

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ

-----o0o-----



PROJECT MÔN HỌC
THIẾT KẾ HỆ THỐNG NHÚNG

GVHD: NGUYỄN PHAN HẢI PHÚ
SINH VIÊN THỰC HIỆN

STT	MSSV	HỌ	TÊN	% ĐIỂM BTL	ĐIỂM BTL	GHI CHÚ
1	2213427	Nguyễn Duy	Thức	100%		
2	2210880	Đinh Trọng	Tiến	100%		
3	2213758	Nguyễn Hoàng	Thiên	100%		

Tp. Hồ Chí Minh, năm 2024

MỤC LỤC

CHƯƠNG I: GIỚI THIỆU THIẾT BỊ	1
I. ĐẶC TẢ HỆ THỐNG	1
II. ĐẶC TẢ KỸ THUẬT	2
III. NĂM VẤN ĐỀ CƠ BẢN CỦA HỆ THỐNG	5
IV. HỢP ĐỒNG NHÓM	10
V. KẾ HOẠCH DỰ ÁN	11
CHƯƠNG 2: THIẾT KẾ	13
I. PHẦN CỨNG	13
II. PHẦN MỀM	16
CHƯƠNG III: THI CÔNG VÀ KẾT QUẢ	25
I. KẾT QUẢ SẢN PHẨM	25

CHƯƠNG I: GIỚI THIỆU THIẾT BỊ

I. ĐẶC TẢ HỆ THỐNG

1. Tên: BlueTemp-Humidity Tracker
2. Mục đích :
 - *Đo nhiệt độ, độ ẩm* : Đo lường nhiệt độ, độ ẩm trong một khu vực nhỏ (phòng ở, nhà kín, trang trại,...).
 - *Truyền tải thông tin* : Giao tiếp qua Bluetooth giữa bộ Master và Slave.
 - *Hiển thị kết quả đo*: Hiển thị các giá trị đo được trên màn hình LCD gắn vào bộ Master.
3. Ngõ vào/ra :
 - Ngõ vào :
 - Nguồn 3.3 - 5V
 - Tín hiệu analog (cảm biến)
 - Module Bluetooth Slave
 - Ngõ ra :
 - LCD 160x128: Hiển thị thông tin, kết quả đo.
 - Module Bluetooth Master
4. Trường hợp sử dụng:
 - Theo dõi nhiệt độ, độ ẩm của phòng chứa vật liệu; khu vườn. Có thể theo dõi thông tin ở vị trí đặt LCD (phòng quản lý). Có thể tích hợp thêm hệ thống khác để có những hỗ trợ thêm khi chỉ số nhiệt độ, độ ẩm vượt mức cho phép.
5. Chức năng :
 - Đo nhiệt độ và độ ẩm: Xác định nhiệt độ và độ ẩm xung quanh hệ thống theo thời gian thực sử dụng cảm biến nhiệt độ trên khối Slave.
 - Truyền tín hiệu sử dụng Bluetooth: truyền dữ liệu đã đo được từ khối Slave sang Master sử dụng một cặp module bluetooth HC-05 đã được cài sẵn.
 - Hiển thị kết quả đo: đưa các dữ liệu vừa nhận được từ HC-05 Master lên LCD để người dùng quan sát.
6. Hiệu năng :
 - Có thể đo và hiển thị đồng thời các giá trị nhiệt độ và độ ẩm, cập nhật mỗi 2s.
 - Hiện thị các dữ liệu một cách rõ ràng và dễ đọc, giúp người dùng dễ dàng theo dõi.
7. Chi phí :
 - Arduino nano V3.0 ATmega328P: 70,000VND x2
 - LCD 1.8inch 160x128 TFT ST7735: 80,000VND
 - DHT22: 15,000VND
 - Module Bluetooth HC-05: 120,000VND x2
 - Tổng : 475,000VND

8. Công suất :

- Dùng nguồn 5V
- Dùng trở chia áp 3.3V cho chân RTX của HC-05
- Để hoạt động cần cấp nguồn liên tục

9. Kích thước/cân nặng:

- Kích thước khoảng 2 x (15x5x5), nặng khoảng 300g

10. Cài đặt :

- Để nơi khô ráo thoáng mát, tránh ánh sáng trực tiếp
- Không để gần lửa tránh cháy nổ

II. ĐẶC TẢ KỸ THUẬT

1. Nguyên lý hoạt động:

a. Khối Slave:

- *Nguyên lý hoạt động:* Cảm biến nhiệt độ - độ ẩm sau khi mạch được cấp nguồn sẽ gửi dữ liệu đã được đo vào Arduino Nano. Với các thư viện thích hợp, ATmega328 hoàn toàn có khả năng mã hoá các dữ liệu đó thành các tín hiệu có thể chuyển đi trong môi trường bluetooth. Vi xử lý và module bluetooth được cài đặt giao tiếp UART, từ đó cho phép các tín hiệu đó được truyền qua chuẩn bị cho việc giao tiếp giữa 2 khối. Module bluetooth phải được cài đặt sẵn sao cho cung địa chỉ và baud rate, và đảm bảo chế độ thích hợp mới có thể giao tiếp.
- *Cấu tạo:*
 - Cảm biến nhiệt độ: Đo nhiệt độ và độ ẩm theo thời gian thực, gửi dữ liệu đo được cho vi điều khiển xử lý.
 - ATmega328: Vi xử lý của khối, chịu trách nhiệm xử lý các dữ liệu có được từ cảm biến và mã hoá chúng để truyền sang khối Master.
 - HC-05 Bluetooth (Slave): Được cài đặt trước sao cho cả 2 module có cùng baud rate và địa chỉ nhằm truyền các dữ liệu đã được xử lý qua khối Master.
- *Quy trình hoạt động:*
 - Sau khi được cấp nguồn, cảm biến sẽ liên tục cập nhật nhiệt độ và độ ẩm xung quanh trong thời gian thực rồi gửi nó qua vi xử lý.
 - Vi xử lý thực hiện mã hoá các dữ liệu vừa có được thành các tín hiệu có thể truyền đi trong môi trường bluetooth để HC-05 Slave thực hiện việc truyền tín hiệu.
 - HC-05 Slave sau khi được thiết lập địa chỉ và baud rate phù hợp sẽ nhận các tín hiệu vừa được xử lý và phát qua khối Master.

b. Khối Master

- *Nguyên lý hoạt động:* Tương tự với khối Slave, một ATmega328 khác được kết nối UART với một module bluetooth được cài đặt ở chế độ Master để thu nhận tín hiệu được phát mỗi 2 giây. Vì xử lý này sau đó giải mã hoá tín hiệu đó trở về các dữ liệu ban đầu và đưa nó đến ngõ ra được cài sẵn LCD, từ đó hiện dữ liệu cho người đọc.
- *Cấu tạo:*
 - ATmega328: tương tự như ở khối Slave, xử lý các dữ liệu nhận được từ khối Slave, giải mã hóa rồi truyền qua LCD để hiện các thông tin đã thu được.
 - HC-05 Bluetooth (Master): Tương tự như ở khối Slave, làm nhiệm vụ nhận dữ liệu được truyền.
 - LCD 160x128 bit: Hiện thị các giá trị đã đo được.
- *Quy trình hoạt động*
 - Sau khi dữ liệu được truyền đi từ khối Slave theo dạng tín hiệu bluetooth, HC-05 Master của khối làm nhiệm vụ thu lấy những tín hiệu này và gửi chúng cho ATmega328 mỗi 2s.
 - ATmega328 sẽ bắt đầu quá trình giải mã tín hiệu đã thu thành dữ liệu số, rồi từ đó chuyển qua LCD để hiện kết quả cho người dùng.

2. Môi Trường Hoạt Động:

Bộ BlueTemp-Humidity Tracker có thể hoạt động trong nhiều môi trường khác nhau, tùy thuộc vào thiết kế của hệ thống và các thành phần được sử dụng. Dưới đây là một số yếu tố quan trọng và điều kiện môi trường cần xem xét:

a. Nhiệt Độ:

- Nhiệt độ hoạt động: Arduino Nano và cảm biến nhiệt độ của hệ thống hoạt động hiệu quả nhất trong phạm vi từ -40°C đến 80°C . Các linh kiện như điện trở, module bluetooth và nguồn cấp phải đảm bảo chịu được dải nhiệt này.

b. Độ Ẩm:

- Độ Ẩm Tương Đối: DHT22 hoạt động tốt ở độ ẩm 100%RH (-40°C đến 80°C), tuy nhiên nên để bảo vệ các linh kiện nên đặt hệ thống ở các môi trường có độ ẩm dưới 85% và nên tránh các môi trường mốc, ủ kín.
- Bảo Vệ: Do hệ thống có khả năng được sử dụng trong các môi trường có độ ẩm cao, cần có vỏ bảo vệ hoặc các biện pháp chống ẩm nhằm bảo vệ các linh kiện điện tử.

