD8	SMI data I/O	30	GPIO 18
	SMI Clock	MDC	VIS	SMI CLOCK	37	GPIO 23
	Interrupt Output  Reference Clock Output	nINT  REFCLKO	VOD8 (PU)   VO8	Hoạt động đầu ra ngắt thấp. Đặt một điện trở bên ngoài kéo lên VDDIO.  Đầu ra đồng hồ 50MHz tùy chọn này có nguồn gốc từ bộ dao động tinh thể 25MHz. REFCLKO có thể lựa chọn thông qua cấu hình nINTSEL.	28	GPIO17  EMAC_CLK_OUT_180 (dùng 4k7 pullup)
				4k7 Pulldown	32	PHY_POWER (NC)

Nếu GPIO17 được kéo LOW trong trình tự khởi động thì bộ nạp khởi động đang chờ chương trình nối tiếp. Nhưng GPIO17 cũng là đầu vào đồng hồ cho khối chức năng EMAC trong ESP32.

Vì vậy, nó phải được đảm bảo rằng GPIO17 được tổ chức **HIGH** trong khi khởi động và 50MHz REFCLK được cung cấp ngay trước khi LAN8720 được khởi tạo. May mắn thay bộ dao động 50MHz có một chốt cho phép. Khi pin này được giữ LOW, đầu ra bị vô hiệu hóa

## 2.7 Giới thiệu module, IC

### 2.7.1 Module Sensor-Node-Lora :



- Board Sensor-Node-Lora sử dụng vi điều khiển STM32F0 , Chip Lora SX1278 433MHz được phát triển bởi công ty CloudFERMI .
  - + Sử dụng dao động nội với tần số hoạt động 48MHz .
  - + Tích hợp Lora SX1278 với khoảng cách truyền xa đến 10km.
  - + Được sử dụng làm node để thu thập dữ liệu và truyền về Gateway thông qua mạng Lora .

### 2.7.2. Module SOM-ESP32-3 :



- Board SOM-ESP32-3 với ESP32 , WIFI , Bluetooth , LAN , MicroSD , LORA SX1278 433MHz .
- Ứng dụng :
  - + Lưu trữ dữ liệu cảm biến .
  - + Thu thập phân tích số liệu .
  - + Dùng làm gateway đưa dữ liệu lên internet , cloud .
  - + Điều khiển từ xa , thông minh .

### 2.7.3. Cảm biến nhiệt độ và độ ẩm ( DHT21/AM2301) :

#### 2.7.3.1. Tổng quát :

Cảm biến độ ẩm, nhiệt độ DHT21 AM2301 tích hợp cảm biến độ ẩm điện dung và cảm biến nhiệt độ có độ chính xác cao, đầu ra tín hiệu số có thể kết nối với một Vi điều khiển 8-bit, Sản phẩm chất lượng cao, đáp ứng nhanh, khả năng chống nhiễu mạnh, giao tiếp duy nhất 1 dây.

Kích thước nhỏ, tiêu thụ điện năng thấp, khoảng cách truyền dẫn tín hiệu lên đến 20m.

Điện năng tiêu thụ cực thấp, khoảng cách truyền dẫn, hiệu chuẩn hoàn toàn tự động, sử dụng các cảm biến độ ẩm điện dung, hoàn toàn hoán đổi cho nhau, tiêu chuẩn kỹ thuật số đầu ra duy nhất- một bus, ổn định lâu dài tuyệt vời, thiết bị đo nhiệt độ chính xác cao.

#### 2.7.3.2. Thông số kỹ thuật :

Áp nguồn: 3.3 - 5V

Dòng tiêu thụ: 300 uA

Kích thước: 58.8 x 26.7 x 13.8 (mm)

Model: AM2301

Độ phân giải chính xác: 0.1

Khoảng đo: 0100% RH

Khoảng đo nhiệt độ: -40 °C ~ 80 °C

Đo lường chính xác độ ẩm:  $\pm 3\%$  RH

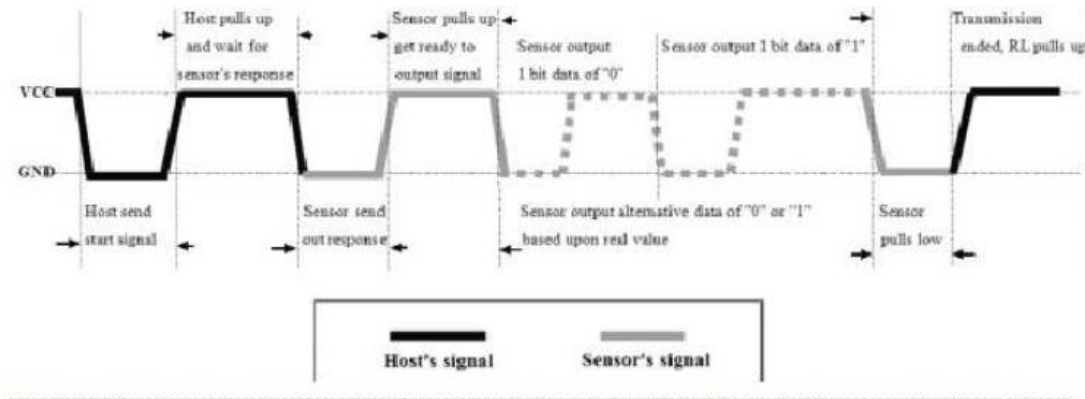
Đo lường chính xác nhiệt độ:  $\pm 0.5$  °C

#### 2.7.3.3. Kích thước và sơ đồ chân :

Cách hoạt động :

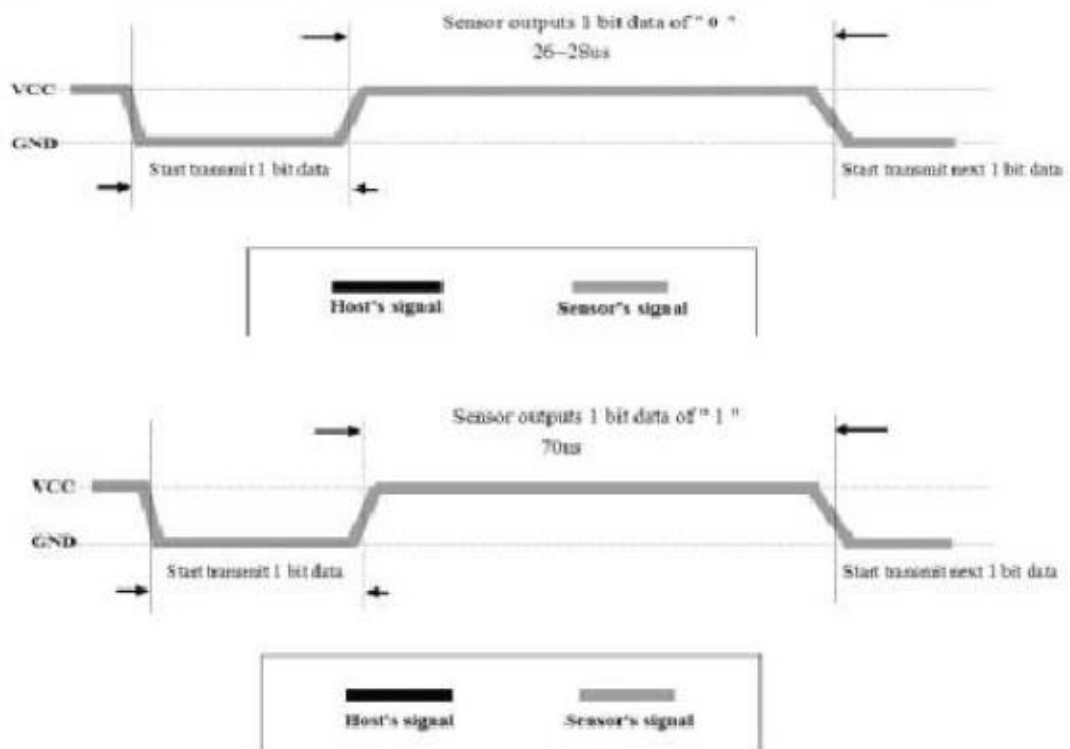
- Dữ liệu nhận được gồm 40 bit , 16 bit nhiệt độ , 16 bit độ ẩm và 8 bit checksum
- Khi MCU gửi dữ liệu bắt đầu đọc tín hiệu . AM2301 từ trạng thái Standby sang trạng thái running . Khi MCU kết thúc gửi tín hiệu bắt đầu , AM2301 sẽ gửi tín hiệu đáp ứng gồm 40bit để phản ánh nhiệt độ và độ ẩm tới MCU. Nếu không có tín hiệu từ MCU , cảm biến sẽ không gửi tín hiệu đáp ứng tới MCU . Một tín hiệu bắt đầu sẽ cho một dữ liệu đáp ứng từ cảm biến . AM2301 sẽ không thay đổi trạng thái Standby nếu nó không nhận được tín hiệu bắt đầu từ MCU lần nữa .

