# LỜI CẢM ƠN

Lời đầu tiên cho em xin trân thành cảm ơn ThS. Trần Xuân Phương, người đã hết lòng chỉ dẫn, truyền đạt những kiến thức chuyên môn cũng như những kinh nghiệm liên quan cho chúng em trong suốt quá trình thực hiện đồ án này.

Xin chân thành cảm ơn đến tất cả quý thầy, cô nhà trường nói chung và các thầy cô bộ môn khoa điện tử nói riêng của trường Đại Học Công Nghiệp Hà Nội đã giảng dạy, trang bị cho em những kiến thức rất bổ ích và quý báu trong suốt quá trình học tập để em có thể áp dụng nghiên cứu hoàn thành đề tài này.

Trong quá trình thực hiện đồ án, với điều kiện thời gian cũng như kinh nghiệm, kiến thức còn hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, em rất mong nhận được những ý kiến đóng góp của thầy, cô và các bạn để em có thêm được nhiều kinh nghiệm vào công việc trong tương lai.

Cuối cùng, chúng em xin chân thành cảm ơn gia đình và bạn bè, đã luôn tạo điều kiện, quan tâm, giúp đỡ, động viên chúng em trong suốt quá trình học tập và hoàn thành đồ án này.

Em xin chân thành cảm ơn!

Hà Nội, ngày…tháng…năm 2022.

Sinh viên thực hiện đề tài

Bùi Thị Hồng

# MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc102901597)

[MỤC LỤC 2](#_Toc102901598)

[DANH MỤC HÌNH ẢNH 4](#_Toc102901599)

[DANH MỤC BẢNG BIỂU 5](#_Toc102901600)

[LỜI MỞ ĐẦU 6](#_Toc102901601)

[CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI 7](#_Toc102901602)

[1.1 Lý do lựa chọn đề tài 7](#_Toc102901603)

[1.2 Đặt vấn đề nghiên cứu 7](#_Toc102901604)

[1.2.1 Tính cấp thiết của đề tài 7](#_Toc102901605)

[1.2.2 Đối tượng và phạm vi nghiên cứu 8](#_Toc102901606)

[1.2.3 Một số mô hình đã có trên thị trường 8](#_Toc102901607)

[1.3 Ý nghĩa khoa học và thực tiễn của đề tài 13](#_Toc102901608)

[1.4 Kết luận chương 1 13](#_Toc102901609)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 14](#_Toc102901610)

[2.1 Các giao tiếp được sử dụng 14](#_Toc102901611)

[2.1.1 Giao tiếp I2C 14](#_Toc102901612)

[2.1.2 Giao tiếp One – Wire 19](#_Toc102901613)

[2.2 Tổng quan về ESP32 23](#_Toc102901614)

[2.3 Cảm biến nhiệt độ độ ẩm 25](#_Toc102901615)

[2.3.1 Cảm biến nhiệt độ độ ẩm DHT21 25](#_Toc102901616)

[2.4 Màn hình LCD 26](#_Toc102901617)

[2.4.1 Màn hình LCD 16x2 26](#_Toc102901618)

[2.4.2 Module chuyển I2C LCD 16x2 27](#_Toc102901619)

[2.5 IC ổn áp LM7805 27](#_Toc102901620)

[2.6 WiFi và Internet 27](#_Toc102901621)

[2.7 Tổng quan về Webserver 27](#_Toc102901622)

[2.7.1 Tạo giao diện Webserver 27](#_Toc102901623)

[2.7.2 Giao thức HTTP 27](#_Toc102901624)

[2.7.3 Xử lý dữ liệu Webserver 28](#_Toc102901625)

[2.8 Các phần mềm được sử dụng 28](#_Toc102901626)

[2.8.1 Phần mềm Arduino IDE 28](#_Toc102901627)

[2.8.2 Phần mềm Visual Studio Code 31](#_Toc102901628)

[2.8.3 Phần mềm Altium Designer 31](#_Toc102901629)

[2.9 Kết luận chương 2 33](#_Toc102901630)

[CHƯƠNG 3: THIẾT KẾ HỆ THỐNG 34](#_Toc102901631)

[3.1 Sơ đồ khối của hệ thống 34](#_Toc102901632)

[3.1.1 Yêu cầu 34](#_Toc102901633)

[3.1.2 Sơ đồ khối 34](#_Toc102901634)

[3.2 Sơ đồ nguyên lý của mạch 35](#_Toc102901635)

[3.2.1 Khối xử lý trung tâm 36](#_Toc102901636)

[3.2.2 Khối nguồn và ổn áp nguồn 36](#_Toc102901637)

[3.2.3 Khối kết nối với ngoại vi 37](#_Toc102901638)

[3.3 Sơ đồ mạch in (PCB) 37](#_Toc102901639)

[3.4 Lập trình vi xử lý ESP32 39](#_Toc102901640)

[3.4.1 Lưu đồ thuật toán 39](#_Toc102901641)

[3.4.2 Các thư viện được sử dụng 39](#_Toc102901642)

[3.4.3 Lập trình Wifi ở chế độ Access Point 40](#_Toc102901643)

[3.5 Lập trình đọc nhiệt độ độ ẩm DHT21 40](#_Toc102901644)

[3.5.1 Lập trình LCD16x2 với giao tiếp I2C 41](#_Toc102901645)

[3.5.2 Lập trình tương tác với giao diện Webserver 42](#_Toc102901646)

[3.6 Lập trình Webserver 43](#_Toc102901647)

[3.6.1 Xây dựng giao diện Web 43](#_Toc102901648)

[3.6.2 Lập trình tương tác Webserver với ESP32 44](#_Toc102901649)

[3.7 Kết luận chương 3 45](#_Toc102901650)

[CHƯƠNG 4: KẾT QUẢ ĐẠT ĐƯỢC 46](#_Toc102901651)

[4.1 Một số hình ảnh thực tế của hệ thống 46](#_Toc102901652)

[4.2 Giao diện điều khiển trên Webserver 46](#_Toc102901653)

[4.3 Đánh giá sự ổn định của hệ thống 46](#_Toc102901654)

[4.4 Kết luận chương 4 46](#_Toc102901655)

[KẾT LUẬN 47](#_Toc102901656)

[TÀI LIỆU THAM KHẢO 49](#_Toc102901657)

[PHỤ LỤC 50](#_Toc102901658)

# DANH MỤC HÌNH ẢNH

# DANH MỤC BẢNG BIỂU

# LỜI MỞ ĐẦU

Với sự phát triển của khoa học và công nghệ cùng với sự bùng nổ của cuộc cách mạng công nghiệp lần thứ tư đang diễn ra, các công nghệ hiện đại đang được phát triển và được ứng dụng trong đời sống rất nhiều. Đặc biệt, với sự phát triển của Internet, công nghệ Internet vạn vật (IoT) đang được phát triển một cách nhanh chóng. IoT cho phép các tổ chức có thể kết nối, kiểm soát và sử dụng các thiết bị IoT. Trong hệ sinh thái này, một tổ chức có thể sử dụng các thiết bị như điện thoại thông minh, máy tính bản, … để gửi đi các hiệu lệnh, hoặc truy cập thông tin từ một mạng lưới các thiết bị IoT khác.

Ngoài ra, trong những năm gần đây, công nghệ vi điện tử phát triển rất mạnh mẽ. Sự ra đời của các vi mạch cỡ lớn, cực lớn với giá thành giảm nhanh, khả năng lập trình ngày càng cao đã mang lại những thay đổi sâu sắc trong ngành kĩ thuật điện tử. Nền công nghiệp thế giới đã đạt được những thành tựu to lớn nhờ ứng dụng những tiến bộ của khoa học kĩ thuật và công nghệ, máy móc đã thay thế con người trong nhiều hoạt động lao động sản xuất.

Sự kết hợp giữa vi điều khiển và công nghệ IoT giúp cho máy móc trở nên thông minh hơn, có nhiều tính năng đa dạng và giải quyết được nhiều vấn đề trong thực tế. Để tìm hiểu về IoT và kết hợp được các kiến thức đã học về vi điều khiển, vi xử lý, các môn học về điều khiển và giám sát không dây, em đã quyết định chọn đề tài: “**Thiết kế hệ thống ứng dụng IoT vào đo nhiệt độ độ ẩm sử dụng ESP32**” là đề tài đồ án tốt nghiệp.

Nội dung quyển báo cáo này bao gồm 3 chương chính:

Chương 1 - Tổng quan về đề tài

Chương 2 - Cơ sở lý thuyết

Chương 3 - Thiết kế hệ thống

Chương 4 - Kết quả đạt được

# TỔNG QUAN VỀ ĐỀ TÀI

* 1. Lý do lựa chọn đề tài

Trong ứng dụng hàng ngày, nhu cầu theo dõi nhiệt độ và độ ẩm ngày càng trở nên phổ biến và thiết thực. Theo dõi và giám sát nhiệt độ và độ ẩm ngày càng trở nên phổ biến và thiết thực. Hệ thống đo nhiệt độ và độ ẩm thường được sử dụng trong: Sản xuất chế biến nông nghiệp, hiển thị và thực thi (Quạt gió, máy sấy, điều hòa…) Lưu trữ và kiểm soát dữ liệu về môi trường tại một khu vực…

Ngày nay, trong thời đại công nghiệp 4.0 ngày càng phát triển, các sản phẩm mang tính thông minh ra đời và mang lại hiệu quả cao đang được chú trọng phát triển. Vì vậy, việc đo và giám sát các thông số như nhiệt độ, độ ẩm kết hợp với các ứng dụng IoT giám sát từ xa đang trở nên phổ biến mà hầu hết các thiết bị hiện nay đều có tích hợp

* 1. Đặt vấn đề nghiên cứu
     1. Tính cấp thiết của đề tài

IoT đang là công nghệ mới và là chủ đề được mọi người quan tâm để ứng dụng vào cuộc sống. Đây là một công nghệ mới và phát triển song song với sự phát triển của Internet và các thiết bị như Smartphone, Table, Laptop …

Việc giám sát nhiệt độ và độ ẩm không có sự kết hợp của IoT gây khó khăn trong việc tổng hợp và giám sát từ xa. Hơn nữa việc thống kê nhiệt độ và độ ẩm gây mất thời gian và hiệu quả không cao.