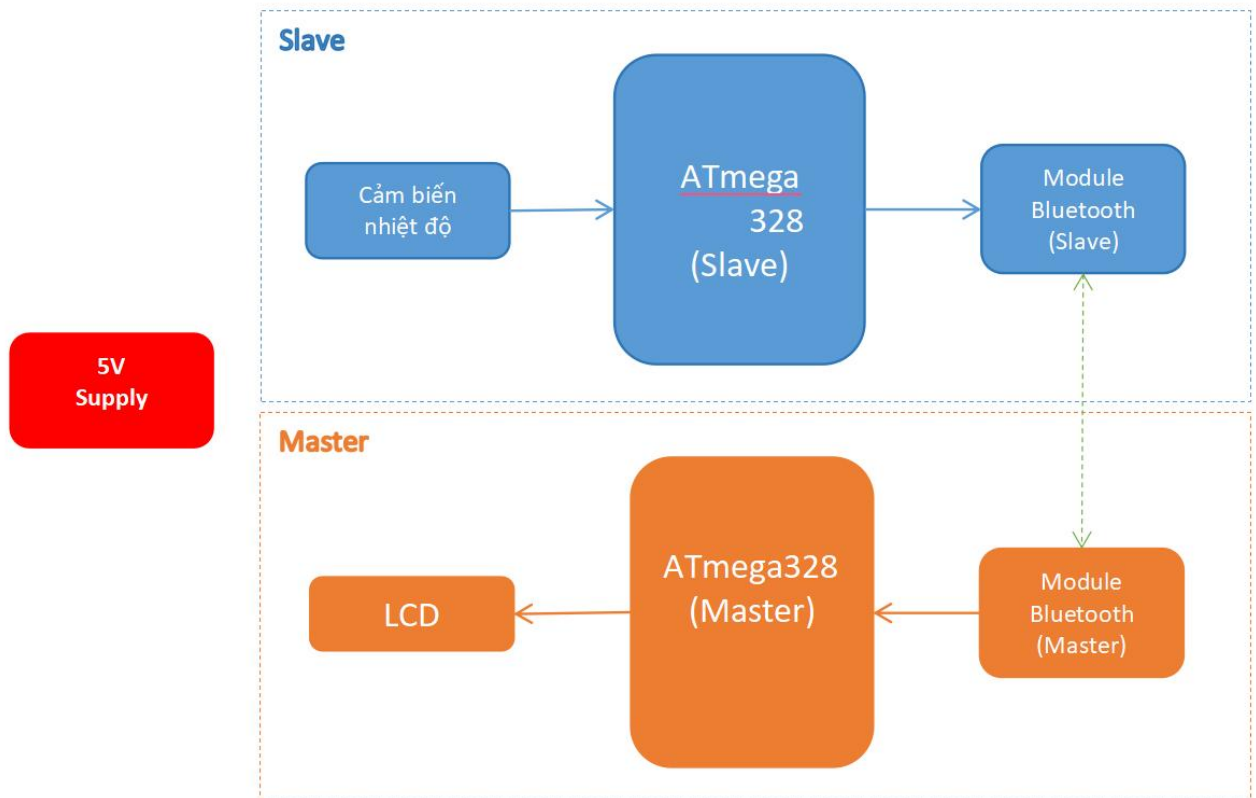
c. Bluetooth:

- Giao tiếp giữa 2 module bluetooth cần được cài đặt trước để có cùng địa chỉ, baud rate (9600) và phải được chia chế tương ứng với khối nó được lắp vào (Slave hoặc Master).
- Đảm bảo khoảng cách giữa hai module Bluetooth không quá xa và không có nhiều vật cản.

d. Môi trường ảnh hưởng trong thực tế:

- Môi trường công nghiệp: Trong nhà máy, nơi có rung động, bụi bẩn, và các tác nhân môi trường khác có thể gây ảnh hưởng đến cảm biến và mạch điện.
- Môi trường ngoài trời: Nếu thiết bị được đặt ngoài trời, cần có các cơ chế bảo vệ chống lại tác động của thời tiết như mưa, nắng, gió, cũng như các yếu tố bụi bẩn.

3. Sơ đồ khối hệ thống:



4. Mô tả các khối chính:

Tên	Chức Năng
5VDC Supply	Cung cấp năng lượng cho hệ thống hoạt động
ATmega328 (Slave + Master)	Xử lý các tín hiệu vào và phát tín hiệu ra
Module Bluetooth (Slave + Master)	Truyền/nhận các dữ liệu giữa 2 khối
Cảm biến nhiệt độ	Đo nhiệt độ và độ ẩm xung quanh theo thời gian thực
LCD 160x128	Hiển thị các giá trị nhiệt độ và độ ẩm đo được

5. Phân chia phần cứng, phần mềm:

a.Hardware:

- +5VDC Supply
- Arduino Nano
- LCD 160X128
- HC-05
- DHT22
- LED
- USB-C
- Resistor

b.Software:

Để lập trình cho Arduino có thể sử dụng các phần mềm phát triển tích hợp (IDE) và công cụ hỗ trợ sau:

- *Arduino IDE*

Arduino IDE là một môi trường phát triển phổ biến và dễ sử dụng cho Arduino. Nó cung cấp các thư viện và công cụ cần thiết để lập trình và nạp mã vào Arduino.

Ưu điểm: Giao diện thân thiện, hỗ trợ nhiều thư viện, cộng đồng người dùng lớn.

- *Thư viện hỗ trợ*

DHT22: Thư viện đặc biệt dành riêng cho cảm biến nhiệt độ, giúp và hỗ trợ vì xử lý đọc được các dữ liệu lấy được từ cảm biến nhiệt độ.

Easy Transfer: thư viện giúp di chuyển các dữ liệu giữa các Arduino một cách dễ dàng.

Adafruit GFX, Adafruit ST7735: Thư viện hỗ trợ cho việc hiển thị LCD.

III. NĂM VẤN ĐỀ CƠ BẢN CỦA HỆ THỐNG

1. Constrain issues:

a. Khỏi Slave:

- Bộ nhớ hạn chế: Arduino Nano ATmega328P có RAM chỉ 2KB và bộ nhớ Flash 32KB, cần tối ưu mã nguồn để không chiếm quá nhiều bộ nhớ.
- Dữ liệu cảm biến: DHT22 lấy mẫu mỗi 2 giây, cần đảm bảo lưu trữ tạm thời trước khi truyền vì dữ liệu buffer chỉ lưu một mẫu.
- Tổng lượng dữ liệu truyền: Mỗi lần truyền dữ liệu chỉ chứa 4-8 byte (2 giá trị nhiệt độ và độ ẩm), cần đảm bảo không vượt quá giới hạn này để tránh mất gói.
- Độ chính xác cảm biến: DHT22 có độ chính xác $\pm 0.5^{\circ}\text{C}$ đối với nhiệt độ và $\pm 2\%$ đối với độ ẩm, cần lưu ý giới hạn này khi hiển thị kết quả.

b. Khỏi Master:

- Hiển thị trên LCD: Độ phân giải TFT 160x128 pixel, bộ nhớ của Arduino chỉ có thể lưu một lượng nhỏ ký tự và hình ảnh trong RAM 2KB.

c. Toàn bộ hệ thống:

- Khoảng cách truyền Bluetooth: Module HC-05 có phạm vi giao tiếp tối đa 10m trong môi trường không có vật cản. Trong môi trường nhiễu, khoảng cách này giảm xuống còn 5-7m.
- Băng thông Bluetooth: HC-05 truyền dữ liệu với tốc độ tối đa 2 Mbps, cần đảm bảo không truyền quá nhiều dữ liệu cùng lúc để tránh quá tải.
- Nguồn cấp: Arduino Nano hoạt động ổn định trong dải điện áp 5V, cần đảm bảo nguồn cấp từ 5-12V để không làm gián đoạn hoạt động của hệ thống.
- Tổng công suất tiêu thụ: Hệ thống cần dưới 500mA để tránh quá tải nguồn cung cấp.

2. Functional Issues:

a. Kết nối Bluetooth không ổn định, có các giải pháp sau:

- Đảm bảo khoảng cách giữa hai module Bluetooth không quá xa và không có nhiều vật cản.
- Kiểm tra và đảm bảo rằng cả hai module Bluetooth đều được cấu hình đúng baud rate.
- Sử dụng các cơ chế kiểm tra và khôi phục kết nối trong mã nguồn.

b. Đọc dữ liệu cảm biến không chính xác, có các giải pháp sau:

- Kiểm tra và đảm bảo rằng cảm biến được kết nối đúng cách và không bị hỏng.
- Đảm bảo rằng cảm biến được đặt ở vị trí không bị ảnh hưởng bởi các yếu tố môi trường khác.
- Thêm các cơ chế kiểm tra lỗi trong mã nguồn để phát hiện và xử lý các giá trị không hợp lệ.

c. Hiển thị LCD không đúng:

- Kiểm tra kết nối giữa Arduino và màn hình LCD.
- Đảm bảo rằng màn hình LCD hoạt động bình thường bằng cách thử với các ví dụ đơn giản trước khi tích hợp vào hệ thống.

d. Tiêu thụ năng lượng cao:

- Sử dụng chế độ ngủ (sleep mode) của Arduino để tiết kiệm năng lượng khi không hoạt động.
- Chọn các linh kiện và module có chế độ tiết kiệm năng lượng.

e. An toàn và tin cậy:

- Giám sát giá trị nhiệt độ và độ ẩm: Nếu cảm biến DHT22 phát hiện nhiệt độ hoặc độ ẩm vượt ngưỡng an toàn (ví dụ: nhiệt độ $>50^{\circ}\text{C}$ hoặc độ ẩm $<20\%$).

hay >90%), hệ thống cần kích hoạt cảnh báo trên màn hình LCD và có thể phát tín hiệu âm thanh hoặc đèn báo.

