



ĐẠI HỌC FPT

VIỆN ĐÀO TẠO QUỐC TẾ FPT

THIẾT KẾ VI MẠCH



HỆ THỐNG CẢM BIẾN CẢNH BÁO

Giáo viên hướng dẫn: Thầy Huỳnh Nhựt Hải, Thầy Trương Huy Hoàng

Lớp: C1.2502.E0

Tên nhóm Nhóm 2

Tên thành viên trong nhóm: _____ **Tên đầy đủ** _____ **Mã số Sinh viên** _____

1.	Dương Minh An	C1S2412009
2.	Nguyễn Thanh Hào	C1S2412005

12-2025

MỤC LỤC

MỤC LỤC	2
DANH MỤC HÌNH ẢNH.....	4
PHẦN 1: GIỚI THIỆU.....	6
1.1 Giới thiệu tổng quan:.....	6
1.2 Ý tưởng thực hiện đề tài:.....	6
1.3 Cấu trúc tổng quan hệ thống:.....	7
1.3.1 Sơ đồ hệ thống:	7
1.3.2. Chức năng hoạt động:	7
PHẦN 2: THIẾT BỊ - PHẦN MỀM	9
2.1 Thành phần linh kiện:	9
2.1.1 EBAZ4205 Development Board:.....	9
2.1.2 DHT11:.....	10
2.1.3 74HC595 4-Digit Display Module:.....	14
2.1.4 Thiết bị hiển thị LCD 16x2:.....	17
2.1.5 Module I2C:	18
2.1.6 ESP32:	21
2.1.7 Cảm Biến Khí Gas MQ-2:	30
2.1.8 Module Led Đơn 8 Kênh:	31
2.1.9 Còi buzz thụ động 5V:.....	31
2.1.10 MKE-M21 SIM768x 4G SMS/CALL IoT Module:	32
2.1.11 Mạch Vietduino Uno USB-C:	33
2.2 Phân mềm và công cụ:.....	34
2.2.1 Thingsboard.cloud:	34
2.2.2 Vivado2025.1:.....	34
2.2.3 Arduino:	35
PHẦN 3: QUÁ TRÌNH THIẾT KẾ	37
3.1 Sơ đồ kết nối:.....	37

3.2 Lắp đặt mô hình:	38
3.2.1 Flowchart:	40
3.2.2 Kết quả thực nghiệm:	41
PHẦN 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	41
4.1 Kết luận:	41
4.2 Hướng phát triển:	41
PHẦN 6: PHÂN CÔNG CÔNG VIỆC NHÓM	42
PHẦN 7: TÀI LIỆU THAM KHẢO	43

DANH MỤC HÌNH ẢNH

<i>Hình 1.1: Sơ đồ hệ thống</i>	7
<i>Hình 2.1: EBAZ4205 Development Board</i>	9
<i>Hình 2.2: Cảm biến DHT11</i>	10
<i>Hình 2.3: Ứng dụng DHT11</i>	11
<i>Hình 2.4: Quy trình giao tiếp tổng thể</i>	12
<i>Hình 2.5: MCU gửi tín hiệu bắt đầu đến DHT</i>	12
<i>Hình 2.6: Dữ liệu chỉ báo "0"</i>	13
<i>Hình 2.7: Dữ liệu chỉ báo "1"</i>	14
<i>Hình 2.8: 74HC595 4-Digit Display Module</i>	14
<i>Hình 2.9: Sơ đồ ứng dụng</i>	16
<i>Hình 2.10: 74HC595 Block Diagram</i>	17
<i>Hình 2.11: Thiết bị hiển thị LCD 16x2</i>	17
<i>Hình 2.12: Module I2C</i>	18
<i>Hình 2.13: Diagram LCD I2C</i>	18
<i>Hình 2.14: Kiểm nghiệm đo giao thức I2C</i>	21
<i>Hình 2.15: Vi điều khiển ESP32</i>	21
<i>Hình 2.16: Đo sóng trong giao thức UART TX</i>	28
<i>Hình 2.17: Đo sóng trong giao thức UART RX</i>	29
<i>Hình 2.18: Cảm Biến Khí Gas MQ-2</i>	30
<i>Hình 2.19: Module Led Đon 8 Kênh</i>	31
<i>Hình 2.20: Còi buzz thụ động 5V</i>	31
<i>Hình 2.21: MKE-M21 SIM768x 4G SMS/CALL IoT Module</i>	32
<i>Hình 2.22: Mạch Vietduino Uno USB-C</i>	33
<i>Hình 2.23: Hình ảnh phần mềm thingsboard</i>	34
<i>Hình 2.24: Hình ảnh Vivado2025.1</i>	34
<i>Hình 2.25: Sơ đồ hệ thống thiết kế</i>	35
<i>Hình 2.26: Sơ đồ luồng DHT11</i>	35
<i>Hình 2.27: Hình ảnh lập trình Arduino</i>	36

<i>Hình 3.1: Sơ đồ kết nối hệ thống</i>	37
<i>Hình 3.2: Quá trình lắp đặt mô hình</i>	38
<i>Hình 3.3: Mô hình hoàn thiện</i>	39
<i>Hình 3.4: Flowchart</i>	40
<i>Hình 3.5: Thực nghiệm</i>	41

PHẦN 1: GIỚI THIỆU

1.1 Giới thiệu tổng quan:

Xu hướng phát triển của Internet of Things (IoT) trong vài năm tới được dự báo sẽ bùng nổ với số lượng thiết bị kết nối lên đến hàng trăm ngàn, thậm chí hàng triệu.

Trong quá trình phát triển kinh tế – xã hội, đặc biệt là giai đoạn đô thị hóa và công nghiệp hóa mạnh mẽ, vấn đề an toàn phòng cháy chữa cháy (PCCC) ngày càng trở thành mối quan tâm hàng đầu. Các vụ hỏa hoạn xảy ra ở nhà dân, chung cư, nhà máy, trung tâm thương mại hay kho hàng không chỉ gây thiệt hại nghiêm trọng về người và tài sản mà còn để lại hậu quả lâu dài đối với môi trường và sự phát triển bền vững. Thực tế cho thấy, nhiều vụ cháy lớn trong những năm gần đây đã làm dấy lên nỗi lo ngại về khả năng ứng phó và hệ thống cảnh báo hiện có.

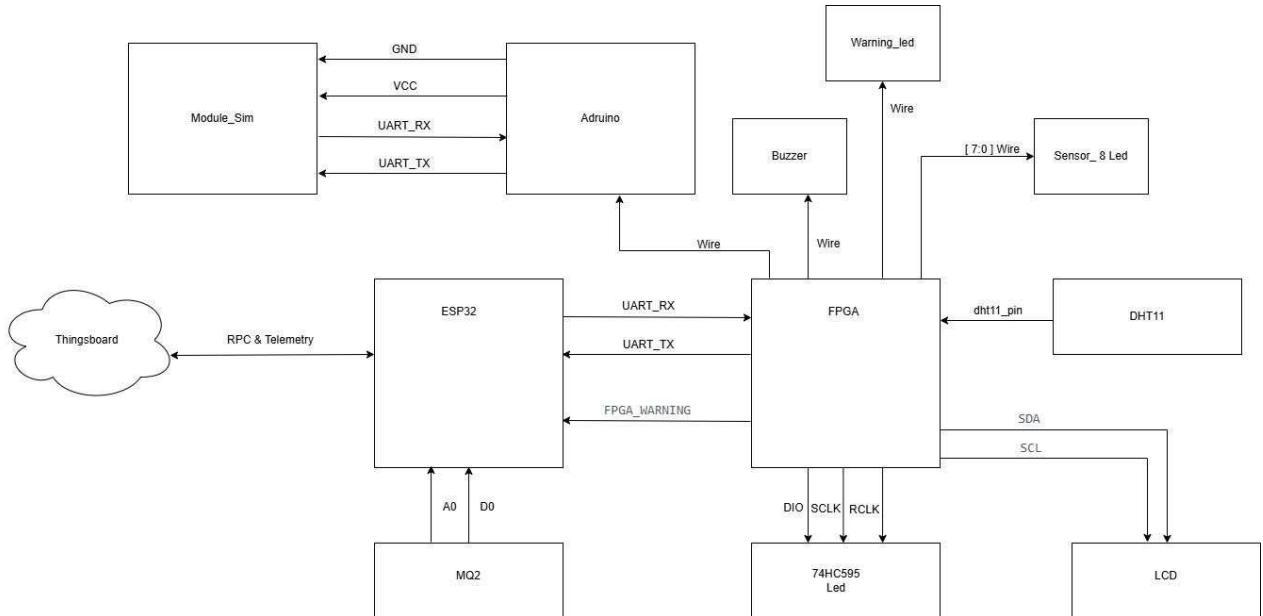
Trong bối cảnh đó, việc nghiên cứu, xây dựng và triển khai hệ thống cảnh báo và giám sát hiện đại thông qua các ứng dụng IoT đã được các tổ chức hay các quốc gia nghiên cứu phát triển. Đây không chỉ là giải pháp kỹ thuật mà còn là công cụ quản lý thông minh, giúp phát hiện sớm nguy cơ, cảnh báo kịp thời và hỗ trợ công tác ứng cứu hiệu quả.

1.2 Ý tưởng thực hiện đề tài:

- Xây dựng hệ thống có khả năng phát hiện sớm các dấu hiệu rủi ro như (khói, nhiệt độ, độ ẩm...) thông qua các cảm biến và bộ điều khiển.
- Tích hợp cảnh báo tự động qua loa, đèn báo, tin nhắn, cuộc gọi.
- Giám sát trực tuyến bằng ứng dụng, giúp quản lý theo dõi tình trạng an toàn từ xa,
- Tối ưu chi phí nhưng vẫn đảm bảo hiệu quả và độ tin cậy cao.
- Hỗ trợ công tác quản lý phòng cháy chữa cháy một cách chủ động, thông minh.
- Ứng dụng trong nhiều lĩnh vực: nhà ở, trường học, kho hàng...

1.3 Cấu trúc tổng quan hệ thống:

1.3.1 Sơ đồ hệ thống:



Hình 1.1: Sơ đồ hệ thống

1.3.2. Các chức năng hoạt động:

Thiết bị	Chức năng
ESP32	<ul style="list-style-type: none"> Nhận tín hiệu từ MQ2, xử lý và truyền dữ liệu lên Thingsboard. Gửi tín hiệu đến FPGA để xử lý.
FPGA	<ul style="list-style-type: none"> Nhận dữ liệu từ ESP32 và DHT11. Hiển thị trạng thái cảnh báo khói qua LED Sensor_8 (dãy LED từ [7:0]). Điều khiển LCD I2C để hiển thị nhiệt độ, độ ẩm. Kích hoạt còi cảnh báo và LED cảnh báo khi phát hiện nguy cơ cháy. Gửi tín hiệu điều khiển đến Vietduino Uno USB-C khi cần thực hiện cảnh báo bằng cuộc gọi.
Vietduino Uno USB-C	<ul style="list-style-type: none"> Nhận tín hiệu từ FPGA qua giao tiếp dây.