Trong thời đại công nghẹ 4.0 với sự bùng nổ của Internet, song song đó là các loại vi xử lý được tích hợp ngay WiFI và Internet giúp việc trao đổi dữ liệu giữa 2 bên ngày càng đơn giản. Nhận thấy sự kết hợp mang tính thông minh này, hệ thống ứng dụng IoT vào đo nhiệt độ độ ẩm sử dụng ESP32 giúp giám sát nhiệt độ và độ ẩm một cách thuận tiện và dễ dàng với Internet và các thiết bị có kết nối Internet

* + 1. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu đề tài.

* Vi điều khiển ESP32
* Lập trình vi điều khiển ESP32 trên Arduino IDE
* Lập trình tạo giao diện Webserver và tương tác Webserver với ESP32
* Cảm biến đo nhiệt độ và độ ẩm DHT21

Phạm vi nghiên cứu.

* Đề tài tập trung về vấn đề đo nhiệt độ và độ ẩm của môi trường
* Hiển thị giá trị nhiệt độ và độ ẩm lên LCD và giao diện Webserver
  + 1. Một số mô hình đã có trên thị trường

*Xiaomi Mijia Smart Home và Xiaomi Home Kit*

Cảm biến Xiaomi Mijia xác định độ ẩm với độ chính xác cao. Cảm biến có thể theo dõi dữ liệu của nhà / văn phòng bằng điện thoại.



Nó cũng có thể liên kết với các sản phẩm khác của Xiaomi giúp chúng hoạt động khi đạt đến nhiệt độ hoặc độ ẩm nhất định giúp không khí căn hộ luôn cân bằng và thông minh



Hệ thống có chức năng theo dõi nhiệt độ và độ ẩm thời gian thực. Tự động báo động nếu nhiệt độ và độ ẩm bất thường. Dễ dàng xem sự thay đổi nhiệt độ và độ ẩm theo thời gian được lưu lại trong nhật ký. Độ chính xác đo nhiệt độ là ± 0,3 ° C và độ chính xác đo độ ẩm: ± 3%

*Máy đo ghi nhiệt độ, độ ẩm không dây Wifi Tuya SHP-V3*

Máy đo ghi nhiệt độ, độ ẩm không dây **Wifi Tuya SHP-LB3v3** được thiết kế màn hình to, rõ ràng, hiển thị thông số nhiệt độ độ ẩm sắc nét, cực kỳ phù hợp cho các ứng dụng theo dõi và giám sát nhiệt độ, độ ẩm tại **kho lạnh**, kho bảo quản thực phẩm tươi sống, kho chứa dược phẩm, tủ bảo quản mẫu, **giám sát nhiệt độ phòng máy chủ**, phòng server, siêu thị…

Cảm biến với đầu dò nhiệt độ độ ẩm không khí cực nhạy được nối dài, liên tục ghi lại sự thay đổi nhiệt độ độ ẩm từ môi trường và so sánh với mức nhiệt ẩm bạn đã cài đặt từ trước khi giá trị vượt quá giới hạn cho phép thiết bị sẽ tự động kêu loa báo động đồng thời cảnh báo về điện thoại qua App Tuya Smart ở bất cứ đâu có wifi, 3G.

Giám sát nhiệt độ độ ẩm cho phòng server, datacenter hay trung tâm dữ liệu vì đây chính là nơi tập trung nhiều thiết bị điện tử hoạt động, tỏa nhiệt rất nhiều, nguy cơ cháy nổ rất cao. Chính vì thế việc giám sát nhiệt độ cũng như độ ẩm của phòng máy chủ cực kỳ quan trọng đảm bảo phòng máy có thể hoạt động ổn định và liên tục hạn chế sự cố do nhiệt độ và độ ẩm cao gây ra.

Sử dụng trong các hệ thống bảo quản thuốc, bảo quản và vận chuyển vắc-xin, hóa chất, dùng trong lĩnh vực công nghệ sinh học, vườn ươm, tạo giống, phòng thí nghiệm…

* Hệ thống giám sát nhiệt độ kho lạnh hiển thị thông số nhiệt độ độ ẩm bằng màn hình led 7 đoạn.
* Cảnh báo tại chỗ khi nhiệt độ độ ẩm không đáp ứng đủ.
* Cảm biến độ chính xác cao cập nhật trạng thái liên tục.

*Thiết Bị Giám Sát Chất Lượng Không Khí Wifi Tuya Treo Tường Air Quality Monitor*

Thiết Bị Giám Sát Chất Lượng Không Khí Wifi Tuya Treo Tường Air Quality Monitor có tới 3 cảm biến là cảm biến CO2, cảm biến nhiệt độ và cảm biến độ ẩm. Thiết bị liên tục ghi lại sự thay đổi về các chỉ số Co2, nhiệt độ độ ẩm từ môi trường cho phép xem trên điện thoại qua App Tuya Smart ở bất cứ đâu có wifi, 3G.

Giám sát chất lượng không khí, nhiệt độ độ ẩm cho văn phòng, trong nhà hay bệnh viện vì đây chính là những nơi mọi người làm việc và sinh hoạt nhiều nên cần quan sát đảm bảo chất lượng không khí để không ảnh hưởng đến vấn đề sức khỏe.

Giám sát các chỉ số cơ bản nhất của không khí với độ chính xác cao. Màn hình lớn hiển thị màu sắc dễ dàng để đọc các kết quả kiểm tra, thuận tiện hơn để sử dụng quan sát chất lượng không khí trong nhà. Dễ dàng để thực hiện, phù hợp cho nhà ở, xe hơi, văn phòng, nhà máy và môi trường khác.

Hiển thị chất lượng không khí để giúp mọi người kiểm soát tình hình sức khỏe tốt hơn. Nó giúp phát hiện các tác nhân có hại HCHO và TVOC xung quanh phòng của bạn trong thời gian thực.

* Hệ thống hiển thị bằng màn hình LCD .
* Sử dụng pin sạc chạy được sau khi mất điện.
* Cảm biến độ chính xác cao cập nhật trạng thái liên tục.
* Phối hợp ngữ cảnh linh hoạt với các thiết bị Tuya khác.
* Dễ dàng chia sẻ cho nhiều điện thoại khác cùng sử dụng thiết bị.

Một số thông số kỹ thuật chính:

* Pin: pin sạc lithium polymer 1200 Mah, cổng Micro USB.
* Màn hình hiển thị LCD có màu.
* Phạm vi đo không khí: CO2: 0-5000ppm
* Thời gian lấy mẫu: 1.5s
* 3 cảm biến: CO2 + Nhiệt Độ + Độ Ẩm
* Khoảng đo nhiệt độ: -10°C - +50°C.
* Lưu trữ nhiệt độ: -10°C - +50°C
* Khoảng đo độ ẩm: 20% - +60%
* Độ chính xác độ ẩm nhiệt độ: 0,5°C, 1%RH.
* Sai số: ± 1°C, ± 3%.
* App Tuya Smart, Smart Life tương thích hệ điều hành IOS và Android.
* Kích thước: 270\*90\*35mm
* Trọng lượng: 700g.
  1. Ý nghĩa khoa học và thực tiễn của đề tài

Đề tài : “**Thiết kế hệ thống ứng dụng IoT vào đo nhiệt độ độ ẩm sử dụng ESP32**” tuy chỉ ngừng tại ở phạm vi nhỏ là đo nhiệt độ độ ẩm, hiển thị nhiệt độ độ ẩm nhưng đề tài có ý nghĩa khá quan trọng trong việc tìm hiểu về IoT, tìm hiểu về giao tiếp giữa cảm biến với vi điều khiển, giữa vi điều khiển với Webserver .

Việc đo nhiệt độ và độ ẩm, thống kê về dữ liệu nhiệt độ và độ ẩm giúp người dùng có cái nhìn trực quan hơn về nhiệt độ trong các phòng cần yêu cầu cao về môi trường làm việc.

Đề tài có nhiều ứng dụng trong thực tế, sinh viên vận dụng những kiến thức tiếp thu được trong quá trình đào tạo tại khoa điện tử và các kiến thức tìm hiểu được bằng cách tự học hỏi.

Đề tài không chỉ dừng lại ở phạm vi nghiên cứu nhất định mà còn có thể mở rộng ra theo rất nhiều hướng khác nhau như: Smarthome, các hệ thống nhà kính, hệ thống tự động …

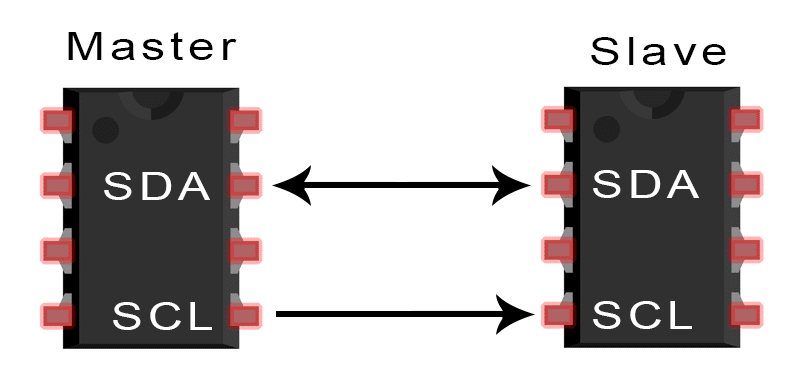
* 1. Kết luận chương 1

Qua tìm hiểu về một số loại mô hình về đo và giám sát nhiệt độ và độ ẩm, em thấy các sản phẩm đều có khuynh hướng đi tới việc thông minh các thiết bị, các thiết bị đều kết nối Internet và giám sát từ xa, đều có kết hợp IoT trong các sản phẩm. Các sản phẩm ngày càng trở nên hiện đại hơn và dễ dàng sử dụng hơn.

Để bắt kịp xu thế của công nghệ, đặc biệt là tổng hợp lại kiến thức đã học ở các môn, hơn nữa mục đích của đề tài giúp việc tìm hiểu và phát triển về vi điều khiển ESP32 - một vi điều khiển giúp xây dựng các ứng dụng về IoT trở nên dễ dàng và nhanh chóng, tạo ra sản phẩm công nghệ cao, có tính ứng dụng cao, có tính mới.