- Phát hiện lỗi cảm biến: Nếu cảm biến cung cấp dữ liệu không hợp lệ hoặc ngừng hoạt động, hệ thống phải thông báo lỗi để người dùng kiểm tra, tránh việc hiển thị sai thông tin có thể gây nguy hiểm.
- Mất kết nối Bluetooth: Nếu mất kết nối giữa master và slave trong thời gian dài, hệ thống cần cảnh báo ngay để người dùng biết và khắc phục, tránh trường hợp không giám sát được điều kiện môi trường.
- Nguồn cấp ổn định: Đảm bảo hệ thống không hoạt động dưới nguồn điện không ổn định, tránh chập cháy hoặc hư hỏng thiết bị, ảnh hưởng đến người sử dụng.

3. Realtime issues:

a. Loại hệ thống:

- Soft Real-time: Hệ thống này được xem là soft real-time vì mặc dù cần thu thập, truyền và hiển thị dữ liệu trong thời gian thực, nhưng việc trễ một chút sẽ không gây hậu quả nghiêm trọng. Tuy nhiên, để đảm bảo thông tin luôn cập nhật, các chu kỳ truyền dữ liệu phải tuân thủ giới hạn thời gian cụ thể.

b. Giới hạn thời gian:

- Thời gian đọc cảm biến: DHT22 có chu kỳ đo khoảng 2 giây, hệ thống cần đảm bảo đọc dữ liệu trong khoảng thời gian này.
- Thời gian truyền Bluetooth: Dữ liệu từ slave phải được truyền đến master trong vòng 100-200ms để không ảnh hưởng đến tính liên tục của thông tin.
- Thời gian cập nhật LCD: Thời gian cập nhật màn hình cần dưới 300ms để đảm bảo người dùng thấy thông tin mới nhất.

c. Vấn đề đồng bộ hóa:

- Đồng bộ dữ liệu giữa slave và master: Dữ liệu từ slave cần được truyền đồng bộ, tránh trường hợp dữ liệu mới ghi đè hoặc xung đột với dữ liệu cũ.
- Đồng bộ hiển thị: Master cần đảm bảo dữ liệu nhận được từ Bluetooth được xử lý và hiển thị ngay lập tức, không bị chậm trễ hoặc gián đoạn.

d. Quản lý tài nguyên:

- Bộ nhớ và băng thông: Với bộ nhớ hạn chế (2KB RAM trên Arduino Nano), hệ thống cần ưu tiên các tác vụ quan trọng, chẳng hạn như xử lý và truyền dữ liệu cảm biến trước khi thực hiện các tác vụ khác.
- Nguồn cấp: Đảm bảo nguồn điện ổn định cho cả master và slave để tránh ngắt quãng khi đang xử lý hoặc truyền dữ liệu.

e. Độ tin cậy và an toàn:

- Phát hiện và xử lý lỗi: Hệ thống cần phát hiện các lỗi như mất kết nối Bluetooth hoặc cảm biến cung cấp dữ liệu không hợp lệ, và đưa ra cảnh báo ngay lập tức.
- An toàn trong trường hợp khẩn cấp: Nếu phát hiện nhiệt độ hoặc độ ẩm vượt ngưỡng nguy hiểm, hệ thống cần phản ứng nhanh và thông báo cho người dùng để kịp thời xử lý.

4. Concurrent issues

a. Hệ thống và môi trường chạy đồng thời:

- Tương tác liên tục: Hệ thống cần theo dõi nhiệt độ và độ ẩm trong môi trường liên tục, đồng thời xử lý và hiển thị dữ liệu trong khi nhận thông tin từ cảm biến.
- Thích ứng với thay đổi: Các điều kiện môi trường (như nhiệt độ, độ ẩm) thay đổi liên tục, hệ thống cần đáp ứng ngay lập tức mà không làm gián đoạn các chức năng khác.

b. Đa nhiệm:

- Slave: Đọc dữ liệu từ cảm biến, xử lý thông tin, và truyền qua Bluetooth phải thực hiện song song mà không ảnh hưởng đến chu kỳ đọc cảm biến (2 giây/lần).
- Master: Vừa nhận dữ liệu từ Bluetooth, vừa cập nhật và hiển thị thông tin trên LCD. Cần đảm bảo các tác vụ này không xung đột hoặc gián đoạn nhau.
- Xử lý sự cố: Ngoài các chức năng chính, hệ thống cần giám sát kết nối và xử lý lỗi khi mất dữ liệu hoặc cảm biến gặp sự cố.

c. Giao diện với các hệ thống khác:

- Bluetooth HC-05: Giao tiếp giữa slave và master qua Bluetooth cần đảm bảo dữ liệu truyền ổn định, không bị trễ hoặc mất mát khi tương tác với môi trường không dây.
- Khả năng mở rộng: Hệ thống có thể được mở rộng để kết nối thêm các thiết bị như cảm biến khác hoặc hệ thống báo động ngoài, cần đảm bảo khả năng giao tiếp mà không làm giảm hiệu suất.

d. Bộ lập lịch quản lý các tác vụ đồng thời:

- Ưu tiên tác vụ: Cần sử dụng bộ lập lịch (scheduler) để ưu tiên các tác vụ quan trọng như đọc cảm biến và truyền dữ liệu Bluetooth.
- Xử lý ngắt: Có thể sử dụng cơ chế ngắt để ưu tiên xử lý các sự kiện quan trọng, như nhận dữ liệu mới từ slave hoặc cập nhật lỗi kết nối.

- Tránh xung đột: Scheduler giúp đảm bảo các tác vụ xử lý song song không chiếm quá nhiều tài nguyên, tránh xung đột giữa việc nhận dữ liệu và hiển thị trên LCD.

5. Reactive issues:

a. Tương tác liên tục và gián đoạn:

- Tương tác liên tục: Hệ thống luôn theo dõi nhiệt độ và độ ẩm, truyền dữ liệu đều đặn từ slave đến master. Master cập nhật LCD liên tục, phản ánh trạng thái hiện tại của môi trường.
- Tương tác gián đoạn: Trong trường hợp mất kết nối Bluetooth hoặc cảm biến không gửi dữ liệu, hệ thống cần xử lý và đưa ra cảnh báo thay vì chờ đợi vô thời hạn.

b. Power on Demand:

- Tối ưu tiêu thụ điện năng: Slave và master chỉ hoạt động các thành phần cần thiết (như đọc cảm biến và truyền Bluetooth), các thành phần không cần thiết nên được tắt hoặc đưa vào chế độ chờ để tiết kiệm năng lượng.
- Khởi động nhanh: Đảm bảo hệ thống có thể khởi động nhanh chóng để cung cấp dữ liệu ngay sau khi có nguồn, tránh trễ quá lâu trong các trường hợp cấp bách.

c. Hệ thống luôn hoạt động:

- Hoạt động liên tục: Hệ thống cần duy trì hoạt động 24/7 để giám sát môi trường liên tục, đảm bảo dữ liệu luôn được cập nhật.
- Khả năng phục hồi: Khi gặp sự cố (như mất điện tạm thời), hệ thống phải tự khởi động lại và tiếp tục hoạt động ngay lập tức.

d. Chấm dứt là hành vi xấu:

- Watchdog Timer: Cần sử dụng watchdog timer để giám sát và reset hệ thống nếu phát hiện treo hoặc ngừng hoạt động, đảm bảo hệ thống không bị dừng hoàn toàn.
- Phòng ngừa chấm dứt đột ngột: Nếu phần mềm hoặc phần cứng gặp lỗi, hệ thống cần tự phục hồi mà không cần can thiệp thủ công.




e. Phản hồi với hệ thống giám sát ngoài hoặc hệ thống thu thập dữ liệu:

- Giao tiếp mở rộng: Hệ thống có thể được thiết kế để gửi dữ liệu đến một hệ thống giám sát ngoài thông qua Bluetooth hoặc giao tiếp khác, đảm bảo giám sát từ xa hoặc tích hợp với các hệ thống IoT.
- Tính tương thích: Đảm bảo hệ thống có thể xử lý và phản hồi dữ liệu từ các thiết bị hoặc cảm biến khác nếu cần mở rộng trong tương lai.

f. Xử lý các sự kiện định kỳ và không định kỳ:

- Sự kiện định kỳ: Đọc dữ liệu từ cảm biến DHT22 và cập nhật LCD là các sự kiện định kỳ, cần được thực hiện theo chu kỳ cố định (ví dụ: mỗi 2 giây).
- Sự kiện không định kỳ: Mất kết nối Bluetooth hoặc lỗi cảm biến là các sự kiện không định kỳ. Hệ thống cần có cơ chế xử lý tức thời để đảm bảo hoạt động không bị gián đoạn

IV. HỢP ĐỒNG NHÓM





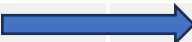



Team Contract		
Team name: TOP 1		Date: 1 Dec.2024
Team member	Roles	Signature
Đinh Trọng Tiến	Leader, Software design, Hardware design	
Nguyễn Duy Thức	Software design, report	
Nguyễn Hoàng Thiên	Hardware design, report	
Task		Responsible member
1. Develop system architecture		Đinh Trọng Tiến
2. Design hardware		Đinh Trọng Tiến, Nguyễn Hoàng Thiên
3. Develop software		Đinh Trọng Tiến, Nguyễn Duy Thức
4. Intergrate and test		All
Team meeting		10PM, Monday
Team rules	1. Participate fully in all meetings. 2. Complete all task before deadlines. 3. Give 3 days notice if you cannot complete the task on time.	