Thiết bị	Chức năng
	<ul style="list-style-type: none"> - Kích hoạt module MKE-M21 SIM768x qua UART để thực hiện cuộc gọi/ nhắn tin cảnh báo đến số điện thoại đã cài đặt trước.
MKE-M21 SIM768x	<ul style="list-style-type: none"> - Khởi động tập lệnh AT để thiết lập thông số SIM và thực hiện cuộc gọi tự động khi có sự cố.
DHT11	<ul style="list-style-type: none"> - Cảm biến đọc nhiệt độ, độ ẩm gửi tín hiệu đến FPGA.
LED Hiện thị trạng thái giám sát (ON/OFF)	<ul style="list-style-type: none"> - Tắt / mở chế độ cảnh báo sớm của hệ thống.
Còi / LED cảnh báo	<ul style="list-style-type: none"> - Kêu / sáng đèn nhấp nháy khi hệ thống phát hiện nguy cơ.
Led 74HC595	<ul style="list-style-type: none"> - Được FPGA điều khiển qua các chân DIO, SCLK, RCLK để hiển thị lên hệ thống từ ESP32 các thông số digital của MQ2.
LCD I2C	<ul style="list-style-type: none"> - Hiển thị nhiệt độ, độ ẩm
Thingsboard	<ul style="list-style-type: none"> - Nhận dữ liệu telemetry từ ESP32. - Hiển thị biểu đồ nhiệt độ, độ ẩm, nồng độ khí. - Thực hiện điều khiển từ xa qua RPC (Remote Procedure Call).

1.3.4 Ý nghĩa thực tiễn:

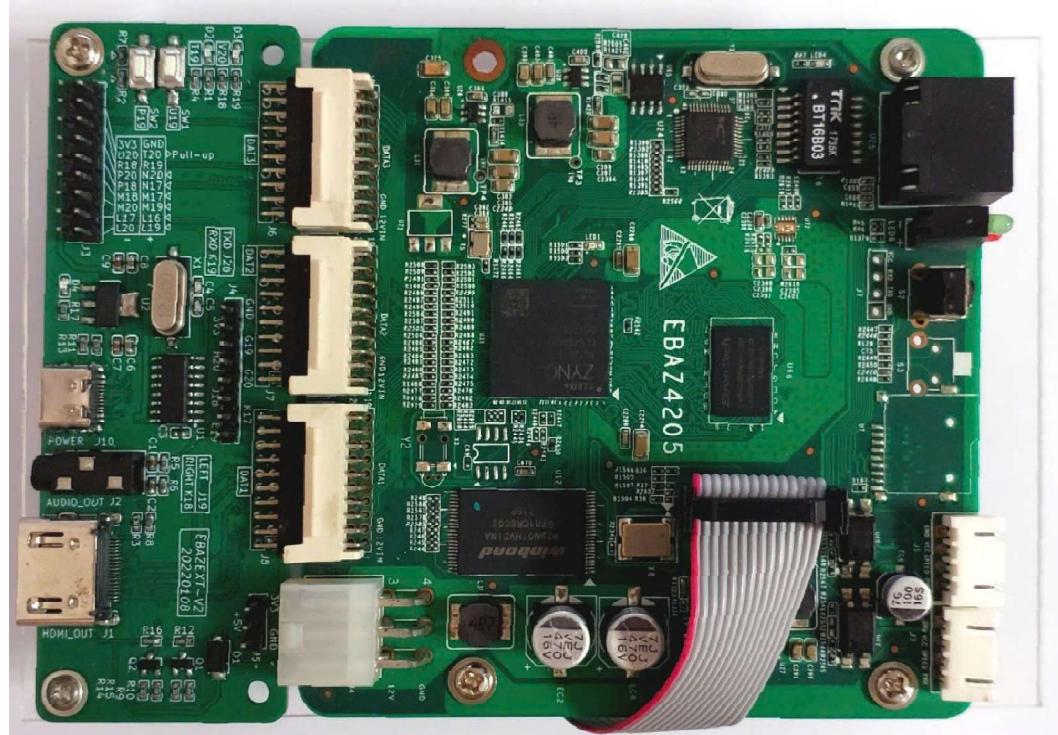
- Kết hợp cảm biến khí và nhiệt độ trên board mạch, dễ dàng thay thế bảo trì.
- Hiển thị trực quan, có thể theo dõi bằng mắt thường nhiệt độ khi trong khu vực hiển thị LCD.
- Cảnh báo đa kênh: tại chỗ (còi, đèn) và từ xa (cuộc gọi, IoT).
- Giám sát thời gian thực: qua nền tảng Thingsboard.
- Tính mở rộng cao: dễ dàng tích hợp thêm cảm biến hoặc thiết bị mới.

- Độ tin cậy: hoạt động song song qua Wi-Fi và SIM, giảm nguy cơ mất kết nối.

PHẦN 2: THIẾT BỊ - PHẦN MỀM

2.1 Thành phần linh kiện:

2.1.1 EBAZ4205 Development Board:



Hình 2.1: EBAZ4205 Development Board

- **Cấu hình phần cứng:**

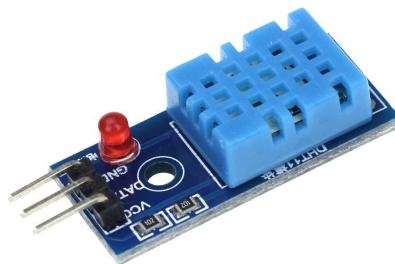
Thành phần	Mô tả
SoC	Xilinx Zynq-7010: gồm FPGA Artix-7 + Dual-core ARM Cortex-A9 @ 666MHz
RAM	256MB DDR3
Flash	128MB NAND

Thành phần	Mô tả
Ethernet	10/100Mbps IP101GA
GPIO	Khoảng 40 chân I/O từ PL (Programmable Logic)
UART, JTAG	Có thể thêm bằng cách hàn thủ công
Nguồn	5V–12V qua cổng DATA hoặc J3/J5 (sau khi gắn diode D24)

- *Ứng dụng chính của EBAZ4205:*

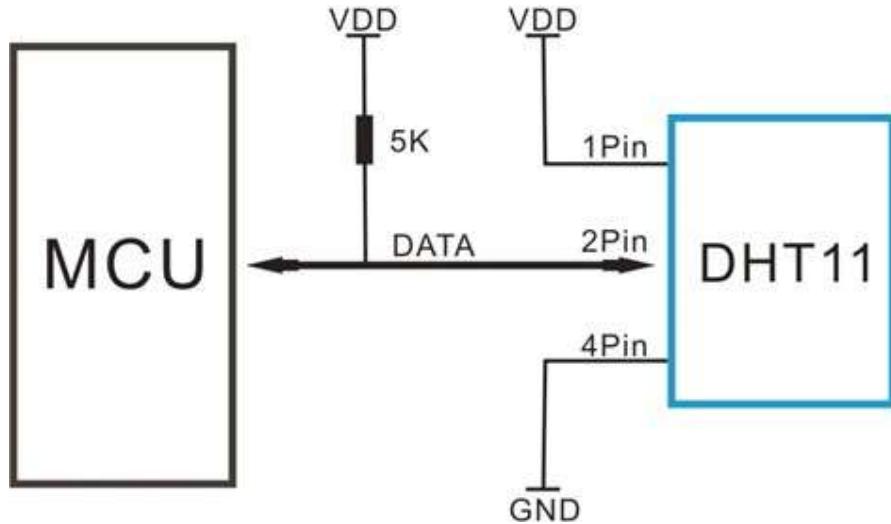
- **Lập trình FPGA:** sử dụng Vivado để thiết kế mạch số, xử lý tín hiệu, giao tiếp ngoại vi.
- **Hệ thống nhúng:** chạy Linux nhúng trên ARM Cortex-A9, phát triển ứng dụng nhúng như điều khiển thiết bị, truyền thông mạng.
- **Học tập SoC Zynq:** kết hợp giữa phần mềm (ARM) và phần cứng (FPGA) trong một chip duy nhất.
- **Thử nghiệm giao tiếp ngoại vi:** UART, GPIO, Ethernet, PWM, LED, cảm biến...

2.1.2 DHT11:



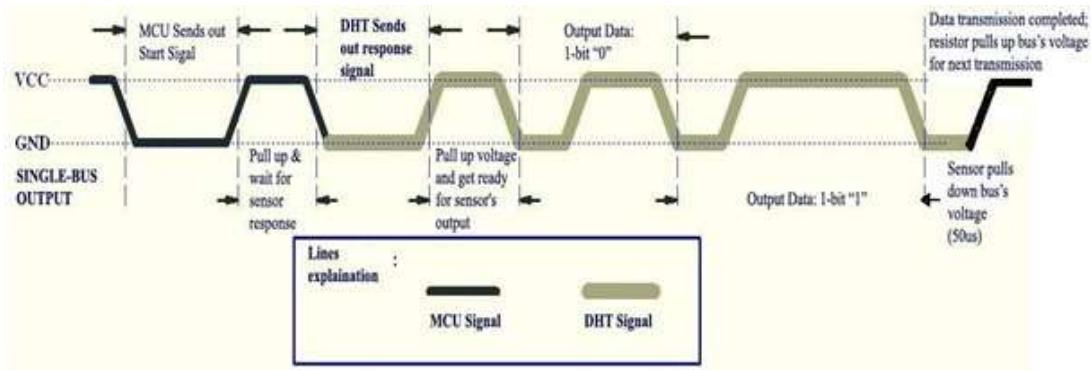
Hình 2.2: Cảm biến DHT11

- Khi cáp kết nối ngắn hơn 20 mét, nên sử dụng điện trở kéo lên 5K; khi cáp kết nối dài hơn 20 mét, hãy chọn điện trở kéo lên phù hợp theo nhu cầu.



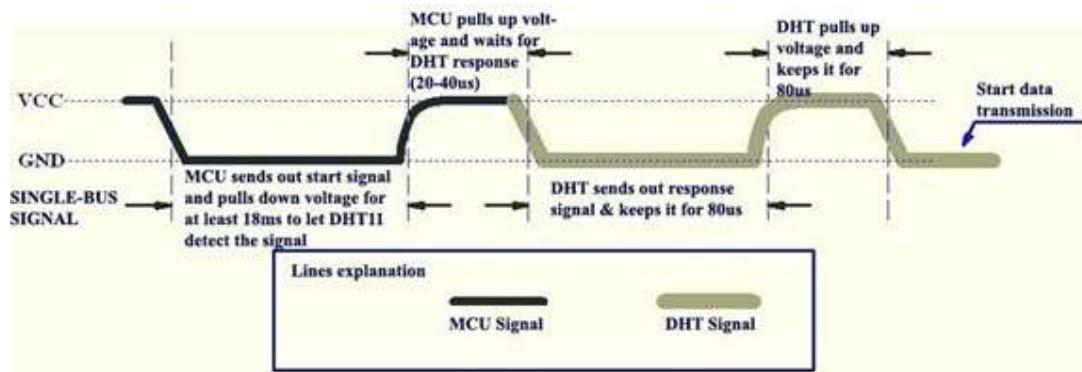
Hình 2.3: Ứng dụng DHT11

- Quy trình giao tiếp tổng thể: Khi MCU gửi tín hiệu bắt đầu, DHT11 chuyển từ chế độ tiêu thụ điện năng thấp sang chế độ hoạt động, chờ MCU hoàn tất tín hiệu bắt đầu. Sau khi hoàn tất, DHT11 gửi tín hiệu phản hồi gồm 40 bit dữ liệu bao gồm thông tin độ ẩm tương đối và nhiệt độ đến MCU. Người dùng có thể chọn thu thập (đọc) một số dữ liệu. Nếu không có tín hiệu bắt đầu từ MCU, DHT11 sẽ không gửi tín hiệu phản hồi đến MCU. Sau khi dữ liệu được thu thập, DHT11 sẽ chuyển sang chế độ tiêu thụ điện năng thấp cho đến khi nhận được tín hiệu bắt đầu từ MCU một lần nữa.