# CƠ SỞ LÝ THUYẾT

* 1. Các giao tiếp được sử dụng
     1. Giao tiếp I2C



*Giới thiệu về giao tiếp I2C*

Giao tiếp I2C là sự kết hợp các tính năng tốt nhất của UART và SPI, trong giao tiếp I2C, một thiết bị có thể đóng vai trò là Master (chủ) kết nối tới nhiều Slave (tớ), hoặc có thể nhiều thiết bị đóng vai trò là Master kết nối tới một hoặc nhiều thiết bị là Slave.

I2C chỉ sử dụng 2 dây để kết nối các thiết bị với nhau:

* SDA (tín hiệu Data) - Master và Slave gửi và nhận dữ liệu.
* SCL (xung Clock) - Tín hiệu xung Clock

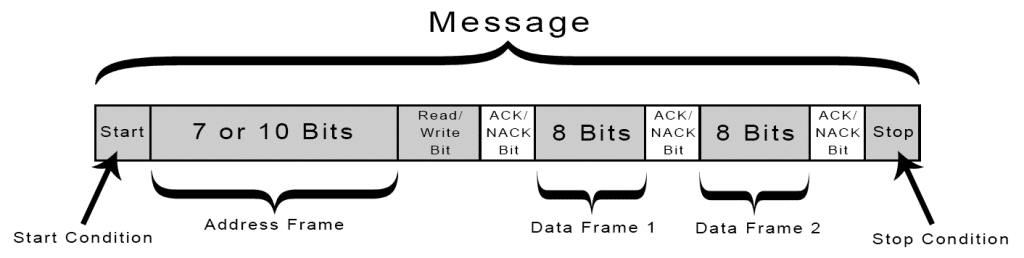
I2C là một giao thức giao tiếp nối tiếp, vì vậy dữ liệu được truyền từng bit theo một dây duy nhất (dây SDA)

Giống như SPI, I2C sử dụng cơ chế đồng bộ do đó, đầu ra của bit được đồng bộ hóa với việc lấy mẫu các bit bằng tín hiệu Clock được chia sẻ giữa Master và Slave. Tín hiệu Clock luôn được kiểm soát bởi Master.

|  |  |
| --- | --- |
| Số lượng dây sử dụng | 2 |
| Tốc độ truyền tối đa | Chế độ tiêu chuẩn = 100 kbps  Chế độ nhanh = 400 kbps  Chế độ tốc độ cao = 3.4 Mbps  Chế độ tốc độ rất cao = 5 Mbps |
| Loại chế độ truyền | Đồng bộ |
| Phương thức truyền | Nối tiếp |
| Số lượng Master tối đa | Không giới hạn về số lượng |
| Số lượng Slave tối đa | 1008 |

*Cách hoạt động của I2C*

Với I2C, dữ liệu được truyền trong các tin nhắn. Tin nhắn được chia thành các khung dữ liệu. Mỗi tin nhắn có một khung địa chỉ chứa địa chỉ nhị phân của địa chỉ slave và một hoặc nhiều khung dữ liệu chứa dữ liệu đang được truyền. Thông điệp cũng bao gồm điều kiện khởi động và điều kiện dừng, các bit đọc / ghi và các bit ACK / NACK giữa mỗi khung dữ liệu:



* Điều kiện khởi động: Đường SDA chuyển từ mức điện áp cao xuống mức điện áp thấp trước khi đường SCL chuyển từ mức cao xuống mức thấp.
* Điều kiện dừng: Đường SDA chuyển từ mức điện áp thấp sang mức điện áp cao sau khi đường SCL chuyển từ mức thấp lên mức cao.
* Khung địa chỉ: Một chuỗi 7 hoặc 10 bit duy nhất cho mỗi slave để xác định slave khi master muốn giao tiếp với nó.
* Bit đọc / ghi: Một bit duy nhất chỉ định master đang gửi dữ liệu đến slave (mức điện áp thấp) hay yêu cầu dữ liệu từ nó (mức điện áp cao).
* Bit ACK / NACK:Mỗi khung trong một tin nhắn được theo sau bởi một bit xác nhận / không xác nhận. Nếu một khung địa chỉ hoặc khung dữ liệu được nhận thành công, một bit ACK sẽ được trả lại cho thiết bị gửi từ thiết bị nhận.

*Địa chỉ*

I2C không có các đường chọn Slave như SPI, vì vậy cần một cách khác để cho slave biết rằng dữ liệu đang được gửi đến slave này chứ không phải slave khác. Nó thực hiện điều này bằng cách định địa chỉ. Khung địa chỉ luôn là khung đầu tiên sau bit khởi động trong một tin nhắn mới.

Master gửi địa chỉ của slave mà nó muốn giao tiếp với mọi slave được kết nối với nó. Sau đó, mỗi slave sẽ so sánh địa chỉ được gửi từ master với địa chỉ của chính nó. Nếu địa chỉ phù hợp, nó sẽ gửi lại một bit ACK điện áp thấp cho master. Nếu địa chỉ không khớp, slave không làm gì cả và đường SDA vẫn ở mức cao.

*Bit đọc / ghi*

Khung địa chỉ bao gồm một bit duy nhất ở cuối tin nhắn cho slave biết master muốn ghi dữ liệu vào nó hay nhận dữ liệu từ nó. Nếu master muốn gửi dữ liệu đến slave, bit đọc / ghi ở mức điện áp thấp. Nếu master đang yêu cầu dữ liệu từ slave, thì bit ở mức điện áp cao.

*Khung dữ liệu*

Sau khi Master phát hiện bit ACK từ Slave, khung dữ liệu đầu tiên đã sẵn sàng được gửi.

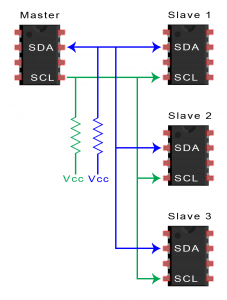
Khung dữ liệu luôn có độ dài 8 bit và được gửi với bit quan trọng nhất trước. Mỗi khung dữ liệu ngay sau đó là một bit ACK / NACK để xác minh rằng khung đã được nhận thành công. Bit ACK phải được nhận bởi Master hoặc Slave (tùy thuộc vào cái nào đang gửi dữ liệu) trước khi khung dữ liệu tiếp theo có thể được gửi.

Sau khi tất cả các khung dữ liệu đã được gửi, Master có thể gửi một điều kiện dừng cho slave để tạm dừng quá trình truyền. Điều kiện dừng là sự chuyển đổi điện áp từ thấp lên cao trên đường SDA sau khi chuyển tiếp từ thấp lên cao trên đường SCL , với đường SCL vẫn ở mức cao.

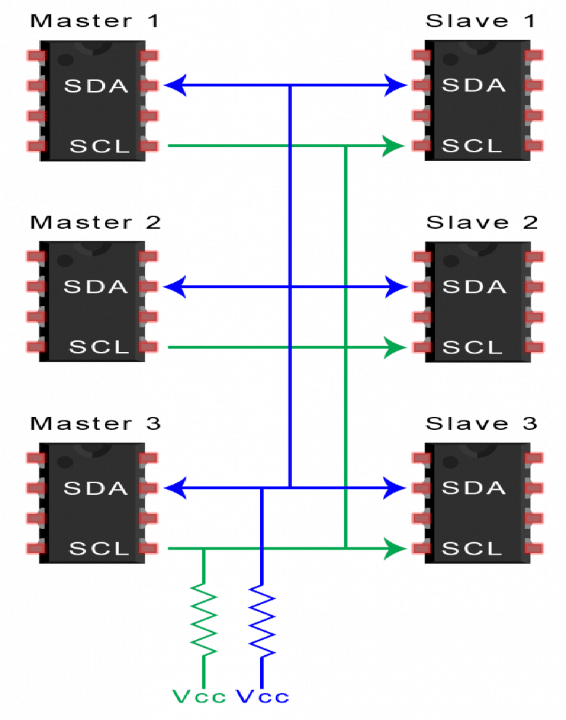
*Các bước truyền dữ liệu I2C*

1. Master gửi điều kiện khởi động đến mọi slave được kết nối bằng cách chuyển đường SDA từ mức điện áp cao sang mức điện áp thấp trước khi chuyển đường SCL từ mức cao xuống mức thấp.
2. Master gửi cho mỗi slave địa chỉ 7 hoặc 10 bit của slave mà nó muốn giao tiếp, cùng với bit đọc / ghi.
3. Mỗi slave sẽ so sánh địa chỉ được gửi từ master với địa chỉ của chính nó. Nếu địa chỉ trùng khớp, slave sẽ trả về một bit ACK bằng cách kéo dòng SDA xuống thấp cho một bit. Nếu địa chỉ từ master không khớp với địa chỉ của slave, slave rời khỏi đường SDA cao.
4. Master gửi hoặc nhận khung dữ liệu.
5. Sau khi mỗi khung dữ liệu được chuyển, thiết bị nhận trả về một bit ACK khác cho thiết bị gửi để xác nhận đã nhận thành công khung.
6. Để dừng truyền dữ liệu, master gửi điều kiện dừng đến slave bằng cách chuyển đổi mức cao SCL trước khi chuyển mức cao SDA.