- Dưới đây là toàn bộ quá trình xử lý , toàn bộ quá trình mất khoảng 2s :



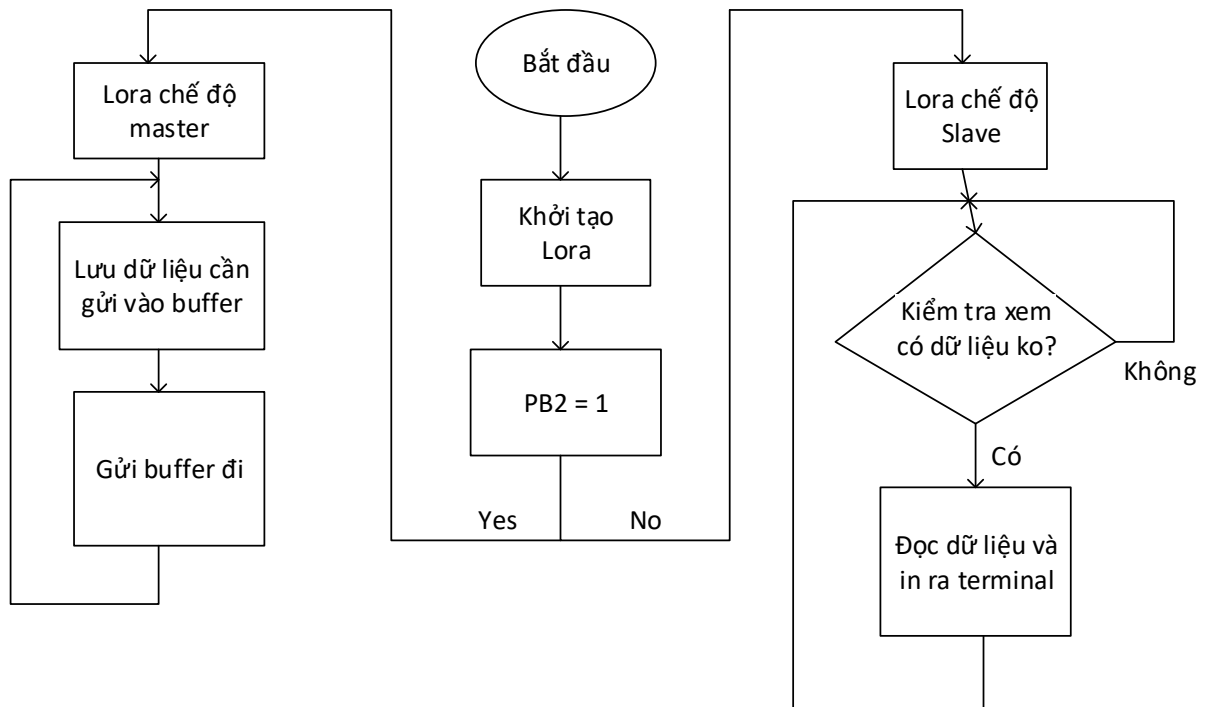
- Khi muốn giao tiếp giữa MCU và AM2301, MCU sẽ kéo đường data xuống thấp và quá trình xử lý này mất ít nhất từ 1-10ms để đảm bảo AM2301 có thể phát hiện tín hiệu từ MCU và chờ khoảng 20-30us để AM2301 đáp ứng.
- Khi AM2301 phát hiện tín hiệu bắt đầu, nó sẽ kéo đường bus xuống thấp khoảng 80us để coi đó là tín hiệu đáp ứng, sau đó sẽ kéo lên cao 80us để chuẩn bị gửi dữ liệu .
- Khi AM2301 gửi dữ liệu tới MCU, mỗi bit được gửi sẽ được bắt đầu với mức thấp khoảng 50us, sau đó sẽ là mức cao, độ dài của mức cao sẽ quyết định bit đó là 0 hay 1.

Dữ liệu của AM2301 được mô tả như hình :



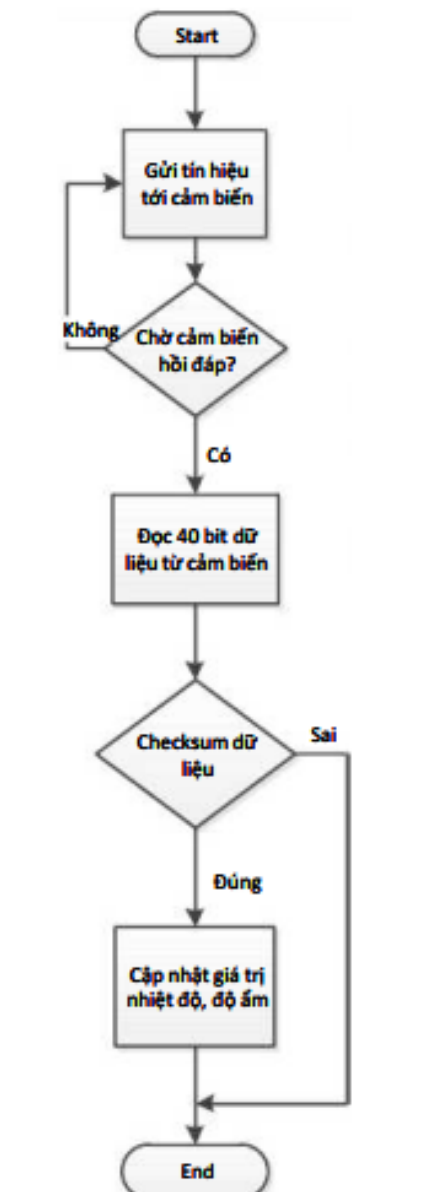
## 2.8 Phần mềm :

### 2.8.1 Giải thuật các Node





Quá trình đọc dữ liệu từ cảm biến :



## 2.8.2 MQTT Broker

+ MQTT broker: tự tạo trên máy tính, server,... phức tạp

+ MQTT broker có sẵn: sử dụng cái sẵn có như dịch vụ MQTT Broker

là [CloudMQTT](https://www.cloudmqtt.com/) (<https://www.cloudmqtt.com/>)

The screenshot shows the 'Create New Instance' form on the CloudMQTT website. It includes fields for 'Name' (placeholder: 'Name to describe your instance'), 'Plan' (dropdown menu showing 'Cute Cat (Free)'), and 'Data center' (dropdown menu showing 'US-East-1 (Northern Virginia)'). Below these is the 'amazon web services' logo and a 'Tags' field with a placeholder 'Type or click here'. A note states 'Admins can manage tag access control.' At the bottom is a 'Create New Instance' button. To the right, under the 'Plan' section, is a 'Cute Cat' emoji and the text 'Cute Cat'. A link 'See the plan page to learn about the different plans.' is also present.

Thông tin MQTT broker sau khi tạo

The screenshot shows the 'Instance info' page for a newly created MQTT broker. It displays the following information:

- Server:** m12.cloudmqtt.com
- User:** cjtdarvp (with a 'Restart' button)
- Password:** wam6k0pGKOYD (with a 'Rotate' button)
- Port:** 18011
- SSL Port:** 28011
- Websockets Port (TLS only):** 38011
- Connection limit:** 5

On the right, under 'Active Plan', is a 'Cute Cat' emoji and the text 'Cute Cat'. Below this is an 'Upgrade Instance' button. At the bottom, there is a 'Reset DB' section with the text 'This will erase all stored messages and sessions. The instance will be restarted.' and a 'Reset DB' button. A red circular icon with a white 'X' is also visible in the bottom right corner.

## Giao diện hiển thị thông tin lên MQTT

## Websocket

### Send message

Topic

Message

Send

### Received messages

Topic	Message
-------	---------

Connected!