Hình 2.4: Quy trình giao tiếp tổng thể

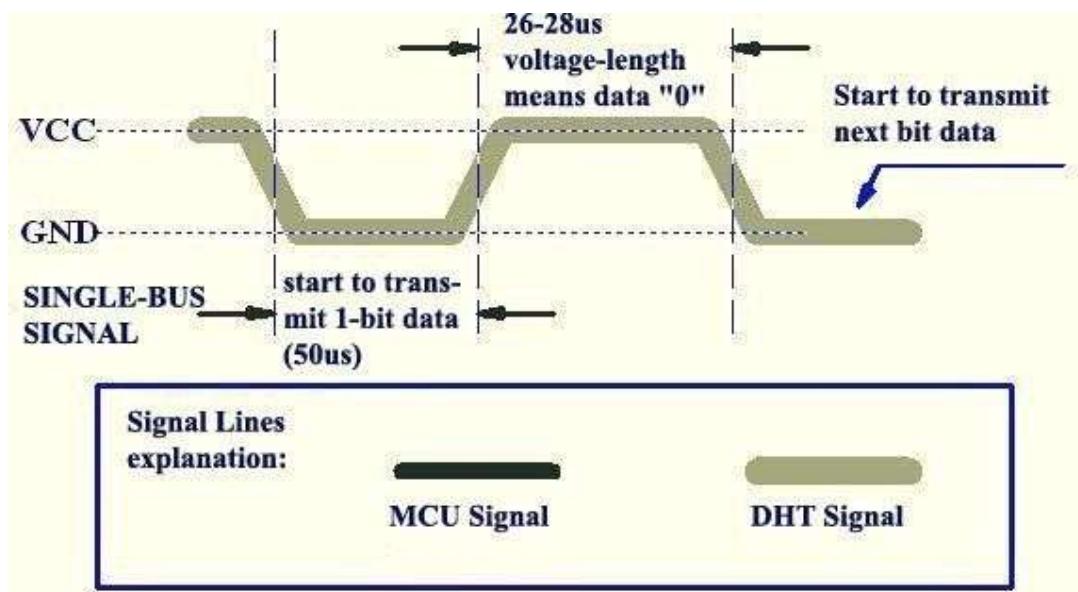
- MCU gửi tín hiệu bắt đầu đến DHT: Trạng thái rảnh của bus đơn dữ liệu ở mức điện áp cao. Khi quá trình giao tiếp giữa MCU và DHT11 bắt đầu, chương trình của MCU sẽ thiết lập mức điện áp bus đơn dữ liệu từ cao xuống thấp và quá trình này phải mất ít nhất 18ms để đảm bảo DHT phát hiện được tín hiệu của MCU, sau đó MCU sẽ kéo điện áp lên và chờ 20-40us để nhận phản hồi từ DHT.



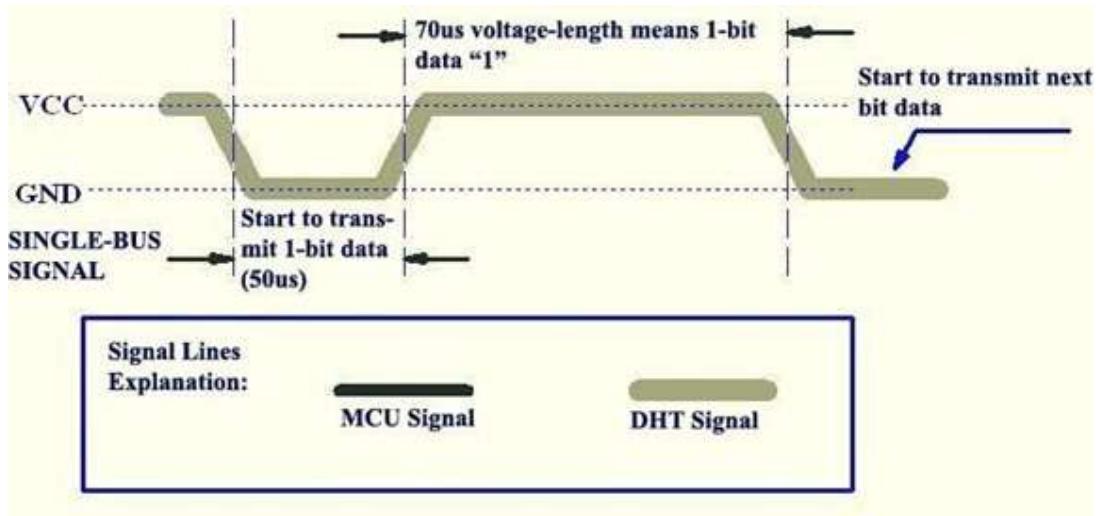
Hình 2.5: MCU gửi tín hiệu bắt đầu đến DHT

- Phản hồi của DHT đến MCU (Hình 2.5)
 - Khi DHT phát hiện tín hiệu bắt đầu, nó sẽ gửi tín hiệu phản hồi ở mức điện áp thấp, kéo dài 80us. Sau đó, chương trình của DHT sẽ thiết lập mức điện áp của bus đơn dữ liệu từ thấp lên cao và duy trì ở mức này trong 80us để DHT chuẩn bị gửi dữ liệu.

- Khi bus đơn dữ liệu ở mức điện áp thấp, điều này có nghĩa là DHT đang gửi tín hiệu phản hồi. Sau khi DHT gửi tín hiệu phản hồi, nó sẽ tăng điện áp và duy trì ở mức này trong 80us để chuẩn bị truyền dữ liệu.
- Khi DHT gửi dữ liệu đến MCU, mỗi bit dữ liệu bắt đầu bằng tín hiệu mức điện áp thấp 50us và độ dài của tín hiệu mức điện áp cao tiếp theo sẽ xác định bit dữ liệu là "0" hay "1" (xem *Hình 2.6* và *Hình 2.7* bên dưới).

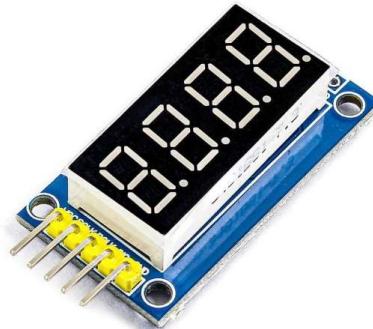


Hình 2.6: Dữ liệu chỉ báo "0"



Hình 2.7: Dữ liệu chỉ báo "1"

2.1.3 74HC595 4-Digit Display Module:



Hình 2.8: 74HC595 4-Digit Display Module

- Tính năng:

- Bộ dịch chuyển nối tiếp 8 bit, đầu vào song song, đầu ra song song
- Dải điện áp hoạt động rộng từ 2 V đến 6 V.
- Đầu ra 3 trạng thái dòng điện cao có thể điều khiển tối đa 15 tải LSTTL.
- Tiêu thụ điện năng thấp: 80 μ A (tối đa) ICC.
- $t_{pd} = 13$ ns (điển hình).

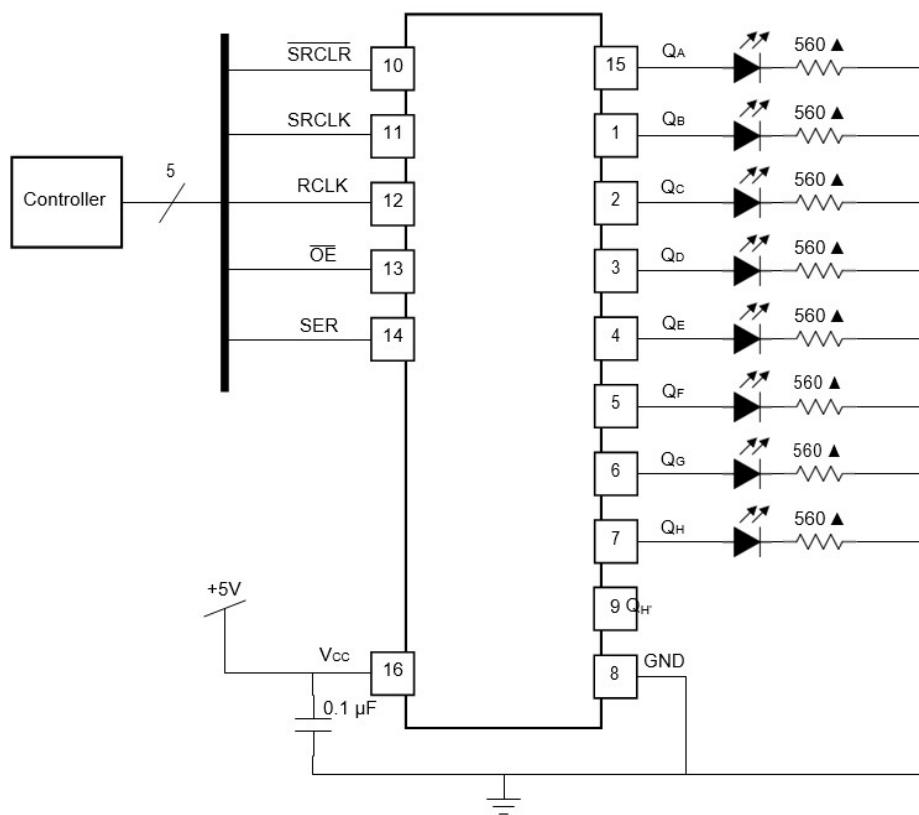
- Dòng điện đầu ra ± 6 mA ở 5 V.
- Dòng điện đầu vào thấp: 1 μ A (tối đa).
- Thanh ghi dịch chuyển có chức năng xóa trực tiếp.
- Đối với các sản phẩm tuân thủ tiêu chuẩn MIL-PRF-38535, tất cả các thông số đều được kiểm tra trừ khi có ghi chú khác. Đối với tất cả các sản phẩm khác, quy trình sản xuất không nhất thiết bao gồm việc kiểm tra tất cả các thông số.

- **Ứng dụng:**

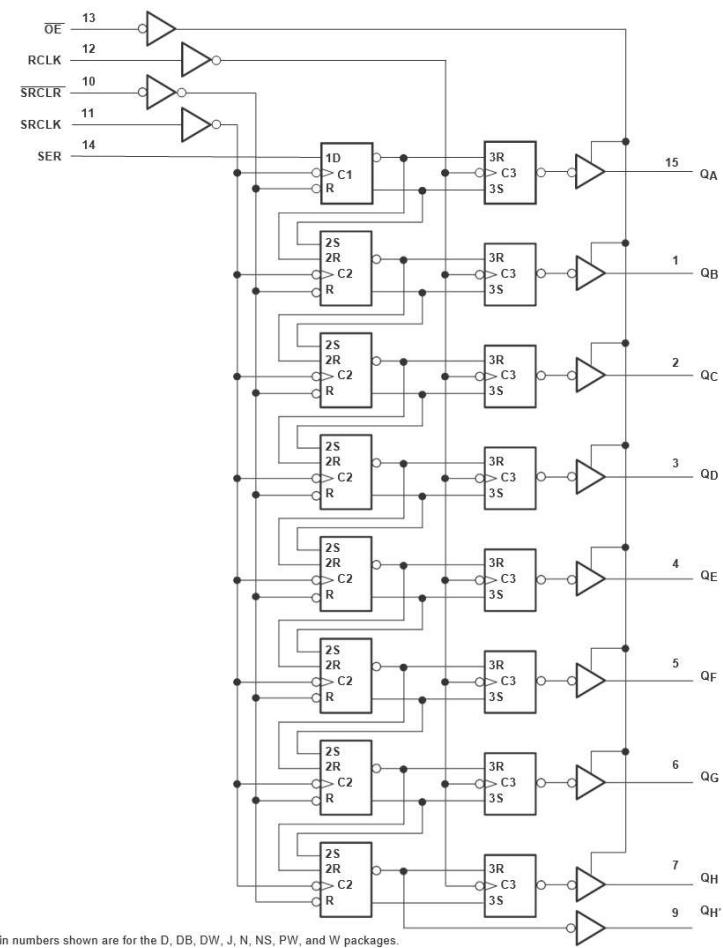
- Bộ chuyển mạch mạng.
- Cơ sở hạ tầng điện.
- Màn hình LED.
- Máy chủ.

- **3 Mô tả:**

- Các thiết bị SNx4HC595 chứa một thanh ghi dịch 8 bit, đầu vào nối tiếp, đầu ra song song, cấp tín hiệu cho một thanh ghi lưu trữ D-type 8 bit. Thanh ghi lưu trữ có đầu ra song song 3 trạng thái. Xung nhịp riêng biệt được cung cấp cho cả thanh ghi dịch và thanh ghi lưu trữ. Thanh ghi dịch có đầu vào xóa ghi đè trực tiếp (SRCLR), đầu vào nối tiếp (SER) và đầu ra nối tiếp để ghép nối tiếp. Khi đầu vào cho phép đầu ra (OE) ở mức cao, các đầu ra ở trạng thái trở kháng cao.
- Cả xung nhịp thanh ghi dịch (SRCLK) và xung nhịp thanh ghi lưu trữ (RCLK) đều được kích hoạt bởi cạnh dương. Nếu cả hai xung nhịp được kết nối với nhau, thanh ghi dịch luôn đi trước thanh ghi lưu trữ một xung nhịp.



Hình 2.9: Sơ đồ ứng dụng



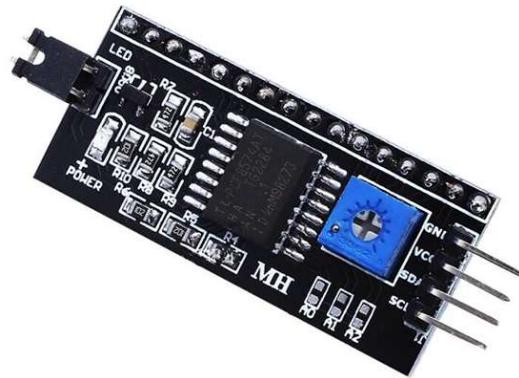
Hình 2.10: 74HC595 Block Diagram

2.1.4 Thiết bị hiển thị LCD 16x2:



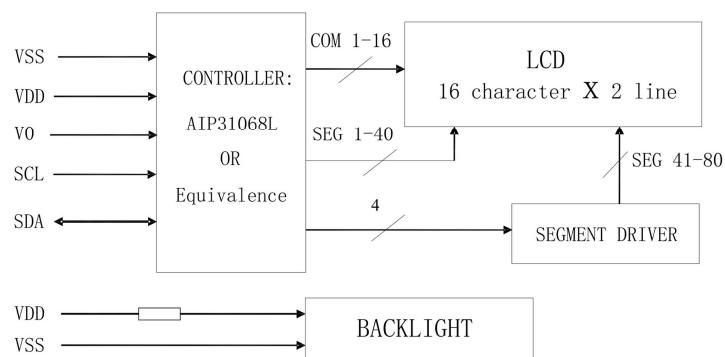
Hình 2.11: Thiết bị hiển thị LCD 16x2

2.1.5 Module I2C:



Hình 2.12: Module I2C

- Diagram:

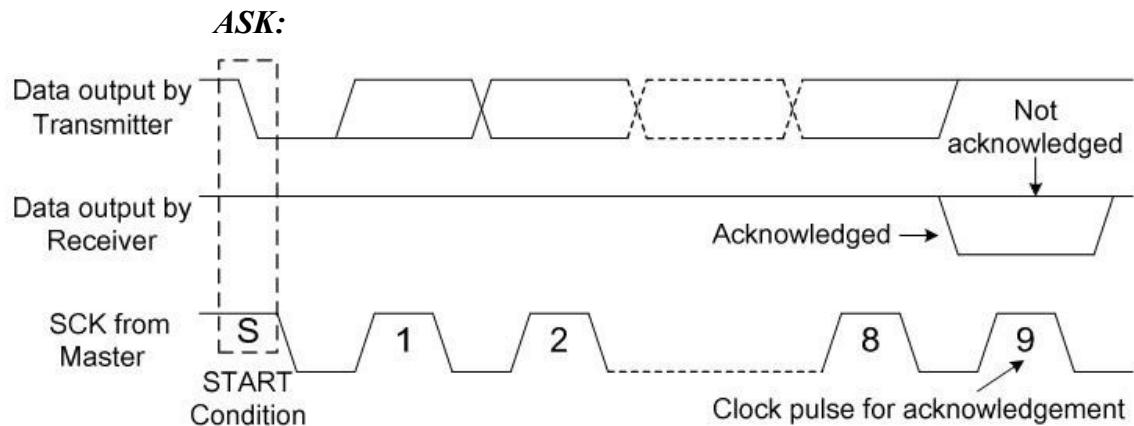
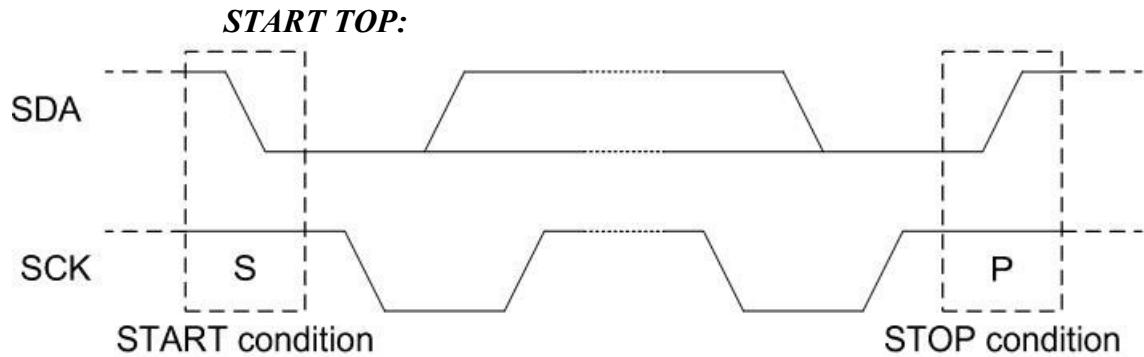


Hình 2.13: Diagram LCD I2C

- Mô tả chân cắm:

Chân	Tên	Mức	Chức năng
1	SDA	H/L	Dữ liệu I2C
2	SCL	H/L	Xung clock I2C
3	GND	0V	Mass
4	VCC	3.3–5V	Nguồn

- **Thời gian ghi MPU:**



- **Khôi phục Module LCD:** Khi bật nguồn VDD và VSS, module LCD sẽ tự động thực hiện quy trình khôi phục. Quá trình này mất khoảng 50ms. Sau khi khôi phục, trạng thái của module LCD sẽ như sau:

- N=1, hiển thị 2 dòng
- Xóa màn hình
- DL=1, giao diện 8 bit
- F=0, phông chữ ký tự 5×8 chấm
- D=0, Tắt màn hình — C=0, Tắt con trỏ — B=0, Tắt nhấp nháy
- I/D=1, Tăng 1
- S=0, Không dịch chuyển

LƯU Ý: Quy trình khôi phục không thể tạo ra Cài đặt cơ bản

- **Sơ đồ bộ nhớ hiển thị:** Có hai vùng bộ nhớ chính trong mô-đun LCD dành cho hiển thị.
 - o RAM tạo ký tự (CGRAM)
 - o RAM dữ liệu hiển thị (DDRAM)
- **RAM tạo ký tự (CGRAM):**
 - o *RAM tạo ký tự dùng để lưu trữ các ký tự do người dùng định nghĩa (phông chữ 5×8 chấm). Có thể tạo tổng cộng 8 ký tự do người dùng định nghĩa (mã ký tự = 00h-07h).*
 - o *Mã ký tự do người dùng định nghĩa là 00h và 07h. Chúng có thể được gọi vào DDRAM như các ký tự thông thường.*

User-defined Character Code	CGRAM Address	CGRAM Data (Font Pattern)	
		D7 ~ D5	D4 ~ D0
00h (08h)	00h 01h ⋮ 06h 07h	Not Use	5 x 8 dots font pattern
01h (09h)	08h 09h ⋮ 0Eh 0Fh	Not Use	5 x 8 dots font pattern
02h (0Ah)	10h 11h ⋮ 16h 17h	Not Use	5 x 8 dots font pattern
03h (0Bh)	18h 19h ⋮ 1Eh 1Fh	Not Use	5 x 8 dots font pattern
04h (0Ch)	20h 21h ⋮ 26h 27h	Not Use	5 x 8 dots font pattern
05h (0Dh)	28h 29h ⋮ 2Eh 2Fh	Not Use	5 x 8 dots font pattern
06h (0Eh)	30h 31h ⋮ 36h 37h	Not Use	5 x 8 dots font pattern
07h (0Fh)	38h 39h ⋮ 3Eh 3Fh	Not Use	5 x 8 dots font pattern

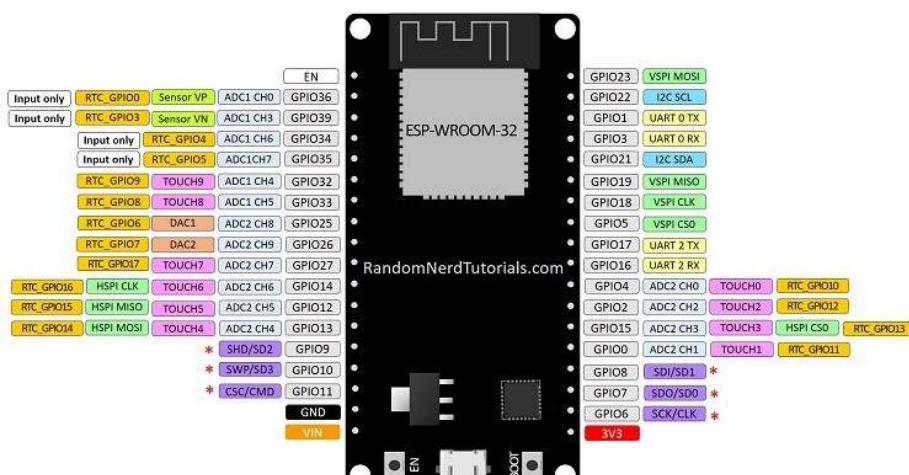
CGRAM Address Map



Hình 2.14: Kiểm nghiệm đo giao thíc I²C

2.1.6 ESP32:

ESP32 DEVKIT V1 – DOIT version with 36 GPIOs



* Pins SCK/CLK, SDO/SD0, SD/SD1, SHD/SD2, SWP/SD3 and SCS/CMD, namely, GPIO6 to GPIO11 are connected to the integrated SPI flash integrated on ESP-WROOM-32 and are not recommended for other uses.

Hình 2.15: Vi điều khiển ESP32

- ***Giới thiệu chung:***
 - o **Chân Input Only:** GPIO từ 34 đến 39 là GPI – chân chỉ đầu vào. Các chân này không có điện trở kéo lên hoặc kéo xuống bên trong. Chúng không thể được sử dụng làm đầu ra, vì vậy chỉ sử dụng các chân này làm đầu vào:
 - GPIO 34
 - GPIO 35
 - GPIO 36
 - GPIO 39
 - o **Chân tích hợp Flash trên ESP32:** GPIO 6 đến GPIO 11 dùng để kết nối Flash SPI, không khuyến khích sử dụng trong các ứng dụng khác
 - GPIO 6 (SCK/CLK)
 - GPIO 7 (SDO/SD0)
 - GPIO 8 (SDI/SD1)
 - GPIO 9 (SHD/SD2)
 - GPIO 10 (SWP/SD3)
 - GPIO 11 (CSC/CMD)
 - o **Chân cảm biến điện dung:** Các chân ESP32 này có chức năng như 1 nút nhấn cảm ứng, có thể phát hiện sự thay đổi về điện áp cảm ứng trên chân. Các cảm biến cảm ứng bên trong đó được kết nối với các GPIO sau:
 - T0 (GPIO 4)
 - T1 (GPIO 0)
 - T2 (GPIO 2)
 - T3 (GPIO 15)
 - T4 (GPIO 13)
 - T5 (GPIO 12)
 - T6 (GPIO 14)
 - T7 (GPIO 27)

- T8 (GPIO 33)
- T9 (GPIO 32)
- **Analog to Digital Converter (ADC):** ESP32 có các kênh đầu vào ADC 18 x 12 bit (trong khi ESP8266 chỉ có ADC 1x 10 bit). Đây là các GPIO có thể được sử dụng làm ADC và các kênh tương ứng:
 - ADC1_CH0 (GPIO 36)
 - ADC1_CH1 (GPIO 37)
 - ADC1_CH2 (GPIO 38)
 - ADC1_CH3 (GPIO 39)
 - ADC1_CH4 (GPIO 32)
 - ADC1_CH5 (GPIO 33)
 - ADC1_CH6 (GPIO 34)
 - ADC1_CH7 (GPIO 35)
 - ADC2_CH0 (GPIO 4)
 - ADC2_CH1 (GPIO 0)
 - ADC2_CH2 (GPIO 2)
 - ADC2_CH3 (GPIO 15)
 - ADC2_CH4 (GPIO 13)
 - ADC2_CH5 (GPIO 12)
 - ADC2_CH6 (GPIO 14)
 - ADC2_CH7 (GPIO 27)
 - ADC2_CH8 (GPIO 25)
 - ADC2_CH9 (GPIO 26)

Các kênh đầu vào ADC có độ phân giải 12 bit. Điều này có nghĩa là bạn có thể nhận được các số đọc tương tự từ 0 đến 4095, trong đó 0 tương ứng với 0V và 4095 đến 3,3V. Bạn cũng có thể lập trình độ phân giải của các kênh của mình trên code.

- **Digital to Analog Converter (DAC):** Có các kênh DAC 2 x 8 bit trên ESP32 để chuyển đổi tín hiệu kỹ thuật số thành đầu ra tín hiệu điện áp

tương tự. Các kênh này chỉ có độ phân giải 8 bit, nghĩa là có giá trị từ 0 – 255 tương ứng với 0 – 3.3V, đây là các kênh DAC:

- DAC1 (GPIO25)
- DAC2 (GPIO26)

- **Các chân thời gian thực RTC:** Các chân này có tác dụng đánh thức ESP32 khi trong chế độ Low Power Mode. Sử dụng như 1 chân ngắt ngoài. Các chân RTC:

- RTC_GPIO0 (GPIO36)
- RTC_GPIO3 (GPIO39)
- RTC_GPIO4 (GPIO34)
- RTC_GPIO5 (GPIO35)
- RTC_GPIO6 (GPIO25)
- RTC_GPIO7 (GPIO26)
- RTC_GPIO8 (GPIO33)
- RTC_GPIO9 (GPIO32)
- RTC_GPIO10 (GPIO4)
- RTC_GPIO11 (GPIO0)
- RTC_GPIO12 (GPIO2)
- RTC_GPIO13 (GPIO15)
- RTC_GPIO14 (GPIO13)
- RTC_GPIO15 (GPIO12)
- RTC_GPIO16 (GPIO14)
- RTC_GPIO17 (GPIO27)

- **Chân PWM:** ESP32 LED PWM có 16 kênh độc lập có thể được định cấu hình để tạo tín hiệu PWM với các thuộc tính khác nhau. Tất cả các chân có thể hoạt động như đầu ra đều có thể được sử dụng làm chân PWM (GPIO từ 34 đến 39 không thể tạo PWM). Để xuất PWM, bạn cần xác định các thông số này trong code:

- Frequency – tần số

- Duty cycle
 - Kênh PWM
 - Chân GPIO nơi bạn muốn xuất tín hiệu
 - **Chân I2C:** ESP32 có hai kênh I2C và bất kỳ chân nào cũng có thể được đặt làm SDA hoặc SCL. Khi sử dụng ESP32 với Arduino IDE, các chân I2C mặc định là:
 - GPIO 21 (SDA)
 - GPIO 22 (SCL)
 - **Chân Ngắt Ngoài:** Tất cả các chân ESP32 đều có thể sử dụng ngắt ngoài
- **Dưới đây là bảng tra cứu khả năng sử dụng của các chân:**

GPIO Pin	Input	Output	Chức năng đặc biệt
0	pulled up	OK	outputs PWM signal at boot
1	TX pin	OK	debug output at boot
2	OK	OK	connected to on-board LED
3	OK	RX pin	HIGH at boot
4	OK	OK	
5	OK	OK	outputs PWM signal at boot
6	X	X	connected to the integrated SPI flash
7	X	X	connected to the integrated SPI flash
8	X	X	connected to the integrated SPI flash
9	X	X	connected to the integrated SPI flash
10	X	X	connected to the integrated SPI flash

GPIO Pin	Input	Output	Chức năng đặc biệt
11	X	X	connected to the integrated SPI flash
12	OK	OK	boot fail if pulled high
13	OK	OK	
14	OK	OK	outputs PWM signal at boot
15	OK	OK	outputs PWM signal at boot
16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	
21	OK	OK	
22	OK	OK	
23	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK		input only
35	OK		input only
36	OK		input only
39	OK		input only

Bảng tra cứu khả năng sử dụng của các chân ESP32

- **Giao thức UART:** (Universal Asynchronous Receiver/Transmitter) là một giao thức truyền thông nối tiếp không đồng bộ được sử dụng rộng rãi trong các hệ thống nhúng, vi điều khiển và thiết bị điện tử. Giao tiếp UART cho phép truyền dữ liệu giữa hai thiết bị chỉ sử dụng hai dây truyền tín hiệu, đơn giản hóa kết nối và tiết kiệm chân vi xử lý.
- **Đặc điểm chính**
 - o Không đồng bộ: Không yêu cầu tín hiệu xung clock chung giữa hai thiết bị.
 - o Truyền nối tiếp: Dữ liệu được truyền từng bit một theo thời gian.
 - o Song song toàn phần: Có thể truyền và nhận dữ liệu đồng thời.
 - o Cấu hình linh hoạt: Có thể thay đổi tốc độ truyền, số bit dữ liệu, bit chẵn lẻ.
- **Ứng dụng thực tế**
 - o Kết nối vi điều khiển với máy tính (qua cổng COM).
 - o Giao tiếp giữa các module: GPS, Bluetooth, WiFi, RFID.
 - o Truyền thông trong hệ thống nhúng.
 - o Debug và cập nhật firmware.
- **Cấu trúc khung dữ liệu UART:** Một khung dữ liệu UART bao gồm các thành phần sau:
 - o *Start Bit(S)*
 - Luôn là mức logic 0 (Low)
 - Báo hiệu bắt đầu truyền một byte dữ liệu
 - Đồng bộ hóa giữa transmitter và receiver
 - o *Data Bits*
 - Chứa 8 bit dữ liệu thực tế cần truyền
 - Bit có trọng số thấp nhất (LSB) được truyền đầu tiên
 - o *Stop Bit(T)*
 - Luôn là mức logic 1 (High)
 - Đảm bảo khoảng thời gian nghỉ giữa các khung dữ liệu.

- *Tốc độ Baud Rate*

- Baud Rate (tốc độ truyền) là số bit được truyền trong một giây, bao gồm cả start bit, data bits, stop bit.
- Các tốc độ phổ biến: 9600 bps (thông dụng nhất), 19200 bps, 38400 bps, 115200 bps.

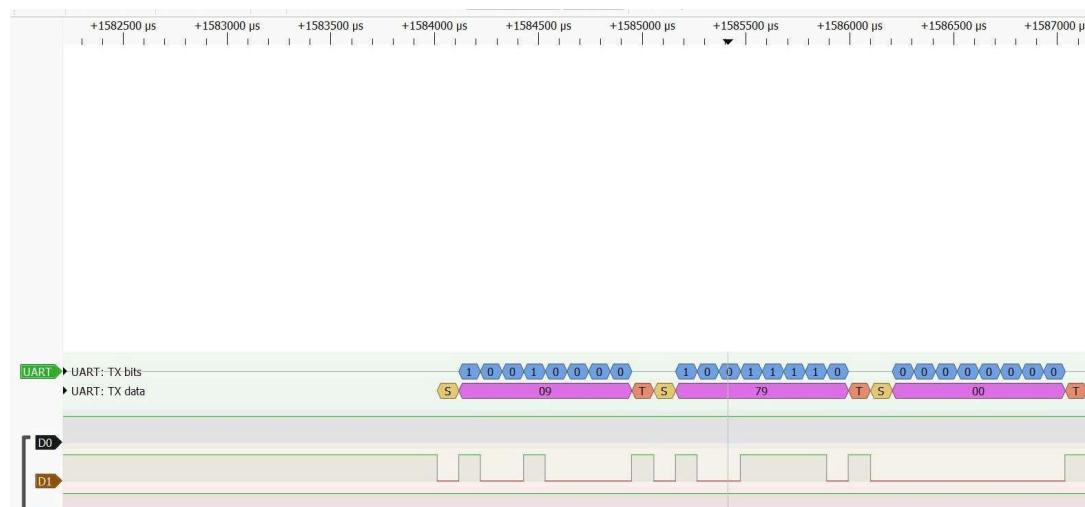
- *Công thức tính thời gian một bit (1 bit kéo dài 1 Baud Rate):*

- Thời gian 1 bit = 1 / Baud Rate
- Ví dụ: Với Baud Rate 9600, thời gian 1 bit = $1/9600 \approx 104.17 \mu\text{s}$.

- **Quy trình truyền dữ liệu:**

- Phía Transmitter (Bộ truyền)

- Chờ đường truyền TX rảnh.
- Gửi start bit (kéo xuống mức 0).
- Gửi lần lượt các bit dữ liệu từ LSB đến MSB.
- Gửi stop bit(s) (kéo lên mức 1).

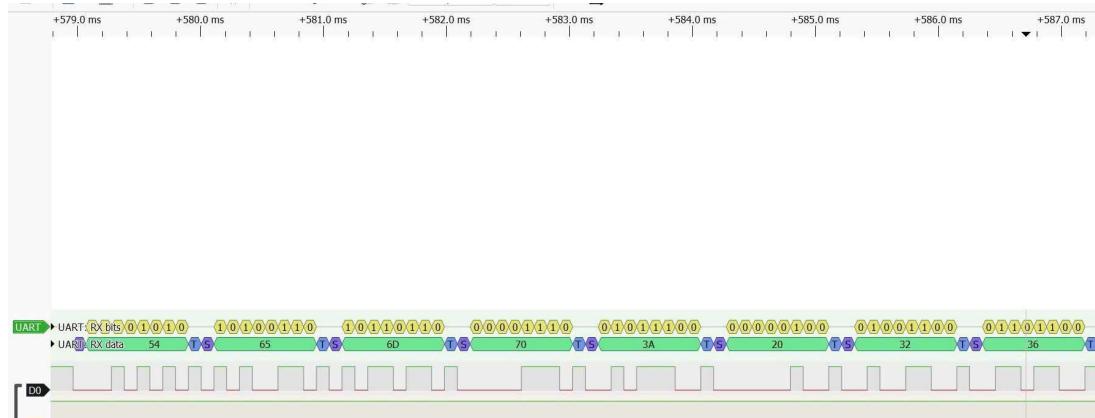


Hình 2.16: Đo sóng trong giao thức UART TX

- Phía Receiver (Bộ nhận)

- Phát hiện start bit (sườn xuống từ 1 xuống 0)
- Đợi 1.5 lần thời gian 1 bit để lấy mẫu ở giữa bit start

- Lấy mẫu các bit dữ liệu ở khoảng giữa mỗi bit
- Xác nhận stop bit(s)



Hình 2.17: Đo sóng trong giao thức UART RX

- **Đồng bộ hóa và lấy mẫu:** Vì UART không có xung clock chung, receiver phải tự đồng bộ bằng cách:

- o Sử dụng baud rate được cấu hình trùng khớp với transmitter
- o Lấy mẫu ở điểm giữa của mỗi bit để tránh méo tín hiệu
- o Sử dụng bộ đếm thời gian chính xác
- o Công thức tính Baud Rate tổng quát:

$$\text{Baud Rate} = \text{UART_input_clock} / (\mathbf{n} \times 16)$$

- o Trong đó:
 - **UART_input_clock:** Tần số xung nhịp đầu vào của module UART
 - **n:** Giá trị divisor trong thanh ghi DLH:DLL
 - **16:** Hệ số oversampling (mỗi bit dùng 16 chu kỳ BCLK)

- **Ưu điểm:**

- o Đơn giản: Chỉ cần 2 dây cho truyền thông 2 chiều
- o Phổ biến: Được hỗ trợ rộng rãi trên hầu hết vi điều khiển
- o Linh hoạt: Có thể cấu hình nhiều thông số khác nhau
- o Chi phí thấp: Không yêu cầu phần cứng phức tạp

- **Nhược điểm**

- o Tốc độ hạn chế: So với các giao thức đồng bộ
- o Khoảng cách ngắn: Thường dưới 15m (có thể tăng với driver)
- o Không có cơ chế xác nhận: Cần thêm protocol layer
- o Yêu cầu cấu hình trùng khớp: Baud rate phải giống nhau

2.1.7 Cảm Biến Khí Gas MQ-2:

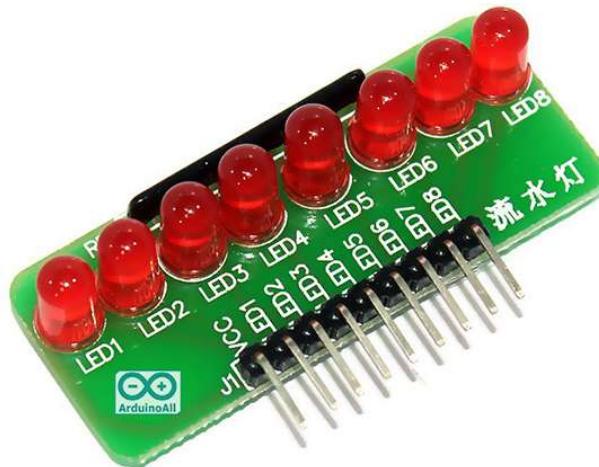


Hình 2.18: Cảm Biến Khí Gas MQ-2

- Cảm biến Vật liệu nhạy cảm của cảm biến khí MQ-2 là SnO₂, có độ dẫn điện thấp trong không khí sạch. Khi có khí dễ cháy cần kiểm tra, độ dẫn điện của cảm biến sẽ cao hơn cùng với sự gia tăng nồng độ khí.
- Cảm biến khí MQ-2 có độ nhạy cao với LPG, Propane và Hydro, cũng có thể được sử dụng cho Methane và các loại khí dễ cháy khác, có chi phí thấp và phù hợp với nhiều ứng dụng khác nhau.
- Đặc điểm cấu hình
 - o Độ nhạy tốt với khí dễ cháy trong phạm vi rộng
 - o Độ nhạy cao với LPG, Propane và Hydro
 - o Tuổi thọ cao và chi phí thấp
 - o Mạch điều khiển đơn giản

- **Ứng dụng**
 - o Máy dò rò rỉ khí trong gia đình
 - o Máy dò khí dễ cháy công nghiệp
 - o Máy dò khí cầm tay.

2.1.8 Module Led Đơn 8 Kênh:



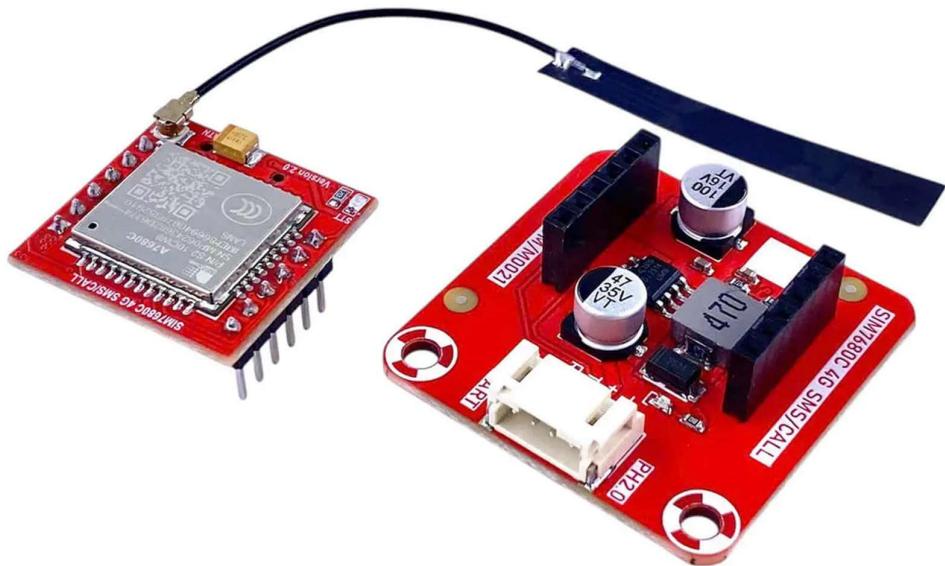
Hình 2.19: Module Led Đơn 8 Kênh

2.1.9 Còi buzz thụ động 5V:



Hình 2.20: Còi buzz thụ động 5V

2.1.10 MKE-M21 SIM768x 4G SMS/CALL IoT Module:



Hình 2.21: MKE-M21 SIM768x 4G SMS/CALL IoT Module

Thông số kỹ thuật

- Model: MKE-M21 SIM768x 4G SMS/CALL IoT Module
- IC chính: SIM768x
- Sử dụng 4G LTE CAT1
- Sử dụng 4G Nano SIM
- Điện áp hoạt động:
 - Cấp nguồn trực tiếp vào khối SIM, không đi kèm khối cấp nguồn: 3.7~4VDC
 - Cấp nguồn qua khối cấp nguồn: 5~24VDC
- Dòng điện tiêu thụ khi hoạt động: Trung bình 300mA, khi gọi hoặc nhắn tin có những thời điểm có thể lên tới 1A.

- Chuẩn giao tiếp: UART, Baudrate được thiết lập mặc định 9600
(Quan trọng: Test thực tế với Arduino set baudrate cao như 115200 truyền nhận rất dễ lỗi khiến module hoạt động kém ổn định)
- Điện áp giao tiếp: TTL 3.3VDC / 5VDC
- Sử dụng trực tiếp an toàn với các board mạch giao tiếp ở cả hai mức điện áp 3.3VDC và 5VDC như: Arduino, Raspberry Pi, Jetson Nano, Micro: bit,
- Bổ sung thêm các thiết kế ổn định, chống nhiễu.
- Chuẩn kết nối: connector XH2.54 4Pins / GPIO
- Chuẩn kết nối Antenna: IpeX
- Thuộc hệ sinh thái phần cứng Robotics MakerEdu, tương thích tốt nhất khi sử dụng với các mạch điều khiển trung tâm của MakerEdu và MakerEdu Shield.

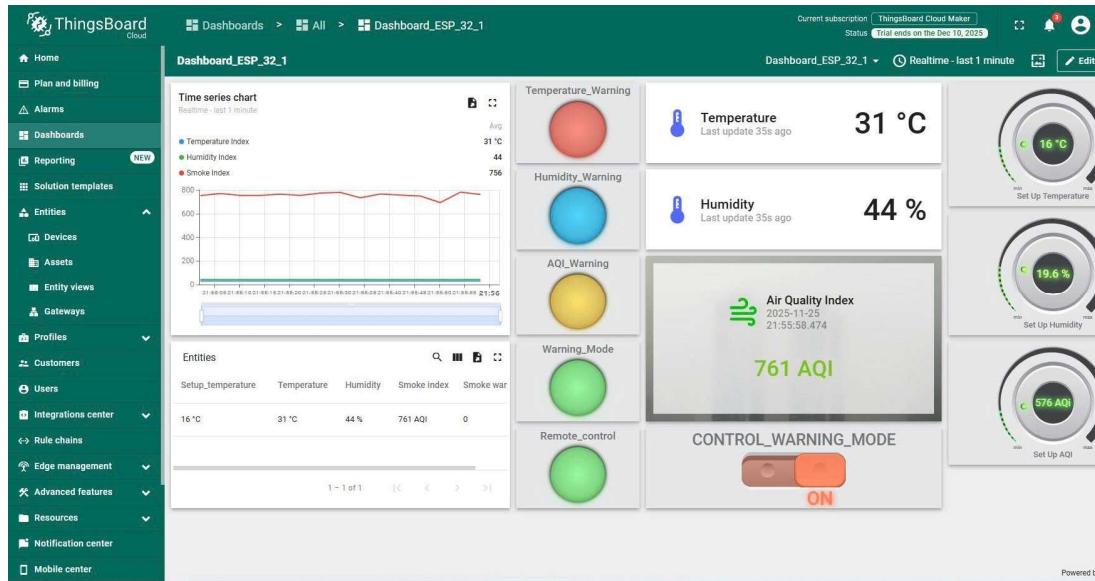
2.1.11 Mạch Vietduino Uno USB-C:



Hình 2.22: Mạch Vietduino Uno USB-C

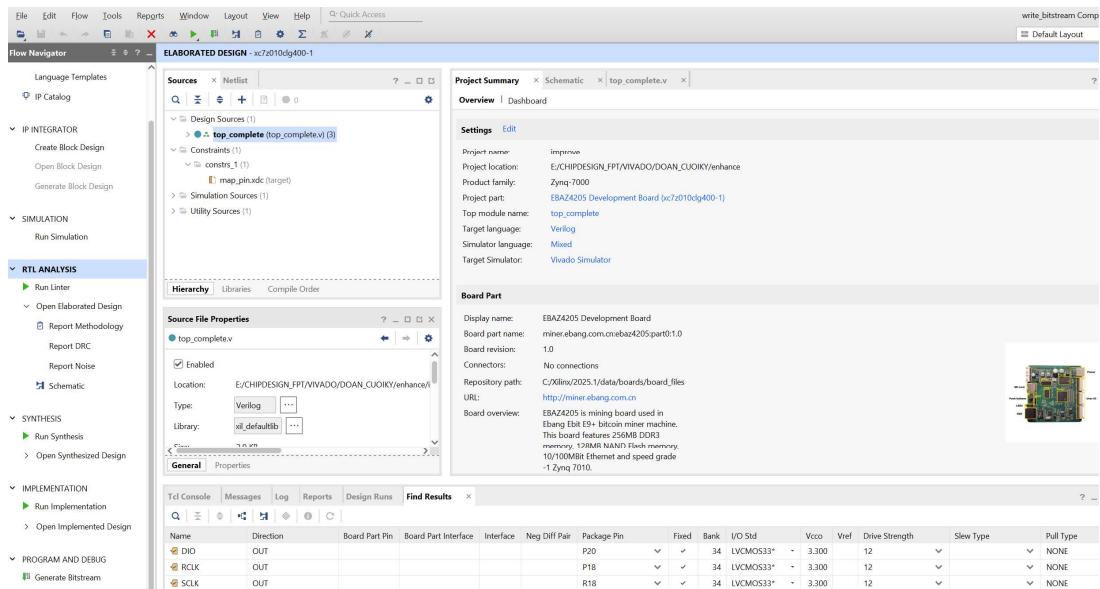
2.2 Phần mềm và công cụ:

2.2.1 Thingsboard.cloud:

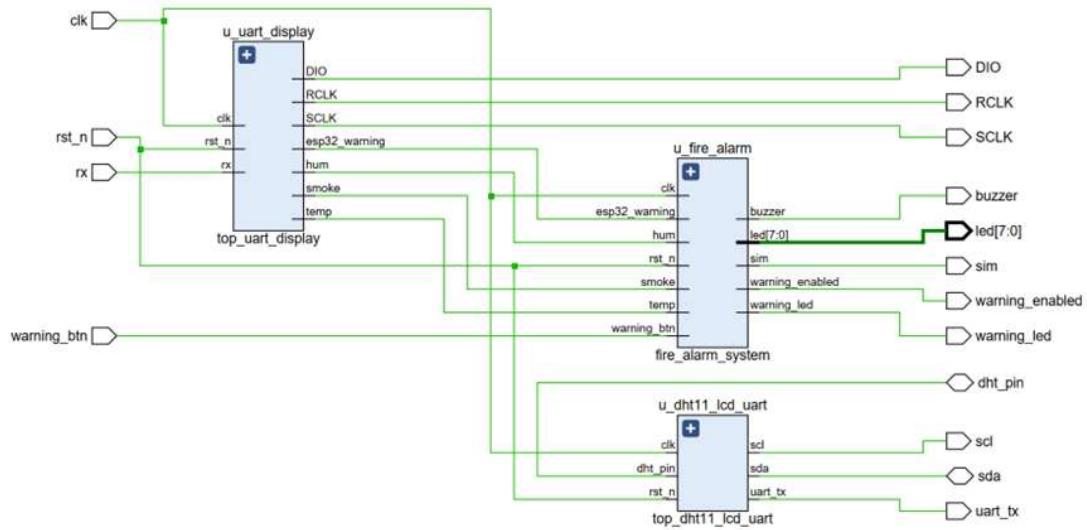


Hình 2.23: Hình ảnh phần mềm thingsboard

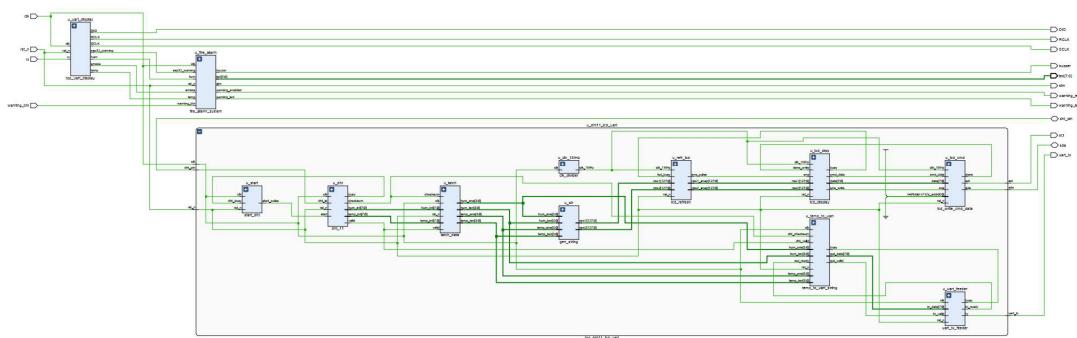
2.2.2 Vivado2025.1:



Hình 2.24: Hình ảnh Vivado2025.1

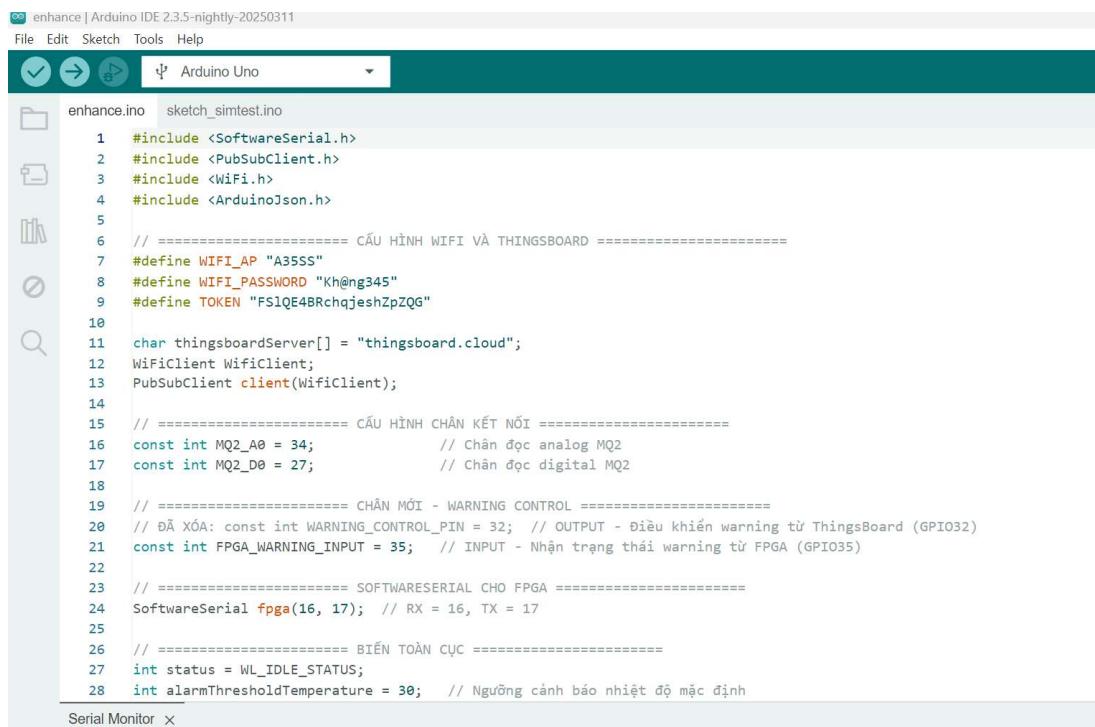


Hình 2.25: Sơ đồ hệ thống thiết kế



Hình 2.26: Sơ đồ luồng DHT11

2.2.3 Arduino:



The screenshot shows the Arduino IDE interface with the title bar "enhance | Arduino IDE 2.3.5-nightly-20250311". The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar has icons for save, upload, and refresh. The sketch name is "Arduino Uno". The code editor displays "enhance.ino" with the following content:

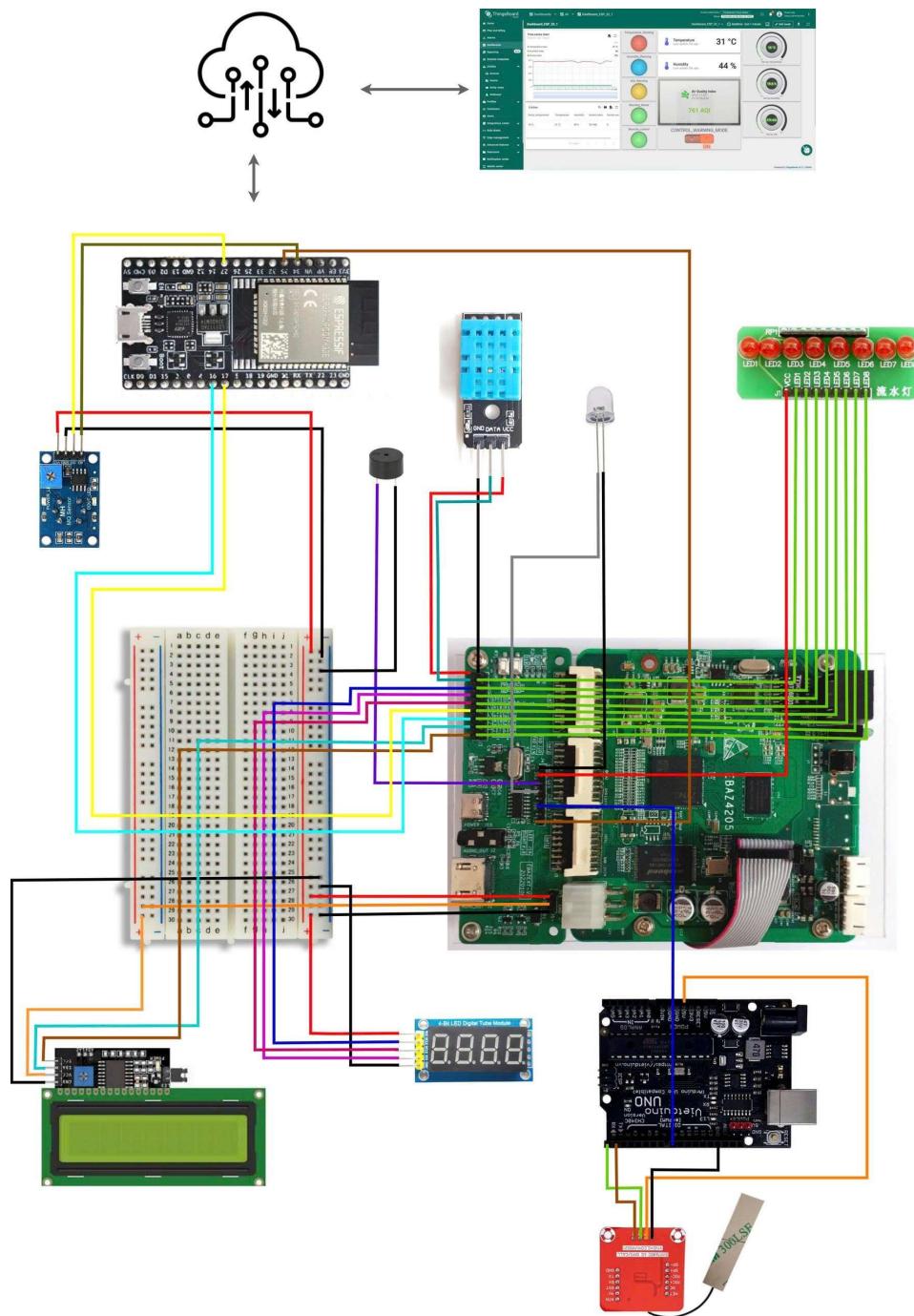
```
enhance.ino sketch_simtest.ino
1 #include <SoftwareSerial.h>
2 #include <PubSubClient.h>
3 #include <WiFi.h>
4 #include <ArduinoJson.h>
5
6 // ===== CẤU HÌNH WIFI VÀ THINGSBOARD =====
7 #define WIFI_AP "A35SS"
8 #define WIFI_PASSWORD "Kh@ng345"
9 #define TOKEN "FS1QE4BRchqjeshZpZQG"
10
11 char thingsboardServer[] = "thingsboard.cloud";
12 WiFiClient wifiClient;
13 PubSubClient client(wifiClient);
14
15 // ===== CẤU HÌNH CHÂN KẾT NỐI =====
16 const int MQ2_A0 = 34;           // Chân đọc analog MQ2
17 const int MQ2_D0 = 27;          // Chân đọc digital MQ2
18
19 // ===== CHÂN MỚI - WARNING CONTROL =====
20 // ĐÃ XÓA: const int WARNING_CONTROL_PIN = 32; // OUTPUT - Điều khiển warning từ ThingsBoard (GPIO32)
21 const int FPGA_WARNING_INPUT = 35; // INPUT - Nhận trạng thái warning từ FPGA (GPIO35)
22
23 // ===== SOFTWARESERIAL CHO FPGA =====
24 SoftwareSerial fpga(16, 17); // RX = 16, TX = 17
25
26 // ===== BIẾN TOÀN CỤC =====
27 int status = WL_IDLE_STATUS;
28 int alarmThresholdTemperature = 30; // Ngưỡng cảnh báo nhiệt độ mặc định
```