4. Focus on quality and results.

V. KẾ HOẠCH DỰ ÁN

Project Planning

Team name	Nhóm 8
Product name	Arduino Nano Bluetooth Temperature & Humidity Sensing
Main features	<ul style="list-style-type: none">- Measure temperature- Measure humidity- Show result on LCD
Estimate time	2,5 months (4 hours per day) Start: 16 Sep. 2024 End: 26 Nov. 2024
Estimate cost	Arduino nano V3.0 ATmega328P: 70,000VND x2 LCD 1.8inch 160x128 TFT ST7735: 80,000VND DHT22: 15,000VND Module Bluetooth HC-05: 120,000VND x2 Total: 475,000 VND

Schedule	Week 1+2	Week 3+4	Week 5+6
1.Design system architecture			
2.Design hardware part			
2.1 Design hardware part			
2.2 Design interface			
2.3 Implement hardware board			
3.Develop software part			
3.1 Develop measure algorithm			
3.2 Develop driver/user interface			
3.3 Develop extra software part			
3.4 Implement software program			

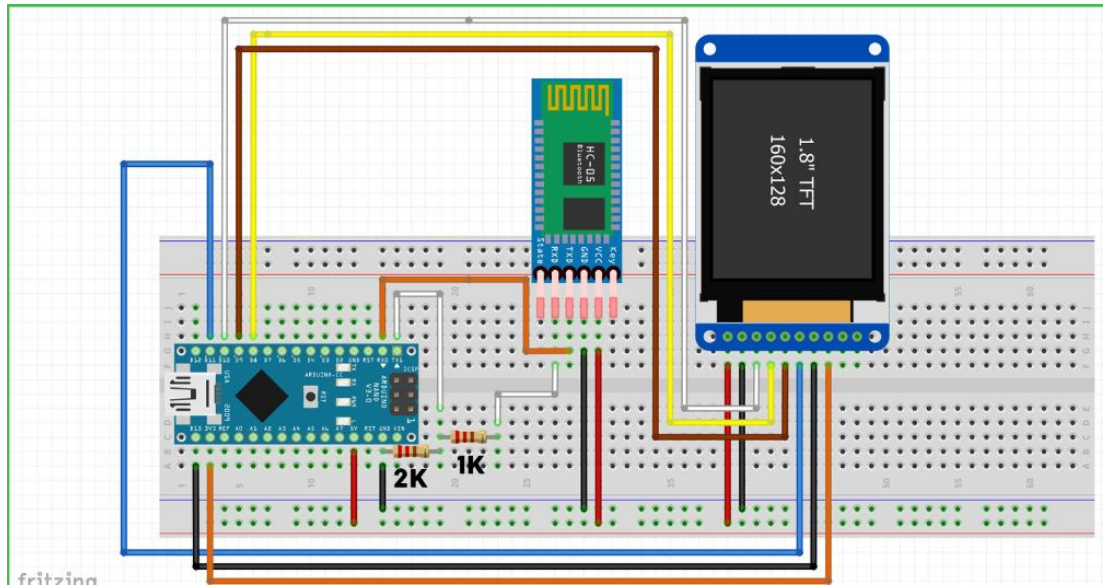
4. Intergrate and test			
4.1 Simulation			
4.2 Verify system			

CHƯƠNG 2: THIẾT KẾ

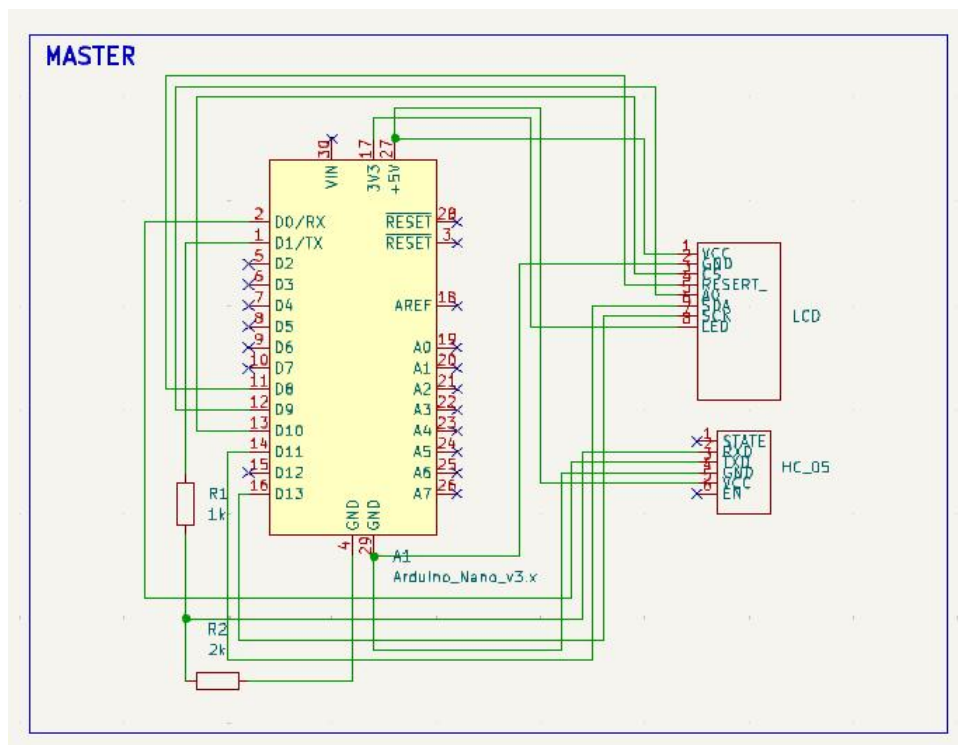
I. PHẦN CỨNG

1. Master:

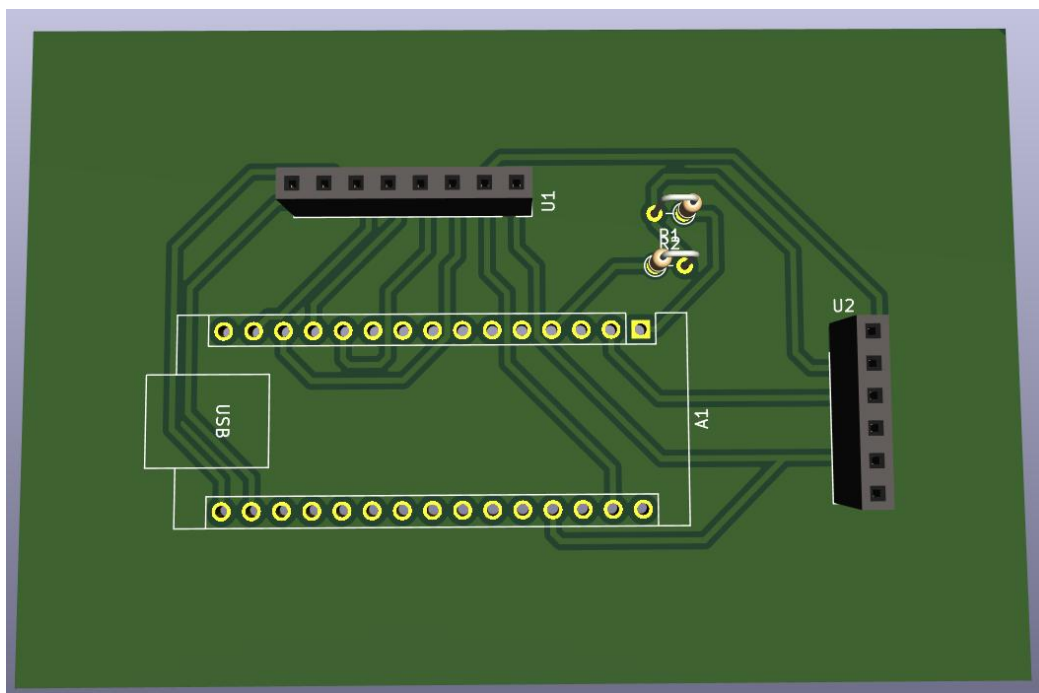
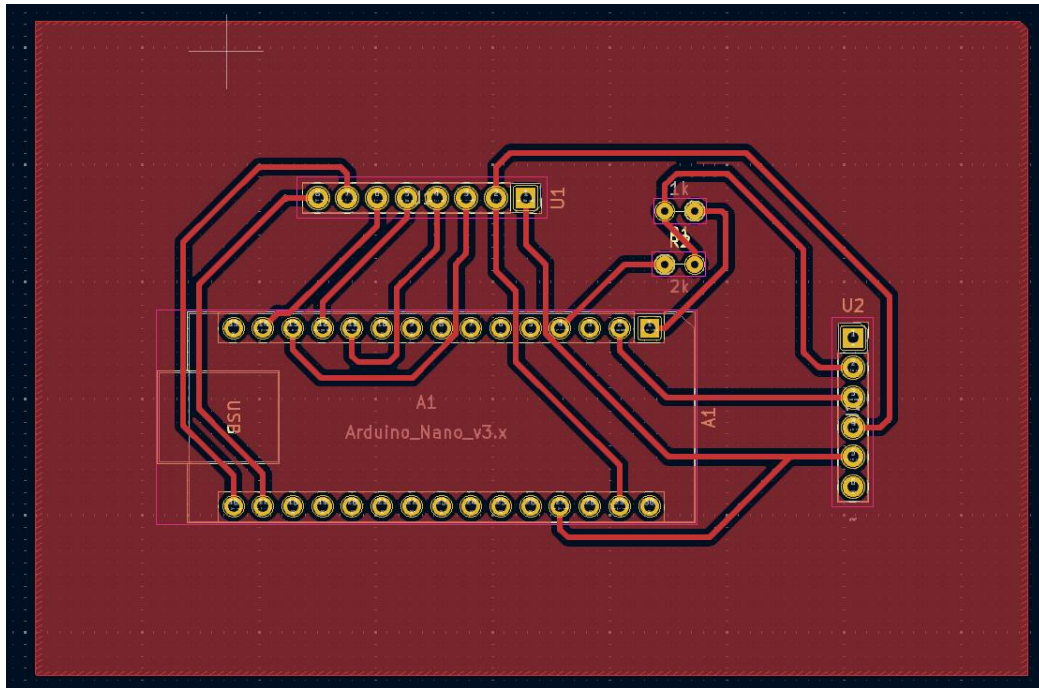
- Sơ đồ kết nối:



- Schematic:

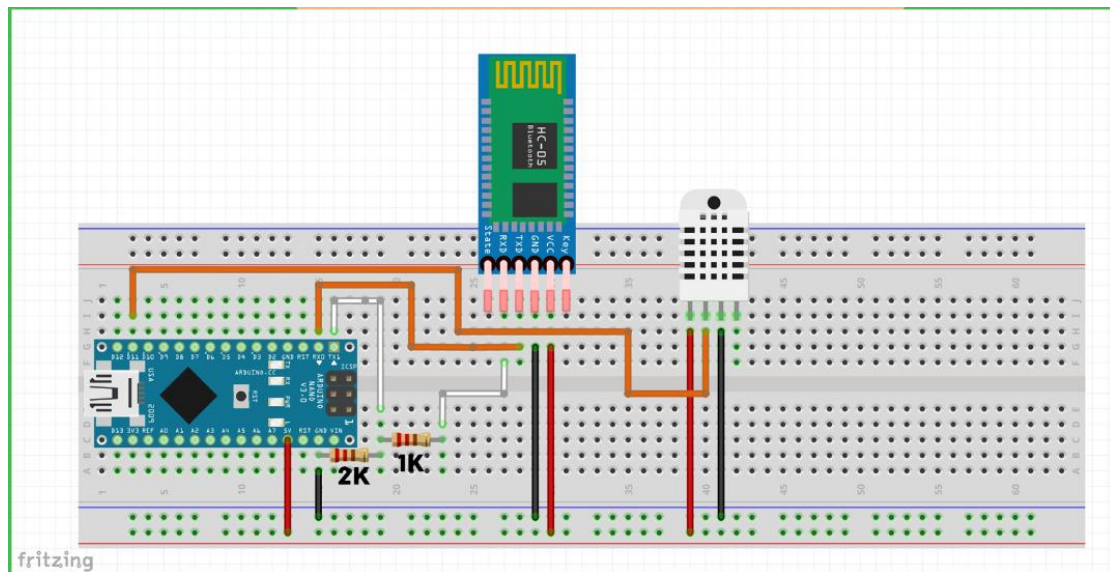


- PCB:

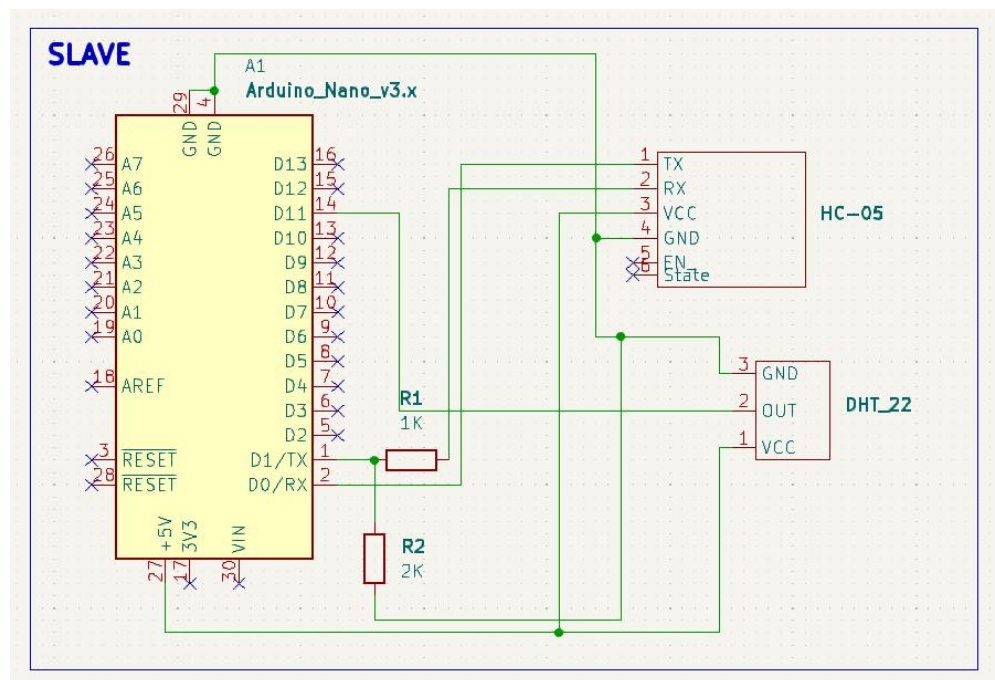


2. Slave:

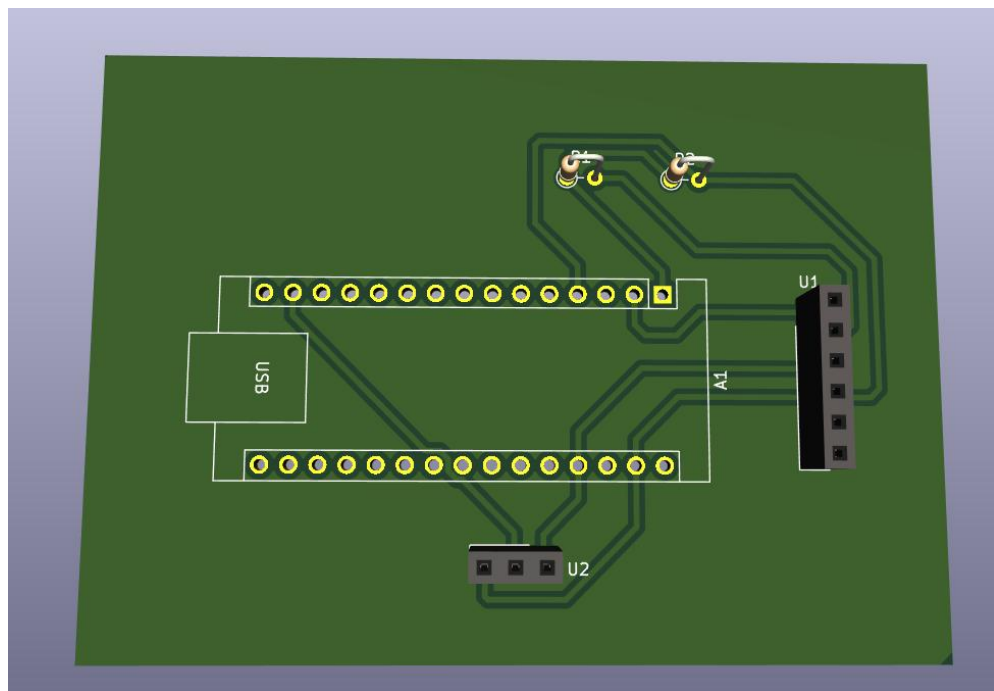
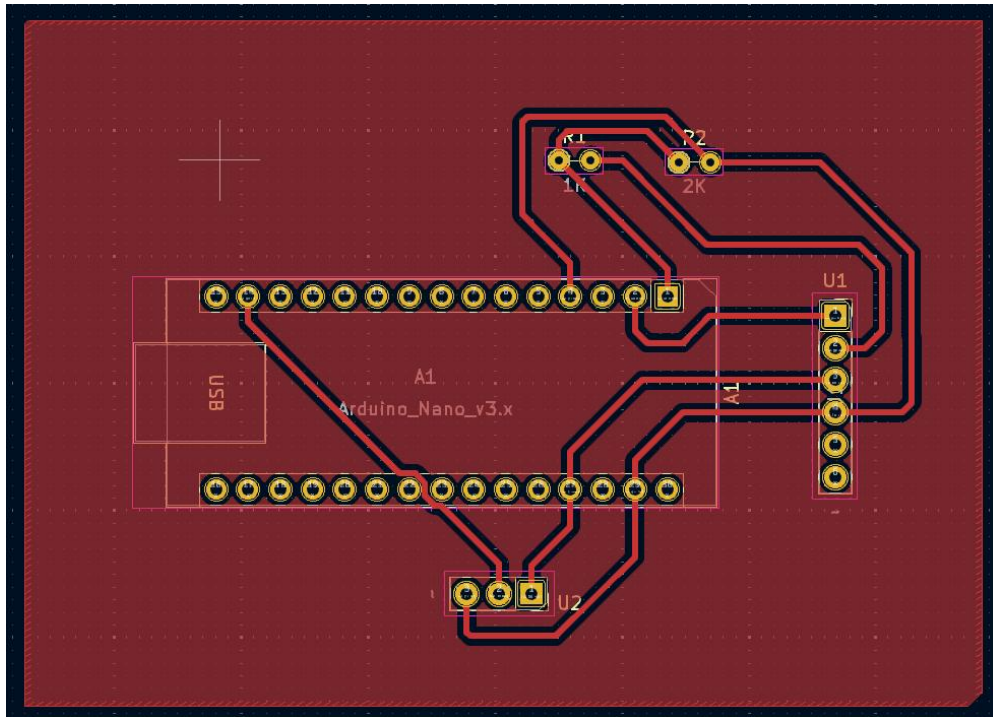
- Sơ đồ kết nối:



- Schematic:



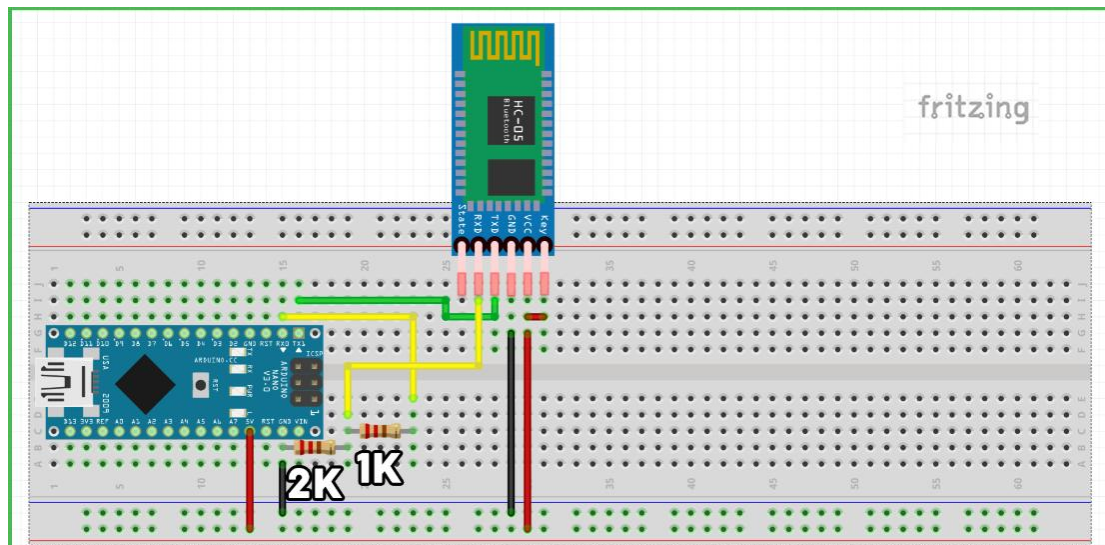
-PCB:



II. PHẦN MỀM

1. Cài đặt chế độ làm việc cho module bluetooth HC-05:

- Sơ đồ kết nối:



- Quy trình thực hiện:

+ Cấu hình module Slave:

- Tải một sketch rỗng lên Arduino. Ngắt dây kết nối serial trước khi upload, sau đó kết nối lại.
- Mở Serial Monitor, chọn Both NL & CR, đặt baud rate 38400.
- Gửi lệnh:
 ` AT: Kiểm tra kết nối (nhận OK).
 AT+UART?: Kiểm tra baud rate (mặc định 9600).
 AT+ROLE?: Kiểm tra chế độ (nhận 0, chế độ Slave).
 AT+ADDR?: Lấy địa chỉ module (ghi lại địa chỉ).

+ Cấu hình module Master:

- Gửi lệnh:
 AT: Kiểm tra kết nối (nhận OK).
 AT+UART?: Kiểm tra baud rate (nếu không phải 9600, dùng AT+UART=9600,0,0 để đặt lại).
 AT+ROLE=1: Đặt chế độ Master.
 AT+CMODE=0: Kết nối với địa chỉ cố định.
 AT+BIND=<địa chỉ module Slave>: Gắn địa chỉ Slave (dùng dấu , thay cho dấu :).

Khởi động lại cả hai module. LED sẽ nhấp nháy mỗi 2 giây khi ghép đôi thành công.

2. Master:

```
#include <EasyTransfer.h> // Thư viện EasyTransfer để giao tiếp giữa các Arduino
#include <Adafruit_ST7735.h> // Thư viện điều khiển màn hình TFT ST7735
#include <Adafruit_GFX.h> // Thư viện hỗ trợ đồ họa cơ bản
// Các chân kết nối của màn hình TFT
#define TFT_CS 10
#define TFT_RST 8
#define TFT_DC 9
#define TFT_SCLK 13
#define TFT_MOSI 11
char temperatureChar[10]; // Chuỗi chứa giá trị nhiệt độ dạng ký tự
char humidityChar[10]; // Chuỗi chứa giá trị độ ẩm dạng ký tự
// Cấu trúc dữ liệu để nhận dữ liệu thời tiết từ Arduino Slave
struct WEATHER_DATA_STRUCTURE
{
    float temperature ; // Giá trị nhiệt độ
    float humidity ; // Giá trị độ ẩm
};
// Cấu trúc dữ liệu để gửi tín hiệu xác nhận lại cho Slave
struct ACKNOWLEDGE
{
    boolean received = false; // Trạng thái xác nhận dữ liệu đã nhận
};
WEATHER_DATA_STRUCTURE data; // Biến chứa dữ liệu thời tiết
ACKNOWLEDGE acknowledge; // Biến chứa tín hiệu xác nhận
EasyTransfer ETin, ETout; // Đối tượng EasyTransfer để nhận và gửi dữ liệu
// Khởi tạo đối tượng điều khiển màn hình TFT
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);
void setup() {
    pinMode(5, OUTPUT); // Cấu hình chân 5 là đầu ra
    Serial.begin(9600); // Khởi tạo giao tiếp Serial với baud rate 9600
```

```

tft.initR(INITR_BLACKTAB); // Khởi tạo màn hình TFT
tft.fillScreen(ST7735_BLACK); // Đổ màu nền đen cho màn hình
printUI(); // Hiển thị giao diện tĩnh trên màn hình
ETin.begin(details(data), &Serial); // Bắt đầu nhận dữ liệu từ Slave
ETout.begin(details(acknowledge), &Serial); // Bắt đầu gửi tín hiệu xác nhận
}

void loop() {
    // Kiểm tra nếu nhận được dữ liệu từ Slave
    if(ETin.receiveData()){
        // Chuyển đổi giá trị nhiệt độ sang chuỗi ký tự và hiển thị
        String temperatureString = String(data.temperature,1); // Chuyển float -> String
        temperatureString.toCharArray(temperatureChar,10); // Chuyển String -> char array
        tft.fillRect(10,26,80,36,ST7735_BLACK); // Xóa khu vực hiển thị cũ
        printText(temperatureChar, ST7735_WHITE,10,26,3); // Hiển thị nhiệt độ mới
        // Chuyển đổi giá trị độ ẩm sang chuỗi ký tự và hiển thị
        String humidityString = String(data.humidity,1); // Chuyển float -> String
        humidityString.toCharArray(humidityChar,10); // Chuyển String -> char array
        tft.fillRect(10,101,80,106,ST7735_BLACK); // Xóa khu vực hiển thị cũ
        printText(humidityChar, ST7735_WHITE,10,101,3); // Hiển thị độ ẩm mới
        acknowledge.received = 1; // Gửi tín hiệu xác nhận dữ liệu đã nhận
        ETout.sendData(); // Gửi dữ liệu xác nhận về Slave
    }

    delay(200); // Tạo độ trễ 200ms
    acknowledge.received = 0; // Reset tín hiệu xác nhận
}

// Hàm hỗ trợ hiển thị chuỗi ký tự trên màn hình TFT
void printText(char *text, uint16_t color, int x, int y,int textSize)
{
    tft.setCursor(x, y); // Đặt vị trí con trỏ
    tft.setTextColor(color); // Đặt màu chữ
    tft.setTextSize(textSize); // Đặt kích thước chữ
    tft.setTextWrap(true); // Bật chế độ tự động xuống dòng
    tft.print(text); // Hiển thị chuỗi

```

```
// Hàm hiển thị giao diện tĩnh (static UI) trên màn hình TFT
```

```
void printUI()
```

```
{
```

```
    // Hiển thị tiêu đề nhiệt độ
```

```
    printText("TEMPERATURE", ST7735_WHITE,30,5,1);
```

```
    printText("o", ST7735_WHITE,90,19,2);
```

```
    printText("C", ST7735_WHITE,105,26,3);
```