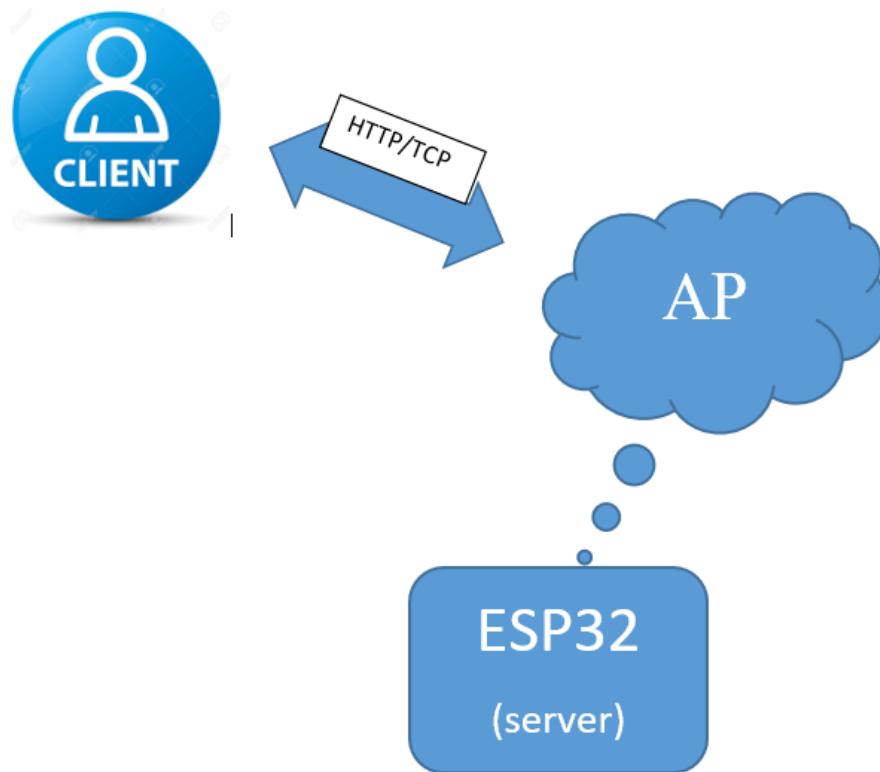
Connecting to m12.cloudmqtt.com...

### Clear session

Clear any data for a client id

Client ID

### 2.8.3 Arduino ESP32 into an Access Point



ESP32 tạo ra một Access Point nó trở thành một điểm phát WiFi để các Client có thể kết nối với nó, bằng cách này ta có thể gửi data đến Server (ESP32). Ở đây ta dùng để gửi thông tin user, pass của router để ESP32 có thể kết nối được với NetWork

Để thiết lập **chế độ WiFi cho ESP32**, chúng tôi sẽ sử dụng chức năng:

- **WiFi.mode (chế độ)** : chế độ có thể là: **WIFI\_OFF** (tắt WiFi), **WIFI\_STA** (Chế độ trạm), **WIFI\_AP** (Chế độ điểm truy cập), **WIFI\_AP\_STA** ( cả hai trạm và chế độ điểm truy cập)
- **WiFi.begin (ssid, mật khẩu)** : sử dụng chức năng này để làm cho ESP32 một khách hàng WiFi kết nối với một mạng có ssid và mật khẩu. Nếu chúng ta sử dụng chức năng này, chúng ta không cần sử dụng WiFi.mode (WIFI\_STA).

- **WiFi.softAP (ssidAP, passwordAP)**: sử dụng chức năng này để tạo ESP32 Access Point có thông tin xác thực là ssidAP và passwordAP. Nếu chúng ta sử dụng chức năng này, chúng ta không cần sử dụng WiFi.mode (WIFI\_AP)

Hãy tạo ESP32 Access Point với thông tin xác thực bên dưới:

- **ssid** là "**AP**"

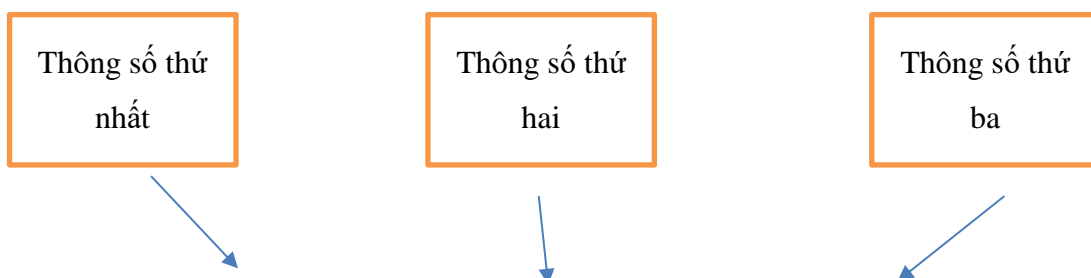
- **mật khẩu** là "**12345678**"

**Lưu ý:** IP mặc định thêm res của ESP32 trong chế độ AP là "**192.168.4.1**"

### 2.8.4 Arduino ESP32 với HTML

Định nghĩa một tuyến đường mà nó sẽ lắng nghe các yêu cầu HTTP đến. Tuyến đường này sẽ được kết hợp với một chức năng xử lý sẽ được thực hiện bất cứ khi nào một yêu cầu đến tuyến đường đó được thực hiện Client

Vì vậy, để ràng buộc một con đường để một chức năng xử lý, chúng tôi chỉ cần gọi **vào** phương pháp trên đối tượng máy chủ



**server.on("/html", HTTP\_GET, handleRoot);**

a) Thông số thứ nhất: nhận được một chuỗi với tuyến đường mà máy chủ nên lắng nghe. Chúng ta sẽ sử dụng tuyến đường **"/ html"** .

b) Thông số thứ hai, nó nhận được một enum kiểu **WebRequestMethod**, cho biết các phương thức HTTP được cho phép trên tuyến đường đó. Chúng tôi sẽ giới hạn các yêu cầu GET, vì vậy chúng tôi nên sử dụng giá trị **HTTP\_GET** .

**Giao thức HTTP định nghĩa một số phương thức (method) truyền đến Server:**

- GET là phương thức yêu cầu dữ liệu đơn giản và thường sử dụng nhất của HTTP. Phương thức **GET** yêu cầu server chỉ trả về dữ liệu bằng việc cung cấp các thông tin truy vấn trên URL, thông thường Server căn cứ vào thông tin truy vấn đó trả về dữ liệu mà không thay đổi nó. path và query trong **URL** chứa thông tin truy vấn.
- POST tương tự như GET, nhưng POST có thể gửi dữ liệu về Server.
- PUT là phương thức yêu cầu tạo mới một dữ liệu, giống POST nhưng đánh dấu cho Server biết, nếu dữ liệu không tồn tại trong cơ sở dữ liệu thì tạo mới, hoặc sửa đổi nó.

- DELETE Tương tự như GET, nhưng báo cho Server biết về việc xóa dữ liệu thông qua URL.

Các phương thức thông thường chỉ dùng GET và POST, các phương thức còn lại thường sử dụng trong API server (RESTful). Một số điểm khác biệt giữa POST và GET.

- GET có thể bị cache (lưu trữ ở trình duyệt và sử dụng lại sau đó), nội dung request có thể lưu trữ ở lịch sử trình duyệt, có thể được đánh dấu (bookmark).
- GET không nên sử dụng để gửi các dữ liệu nhạy cảm.
- GET bị giới hạn độ lớn dữ liệu cần gửi.
- GET chỉ nên dùng để lấy dữ liệu về.
- POST không bị catch, không tồn tại dữ liệu gửi trong lịch sử trình duyệt, không thể đánh dấu (bookmark).
- POST không giới hạn bởi độ lớn dữ liệu cần gửi.

c) Thông số thứ 3: phương thức **on** nhận hàm xử lý sẽ được thực thi khi các yêu cầu được nhận.

```
void handleRoot() {

    server.send (200, "text/html", "<form action=\"/login\"
method=\"POST\"><label class=\"label\">Network Name</label><input
type=\"text\" name=\"username\"
placeholder=\"Username\"></br><label>Password</label><input
type=\"text\" name=\"password\" placeholder=\"Password\"></br><input
type=\"submit\" value=\"Login\"></form>");

}
```

+ HTML Code: Trả về mã HTTP OK, tương ứng với **200** .