**Một Master với nhiều Slave**

Vì I2C sử dụng định địa chỉ nên nhiều Slave có thể được điều khiển từ một master duy nhất. Với địa chỉ 7 bit sẽ có 128 (2 mũ 7) địa chỉ duy nhất. Việc sử dụng địa chỉ 10 bit không phổ biến, nhưng nó cung cấp 1.024 (2 mũ 10) địa chỉ duy nhất. Để kết nối nhiều slave đến một master duy nhất, bạn có thể đấu dây như thế này, với điện trở kéo lên 4,7K Ohm kết nối đường SDA và SCL với Vcc:

*Nhiều master với nhiều slave*

Nhiều master có thể được kết nối với một slave hoặc nhiều slave. Sự cố với nhiều master trong cùng một hệ thống xảy ra khi hai master cố gắng gửi hoặc nhận dữ liệu cùng một lúc qua đường SDA. Để giải quyết vấn đề này, mỗi master cần phải phát hiện xem đường SDA thấp hay cao trước khi truyền tin nhắn. Nếu đường SDA thấp, điều này có nghĩa là một master khác có quyền điều khiển bus và master đó phải đợi để gửi tin nhắn. Nếu đường SDA cao thì có thể truyền tin nhắn an toàn. Để kết nối nhiều master với nhiều slave, hãy sử dụng sơ đồ sau, với các điện trở kéo lên 4,7K Ohm kết nối các đường SDA và SCL với Vcc:

*Ưu điểm*

* Chỉ sử dụng hai dây
* Hỗ trợ nhiều Master và nhiều Slave
* Bit ACK / NACK xác nhận mỗi khung được chuyển thành công
* Phần cứng ít phức tạp hơn so với UART
* Giao thức nổi tiếng và được sử dụng rộng rãi

*Nhược điểm*

* Tốc độ truyền dữ liệu chậm hơn SPI
* Kích thước của khung dữ liệu bị giới hạn ở 8 bit
* Cần phần cứng phức tạp hơn để triển khai so với SPI
  + 1. Giao tiếp One – Wire

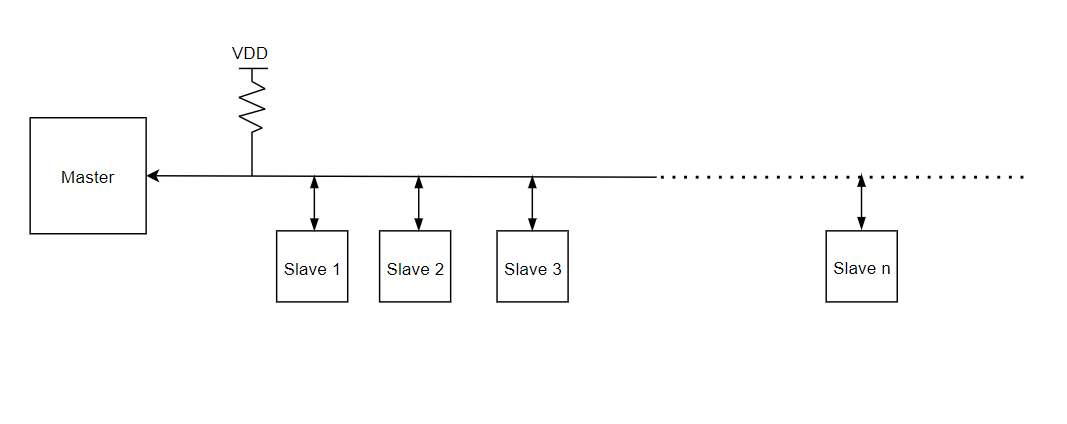
*Giới thiệu về giao tiếp One – Wire ( 1 – Wire)*

One - Wire (1-Wire) là hệ thống bus giao tiếp được thiết kế bởi Dallas Semiconductor Corp. Giống như tên gọi, hệ thống bus này chỉ sử dụng 1 dây để truyền nhận dữ liệu.

Chính vì chỉ sử dụng 1 dây nên giao tiếp này có tốc độ truyền thấp nhưng dữ liệu lại truyền được khoảng cách xa hơn.

OneWire chủ yếu sử dụng để giao tiếp với các thiết bị nhỏ, thu thập và truyền nhận dữ liệu thời tiết, nhiệt độ,… các công việc không yêu cầu tốc độ cao.

Giống như các chuẩn giao tiếp khác, 1-Wire cho phép truyền nhận dữ liệu với nhiều Slave trên đường truyền. Tuy nhiên chỉ có thể có 1 Master (giống với SPI).

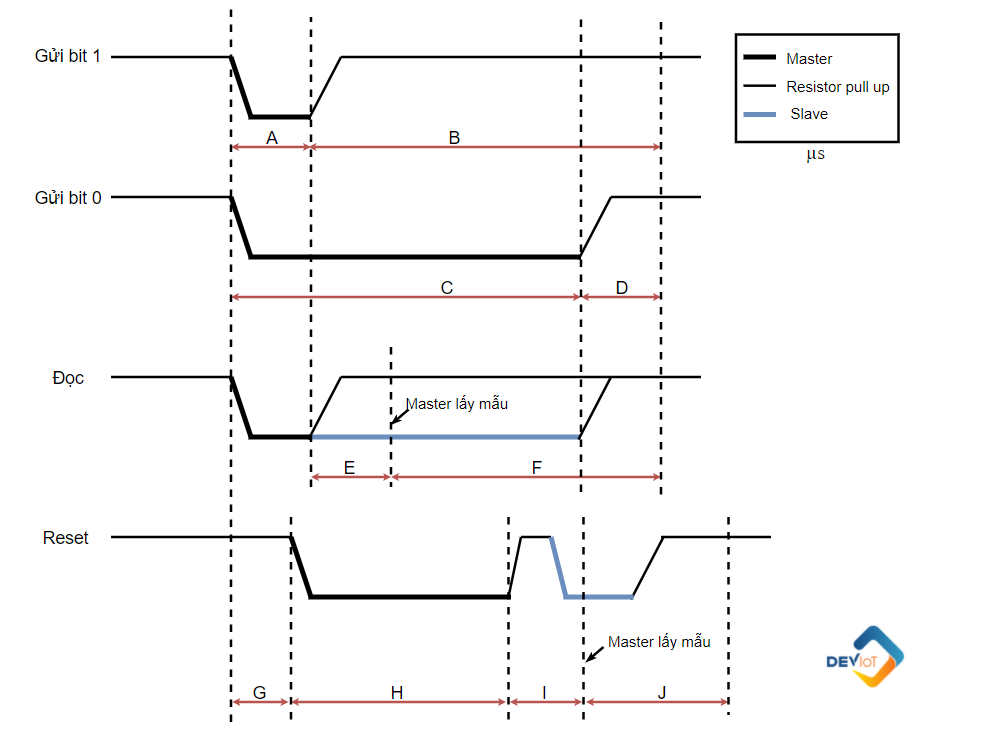


*Cách hoạt động của giao tiếp One – Wire*

So với các chuẩn giao tiếp cơ bản như UART, SPI, I2C mà chúng ta đã biết, cách thức hoạt động của OneWire có hơi “ lạ “ 1 chút.

Như chúng ta thấy ở hình…, đường dây luôn được giữ ở mức cao (High). Các thao tác hoạt động cơ bản của bus sẽ được quy định bởi thời gian kéo đường truyền xuống mức thấp (Low) như hình vẽ dưới.

Có 4 thao tác cơ bản như sau:



Giải thích ý nghĩa :

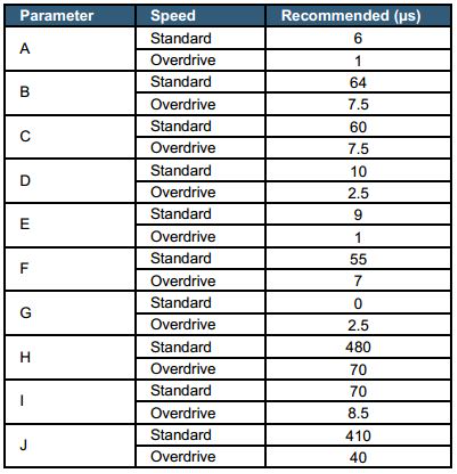
Gửi bit 1: Khi muốn gửi đi bit 1, thiết bị Master sẽ kéo bus xuống mức 0 trong một khoảng thời gian A (µs) và trở về mức 1 trong khoảng B (µs).

Gửi bit 0: Thiết bị Master kéo bus xuống mức 0 trong một khoảng thời gian C (µs) và trở về mức 1 trong khoảng D (µs).

Đọc bit: Thiết bị Master kéo bus xuống 1 khoảng A (µs). Trong khoảng thời gian E (µs) tiếp theo, thiết bị master sẽ tiến hành lấy mẫu. Có nghĩa trong E (µs) này, nếu bus ở mức 1, thiết bị master sẽ đọc bit 1. Ngược lại, nếu bus ở mức 0 thì master sẽ đọc được bit 0.

Reset: Thiết bị Master kéo bus xuống 1 khoảng thời gian H (µs) và sau đó về mức 1. Khoảng thời gian này gọi là tín hiệu reset. Trong khoảng thời gian I (µs) tiếp theo, thiết bị master tiến hành lấy mẫu. Nếu thiết bị slave gắn với bus gửi về tín hiệu 0, (tức bus ở mức 0), master sẽ hiểu rằng slave vẫn có mặt và quá trình trao đổi dữ liệu lại tiếp tục. Ngược lại nếu slave gửi về tin hiệu 1 ( bus ở mức 1) thì master hiểu rằng không có thiết bị slave nào tồn tại và dừng quá trình.

Bảng giá trị thời gian



*Chế độ hoạt động:*

Chế độ Standard (Chế độ tiêu chuẩn)

* 15.4 Kb/s
* 65 µs bit

Chế độ Overdrive (Chế độ tốc độ nhanh)

* 125 Kb/s
* 8 µs bit
  1. Tổng quan về ESP32

ESP32 là một trong các loại vi điều khiển trên thị trường có giá rẻ, năng lượng thấp có hỗ trợ WiFi và Bluetooth chế độ kép. Dòng ESP32 sử dụng bộ vi xử lý Tensilica Xtensa LX6 ở cả hai biến thể lõi kép và lõi đơn, và bao gồm các công tắc antenna tích hợp, RF balun, bộ khuếch đại công suất, bộ khuếch đại thu nhiễu thấp, bộ lọc và module quản lý năng lượng.