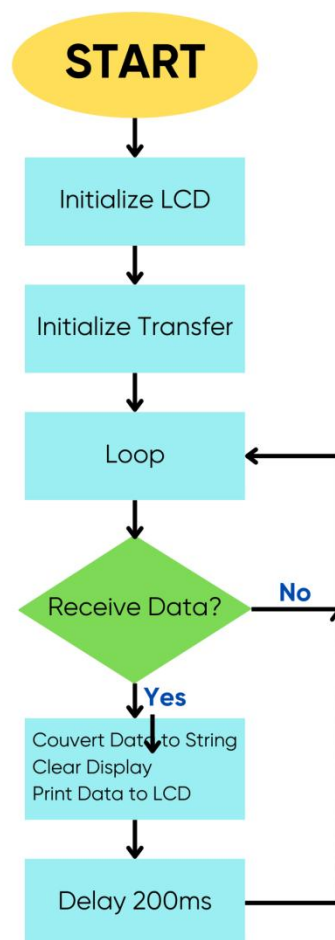
```
    // Hiển thị tiêu đề độ ẩm
```

```
    printText("HUMIDITY", ST7735_WHITE,30,86,1);
```

```
    printText("%", ST7735_WHITE,90,101,3);
```

```
}
```

- Lưu đồ giải thuật:



-Giải thích lưu đồ thuật toán:

+ Khởi tạo hệ thống bao gồm khai báo các thư viện cần thiết như EasyTransfer, Adafruit_ST7735 và Adafruit_GFX. Các biến toàn cục được khai báo để lưu trữ dữ liệu nhiệt độ, độ ẩm và trạng thái xác nhận. Màn hình TFT LCD được khởi tạo và thiết lập các chân kết nối cần thiết. Trong phần cài đặt hệ thống, thiết lập các chân đầu ra và bắt đầu giao tiếp Serial với tốc độ 9600 baud.

+ Màn hình TFT LCD được làm mới và vẽ giao diện người dùng với các chỉ số nhiệt độ và độ ẩm.

+ Các module EasyTransfer cho việc nhận (ETin) và gửi (ETout) dữ liệu được khởi tạo với cấu trúc dữ liệu tương ứng. Trong vòng lặp loop(), chương trình kiểm tra xem có dữ liệu từ DHT22 được gửi qua EasyTransfer hay không. Nếu dữ liệu được nhận, dữ liệu nhiệt độ và độ ẩm được chuyển đổi thành chuỗi ký tự và hiển thị lên màn hình TFT.

+ Biến trạng thái xác nhận acknowledge.received được đặt thành 1 và gửi ngược lại qua EasyTransfer để xác nhận rằng dữ liệu đã được nhận. Sau đó, chương trình đợi trong 200ms và đặt lại biến trạng thái xác nhận acknowledge.received về 0 để chuẩn bị cho lần nhận dữ liệu tiếp theo.

+ Hàm printText(char *text, uint16_t color, int x, int y, int textSize) dùng để đặt vị trí con trỏ, màu sắc, kích thước chữ và in chuỗi ký tự lên màn hình TFT.

+ Hàm printUI() dùng để vẽ giao diện người dùng tĩnh với các tiêu đề TEMPERATURE và HUMIDITY

3. Slave:

```
#include "DHT.h" // Thư viện điều khiển cảm biến DHT (nhiệt độ, độ ẩm)

#include <EasyTransfer.h> // Thư viện EasyTransfer để giao tiếp dữ liệu giữa hai Arduino

// Định nghĩa chân kết nối cảm biến DHT và loại cảm biến (DHT22)

#define DHTPIN 11 // Chân đọc dữ liệu từ cảm biến DHT

#define DHTTYPE DHT22 // Loại cảm biến DHT sử dụng (DHT22)

// Cấu trúc dữ liệu để gửi thông tin nhiệt độ và độ ẩm

struct SEND_DATA_STRUCTURE

{

float temperature; // Giá trị nhiệt độ

float humidity; // Giá trị độ ẩm

};

// Cấu trúc dữ liệu để nhận tín hiệu xác nhận từ Arduino Master

struct ACKNOWLEDGE

{

boolean received = false; // Trạng thái xác nhận dữ liệu đã nhận

};

int counter = 0; // Bộ đếm để xác định thời điểm gửi dữ liệu

SEND_DATA_STRUCTURE data; // Biến chứa dữ liệu nhiệt độ và độ ẩm

ACKNOWLEDGE acknowledge; // Biến chứa tín hiệu xác nhận

DHT dht(DHTPIN, DHTTYPE); // Đối tượng điều khiển cảm biến DHT

EasyTransfer ETin, ETout; // Hai đối tượng EasyTransfer: một để nhận và một để gửi dữ liệu

void setup() {

Serial.begin(9600); // Thiết lập tốc độ baud cho giao tiếp Serial (Bluetooth)

dht.begin(); // Khởi tạo cảm biến DHT

ETout.begin(details(data), &Serial); // Khởi tạo EasyTransfer để gửi dữ liệu

ETin.begin(details(acknowledge), &Serial); // Khởi tạo EasyTransfer để nhận tín hiệu xác nhận

}

void loop() {

// Kiểm tra nếu nhận được tín hiệu xác nhận từ Master

if(ETin.receiveData())

{

// Nếu tín hiệu xác nhận đã nhận thành công

if(acknowledge.received == true)

{

digitalWrite(13, HIGH); // Bật LED trên chân 13 để báo hiệu nhận thành công
```



```

delay(100); // Giữ LED sáng trong 100ms

digitalWrite(13, LOW); // Tắt LED

}

}

counter++; // Tăng bộ đếm

if(counter == 8) // Cứ sau 8 lần lặp (2 giây) thì gửi dữ liệu

{

    data.temperature = dht.readTemperature(); // Đọc giá trị nhiệt độ từ cảm biến DHT

    data.humidity = dht.readHumidity(); // Đọc giá trị độ ẩm từ cảm biến DHT

    ETout.sendData(); // Gửi dữ liệu đến Arduino Master

    counter = 0; // Reset bộ đếm

}

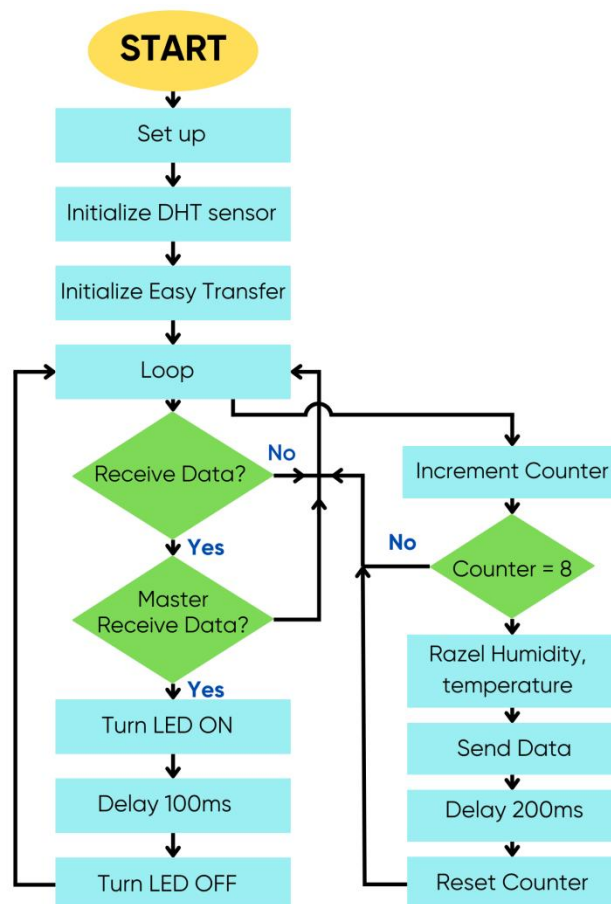
delay(250); // Độ trễ 250ms giữa mỗi lần lặp

acknowledge.received = false; // Reset tín hiệu xác nhận

}

```

- Lưu đồ giải thuật:



- Giải thích thuật toán:

+ Khởi tạo:

- Khai báo các thư viện cần thiết và các biến toàn cục. Bao gồm các thư viện DHT, DHT_U và EasyTransfer, cùng với các biến để lưu trữ nhiệt độ, độ ẩm và trạng thái xác nhận.
- Khởi Tạo Cảm Biến DHT: Cảm biến DHT được khởi tạo để bắt đầu đọc dữ liệu nhiệt độ và độ ẩm.
- Khởi Tạo EasyTransfer: Giao thức EasyTransfer được khởi tạo cho cả việc gửi (ETout) và nhận (ETin) dữ liệu.