+ Vì chúng ta sẽ trả về HTML, chúng ta định nghĩa kiểu nội dung là " **text / html** ".

+ Cuối cùng, chúng tôi chỉ định nội dung HTML của chúng tôi dưới dạng chuỗi.

Chúng tôi sẽ cần phải phục vụ mã HTML của chúng tôi từ ESP32, chúng tôi sẽ lập trình bằng C ++ / Arduino. Vì vậy, chúng tôi sẽ cần phải chuyển đổi nó thành một đại diện mà vi điều khiển của chúng tôi hiểu và có thể làm việc.

trước tiên chúng ta sẽ nén mã vào một dòng đơn, loại bỏ tất cả các khoảng trống và các dòng không cần thiết (vì vậy nó chiếm ít không gian hơn trong ESP32), tiếp theo là thoát nó vào một chuỗi C.

[https://www.w3schools.com/html/tryit.asp?filename=tryhtml\\_default](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_default)

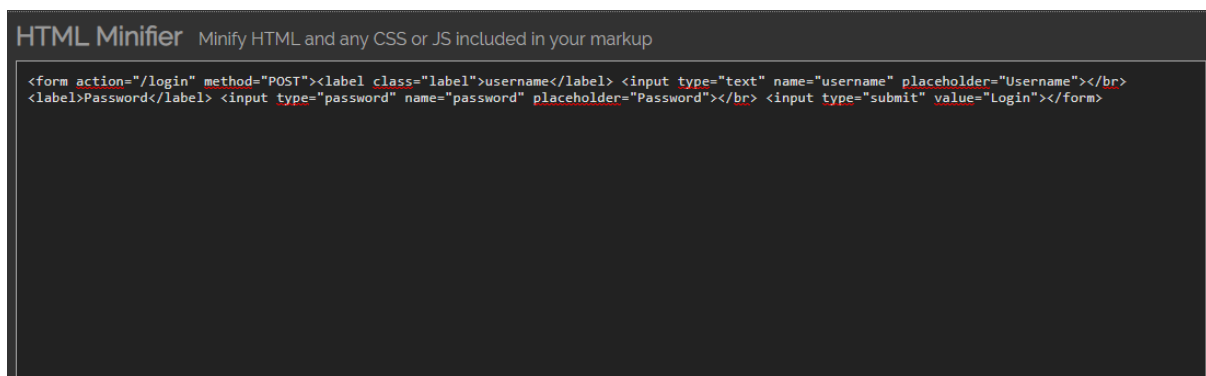


The screenshot shows the W3Schools HTML Tryit editor. On the left, there is a code editor with the following HTML code:

```
<form action="/login" method="POST">
  <label class="label">username</label>
  <input type="text" name="username" placeholder="Username"></br>
  <label>Password</label>
  <input type="password" name="password" placeholder="Password"></br>
  <input type="submit" value="Login">
</form>
<p>
```

On the right, there is a preview of the rendered form. It features two input fields: one for 'username' with a placeholder 'Username' and one for 'Password' with a placeholder 'Password'. Below the password field is a 'Login' button. The top right corner of the preview area indicates 'Result Size: 668 x 492'.

<https://www.willpeavy.com/minifier/>



The screenshot shows the HTML Minifier tool. The title is 'HTML Minifier' with a subtitle 'Minify HTML and any CSS or JS included in your markup'. The main area displays the minified HTML code for the login form:

```
<form action="/login" method="POST"><label class="label">username</label> <input type="text" name="username" placeholder="Username"></br>
<label>Password</label> <input type="password" name="password" placeholder="Password"></br> <input type="submit" value="Login"></form>
```



[http://tomeko.net/online\\_tools/cpp\\_text\\_escape.php?lang=en](http://tomeko.net/online_tools/cpp_text_escape.php?lang=en)

## Text -> C/C++ string converter

Converting text into C-like literal, escaping newlines, tab, double quotes, backslash.

Source text:

```
<form action="/login" method="POST"><label class="label">username</label>
<input type="text" name="username" placeholder="Username"></br>
<label>Password</label> <input type="password" name="password"
placeholder="Password"></br> <input type="submit" value="Login"></form>
```

Options:

☒ split output into multiple lines

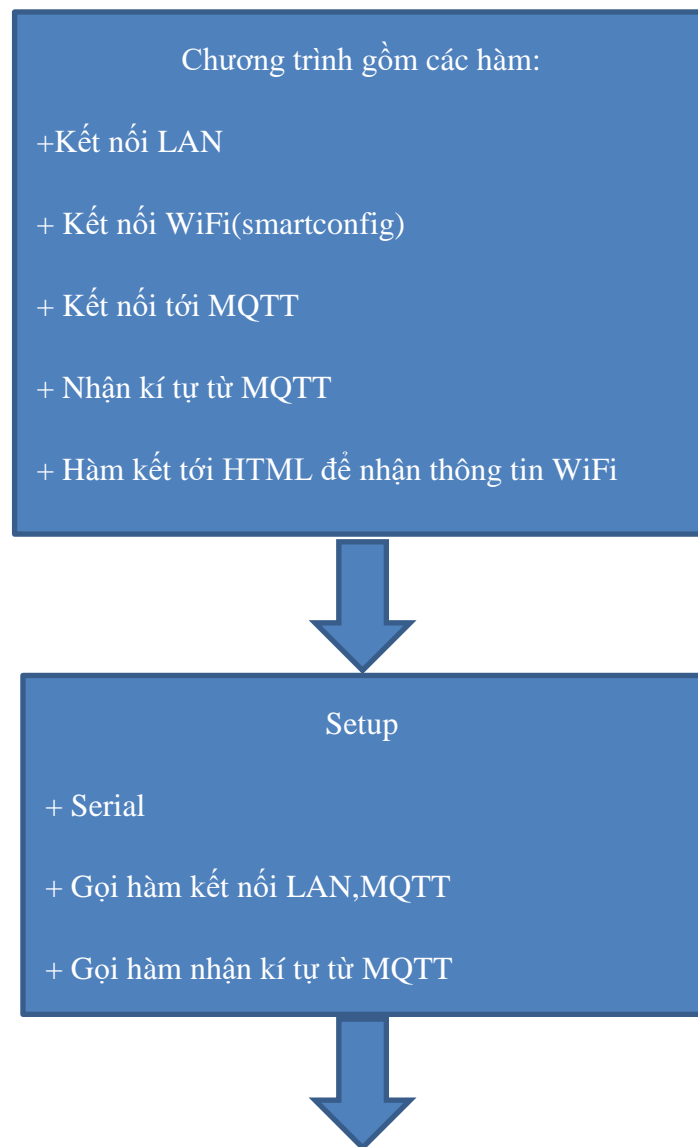
Convert

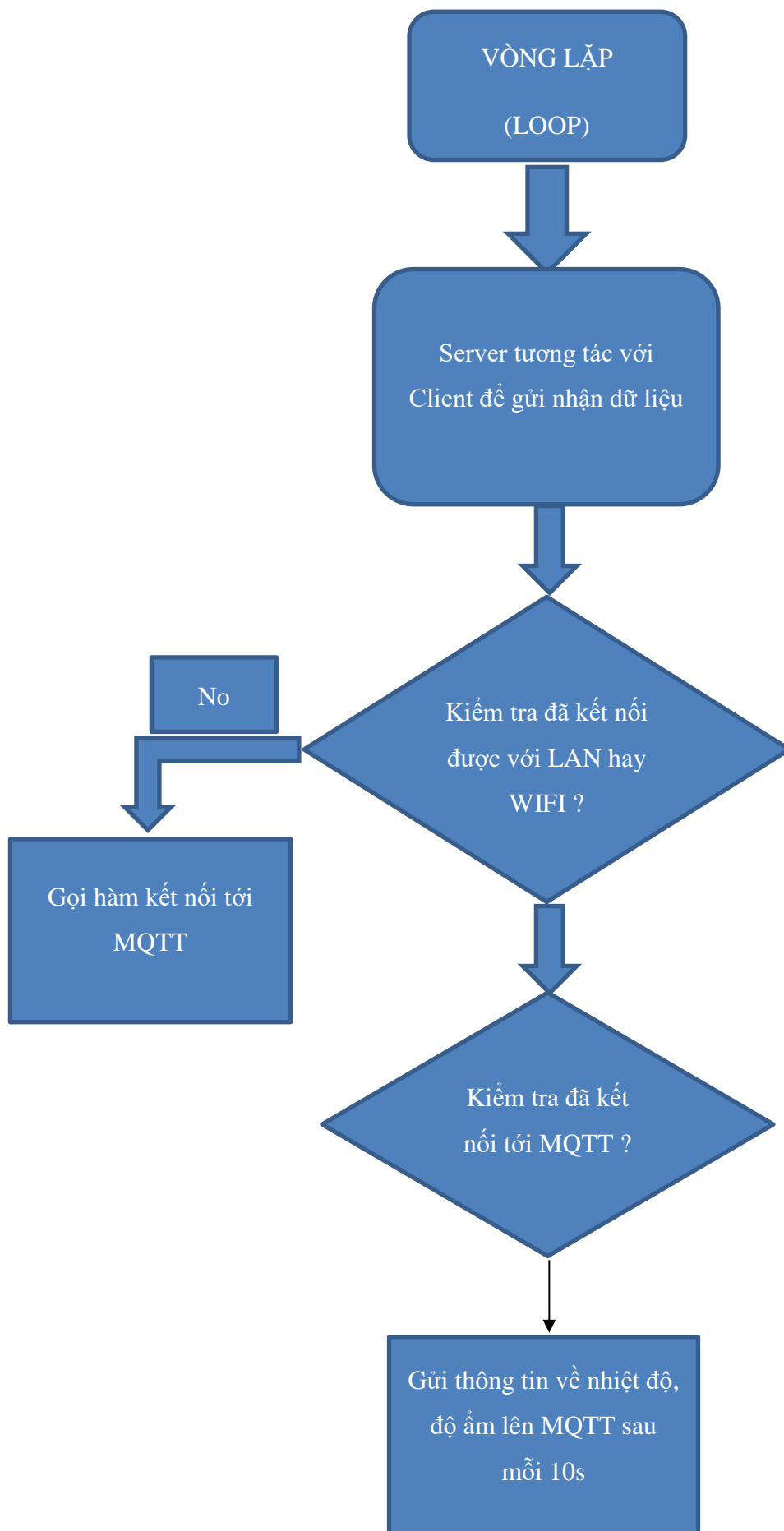
C/C++ string

```
"<form action=\"/login\" method=\"POST\"><label
class=\"label\">username</label> <input type=\"text\" name=\"username\"
placeholder=\"Username\"></br> <label>Password</label> <input
type=\"password\" name=\"password\" placeholder=\"Password\"></br> <input
type=\"submit\" value=\"Login\"></form>"
```

## 2.9 Lưu đồ giải thuật:

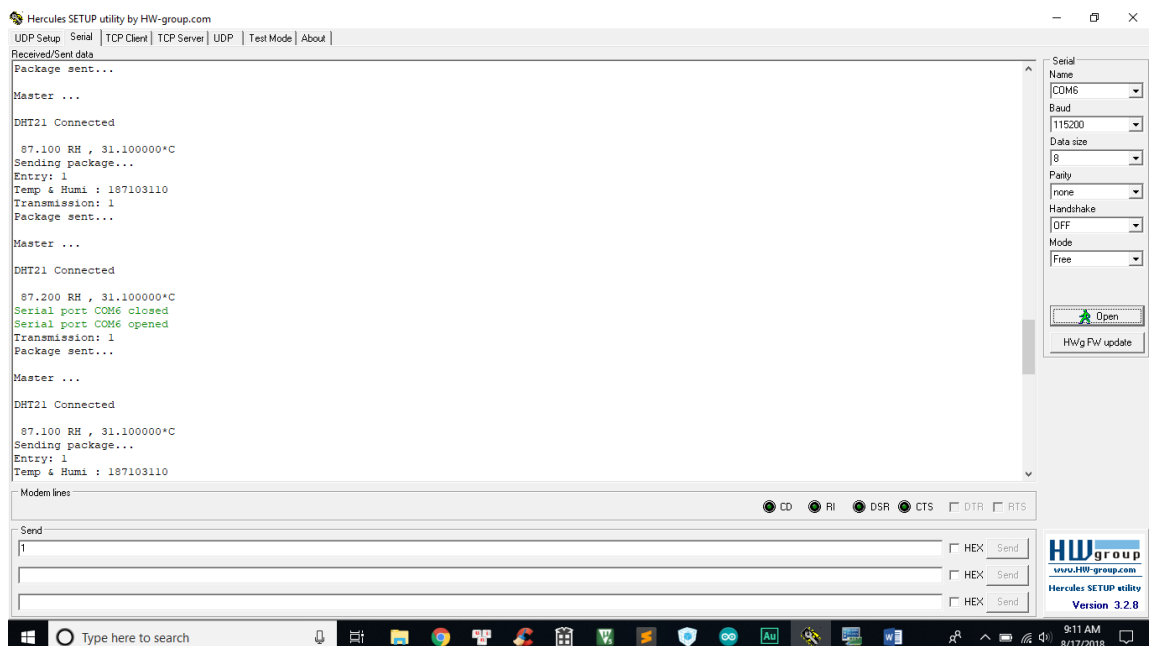
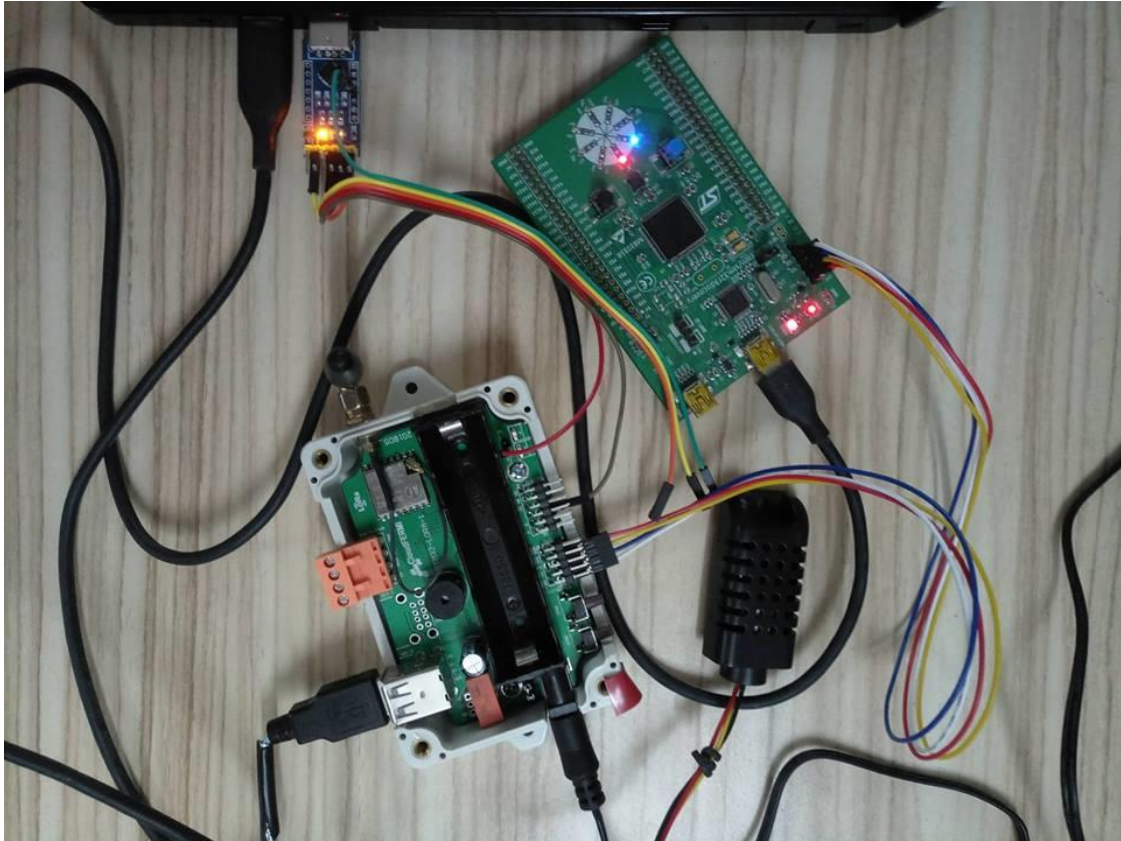
có thể kết nối Lan, kết nối wifi khi đến địa điểm khác, gửi dữ liệu nhiệt độ, độ ẩm lên MQTT( Website hoặc App)