ESP32 được chế tạo và phát triển bởi Espressif Systems, một công ty Trung Quốc có trụ sở tại Thượng Hải, và được sản xuất bởi TSMC bằng cách sử dụng công nghệ 40 nm. ESP32 là sản phẩm kế thừa từ vi điều khiển ESP8266. Các tính năng của ESP32 bao gồm:

*Bộ xử lý:*

* CPU: Bộ vi xử lý Xtensa lõi kép (hoặc lõi đơn) 32-bit LX6, hoạt động ở tần số 240 MHz (160 MHz cho ESP32-S0WD và ESP32-U4WDH) và hoạt động ở tối đa 600 MIPS (200 MIPS với ESP32-S0WD/ESP32-U4WDH)
* Bộ đồng xử lý (co-processor) công suất cực thấp (Ultra low power, viết tắt: ULP)

*Hệ thống xung nhịp: CPU Clock, RTC Clock và Audio PLL Clock*

*Bộ nhớ nội:*

* 448 KB bộ nhớ ROM cho việc booting và các tính năng lõi
* 520 KB bộ nhớ SRAM trên chip cho dữ liệu và tập lệnh

*Kết nối không dây:*

* Wi-Fi: 802.11 b/g/n
* Bluetooth: v4.2 BR/EDR và BLE (chia sẻ sóng vô tuyến với Wi-Fi)

*34 GPIO vật lý với các ngoại vi*

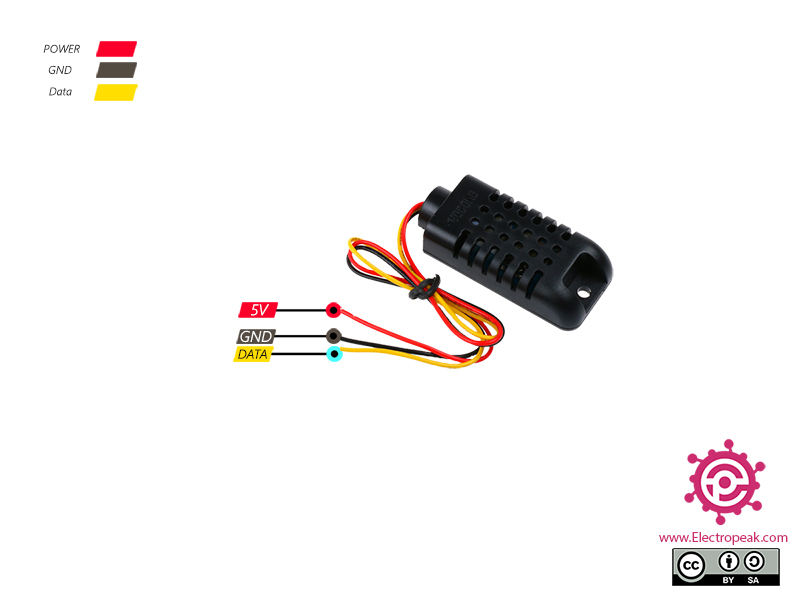
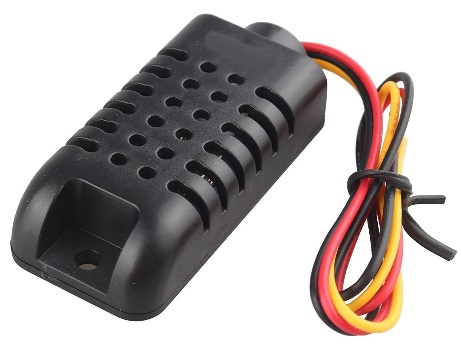
* ADC SAR 12 bit, 18 kênh
* DAC 2 × 8-bit
* Cảm biến cảm ứng (touch sensor) (GPIO cảm ứng điện dung)
* 3 SPI (SPI, HSPI và VSPI) hoạt động ở cả 2 chế độ master/slave. Module ESP32 hỗ trợ 4 ngoại vi SPI với SPI0 và SPI1 kết nối đến bộ nhớ flash của ESP32 còn SPI2 và SPI3 tương ứng với HSPI và VSPI.
* 2 I²S
* 2 I²C, hoạt động được ở cả chế độ master và slave, với chế độ Standard mode (100 Kbit/s) và Fast mode (400 Kbit/s). Hỗ trợ 2 chế độ định địa chỉ là 7-bit và 10-bit.Các GPIO đều có thể được dùng để triển khai I²C.
* 3 UART (UART0, UART1, UART2) với tốc độ lên đến 5 Mbps
* SD/SDIO/CE-ATA/MMC/eMMC host controller
* SDIO/SPI slave controller
* Ethernet MAC interface cho DMA và IEEE 1588 Precision Time Protocol (tạm dịch: Giao thức thời gian chính xác IEEE 1588)
* CAN bus 2.0
* Bộ điều khiển hồng ngoại từ xa (TX/RX, lên đến 8 kênh)
* PWM cho điều khiển động cơ
* LED PWM (lên đến 16 kênh)
* Cảm biến hiệu ứng Hall
* Bộ tiền khuếch đại analog công suất cực thấp (Ultra low power analog pre-amplifier)

*Bảo mật:*

* Hỗ trợ tất cả các tính năng bảo mật chuẩn IEEE 802.11, bao gồm WFA, WPA/WPA2 và WAPI.
* Secure boot (tạm dịch: khởi động an toàn)
* Mã hóa flash
* 1024-bit OTP, lên đến 768-bit cho khách hàng
* Tăng tốc mã hóa phần cứng: AES, SHA-2, RSA, elliptic curve cryptography (ECC, tạm dịch: mật mã đường cong ellip), trình tạo số ngẫu nhiên (random number generator)

*Quản lý năng lượng*

* Bộ ổn áp nội với điện áp rơi thấp (internal low-dropout regulator)
* Miền nguồn riêng (individual power domain) cho RTC
* Dòng 5 μA cho chế độ deep sleep
* Trở lại hoạt động từ ngắt GPIO, timer, đo ADC, ngắt với cảm ứng điện dung
  1. Cảm biến nhiệt độ độ ẩm
     1. Cảm biến nhiệt độ độ ẩm DHT21



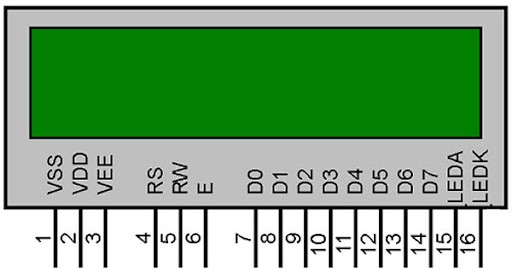
DHT-21 (AM2301) là cảm biến độ ẩm và nhiệt độ tương đối đầu ra kỹ thuật số. Nó sử dụng cảm biến độ ẩm điện dung và nhiệt điện trở để đo không khí xung quanh và phun ra tín hiệu kỹ thuật số trên chân dữ liệu

* 1. Màn hình LCD
     1. Màn hình LCD 16x2

*Giới thiệu*

LCD có rất nhiều ưu điểm so với các dạng hiển thị khác: Nó có khả năng hiển thị kí tự đa dạng, trực quan dễ dàng đưa vào mạch ứng dụng theo nhiều giao thức giao tiếp khác nhau, tốn ít tài nguyên hệ thống và giá thành rẻ.

*Cấu tạo*



LCD 16x2 có 2 hàng, mỗi hàng 16 ký tự, trong 16 chân của LCD được chia làm 3 dạng tín hiệu như sau:

* Các chân cấp nguồn: Chân số 1 nối mass (0V), chân số 2 là VDD nối với nguồn 5V, chân số 3 dùng để chỉnh contrast thường nối với biến trở.
* Các chân điều khiển: Chân số 4 là chân RS dùng để điều khiển lựa chọn thanh ghi. Chân R/W dùng để điều khiển quá trình đọc và ghi. Chân E là chân cho phép dạng xung chốt.
* Các chân dữ liệu DB0 - DB7: Là chân từ số 7 đến 14 dùng để trao đổi dữ liệu giữa thiết bị điều khiển và LCD.
* Chân 15 nối nguồn +5V hoặc 4.2V nối với led, chân 16 nối GND.

*Ứng dụng*

LCD thường được sử dụng trong các mạch điện tử, hiển thị thời gian thực, giá trị, kết quả, hiệu ứng.

* + 1. Module chuyển I2C LCD 16x2
  1. IC ổn áp LM7805
  2. WiFi và Internet
  3. Tổng quan về Webserver
     1. Tạo giao diện Webserver

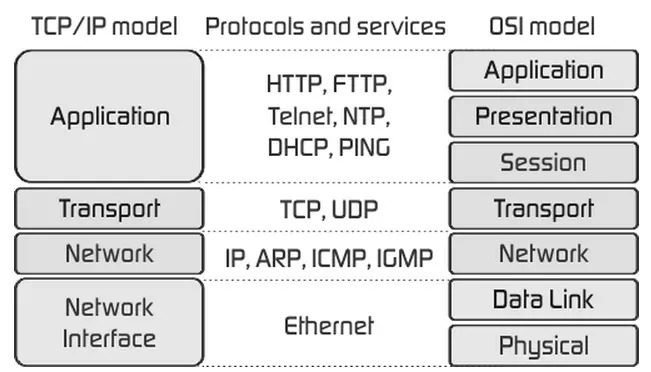
*HTML*

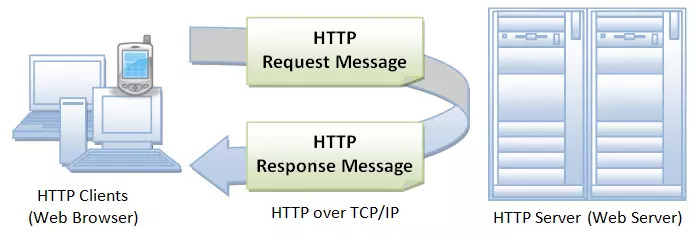
*CSS*

*JavaScript*

* + 1. Giao thức HTTP

HTTP (HyperText Transfer Protocol – Giao thức truyền tải siêu văn bản) là một trong các giao thức chuẩn về mạng Internet, được dùng để liên hệ thông tin giữa Máy cung cấp dịch vụ (Web server) và Máy sử dụng dịch vụ (Web client), là giao thức Client/Server dùng cho World Wide Web – WWW



Giao thức HTTP hoạt động dựa trên mô hình Client – Server. Thông thường khi các bạn lướt web, các máy tính của người dùng sẽ đóng vai trò làm máy khách (Client). Sau một thao tác nào đó của người dùng, các máy khách sẽ gửi yêu cầu đến máy chủ (Server) và chờ đợi câu trả lời từ những máy chủ này.