+ Vòng lặp chính:

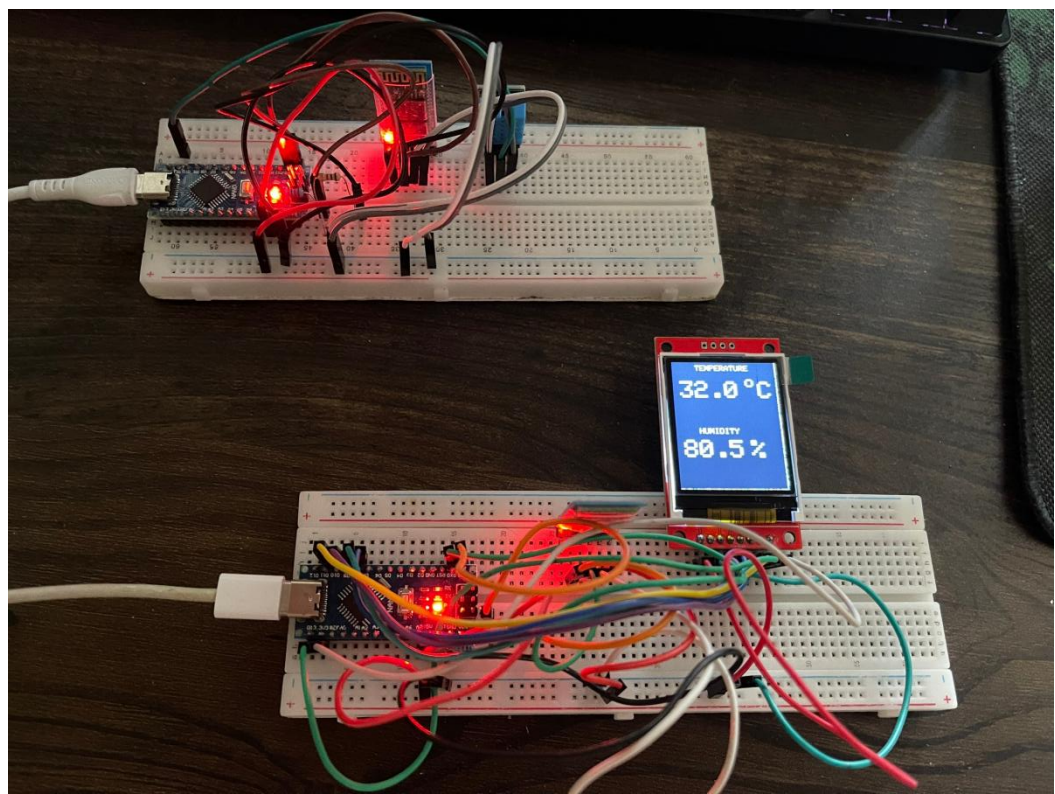
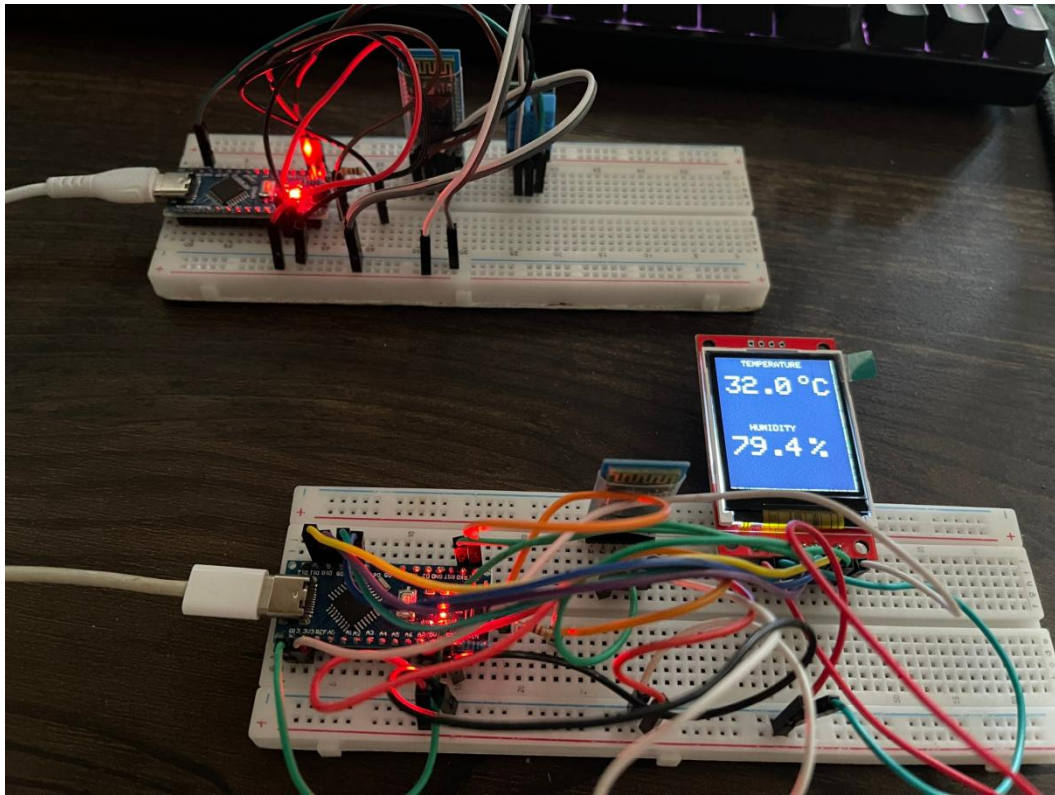
- Vòng lặp chính liên tục kiểm tra dữ liệu mới và xử lý giao tiếp giữa các module master và slave.
- Nhận Dữ Liệu: Thuật toán kiểm tra xem có dữ liệu nào được nhận từ module master thông qua EasyTransfer hay không.
- Acknowledge.received == true: Nếu dữ liệu được nhận và trạng thái xác nhận là true, điều này cho thấy master đã gửi dữ liệu thành công.
 - Bật Đèn LED: Đèn LED được bật lên để báo hiệu dữ liệu đã được nhận thành công.
 - Delay 100ms: Đèn LED sáng trong 100 mili giây.
 - Tắt Đèn LED: Sau đó đèn LED được tắt.

+ Xử lý dữ liệu:

- Tăng Biến Đếm: Biến đếm được tăng lên để theo dõi khoảng thời gian gửi dữ liệu.
- Counter == 8: Kiểm tra biến đếm xem có đạt đến 8 (tương đương 2 giây) hay không.
 - Đọc Nhiệt Độ: Nếu biến đếm bằng 8, dữ liệu nhiệt độ được đọc từ cảm biến DHT.
 - Đọc Độ Ẩm: Tương tự, dữ liệu độ ẩm được đọc từ cảm biến DHT.
 - Gửi Dữ Liệu Qua EasyTransfer: Dữ liệu nhiệt độ và độ ẩm sau đó được gửi đến module master qua EasyTransfer.
 - Đặt Lại Biến Đếm: Biến đếm được đặt lại về 0 sau khi gửi dữ liệu.
- Delay 250ms: Thuật toán chèn thêm một khoảng delay 250 mili giây để xử lý và truyền dữ liệu.
- Đặt lại acknowledge.received = false: Trạng thái xác nhận được đặt lại về false để chuẩn bị cho chu kỳ nhận dữ liệu tiếp theo.

CHƯƠNG III: THI CÔNG VÀ KẾT QUẢ

I. KẾT QUẢ SẢN PHẨM



1. Nhóm tự đánh giá sản phẩm

a. Khoảng cách truyền nhận

- Hệ thống thử nghiệm hoạt động ổn định trong phạm vi khoảng 5-10m trong không gian mở. Tuy nhiên, trong môi trường có vật cản (tường bê tông, đồ nội thất), phạm vi truyền nhận giảm xuống còn 3-5m. Kết quả này phù hợp với khả năng của module Bluetooth HC-50 được sử dụng trong thiết kế.

b. Nhận xét về hoạt động:

- Hệ thống hoạt động ổn định và dữ liệu nhiệt độ, độ ẩm được truyền nhận chính xác, không có lỗi mất dữ liệu trong các thử nghiệm thực tế.

- Màn hình LCD hiển thị thông tin nhanh chóng, các giá trị thay đổi gần như theo thời gian thực, với độ trễ chỉ khoảng 1-2 giây.

c. Ưu điểm:

- Thiết kế gọn gàng, dễ dàng lắp đặt trong thực tế.
- Sử dụng các linh kiện phổ biến, dễ tìm, giảm chi phí và thời gian lắp ráp.
- Giao diện hiển thị trực quan, dễ đọc, hỗ trợ tốt cho người dùng trong việc quan sát thông số môi trường.
- Có khả năng mở rộng, ví dụ: thêm các cảm biến khác (chất lượng không khí, ánh sáng, v.v.) hoặc tích hợp IoT để gửi dữ liệu lên cloud.

d. Khuyến nghị

- Nâng cấp module truyền nhận: Sử dụng các module truyền thông có tầm xa hơn, chẳng hạn như Wi-Fi hoặc LoRa, để tăng phạm vi kết nối.
- Tối ưu hóa năng lượng: Thêm các chế độ tiết kiệm năng lượng cho cảm biến và module truyền nhận để tăng hiệu suất sử dụng pin.
- Tích hợp thêm chức năng thông minh: Kết hợp hệ thống cảnh báo qua tin nhắn hoặc ứng dụng điện thoại khi thông số môi trường vượt ngưỡng an toàn.
- Mở rộng số lượng cảm biến: Tích hợp nhiều loại cảm biến hơn để thu thập dữ liệu môi trường đa dạng và chi tiết hơn.

e. Khả năng cải tiến:

- Nâng cấp module truyền nhận: Sử dụng các module truyền thông có tầm xa hơn, chẳng hạn như Wi-Fi hoặc LoRa, để tăng phạm vi kết nối.
- Tối ưu hóa năng lượng: Thêm các chế độ tiết kiệm năng lượng cho cảm biến và module truyền nhận để tăng hiệu suất sử dụng pin.

- Tích hợp thêm chức năng thông minh: Kết hợp hệ thống cảnh báo qua tin nhắn hoặc ứng dụng điện thoại khi thông số môi trường vượt ngưỡng an toàn.

- Mở rộng số lượng cảm biến: Tích hợp nhiều loại cảm biến hơn để thu thập dữ liệu môi trường đa dạng và chi tiết hơn.