The bottom of the code editor shows "Serial Monitor X".

Hình 2.27: Hình ảnh lập trình Arduino

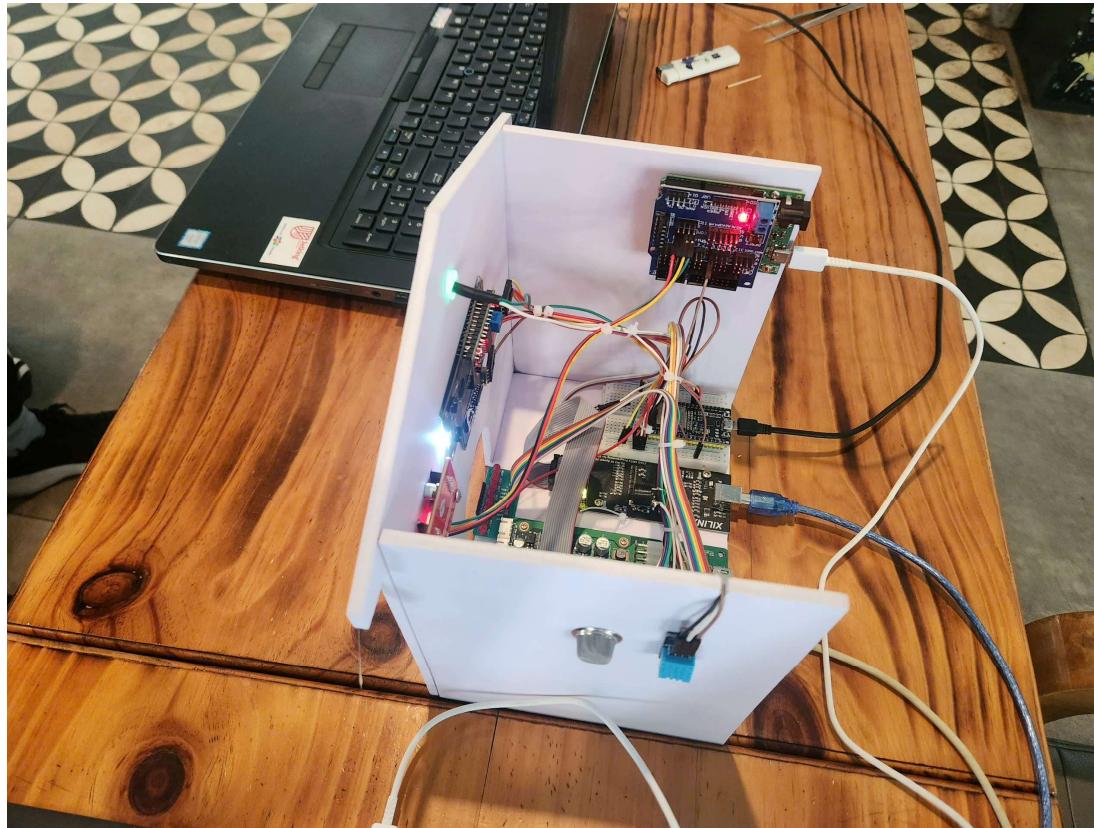
PHẦN 3: QUÁ TRÌNH THIẾT KẾ

3.1 Sơ đồ kết nối:



Hình 3.1: Sơ đồ kết nối hệ thống

3.2 Lắp đặt mô hình:



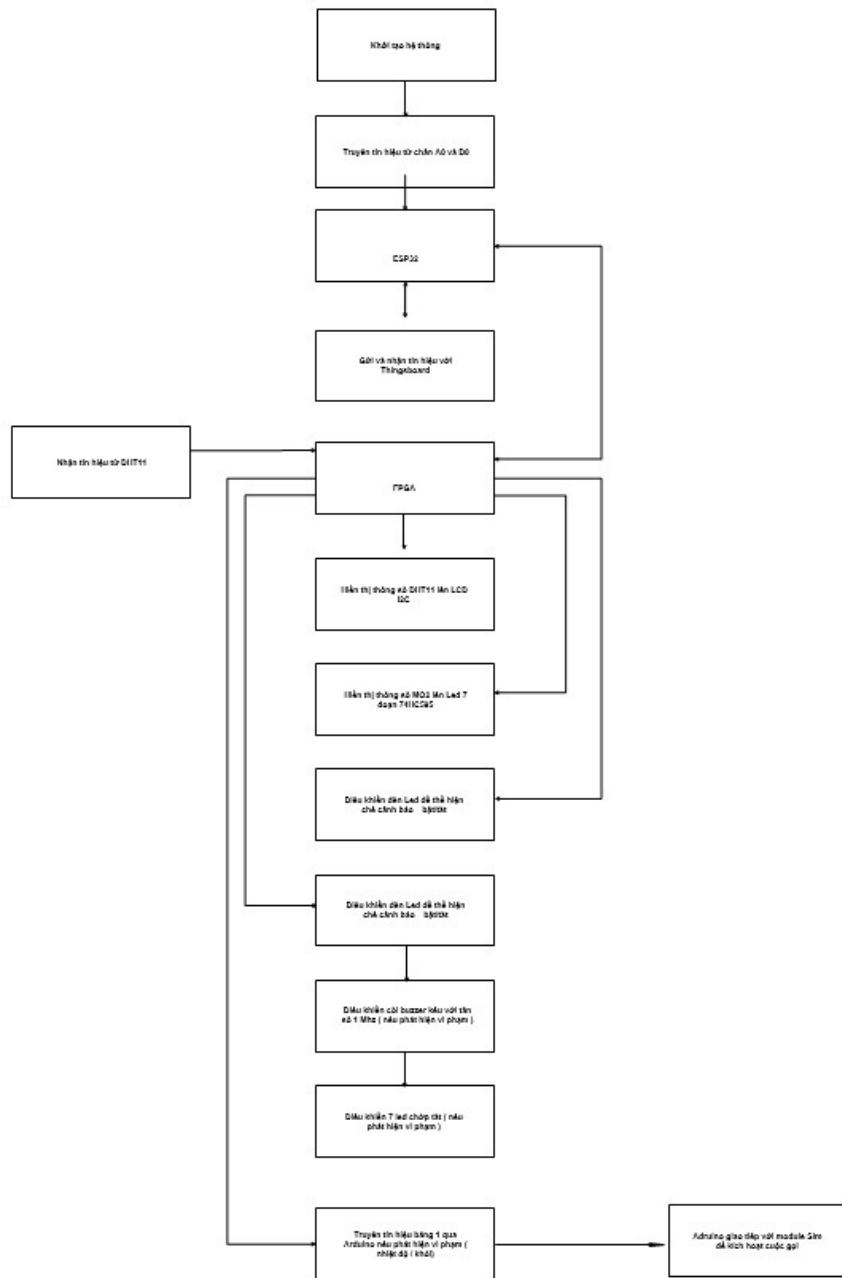
Hình 3.2: Quá trình lắp đặt mô hình



Hình 3.3: Mô hình hoàn thiện

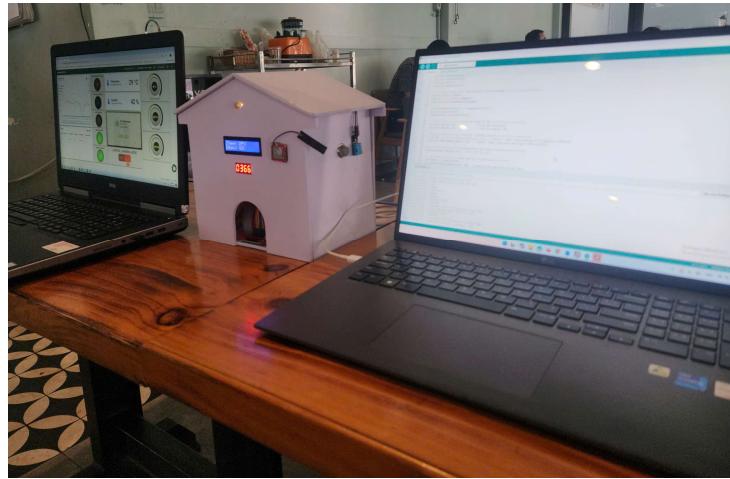
3.3 Flowchart và code:

3.2.1 Flowchart:



Hình 3.4: Flowchart

3.2.2 Kết quả thực nghiệm:



Hình 3.5: Thực nghiệm

PHẦN 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1 Kết luận:

- Năm bắt kiến trúc IoT, giao tiếp UART, và tích hợp phần cứng/phần mềm.
- Hiểu được nguyên lý của các giao thức I2C, UART.

4.2 Hướng phát triển:

- Cải thiện lý thuyết IoT: Nghiên cứu sâu hơn về các giao thức.
- Khả năng tích hợp thêm các module phần cứng để mở rộng ứng dụng trong thực tế.
- Ứng dụng thingsboard: Xây dựng app thực tế để hiển thị thêm các dữ liệu cuộc gọi để lưu trữ và truy xuất khi cần thiết.

PHẦN 6: PHÂN CÔNG CÔNG VIỆC NHÓM

Họ và tên	Nhiệm vụ phụ trách
Dương Minh An	Code, trình bày file word báo cáo
Nguyễn Thanh Hào	Code, trình bày file PP báo cáo

PHẦN 7: TÀI LIỆU THAM KHẢO

- [1] <https://github.com/>
- [2] <https://digilent.com/>
- [3] <https://www.alldatasheet.com/>