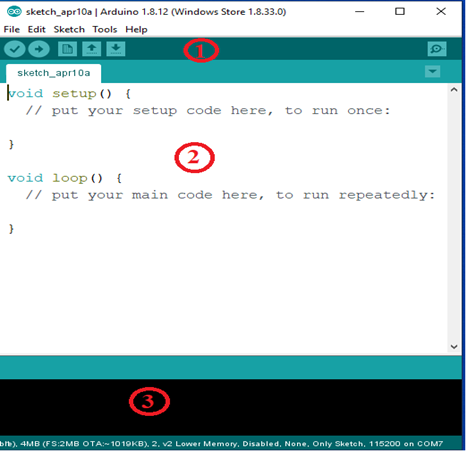
Ngoài ra, khi các hệ thống trao đổi dữ liệu với nhau, chúng cũng sử dụng giao thức này nhưng 2 bên đều là server.

* + 1. Xử lý dữ liệu Webserver
  1. Các phần mềm được sử dụng
     1. Phần mềm Arduino IDE

Sử dụng phần mềm Arduino IDE để lập trình cho ESP32

Arduino IDE là môi trường để lập trình và nạp code cho các dòng Arduino. Arduino IDE được xây dựng trên miền nền tảng Java nên hỗ trợ hầu hết các hệ điều hành hiện nay.

Giao diện của Arduino IDE được chia thành 3 vùng chính (hình …)



Vùng 1: Các phím chức năng được nêu như bảng …

|  |  |  |
| --- | --- | --- |
| Icon |  | Chức năng |
|  |  | Biên dịch chương trình soạn thảo để kiểm tra các lỗi. |
|  |  | Biên dịch và upload chương trình đang soạn thảo. |
|  |  | Mở một trang soạn thảo mới. |
|  |  | Mở các chương trình đã lưu. |
|  |  | Lưu chương trình đang soạn thảo. |
|  |  | Mở cửa sổ Serial Monitor để gửi và nhận dữ liệu giữa máy tính và board vi điều khiển. |

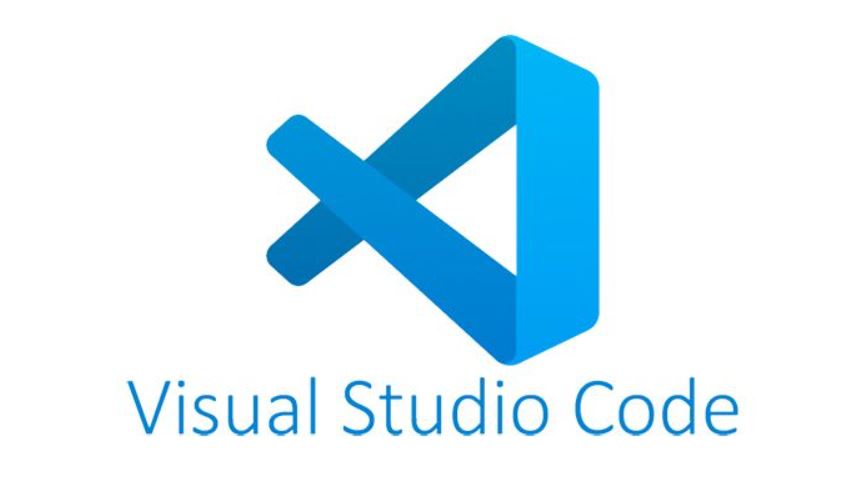
Vùng 2: Cửa sổ để viết chương trình

* Chương trình Code sẽ được viết tại đây. Ở đây có hai hàm quan trọng là setup() và loop().
* Hàm setup() được khởi chạy một lần duy nhất. Chức năng của hàm này dùng để khởi tạo các biến, khai báo chức năng các chân, khởi tạo các thông số bán đầu
* Hàm loop() là nơi chương trình được chạy lặp đi lặp lại đến khi ngắt vi điều khiển.
* Các dấu “//” dùng để tạo chú thích, giúp cho việc đọc code được dễ dàng hơn
* Trong lập trình có phân biệt ký tự hoa, thường, tuyệt đối phải đánh chích xác, đồng thời cuối mỗi câu lệnh cần phải có dấu chấm phẩy (;), trừ lệnh khai báo thư viện.

Vùng 3: Hiển thị các thông tin liên quan đến chương trình

Là cửa sổ để hiển thị về việc build chương trình, nạp chương trình thành công xuống vi điều khiển và các cảnh báo khác liên quan đến chương trình và điều khiển của chúng ta. Lưu ý, mọi thông báo và trạng thái của cả quá trình viết chương trình (write code), xây dựng chương trình (build code) và nạp chương trình (program code) đều được hiển thị tại đây. Cửa sổ này được gọi là cửa sổ debug.

* + 1. Phần mềm Visual Studio Code

Visual Studio Code là phần mềm lập trình được phát triển bởi Microsoft trên ba nền tảng khác nhau và hỗ trợ đa ngôn ngữ. Ngoài ra, Visual Studio Code còn có chức năng tự hoàn thành lệnh thông minh và cải tiến mã nguồn.

Phần mềm tải Visual Studio Code hỗ trợ đa ngôn ngữ, bao gồm HTML, CSS, C#, F#, C/C++, JSON, JavaScript. Bạn có thể lựa chọn ngôn ngữ mình muốn, sau đó nhấn chọn mục Install và bắt đầu sử dụng. Tại danh mục Popular này, bạn còn có thể lựa chọn các ngôn ngữ lập trình khác như Python, ESLint, Java,... để sử dụng.

Một ưu điểm vượt trội khác nữa của Visual Studio Code là tại phần mềm này, người dùng có nền tảng Windows, Mac hoặc Linux đều có thể sử dụng được. Với ba nền tảng trên, bạn đừng lo lắng máy mình không cài đặt được nhé.

* + 1. Phần mềm Altium Designer

Altium Designer trước kia có tên gọi quen thuộc là Protel DXP, là một trong những công cụ vẽ mạch điện tử mạnh nhất hiện nay. Được phát triển bởi hãng Altium Limited. Altium designer là một phần mềm chuyên nghành được sử dụng trong thiết kế mạch điện tử. Nó là một phần mềm mạnh với nhiều tính năng thú vị, tuy nhiên phần mềm này còn được ít người biết đến so với các phần mềm thiết kế mạch khác như Orcad hay Proteus

Altium Designer có một số đặc trưng sau:

- Giao diện thiết kế, quản lý và chỉnh sửa thân thiện, dễ dàng biên dịch, quản lý file, quản lý phiên bản cho các tài liệu thiết kế.

- Hỗ trợ mạnh mẽ cho việc thiết kế tự động, đi dây tự động theo thuật toán tối ưu, phân tích lắp ráp linh kiện. Hỗ trợ việc tìm các giải pháp thiết kế hoặc chỉnh sửa mạch, linh kiện, netlist có sẵn từ trước theo các tham số mới.

- Mở, xem và in các file thiết kế mạch dễ dàng với đầy đủ các thông tin linh kiện, netlist, dữ liệu bản vẽ, kích thước, số lượng…

- Hệ thống các thư viện linh kiện phong phú, chi tiết và hoàn chỉnh bao gồm tất cả các linh kiện nhúng, số, tương tự…

- Đặt và sửa đối tượng trên các lớp cơ khí, định nghĩa các luật thiết kế, tùy chỉnh các lớp mạch in, chuyển từ schematic sang PCB, đặt vị trí linh kiện trên PCB.

- Mô phỏng mạch PCB 3D, đem lại hình ảnh mạch điện trung thực trong không gian 3 chiều, hỗ trợ MCAD-ECAD, liên kết trực tiếp với mô hình STEP, kiểm tra khoảng cách cách điện, cấu hình cho cả 2D và 3D

- Hỗ trợ thiết kế PCB sang FPGA và ngược lại.

Từ đó, chúng ta thấy Altium Designer có nhiều điểm mạnh so với các phần mềm khác như đặt luật thiết kế, quản lý đề tài mô phỏng dễ dàng, giao diện thân thiện,…

Việc thiết kế mạch điện tử trên phần mềm altium designer có thể được tóm tắt gồm các bước như sau:

- Đặt ra các yêu cầu bài toán.

- Lựa chọn linh kiện.

- Thiết kế mạch nguyên lý.

- Lựa chọn các chân linh kiện để chuyển sang mạch in Update mạch nguyên lý sang mạch in.

- Lựa chọn kích thước mạch in Sắp sếp các vị trí các loại linh kiện như điện trở , tụ điện, IC...

- Đặt kích thước các loại dây nối.

- Đi dây trên mạch.

- Kiểm tra toàn mạch.

* 1. Kết luận chương 2

Chương 2 nói về việc sử dụng các thiết bị, các phần mềm và các công cụ đã dùng trong quá trình hoàn thành sản phẩm và các kiến trúc tổng quan phục vụ cho việc giải bài toán được đặt ra. Cùng với đó, chương 2 giới thiệu cũng như cách sử dụng các công cụ để lập trình và debug chương trình vi điều khiển ESP32. Ngoài ra chương 2 cũng giới thiệu về các công cụ lập trình bên phía Webserver như HTML, CSS, JavaScript.

# THIẾT KẾ HỆ THỐNG

* 1. Sơ đồ khối của hệ thống
     1. Yêu cầu

Mô hình hoàn thiện đảm bảo các chức năng đo đúng nhiệt độ và độ ẩm môi trường. Hiển thị đúng các thông số về nhiệt độ và độ ẩm lên màn hình LCD, tương tác với Webserver và gửi dữ liệu lên Webserver.

* + 1. Sơ đồ khối

Giao tiếp giữa ESP32 và điện thoại/máy tính:



**Mạch ESP32**



**Request**

**Response**

Khi đó ESP32 đóng vai trò là Server và điện thoại/máy tính đóng vai trò là Client.

Điện thoại/máy tính sẽ gửi Request tới Server của ESP32 và ESP32 sẽ trả về dữ liệu nhiệt độ, độ ẩm.