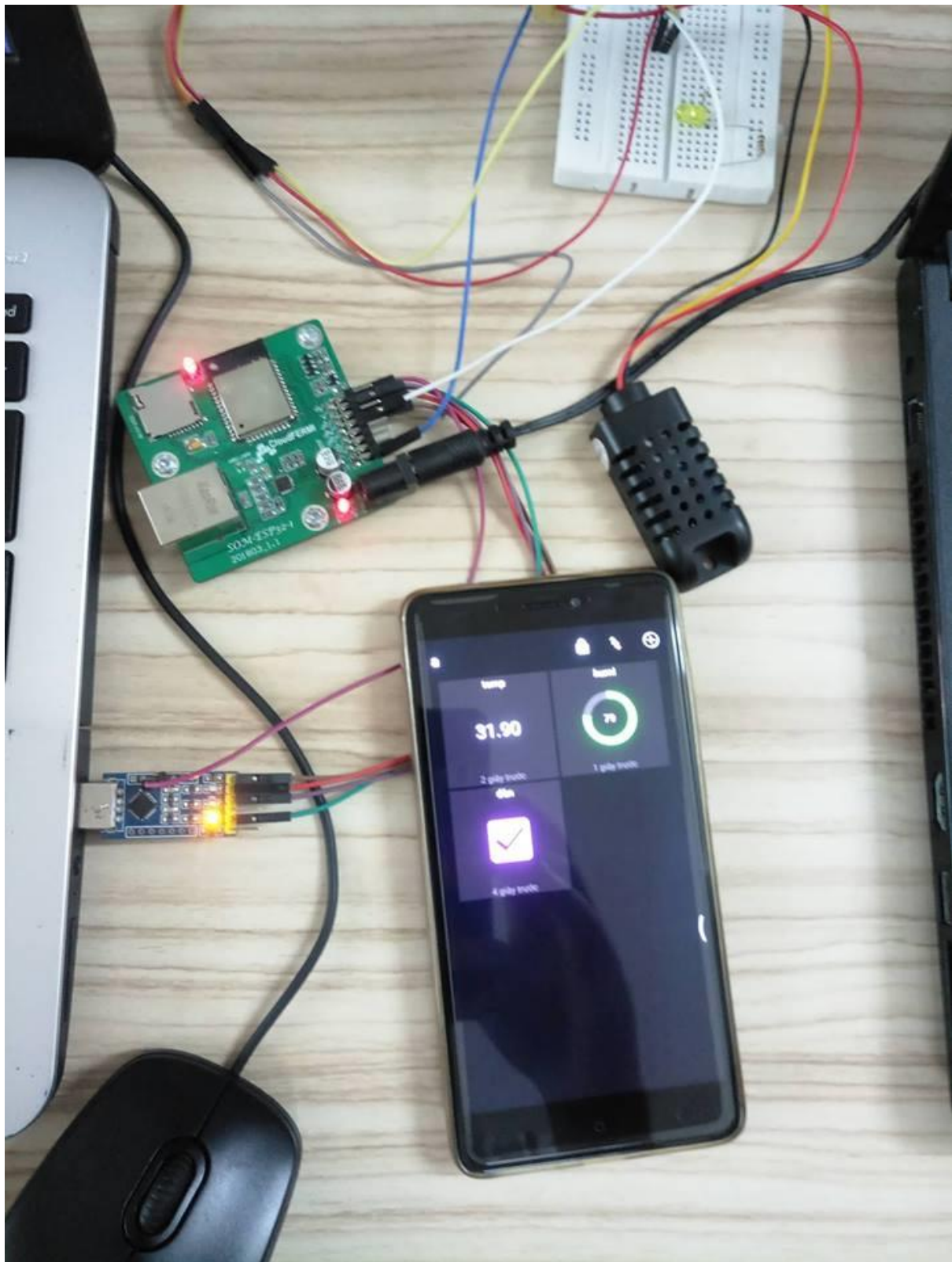


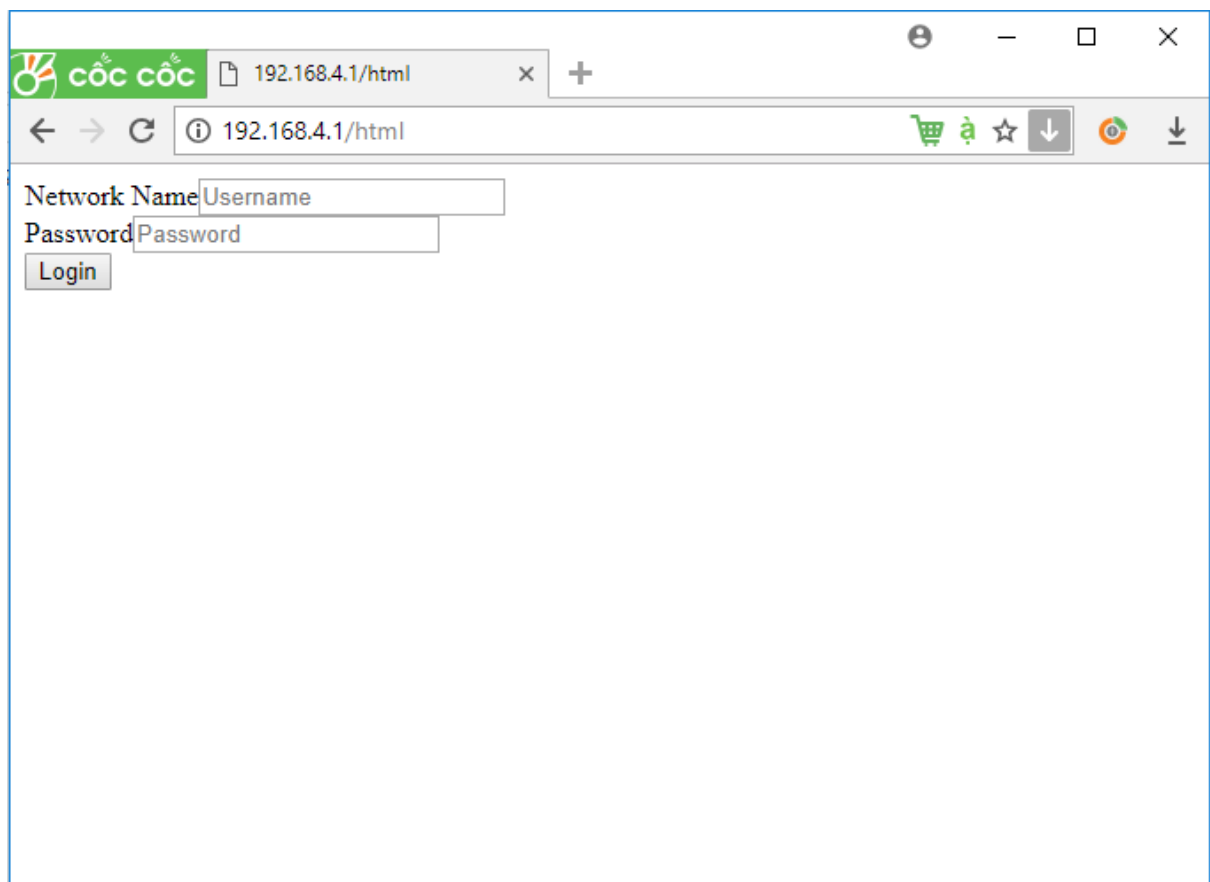
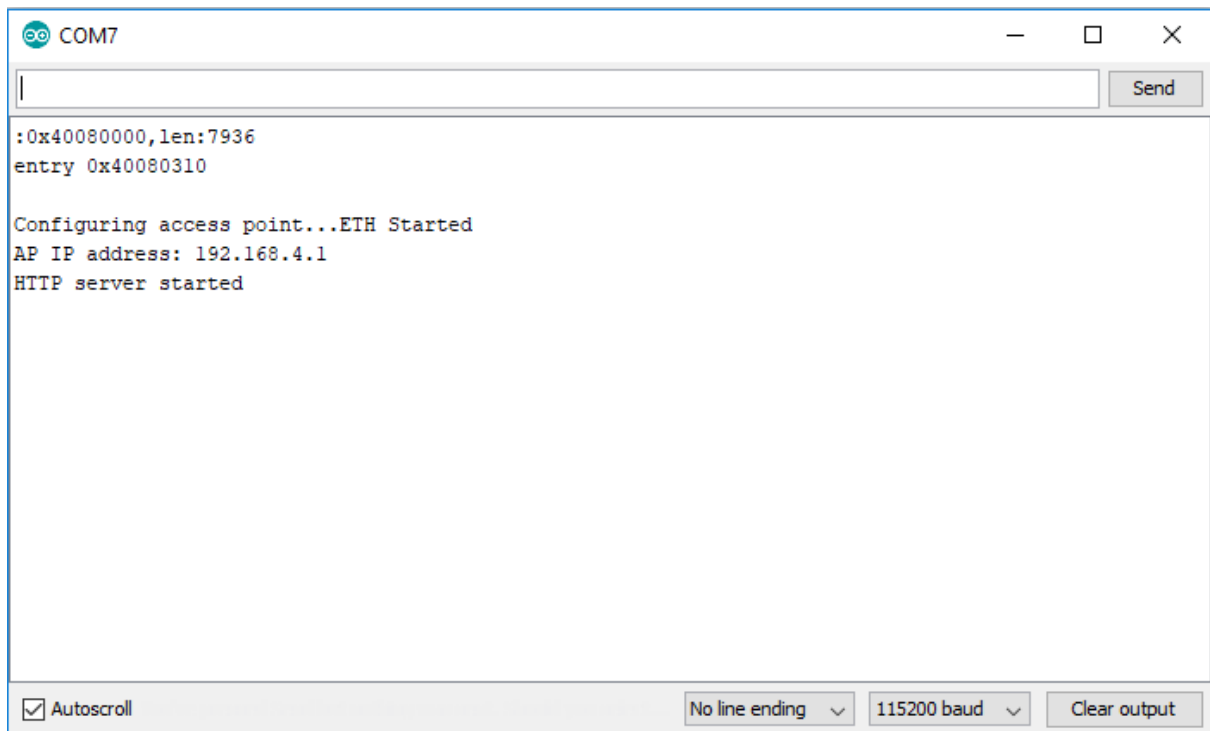
## 2.10 Kết quả thực hiện

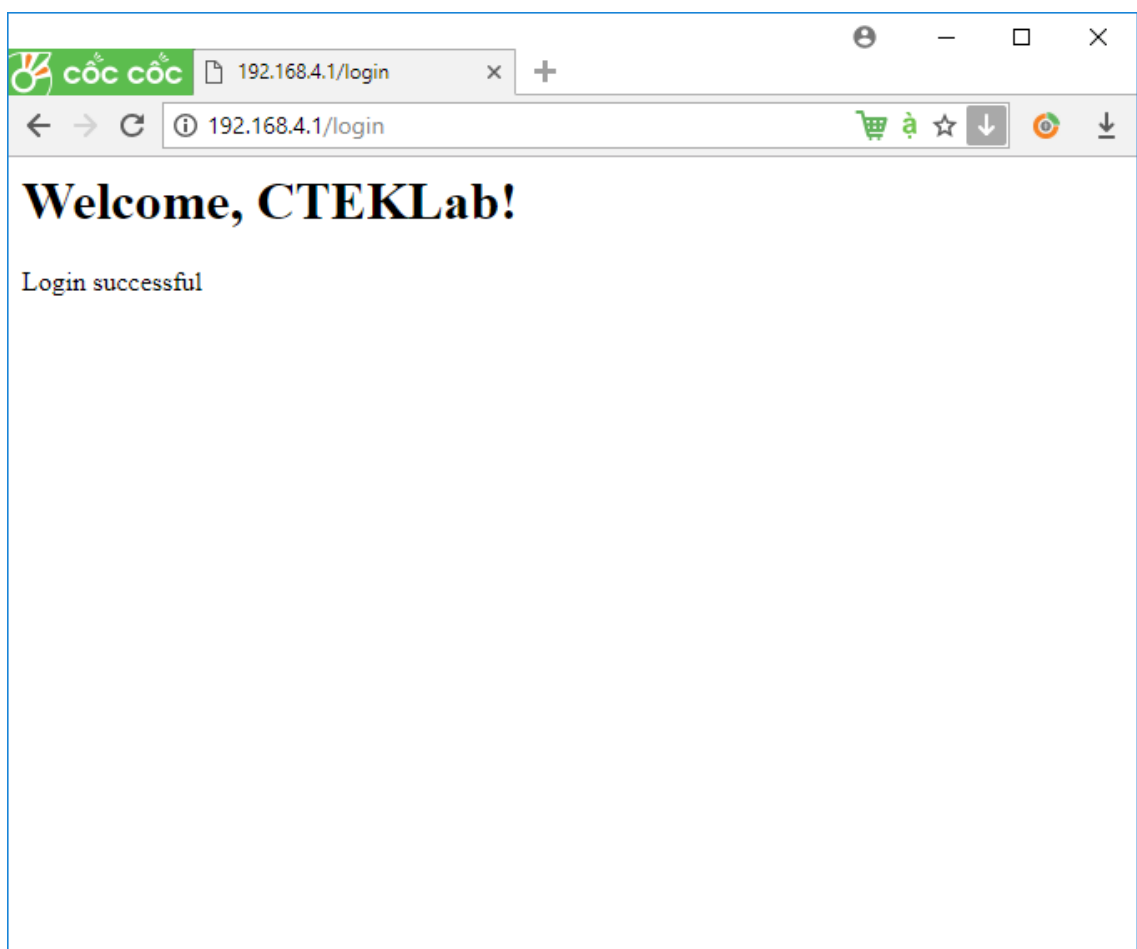
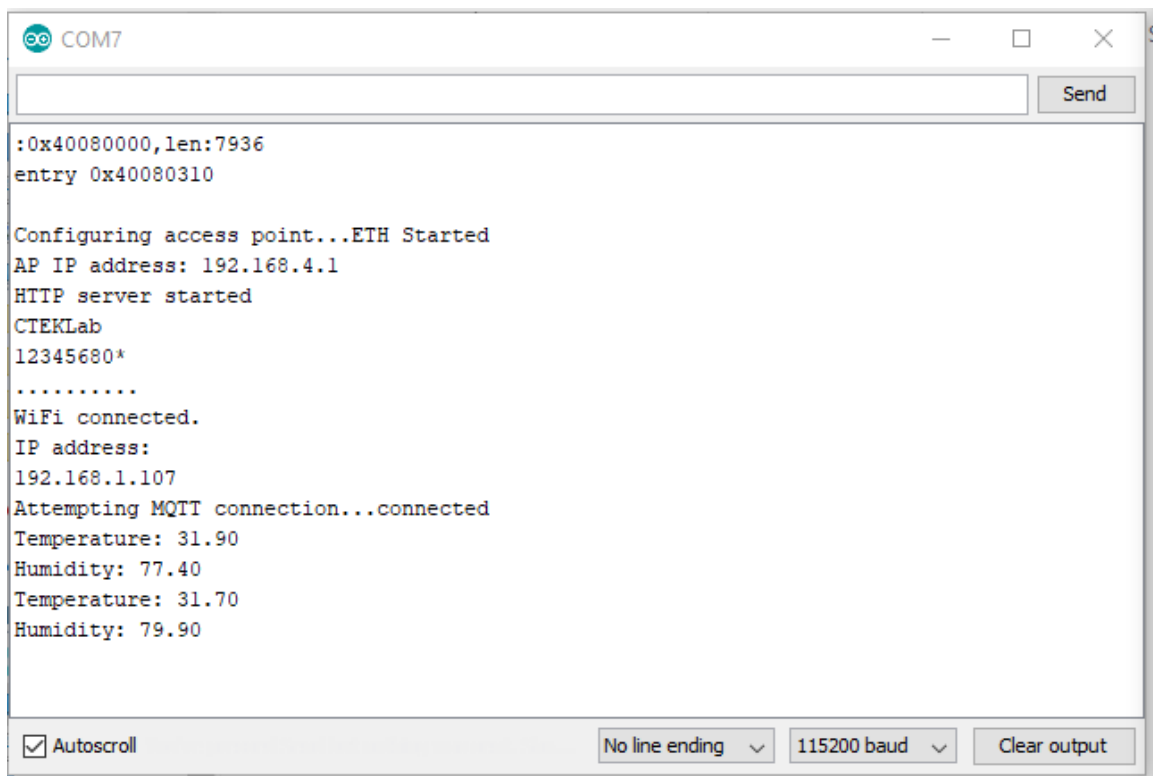
Đọc dữ liệu từ Sensor sử dụng Module STM32-LORA-NODE và gửi đến gateway thông qua mạng LORA