Khối nguồn và ổn áp 5v

Khối xử lý trung tâm (ESP32)

Màn hình LCD

Khối cảm biến

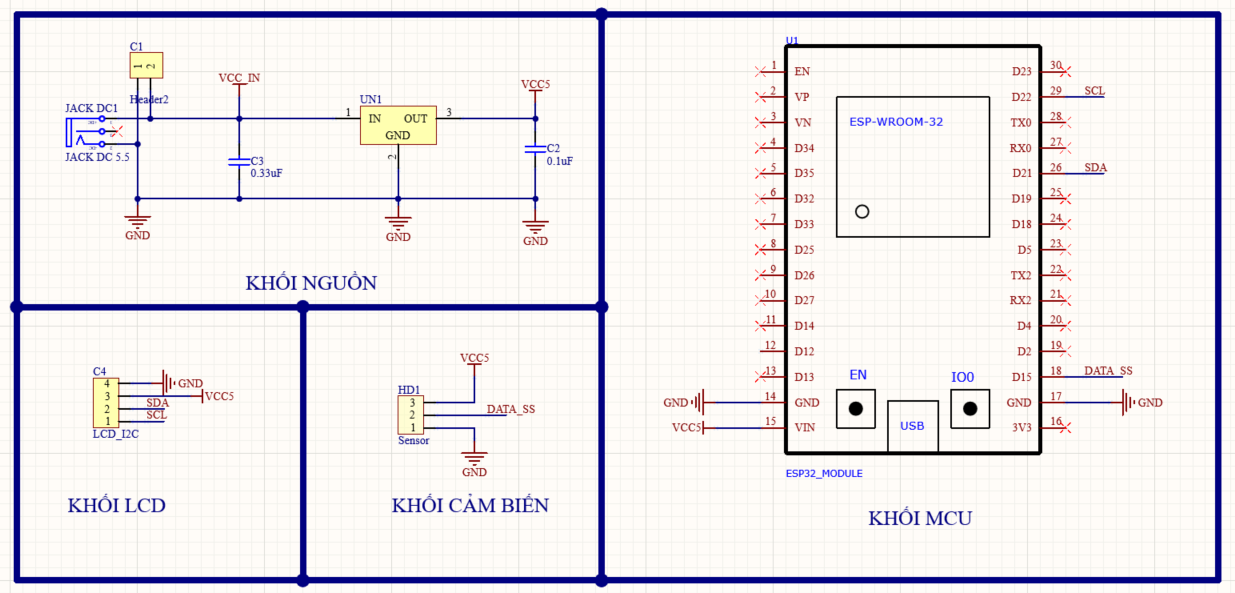
Mạch ESP32 bao gồm 4 khối chính:

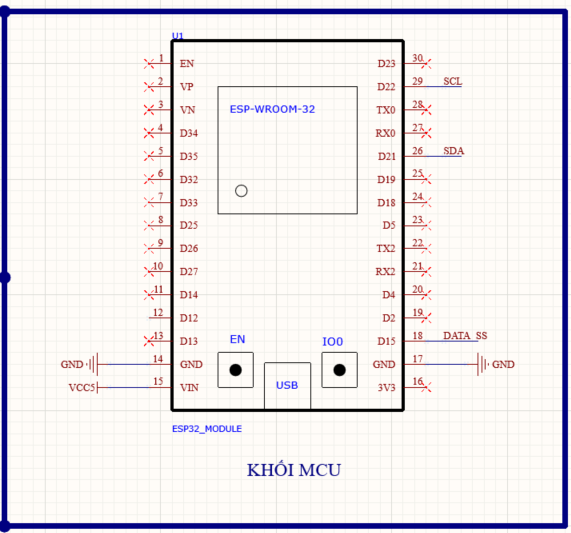
Khối nguồn và ổn áp 5V: mạch được cấp nguồn 12V, sau đó nguồn 12V sẽ được đi qua khối ổn áp về 5V để cung cấp cho vi điều khiển và cảm biến

Khối xử lý trung tâm: Nhận và xử lý tín hiệu từ cảm biến, hiển thị giá trị ra LCD và gửi tín hiệu khi Client Request vào.

Màn hình LCD: Hiển thị giá trị nhiệt độ và độ ẩm

* 1. Sơ đồ nguyên lý của mạch



* + 1. Khối xử lý trung tâm

Khối xử lý trung tâm gồm có 1 vi xử lý chính là ESP32. ESP32 được sử dụng trong mạch ở dạng Kit cắm có tên là ESP32 Devkit V1.

* + 1. Khối nguồn và ổn áp nguồn

*Khối nguồn:*

Các Tính năng

Điện áp ngõ ra: 12V

Điện áp ngõ vào: 100~240VAC

Đầu cắm AC: 2 chân dẹt Mỹ

Đầu DC ngõ ra: 5.5x2.5mm

Dòng điện ngõ ra: 1A

*Khối ổn áp nguồn*

IC chính được sử dụng để ổn áp mạch 5V là LM7805

Tụ C3 được thêm vào để lọc nguồn đầu và cải thiện nguồn nếu LM7805 nằm cách bộ lọc của nguồn cung cấp một khoảng cách đáng kể.

Tụ C2 giúp bộ ổn áp nguồn cải thiện được độ ổn định và phản ứng nhất thời.

* + 1. Khối kết nối với ngoại vi

Các header XH2.54 để kết nối tới cảm biến và màn hình:

Kết nối tới màn hình LCD I2C:

Gồm 4 chân: GND, VCC5, SDA, SCL

Kết nối cảm biến:

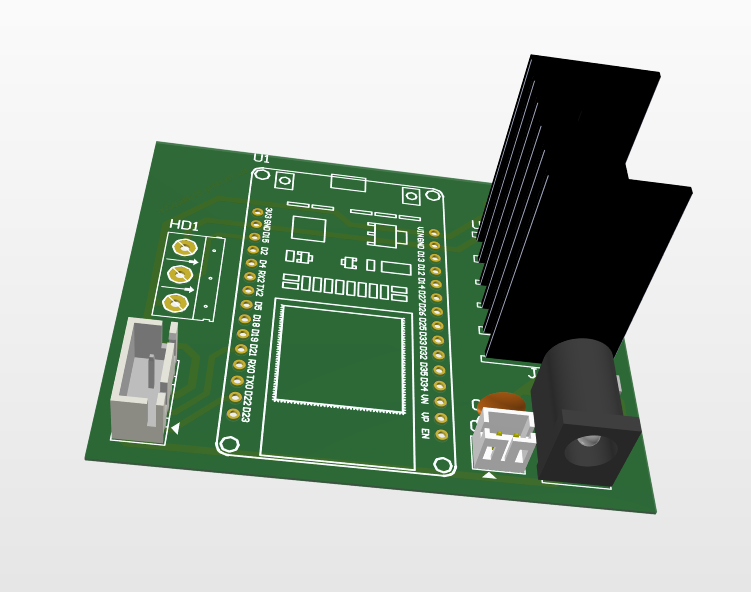
Gồm 3 chân: VCC5, DATA\_SS và GND

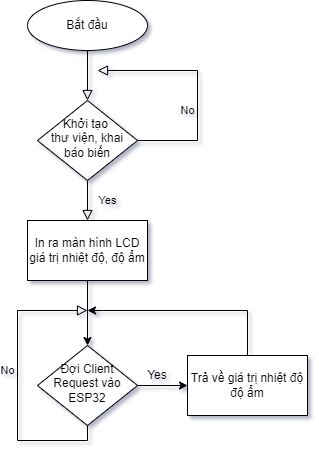
* 1. Sơ đồ mạch in (PCB)

Sử dụng phần mềm Altium Designer 21.2.1 để thiết kế mạch in (PCB) cho đề tài.

Dạng 2D của mạch

Dạng 3D của mạch:



* 1. Lập trình vi xử lý ESP32
     1. Lưu đồ thuật toán

Khởi tạo và khai báo các thư viện sử dụng

In ra màn hình LCD 5s/1 lần liên tục khi bật hệ thống

Đợi Client Request vào Server khi truy cập vào địa chỉ IP, từ đó trả về giá trị nhiệt độ và độ ẩm

* + 1. Các thư viện được sử dụng
* Thư viện Arduino.h cung cấp các hàm dạng Arduino cho vi điều khiển ESP32: #include <Arduino.h>
* Thư viện Wifi.h cung cấp các hàm về kết nối WiFi cho ESP32:   
  #include <WiFi.h>
* Thư viện giao tiếp I2C dùng để giao tiếp giữa màn hình LCD và ESP32: #include <Wire.h>
* Thư viện giao tiếp giữa cảm biến DHT21 và ESP32: #include <DHT.h>
* Thư viện các file hệ thống SPIFFS #include <SPIFFS.h>
  + 1. Lập trình Wifi ở chế độ Access Point

Khi ESP32 được cấu hình ở chế độ Access Point, ESP32 sẽ phát ra một mạng Wifi với địa chỉ IP là: 192.168.4.1 để kết nối với các thiết bị khác.

Lập trình ESP32 ở chế độ Access Point

WiFi.softAP(ssid, password); //Khởi tạo SSID và PASSWORD

Serial.print("IP address: ");

Serial.println(WiFi.softAPIP()); //In ra màn hình địa chỉ IP của mạch

* 1. Lập trình đọc nhiệt độ độ ẩm DHT21

Tạo hàm đọc nhiệt độ reaDHTHum() với kiểu trả về dạng String

String readDHTHum()

{

humi = dht.readHumidity();

if (isnan(humi)) {

Serial.println("khong doc duoc do am");

}

else {

Serial.println(humi);

return String(humi);

}

Tạo hàm đọc nhiệt độ reaDHTHum() với kiểu trả về dạng String

String readDHTTemp()

{

temper = dht.readTemperature();

if (isnan(temper)) {

Serial.println("khong doc duoc nhiet do");

}

else {

Serial.println(temper);

return String(temper);

}

* + 1. Lập trình LCD16x2 với giao tiếp I2C

*Khai báo LCD với địa chỉ I2C là 0x27, loại 16 cột và 2 hàng*

LiquidCrystal\_I2C lcd(0x27, 16, 2); //Khai báo LCD 16x2

*Khởi tạo LCD*

lcd.init();//khởi tạo LCD

lcd.backlight();//bật đèn nền LCD

*Set con trỏ và in ra dữ liệu lên màn hình*

lcd.setCursor(0, 0); //để con trỏ về đầu màn hình hàng 0

lcd.print("Temp: "); //in ra chữ Temp:

lcd.setCursor(6, 0); //để con trỏ về cột thứ 6, hàng thứ 0 của màn hình

lcd.print(temper); //in ra giá trị nhiệt độ:

lcd.setCursor(13, 0); //để con trỏ về cột thứ 13, hàng thứ 0 của màn hình

lcd.write(0xDF); //in ra ký tự độ trong nhiệt độ C

lcd.print("C");); //in ra chữ C

lcd.setCursor(0,1); //để con trỏ về đầu màn hình hàng 1

lcd.print("Humi: "); //in ra chữ Humi:

lcd.setCursor(6, 1);

lcd.print(humi); //in ra giá trị độ ẩm

lcd.setCursor(13, 1); //để con trỏ về đầu màn hình hàng 1 cột 13

lcd.print("%");//in ra ký tự %

* + 1. Lập trình tương tác với giao diện Webserver

Trả về file index.html khi trình duyệt request vào trang chính (192.168.4.1):

server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/index.html");

});

Trả về file chứa dữ liệu nhiệt độ khi trình duyệt request vào trang /temperature (192.168.4.1/temperature):

server.on("/temperature", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", temperStr.c\_str());

});

Trả về file chứa dữ liệu độ ẩm khi trình duyệt request vào trang /temperature (192.168.4.1/temperature):

server.on("/humidity", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", humiStr.c\_str());

});

* 1. Lập trình Webserver
     1. Xây dựng giao diện Web

Giao diẹn được chia làm 3 phần:

Phần HTML được đưa vào trong thẻ <body> </body>

<body>

  <h2>Bui Thi Hong 2018605226</h2>

  <h2>Web display temperature and humidity</h2>

  <div id="chart-temperature" class="container"></div>

  <div id="chart-humidity" class="container"></div>

</body>

Phần CSS nằm trong thẻ <style> </style>

  <style>

    body {

      min-width: 310px;

      max-width: 800px;

      height: 400px;

      margin: 0 auto;

    }

    h2 {

      font-family: Arial;

      font-size: 2.5rem;

      text-align: center;

    }

  </style>

Phần JavaScript: tạo đồ thị với thư viện HighChart.js

var chartT = new Highcharts.Chart({

  chart:{ renderTo : 'chart-temperature' },

  title: { text: 'DHT21 Temperature' },

  series: [{

    showInLegend: false,

    data: []

  }],

  plotOptions: {

    line: { animation: false,

      dataLabels: { enabled: true }

    },

    series: { color: '#059e8a' }

  },

  xAxis: { type: 'datetime',

    dateTimeLabelFormats: { second: '%H:%M:%S' }

  },

  yAxis: {

    title: { text: 'Temperature (C)' }

  },

  credits: { enabled: false }

});

* + 1. Lập trình tương tác Webserver với ESP32

Gọi hàm setInterval() để Request tới Server của ESP32 với thời gian 5S request 1 lần về giá trị nhiệt độ

setInterval(function ( ) {

  var xhttp = new XMLHttpRequest();

  xhttp.onreadystatechange = function() {

    if (this.readyState == 4 && this.status == 200) {

      var x = ((new Date()).getTime()) + + 7\*3600000,

          y = parseFloat(this.responseText);

      //console.log(this.responseText);

      if(chartT.series[0].data.length > 40) {

        chartT.series[0].addPoint([x, y], true, true, true);

      } else {

        chartT.series[0].addPoint([x, y], true, false, true);

      }

    }

  };

  xhttp.open("GET", "/temperature", true);

  xhttp.send();

}, 5000 ) ;

Gọi hàm setInterval() để Request tới Server của ESP32 với thời gian 5S request 1 lần về giá trị độ ẩm

setInterval(function ( ) {

  var xhttp = new XMLHttpRequest();

  xhttp.onreadystatechange = function() {

    if (this.readyState == 4 && this.status == 200) {

      var x = ((new Date()).getTime()) + + 7\*3600000,

          y = parseFloat(this.responseText);

      if(chartP.series[0].data.length > 40) {

        chartP.series[0].addPoint([x, y], true, true, true);

      } else {

        chartP.series[0].addPoint([x, y], true, false, true);

      }

    }

  };

  xhttp.open("GET", "/humidity", true);

  xhttp.send();

}, 5000 ) ;

* 1. Kết luận chương 3

Chương 3 thể hiện toàn bộ quá trình thiết kế, hoàn thiện và thử nghiệm “Mô hình xe robot dò line và tránh vật cản tự động”. Kết thúc chương, em đã rút ra nhiều kết luận:

* Tìm hiểu các bước thực hiện và giải quyết một bài toán ứng dụng cụ thể.
* Thiết kế mạch và lập trình ứng dụng vào mạch thực tế.
* Khắc phục lỗi, sự cố của sản phẩm.
* Thử nghiệm và đánh giá kết quả sản phẩm.

# KẾT QUẢ ĐẠT ĐƯỢC

* 1. Một số hình ảnh thực tế của hệ thống
  2. Giao diện điều khiển trên Webserver
  3. Đánh giá sự ổn định của hệ thống
  4. Kết luận chương 4

# KẾT LUẬN

Qua quá trình thực hiện làm đồ án, em đã trình bày các cơ sở lý thuyết liên quan và chạy thành công hệ thống “Thiết kế hệ thống điều khiển và giám sát các thiết bị từ xa bằng WiFi sử dụng ESP32”. Tuy thời gian làm đồ án thực sự không quá dài nhưng được sự giúp đỡ tận tình của Th.S Trần Xuân Phương cùng với sự nỗ lực và cố gắng của bản thân, sự chỉ bảo của các Thầy Cô trong khoa Điện tử em đã hoàn thành đề tài theo yêu cầu và đúng thời gian quy định với những nội dung sau:

* Nghiên cứu và tìm hiểu về các hệ thống điều khiển và giám sát trên thực tế, ưu điểm và nhược điểm của từng hệ thống
* Nghiên cứu và tìm hiểu về vi điều khiển ESP32, các ngoại vi và lập trình ESP32
* Tìm hiểu về chuẩn truyền thông không dây WiFi, ứng dụng nó vào trong việc điều khiển, truyền nhận tín hiệu và giám sát
* Nắm được các hệ thống điều khiển và giám sát từ xa hoạt động
* Xây dựng được giao diện điều khiển và giám sát
* Thiết kế và vận hành thành công mạch giám sát và điều khiển thiết bị qua WiFi sử dụng vi điều khiển ESP32

Hướng phát triển đề tài:

* Tạo giao diện phong phú và đa dạng hơn
* Giám sát và điều khiển các thiết bị không chỉ ON/OFF mà cả các tín hiệu nhiều dạng khác nhau
* Phát triển đa nền tảng (Android, IOS…)
* Điều khiển và giám sát thêm các thiết bị khác, các loại thiết bị có công suất lớn

Thông qua quá trình làm đồ án, em đã được vận dụng những kiến thức chuyên ngành trong 4 năm học. Qua đó đã giúp cho em rèn luyện được kỹ năng, cách tiếp cận với các vấn đề, các bài toán thực tế phức tạp tại các doanh nghiệp, nhà máy khi ra trường làm việc.

Việc xây dựng mô hình đã đáp ứng được yêu cầu đặt ra, tuy nhiên do trình độ và kinh nghiệm thực tiễn còn hạn chế nên không thể tránh khỏi sai sót và thiếu hoàn chỉnh. Rất mong được đón nhận sự đóng góp ý kiến từ thầy cô và các bạn.

Em xin chân thành cảm ơn!

# TÀI LIỆU THAM KHẢO

# PHỤ LỤC

Chương trình ESP32:

#include <WiFi.h>

#include <ESPAsyncWebServer.h>

#include <SPIFFS.h>

#include <Wire.h>

#include <DHT.h>

#include <AsyncElegantOTA.h>

#include <LiquidCrystal\_I2C.h>

#define ssid "Hong"

#define password "hongdt4"

//khai bao bien luu nhiet do va do am

float temper = 0.0;

float humi = 0.0;

float temperPre = 0.0;

float humiPre = 0.0;

String temperStr = " ";

String humiStr = " ";

//init lcd

LiquidCrystal\_I2C lcd(0x27, 16, 2); //lcd i2c 16x2

#define DHTPIN 15 // what pin we're connected to

#define DHTTYPE DHT21 // DHT 22 (AM2302)

DHT dht(DHTPIN, DHTTYPE); //// Initialize DHT sensor for normal 16mhz Arduino

// Create AsyncWebServer object on port 80

AsyncWebServer server(80);

String readDHTTemp()

{

// Read temperature as Celsius (the default)

temper = dht.readTemperature();

if(temper > 0)

{

temperPre = temper;

}

if (isnan(temper)) {

Serial.println("khong doc duoc nhiet do");

temper = temperPre;

return String(temper);

}

else {

Serial.println(temper);

return String(temper);

}

}

String readDHTHum()

{

humi = dht.readHumidity();

if(humi > 0)

{

humiPre = humi;

}

if (isnan(humi)) {

Serial.println("khong doc duoc do am");

humi = humiPre;

return String(humi);

}

else {

Serial.println(humi);

return String(humi);

}

}

void setup()

{

Serial.begin(115200);

dht.begin();

//khoi tao lcd

lcd.init();

// bat led nen

lcd.backlight();

if(!SPIFFS.begin()){

Serial.println("An Error has occurred while mounting SPIFFS");

return;

}

// Connect to Wi-Fi

WiFi.softAP(ssid, password);

Serial.print("IP address: ");

Serial.println(WiFi.softAPIP());

// Route for root / web page

server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/index.html");

});

server.on("/temperature", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", temperStr.c\_str());

});

server.on("/humidity", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", humiStr.c\_str());

});

AsyncElegantOTA.begin(&server);

server.begin();

}

void loop(){

temperStr = readDHTTemp();

humiStr = readDHTHum();

//set con tro hang thu nhat, cot thu nhat

lcd.setCursor(0, 0);

//in ra man hinh

lcd.print("Temp: ");

lcd.setCursor(6, 0);

lcd.print(temper);

lcd.setCursor(13, 0);

lcd.write(0xDF);

lcd.print("C");

//set con tro hang thu hai, cot thu nhat

lcd.setCursor(0,1);

lcd.print("Humi: ");

lcd.setCursor(6, 1);

lcd.print(humi);

lcd.setCursor(13, 1);

lcd.print("%");

delay(5000);

}