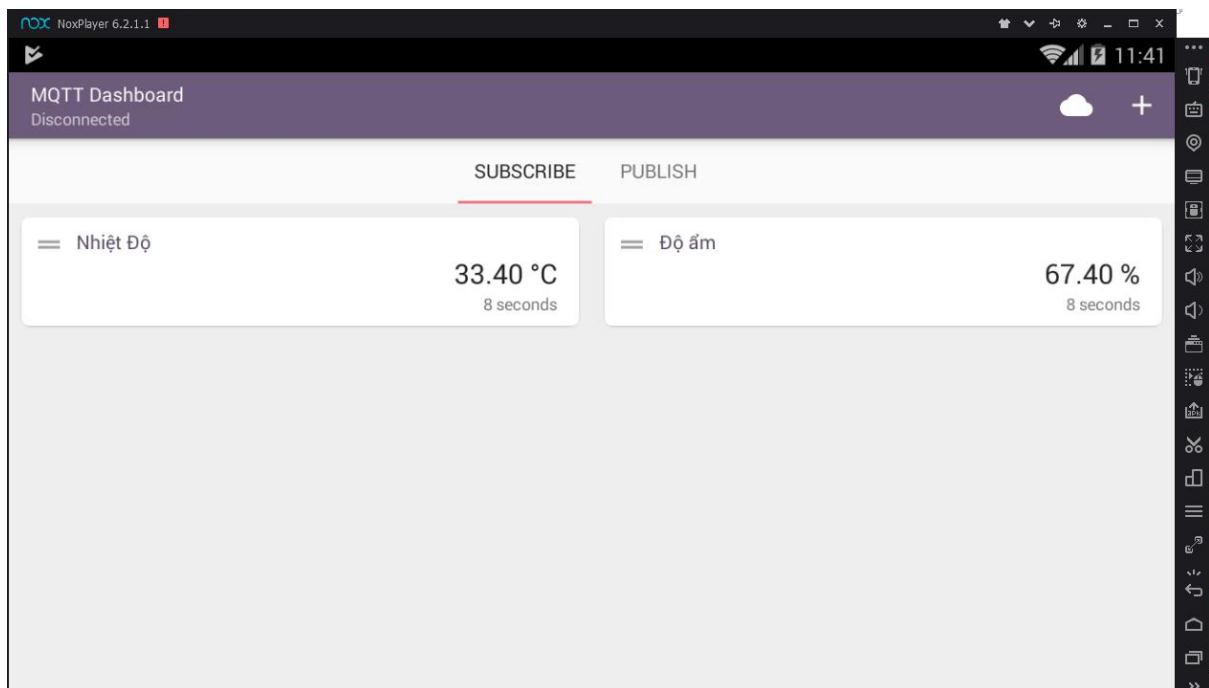
Độc cảm biến trực tiếp sử dụng Module SOM-ESP32-1 rồi gửi lên Cloud



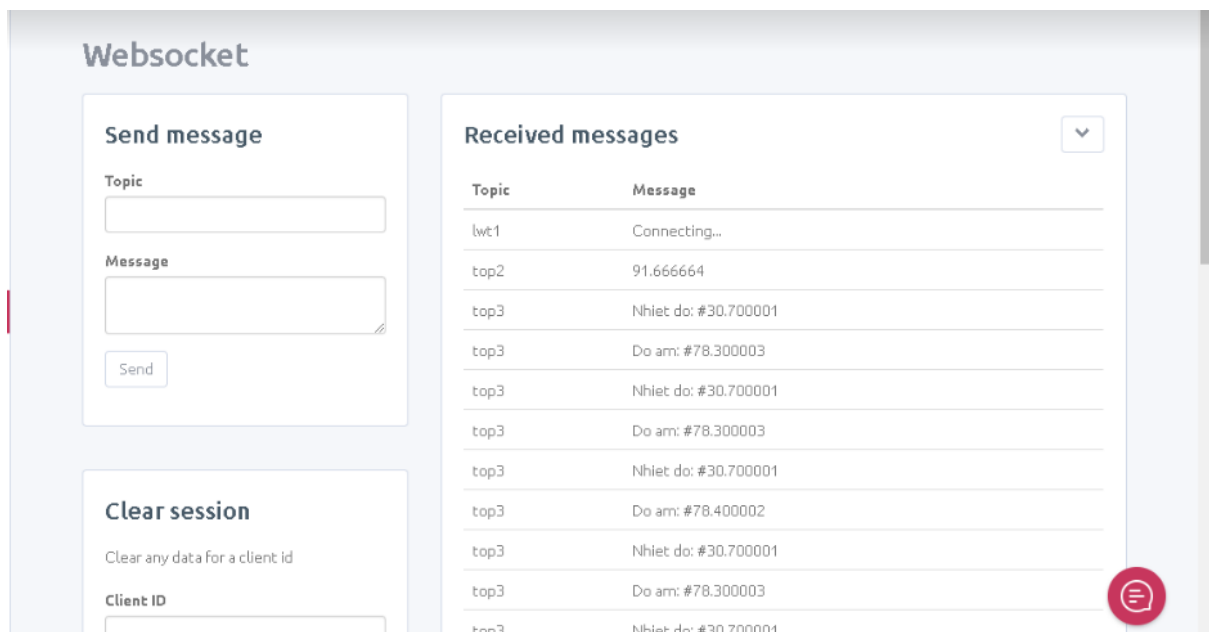




## Kết quả thông tin nhiệt độ, độ ẩm hiển thị trên App MQTT trên điện thoại Android



## Kết quả thông tin nhiệt độ, độ ẩm trên Web





## 3. TỔNG KẾT CÔNG VIỆC THỰC TẬP

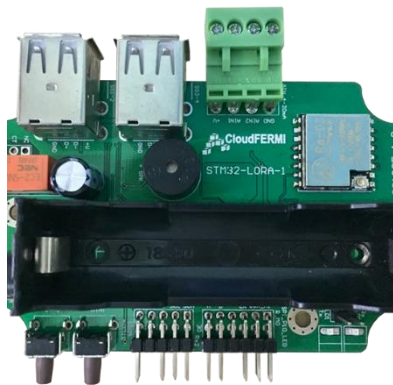
### 3.1. Kết quả công việc thực tập

Đề tài : Giám sát nhiệt độ và độ ẩm gửi lên web server qua mạng Lora vẫn chưa hoàn thành. Đề tài đã đạt được các nhiệm vụ sau:

- Tìm hiểu về VĐK STM32 , cảm biến DHT21 , mạng LORA, ESP32 , WIFI , LAN và giao thức MQTT .
- Tiến hành đọc nhiệt độ ,độ ẩm từ DHT21 về VĐK STM32 .
- dùng ESP32 đọc dữ liệu và gửi lên WEBSERVER thông qua giao thức MQTT . Giám sát các thông số và điều khiển thiết bị thông qua APP điện thoại.

Nhiệm vụ chưa hoàn thành : truyền nhận dữ liệu từ node về gateway Lora.

Module Node



Module Gateway



### 3.2. Kinh nghiệm học được sau khi thực tập

Kinh nghiệm : Học hỏi được nhiều kiến thức mới liên quan đến mạng lora, esp32, MQTT, STM32.

Kỹ năng làm việc nhóm.

## 4. TÀI LIỆU THAM KHẢO

1. [IoT] Những điều cần biết về giao thức MQTT
2. Lập trình STM32/ HocARM.
3. Tìm hiểu về Lora
4. ESP32 MQTT – Publish and Subscribe with Arduino IDE
5. Datasheet STM32F4-discovery
6. Open Source

<https://github.com/IoTThinks>

<https://github.com/knolleary/pubsubclient>

7. Example:

❶ Examples For ESP32 Dev Module

❷ <http://www.iotsharing.com/search?updated-max=2017-05-15T16:57:00-07:00&max-results=8&start=44&by-date=false>

❸ <https://techtutorialsx.com/2018/01/06/esp32-arduino-http-server-serving-html-and-javascript/>

## 5. PHỤ LỤC

- Sơ đồ:

