IOT102 - Báo cáo chuyên đề

HỆ THỐNG SMARTHOME

Lớp	SE1509	
Nhóm	5	
Thành viên	Vũ Thiên Ân	SE160296
	Trần Thành Công	SE150167
	Nguyễn Vĩ Khang	SE160221
	Nguyễn Đăng Lộc	SE160199
	Lê Nam Phú	SE160467
	Phạm Khắc Triệu	SE160232

Tháng 3, 2022

MỤC LỤC

MỤC LỤC **2**
ỨNG DỤNG CỦA IOT TRONG NÔNG NGHIỆP **4**
HỆ THỐNG SMARTHOME TRÊN ARDUINO **10**
1. Giới thiệu đề tài **10**
2. Hệ thống quạt tự động **11**
2.1. Mô tả 11
a) Tên dự án 11
b) Tổng quát chức năng 11
c) Nguyên lý hoạt động 11
2.2. Phần cứng 11
a) Các thành phần 11
b) Schematic design 12
2.3. Phần mềm 13
a) Flowchart 13
b) Source code 15
2.4. Triển khai lắp đặt và chạy thử 18
3. Hệ thống đèn LED RGB **20**
3.1. Mô tả 20
a) Tên dự án 20
b) Ứng dụng 20
c) Yêu cầu 20
d) Cách sử dụng 20
e) Ưu điểm 20
f) Nhược điểm 20
3.2. Phần cứng 20
a) Các thành phần 20
b) Đặc tính kỹ thuật 21
c) Schematic design 21
3.3. Phần mềm 21
a) Giải thích source code 21
b) Flowchart 22
c) Source code 23
3.4. Triển khai lắp đặt và chạy thử 24
a) Thiết kế lắp đặt 24
b) Lỗi có thể phát sinh trong lắp đặt: 26

c) Các bộ thử: 26
d) Hướng phát triển 26
4. Hệ thống phơi đồ thông minh 27
4.1. Mô tả 27
a) Nội dung dự án 27
b) Ứng dụng 27
c) Yêu cầu 27
d) Cách sử dụng 27
4.2. Phần cứng 29
a) Các thành phần 29
b) Đặc tính kỹ thuật 30
c) Schematic design 32
4.3. Phần mềm 33
a) Giải thích source code 33
b) Flowchart 36
c) Source code 38
4.4. Triển khai lắp đặt và chạy thử 49
a) Thiết kế lắp đặt 49
b) Chạy thử 51
4.5. Tổng kết 51
a) Kết luận 51
b) Hướng phát triển 52
PHỤ LỤC 53
Tổng hợp file dự án 53
Link video demo sản phẩm 53
Link flowchart 53
Link code Arduino của dự án 53
Link mạch arduino mô phỏng 53
TÀI LIỆU THAM KHẢO 54

ỨNG DỤNG CỦA IOT TRONG NÔNG NGHIỆP

Ngành nông nghiệp hiện nay lấy dữ liệu làm trung tâm, chính xác và thông minh hơn. Sự xuất hiện nhanh chóng của các công nghệ dựa trên IoT (Internet-of-Things) đã chuyển đổi ngành nông nghiệp thông minh từ phương pháp tiếp cận thống kê sang định lượng.

Tác động của IoT đối với nông nghiệp thông minh

Nông nghiệp thông minh là một hệ thống nông nghiệp ứng dụng các công nghệ số để giám sát, điều khiển và chăm sóc cây trồng tự động, giúp tối đa hóa năng suất và chất lượng nông sản.

Bằng cách triển khai các cảm biến, thiết bị IoT trong thực tiễn đã làm thay đổi mọi khía cạnh của phương pháp canh tác truyền thống. IoT giúp cải thiện các giải pháp canh tác truyền thống, ứng phó với hạn hán, tối ưu hóa năng suất, tính phù hợp đất đai, tưới tiêu và kiểm soát dịch hại. Có thể kể đến những cách thức ứng dụng IoT trong nông nghiệp phổ biến nhất như:

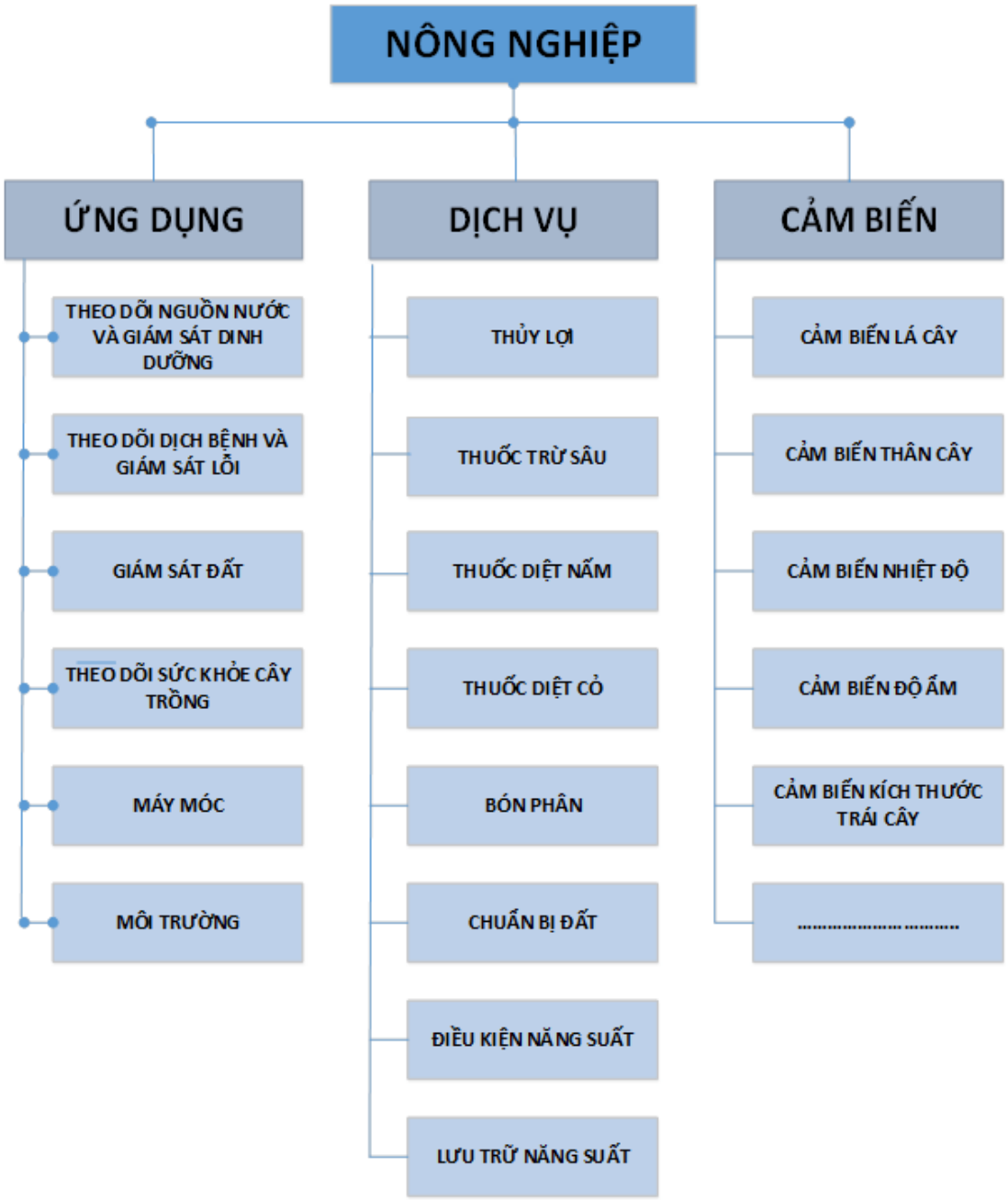
- Thu thập lượng dữ liệu khổng lồ bằng cảm biến nông nghệp thông minh (điều kiện thời tiết, chất lượng đất, nước, không khí,…)
- Phân tích dữ liệu, dự đoán các khả năng trong quá trình canh tác, thu hoạch và phân phối, giảm thiểu rủi ro cho đầu ra.
- Quản lý rủi ro tốt hơn trong quá trình trồng trọt, chăn nuôi nhờ dữ kiện, thông tin trích xuất bởi các thiết bị IoT.
- Tự động hoá các quy trình vận hành trang trại, vườn, nhà kính, tiến hành cùng lúc nhiều công việc.
- Tăng chất lượng, sản lượng của sản phẩm đầu ra nhờ ứng dụng công nghệ kỹ thuật hiện đại xuyên suốt quá trình.

Ứng dụng cụ thể

1. Giám sát điều kiện khí hậu

Tiện ích IoT phổ biến nhất là các trạm thời tiết, kết hợp các cảm biến canh tác thông minh khác nhau. Thiết bị IoT thu thập nhiều dữ liệu từ môi trường và

gửi lên đám mây. Các phép đo đạc, tính toán được cung cấp được sử dụng để lập bản đồ các điều kiện khí hậu, chọn loại cây trồng phù hợp và thực hiện các biện pháp cần thiết để cải thiện năng suất.



Hình 1-1. Hệ thống phân cấp chung của các ứng dụng, dịch vụ và cảm biến có cho nông nghiệp thông minh.

2. Lấy mẫu và bản đồ đất

Các bộ công cụ, cảm biến hỗ trợ nông dân theo dõi chất lượng đất, dựa trên dữ liệu này để đề xuất các biện pháp khắc phục tình trạng suy thoái đất. Các bộ công cụ này cho phép theo dõi các đặc tính của đất, chẳng hạn như kết cấu, khả

năng giữ nước và tỷ lệ hấp thụ để giúp giảm thiểu xói mòn, dày đặc, nhiễm mặn, axit hóa và ô nhiễm bằng cách tránh sử dụng quá nhiều phân bón.

Ví dụ:

- Vệ tinh Độ ẩm của đất và độ mặn của đại dương SMOS (Soil Moisture and Ocean Salinity) đã được phóng vào năm 2009 để cung cấp bản đồ độ ẩm của đất sau 1-2 ngày.

3. Thủy lợi

Các phương pháp tưới tiêu truyền thống được sử dụng như nhỏ giọt và tưới phun không đều dẫn đến cây trồng bị thiếu nước, giảm chất dinh dưỡng trong đất, gây ra các bệnh vi sinh. Vì vậy, việc sử dụng các công nghệ kỹ thuật dựa trên IoT như: các cảm biến không dây, hệ thống giám sát và các ứng dụng phần mềm thông minh để thu thập, phân tích, kiểm tra độ ẩm đất và không khí sẽ tối ưu hóa được lượng nước, ước tính chính xác nhu cầu nước của cây trồng và khả năng giữ ẩm của đất tại từng thời điểm trong năm, cải thiện hiệu quả sử dụng nước.

4. Phân bón

Bón phân trong nông nghiệp thông minh giúp ước tính chính xác liều lượng chất dinh dưỡng cần thiết, giảm thiểu tác động tiêu cực đối với môi trường. Bón phân đòi hỏi các phép đo mức độ dinh dưỡng của đất cụ thể dựa trên các yếu tố khác nhau, chẳng hạn như loại cây trồng, loại đất, khả năng hấp thụ của đất, năng suất sản phẩm, loại màu mỡ, tỷ lệ sử dụng và điều kiện thời tiết,... Các phương pháp bón phân dựa trên IoT mới giúp ước tính các mô hình không gian về nhu cầu của chất dinh dưỡng với độ chính xác cao hơn.

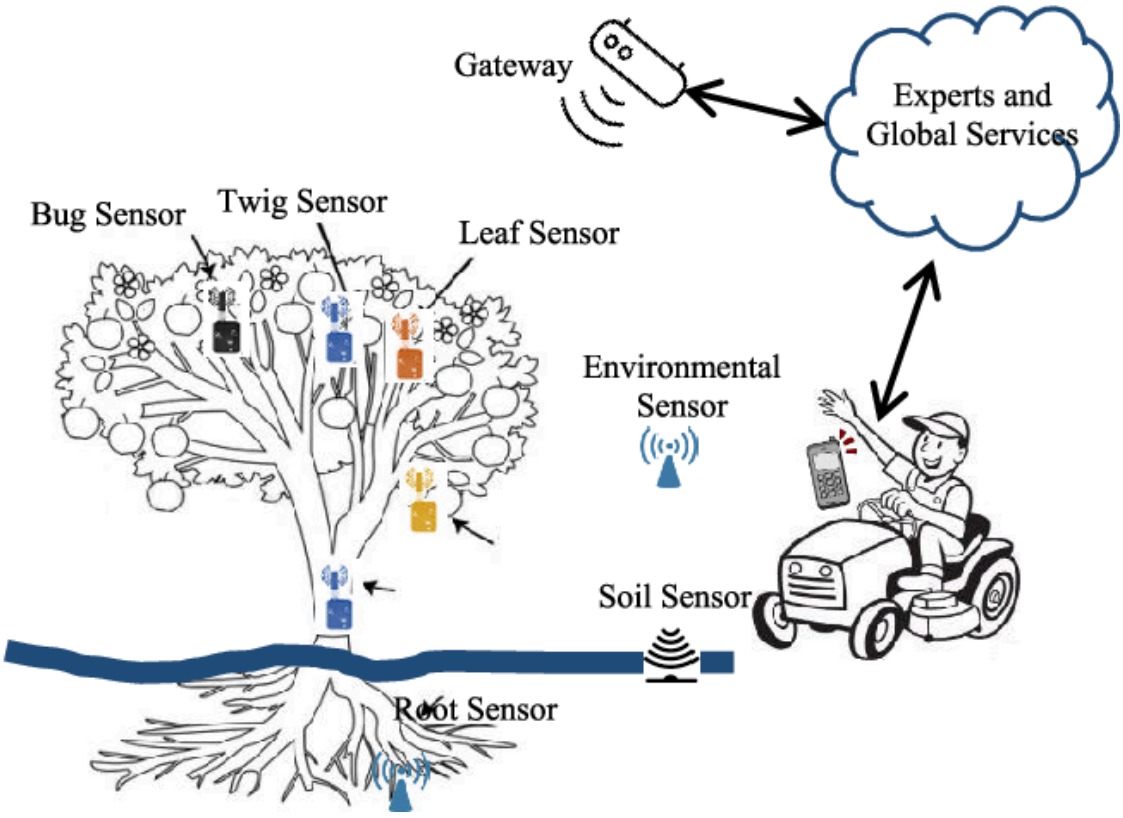
5. Quản lý bệnh cây trồng

Các thiết bị IoT như cảm biến không dây, robot phát hiện khá chính xác sâu bệnh của cây trồng. So với quy trình kiểm soát dịch hại dựa trên truyền thống thì quản lý dịch hại dựa trên IoT cung cấp khả năng giám sát, mô hình hóa, dự báo dịch bệnh theo thời gian thực hiệu quả hơn. Các phương pháp tiếp cận nhận

dịch bệnh và sâu bệnh tiên tiến dựa trên việc xử lý hình ảnh, trong đó các hình ảnh thô được thu thập bằng cách sử dụng cảm biến hiện trường, máy bay không người lái UAV (Unmanned Aerial Vehicle) hoặc vệ tinh viễn thám. Ví dụ: bẫy tự động dựa trên IoT có thể bắt, đếm và thậm chí xác định đặc điểm của các loại côn trùng, sau đó đẩy dữ liệu về sâu bệnh lên đám mây.

6. Theo dõi, dự báo và thu hoạch

Dự báo mùa màng là một nghệ thuật để dự đoán năng suất và sản lượng (tấn/ha) trước khi vụ thu hoạch diễn ra. Dự báo này giúp người nông dân lập kế hoạch và ra quyết định về chất lượng sản phẩm để xác định thời điểm thu hoạch thích hợp. Việc giám sát này bao gồm các giai đoạn phát triển khác nhau. Dự đoán đúng thời điểm thu hoạch không chỉ giúp tối đa hóa sản lượng và chất lượng cây trồng mà còn tạo cơ hội điều chỉnh chiến lược quản lý. Hình 4 thể hiện ảnh chụp nhanh của mạng khu vực nông trại (FAN) có thể hiển thị toàn bộ trang trại của người nông dân.



Hình 1-2. Mạng khu vực trang trại dựa trên IoT FAN (Farm Area Network)

Một bộ theo dõi năng suất, có thể được cài đặt trên bất kỳ tổ hợp máy gặt nào và được liên kết với ứng dụng di động FarmRTX, hiển thị dữ liệu thu hoạch

trực tiếp và tự động tải lên nền tảng dựa trên web của nhà sản xuất. Ứng dụng này có khả năng tạo bản đồ năng suất và chia sẻ các bản đồ này với nhà nông học và người nông dân, sau đó được tùy chọn xuất sang phần mềm quản lý trang trại khác để phân tích chúng.

7. Giám sát và quản lý gia súc

Giống như theo dõi cây trồng, có cảm biến nông nghiệp IoT có thể gắn vào các động vật trong trang trại để theo dõi hiệu suất sức khỏe và nhật ký của chúng.

8. Máy bay không người lái trong nông nghiệp

Có lẽ một trong những tiến bộ Agritech hứa hẹn nhất là việc sử dụng máy bay không người lái nông nghiệp trong chăn nuôi thông minh. Còn được gọi là UAV (Xe không khí không người lái), máy bay không người lái được trang bị tốt hơn so với máy bay và vệ tinh để thu thập dữ liệu nông nghiệp. Ngoài khả năng giám sát, máy bay không người lái cũng có thể thực hiện một số lượng lớn các nhiệm vụ cần thiết trước đây cần sức người: trồng trọt, chống sâu bệnh, phun nông nghiệp, giám sát cây trồng, v.v…

Sự tiến bộ của công nghệ robot mang đến những giải pháp mới. Khi trang bị cho một robot nông nghiệp với các thiết bị cảm biến đa hình ảnh và vòi phun chính xác, có thể xác định vị trí và phương pháp đối phó các vấn đề dịch hại chính xác hơn dưới sự điều khiển của hệ thống quản lý dịch bệnh IoT từ xa.

9. Hệ thống quản lý trang trại End-to-end

Một cách tiếp cận phức tạp hơn đối với các sản phẩm IoT trong nông nghiệp có thể được đại diện bởi cái gọi là hệ thống quản lý năng suất trang trại. Chúng thường bao gồm một số thiết bị và cảm biến IoT nông nghiệp, được cài đặt trong khuôn viên cũng như bảng điều khiển mạnh mẽ với khả năng phân tích và các tính năng kế toán / báo cáo được xây dựng.

- *Nhà kính thông minh:* Kiểm soát ánh sáng, nhiệt độ, độ ẩm, khí CO_2 của nhà kính thông qua máy tính/điện thoại theo thời gian thực để duy trì

tối đa môi trường phát triển của vụ mùa bằng cách vận hành tự động từ xa đóng mở cửa sổ, cung cấp chất dinh dưỡng.

- *Vườn cây thông minh:* Kiểm soát từ xa, tự động tưới và kiểm soát lượng thuốc trừ sâu.
- *Chuồng trại thông minh:* Kiểm soát môi trường chuồng trại như nhiệt độ, độ ẩm thông qua máy tính/điện thoại; Tự động hệ thống hẹn giờ và lượng cung cấp thức ăn và nước uống.

Kết luận

Để tạo ra nền nông nghiệp bền vững nói riêng và sự phát triển kinh tế bền vững nói chung, việc ứng dụng IoT sẽ là trung tâm hàng đầu trong các hoạt động nông nghiệp. IoT sắp xếp hợp lý cách làm việc từ sử dụng tài nguyên nước, khí đất và điện, vận chuyển cây trồng, cảnh báo vận hành, bảo trì máy móc nông trại. IoT đã chứng tỏ một bước đột phá và tiếp tục thay đổi cách nhìn vào các hoạt động nông nghiệp khác nhau, trên thế giới ước tính có hơn 75 triệu thiết bị dựa trên IoT sẽ hoạt động trong ngành nông nghiệp vào năm 2020. Trong tương lai, IoT có thể được định hình bởi những tiến bộ vượt bậc trong Mạng cảm biến không dây WSN (Wireless Sensor Network - WSN) và thế hệ thứ 5 của công nghệ thông tin di động (5G) để cung cấp cho nông dân dữ liệu và thông tin theo thời gian thực mọi lúc mọi nơi trên đất của họ.

Các công cụ phân tích dữ liệu giúp làm cho nông nghiệp, vốn có phụ thuộc nhiều vào điều kiện thời tiết, dễ quản lý và có thể dự đoán hơn.

HỆ THỐNG SMARTHOME TRÊN ARDUINO

1. Giới thiệu đề tài

Internet of Things (IoT) là xu hướng đang được các doanh nghiệp trong lĩnh vực công nghệ quan tâm và đầu tư nguồn lực nghiên cứu. Cuộc đua IoT đã và đang diễn ra mạnh mẽ giữa các doanh nghiệp, các quốc gia trên toàn thế giới. Một trong những ứng dụng lớn lao và thông dụng nhất của IoT chính là Smarthome. Ngày càng có nhiều công trình nhà ở, nhiều hộ gia đình lựa chọn xây dựng ngôi nhà của mình theo hướng tích hợp các thiết bị IoT để biến ngôi nhà thành một thứ thật tiện nghi và thông minh, phụ giúp cho con người bằng những cơ chế tự động. Sự xuất hiện của Arduino vào năm 2005 tại Italia đã mở ra một hướng đi mới cho lập trình vi điều khiển, là cơ sở để phát triển nên các thiết bị Smarthome. Arduino đã hỗ trợ cho con người rất nhiều trong lập trình và thiết kế mạch điện tử, nhất là đối với những người bắt đầu tìm tòi về vi điều khiển mà không có quá nhiều kiến thức, hiểu biết sâu sắc về vật lý và điện tử . Phần cứng của thiết bị đã được tích hợp nhiều chức năng cơ bản và là mã nguồn mở, tương thích với ngôn ngữ C và hệ thư viện rất phong phú và được chia sẻ miễn phí. Chính vì những lý do như vậy nên Arduino đang dần phổ biến và phát triển ngày càng mạnh mẽ trên toàn thế giới.

Trên cơ sở kiến thức đã học trong môn học: IOT102 – Internet of Things cùng sự tự tìm hiểu, nghiên cứu về các thiết bị điện tử, nhóm đã thực hiện đề tài: Hệ thống Smarthome trên board Arduino với 3 ứng dụng cơ bản cụ thể. Đề tài gồm các nội dung sau:

Chương 1. Hệ thống quạt thông tự động

Chương 2. Hệ thống đèn LED RGB tự động

Chương 3. Hệ thống phơi đồ thông minh

2. Hệ thống quạt tự động

Link mô phỏng trên Tinkercad: Automatic Smart Fan | Tinkercad

Link video demo: Demo hệ thống quạt tự động

2.1. Mô tả

a) Tên dự án

Quạt thông minh tự động điều khiển dựa trên điều kiện nhiệt độ

b) Tổng quát chức năng

Hệ thống quạt tự điều chỉnh theo nhiệt độ môi trường hoặc theo tín hiệu từ remote.

Thông tin được hiển thị trên LCD bao gồm:

- Nhiệt độ môi trường theo đơn vị Celsius ($°C$)
- Chế độ tự động đang bật hay tắt
- Nguồn của quạt đang bật hay tắt
- Tốc độ của quạt.

Kết quả hiển thị trên màn hình LCD, được cập nhật mỗi 1 giây.

c) Nguyên lý hoạt động

Chế độ điều khiển tự động:

- Tự động bật quạt nếu như nhiệt độ > $25°C2$
- Thay đổi tốc độ của quạt theo nhiệt độ T:
  + *Tốc độ 1:* $25°C < T \leq 30°C$
  + *Tốc độ 2:* $30°C < T \leq 35°C$
  + *Tốc độ 3:* $T > 35°C$

Chế độ điều khiển với remote:

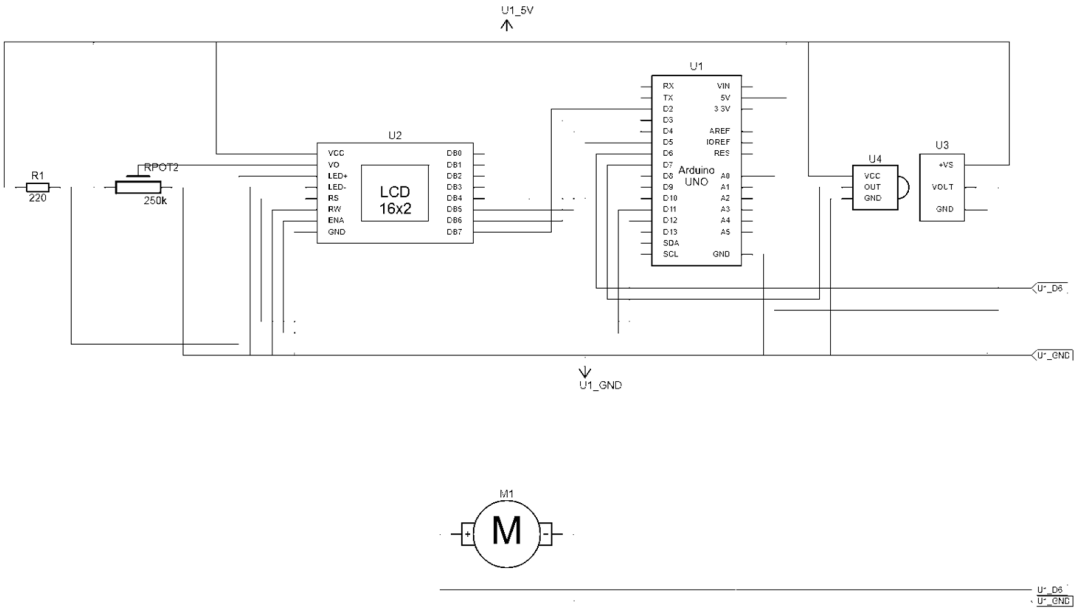
- Nhấn nút đỏ trên remote để bật/tắt nguồn cho quạt
- Nhấn nút "FUNC/STOP" để bật/tắt chế độ điều khiển tự động
- Nhấn "<<" để giảm tốc quạt,
- Nhấn ">>" để tăng tốc độ quạt, tốc độ tối đa là 3

2.2. Phần cứng

a) Các thành phần

- 01 Bộ mạch Arduino
- 01 Breadboard
- 01 DC Motor
- 01 Cảm biến nhiệt độ TMP36
- 01 LCD 16 x 2
- 01 Potentiometer 250kΩ
- 01 Resistor 220Ω
- 01 IR remote
- 01 IR sensor

b) Schematic design



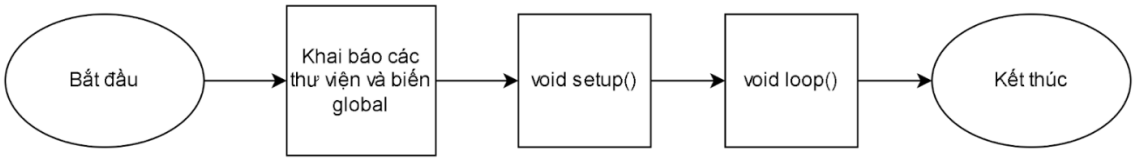
Hình 2-1. Schematic design của hệ thống quạt tự động

2.3. Phần mềm

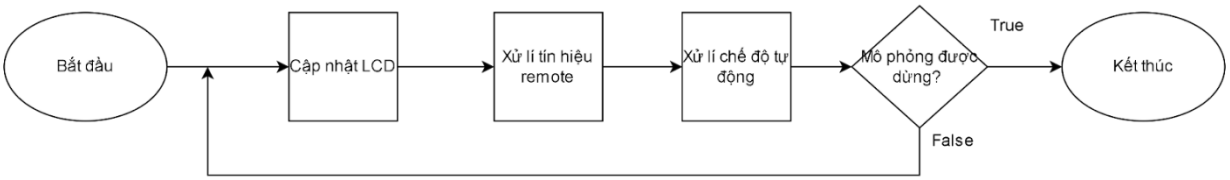
a) Flowchart

Link: [Flowchart hệ thống quạt tự động]

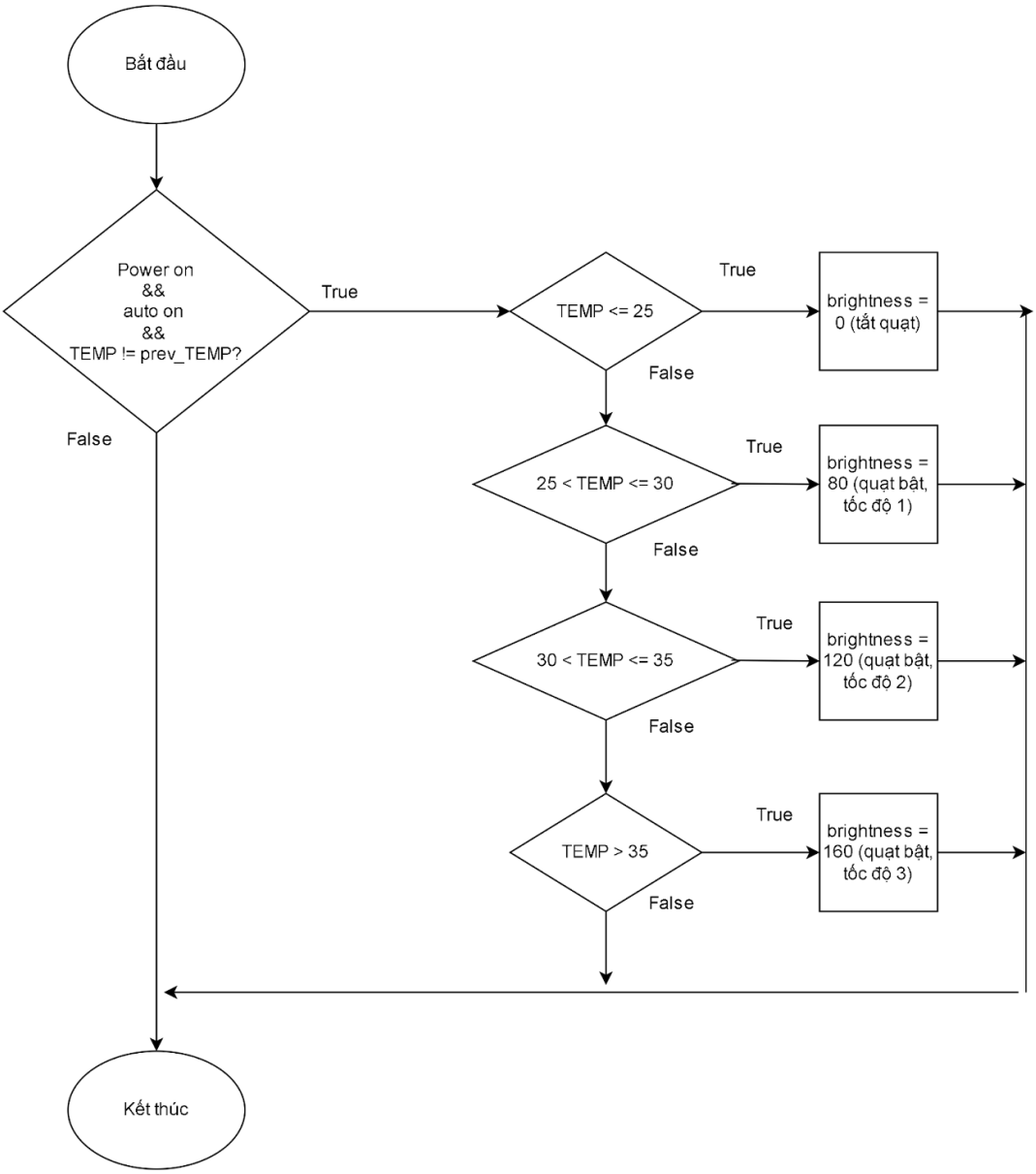
Flowchart tổng quát cho toàn mô phỏng (simulation):



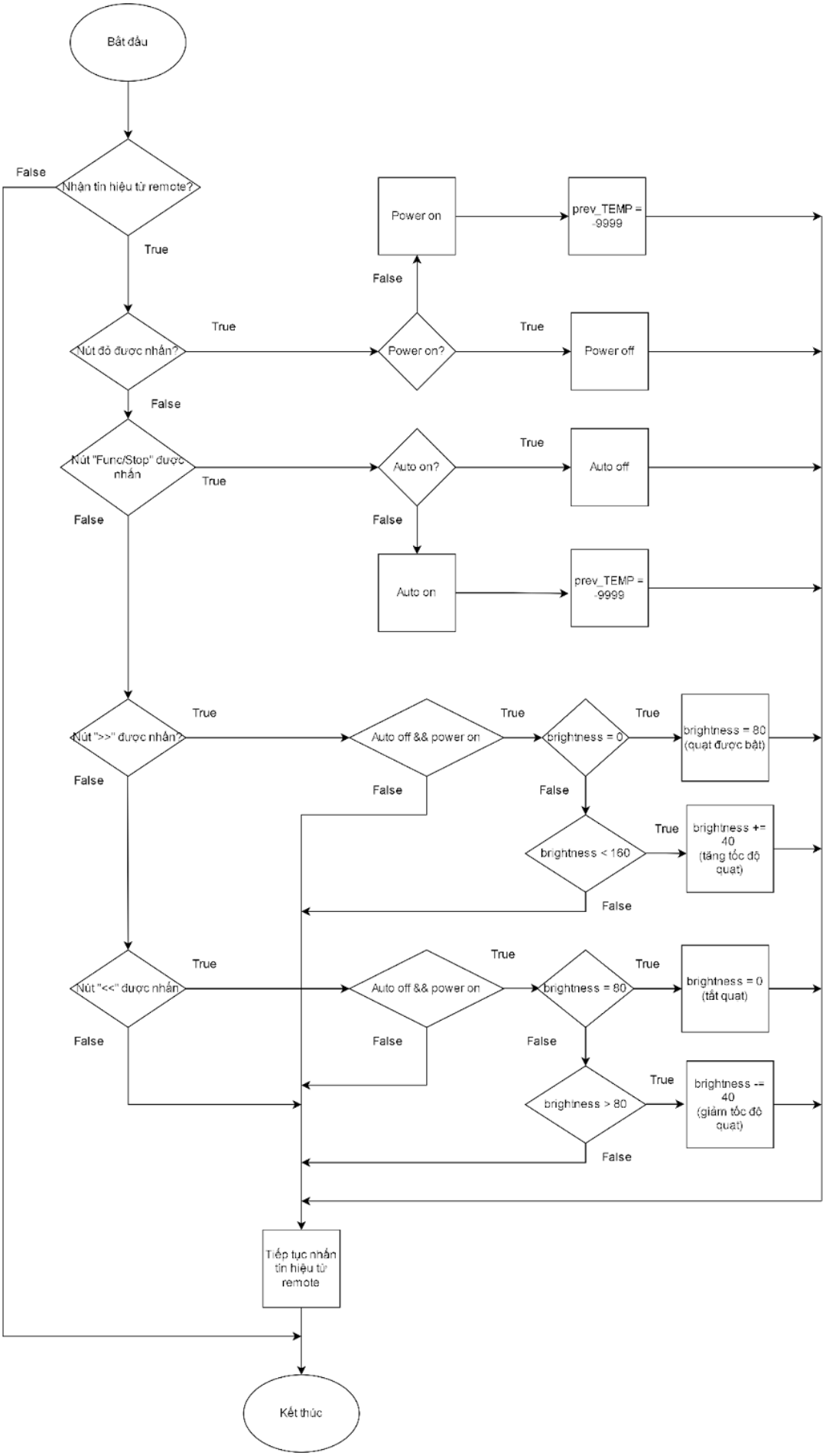
Flowchart cho function void loop()



Xử lý chế độ tự động



Xử lý tín hiệu remote



b) Source code

Ngôn ngữ: C++

Khai báo thư viện và các biến toàn cục:

```
//  ========== INCLUDE THƯ VIỆN ==========
#include <LiquidCrystal.h>  // thư viện cho lcd
#include <IRremote.h>   // thư viện cho remote

// ========== BIẾN CHO ỨNG DỤNG BẬT QUẠT ==========
float TEMP;     // biến cho nhiệt độ từ cảm biến
float prev_TEMP = -9999;  // biến lưu lại giá trị nhiệt độ cũ để so sánh
int motorPin = 6;   // kết nối motor với cổng ~6
int brightness = 0;     // biến cho power chuyển cho motor
bool power_on = true;     // kiem soat on/off nguồn cho quạt, default true ngay
tu dau
bool auto_on = true;      // kiem soát on/off che do auto, default true ngay tu
dau
int fan_speed = 48;       // fan speed ban đầu là 0, mã ascii 48

// ========== BIẾN CHO REMOTE ==========
const int RECV_PIN = 7;   // khai báo pin 7 nhận tín hiệu từ IR sensor
IRrecv irrecv(RECV_PIN);  // khai báo IRrecv object nhận input là pin 7
decode_results results;   // khai báo decode_results object

// ========== INITIALIZE THƯ VIỆN CHO LCD ==========
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

Hàm setup()

```
void setup() {
  // gửi OUTPUT đến pin cho motor
  pinMode(motorPin, OUTPUT);

  // bắt đầu serial monitor để theo dõi
  Serial.begin(9600);

  // chuẩn bị receiver cho IR remote
  irrecv.enableIRIn();    // bắt dầu nhận tín hiệu từ pin 7
  irrecv.blink13(true);   // blink led trên adruino khi nhận tín hiệu

  // chuẩn bị lcd
  lcd.clear();
  lcd.begin(16, 2);   // chuẩn bị số lượng (cột, hàng) cho lcd
  lcd.setCursor(0, 0);    // di chuyển con trỏ đến đầu hàng 1
  lcd.print("Dieu chinh quat");
  lcd.setCursor(0, 1);    // di chuyển con trỏ đến đầu hàng 2
  lcd.print("theo nhiet do");
  delay(2000);
  lcd.clear();
}
```

Hàm loop()

```
void loop() {

  // ========== C P NH T LCD ==========
  float T = analogRead(0) * 0.004882814; // đọc giá trị từ cảm biến nhiệt
```

```
  // rồi nhân với (5V / 1024) để chuyển từ analog thành digital
  float TEMP = (T - 0.5) * 100;   // Tính TEMP theo Celsius để hiện lên LCD
  lcd.setCursor(0, 0);
  lcd.print("T: ");
  lcd.setCursor(3, 0);
  lcd.print(TEMP);
  lcd.print(" do C");
  delay(1000);
  lcd.clear();

  lcd.setCursor(0, 0);
  if (power_on) {
    lcd.print("Pow ON");
  } else {
    lcd.print("Pow OFF");
  }
  if (auto_on) {
    lcd.print(" Aut ON");
  } else {
    lcd.print(" Aut OFF");
  }

  lcd.setCursor(0, 1);
  if (brightness != 0) {
    lcd.print("Fan ON Speed ");
    lcd.print((char)fan_speed);
  } else {
    lcd.print("Fan OFF");
  }
  delay(1500);
  lcd.clear();
```

Xử lý tín hiệu remote

```
// ========== XỬ LÝ TÍN HIỆU REMOTE ==========
  if (irrecv.decode(&results)) {
    switch (results.value) {

      // chỉnh on/off quạt, nút đỏ
      case 16580863:
        if (power_on) {
          power_on = false;
          Serial.println("power off");
          // cập nhật brightness = 0 => tắt quạt
          brightness = 0;
          fan_speed = 48;
          // cập nhật power cho motorPin
          analogWrite(motorPin, brightness);
          delay(500);
        } else {
          power_on = true;
          Serial.println("power on");
          prev_TEMP = -9999;  // reset prev_TEMP khi power on trở lại
        }
        break;

      // chỉnh on/off chế độ auto
      case 16597183:
        if (auto_on) {
          // tat auto => tat quat
```

```
          auto_on = false;
          brightness = 0;
          fan_speed = 48;
          // cập nhật power cho motorPin
          analogWrite(motorPin, brightness);
          delay(500);
          Serial.println("auto off");
        } else {
          auto_on = true;
          Serial.println("auto on");
          prev_TEMP = -9999;  // reset prev_TEMP khi auto on trở lại
        }
        break;

      // tăng tốc độ quạt với nút >> nếu auto off và power on
      case 16605343:
        if (!auto_on && power_on) {
          if (brightness == 0) {  // nêu quạt đang tắt => bật quạt
            brightness = 80;
            fan_speed += 1;
          } else {
            if (brightness < 160) { // chỉ tăng nếu như brightness < 160
              brightness += 40;
              fan_speed += 1;
            }
          }
          // cập nhật power cho motorPin
          analogWrite(motorPin, brightness);
          delay(500);
        }
        break;

      // giảm tốc độ quạt với nút << nếu auto off và power on
      case 16589023:
        if (!auto_on && power_on) {
          if (brightness == 80) { // nếu ở mức thấp nhất => tắt quạt
            brightness = 0;
            fan_speed -= 1;
          } else {
            if (brightness > 80) {
              brightness -= 40;
              fan_speed -= 1;
            }
          }
          // cập nhật power cho motorPin
          analogWrite(motorPin, brightness);
          delay(500);
        }
        break;
    }
    // remote receiver tiếp tục nhận tín hiệu
    irrecv.resume();
  }
```

Xử lý chế độ tự động

```
// ========== XỬ LÍ CHẾ ĐỘ AUTO ==========
  // Kiểm tra power và auto đều on hay không, nhiệt độ từ sensor
  // có đổi hay không (prev_TEMP != TEMP)
  if (power_on && auto_on && prev_TEMP != TEMP) {

    if (TEMP <= 25) {
      brightness = 0;  // tắt quạt
```

```
      fan_speed = 48; // cập nhật fan_speed để in ra lcd
    }

    if (TEMP > 25 && TEMP <= 30) {
      brightness = 80;  // đổi tốc độ quạt thành 1
      fan_speed = 49; // cập nhật fan_speed để in ra lcd
    }

    if (TEMP > 30 && TEMP <= 35) {
      brightness = 120; // đổi tốc độ quạt thành 2
      fan_speed = 50; // cập nhật fan_speed để in ra lcd
    }

    if (TEMP > 35) {
      brightness = 160; // đổi tốc độ quat thành 3
      fan_speed = 51; // cập nhật fan_speed để in ra lcd
    }

    // cập nhật power cho motorPin
    analogWrite(motorPin, brightness);
    delay(500);
    // cập nhật giá trị cho prev_TEMP
    prev_TEMP = TEMP;
  }
}
```

2.4. Triển khai lắp đặt và chạy thử

Lắp đặt LCD

- Từ LCD, nối chân LED Anode với resistor (resistor nối với nguồn 5V) và LED Cathode nối GND
- Nối chân DB4, DB5, DB6, DB7 từ mạch LCD đến Arduino thông qua mạch nhựa vào pin digital số 5,4,3,2 theo đúng thứ tự
- Nối chân E vào pin số 11 của Arduino
- Nối chân RW với GND và RS nối vào pin số 12

Lắp đặt potentiometer

- Lắp Potentiometer vào mạch nhựa
- Nối chân terminal 1 với nguồn 5V
- Nối chân terminal 2 với chân GND của LCD
- Nối chân wiper với chân V0 của LCD

Lắp đặt cảm biến nhiệt TMP36

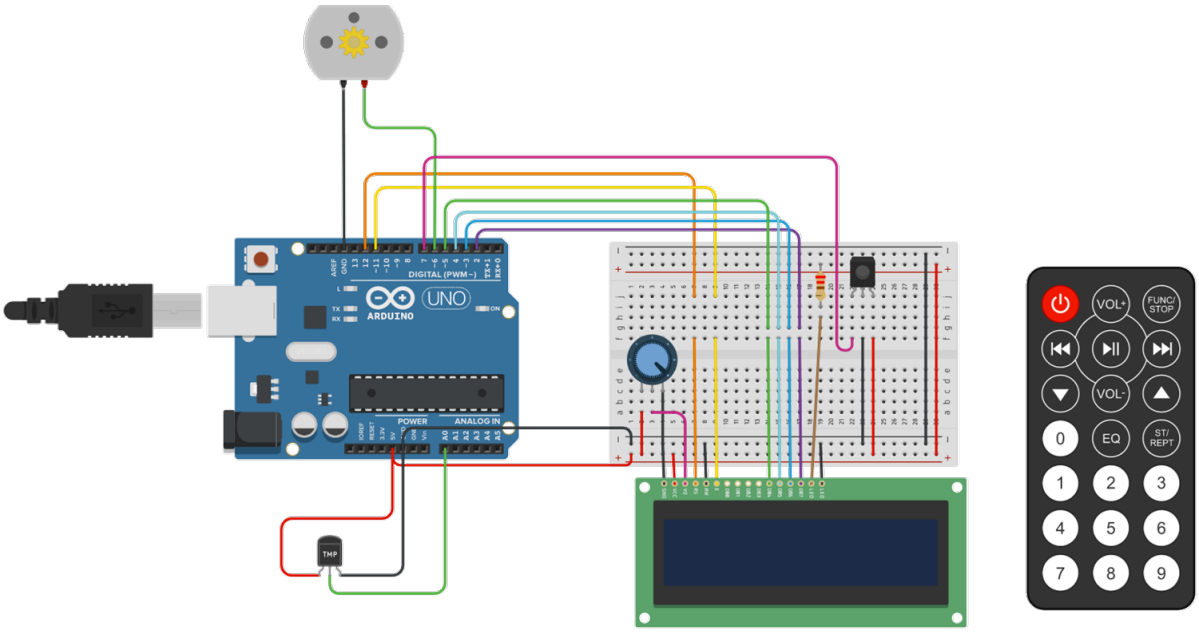
- Nối chân power với pin nguồn 5V của Arduino
- Nối chân GND với pin GND của Arduino
- Nối chân Vout với pin A0 của Arduino

Lắp đặt DC motor

- Nối chân terminal 1 với pin GND của Arduino
- Nối chân terminal 2 với pin digital 6 của Arduino

Lắp đặt IR sensor

- Nối chân out với pin digital số 7 của Arduino
- Nối 2 chân còn lại với nguồn 5V và GND



Hình 2-2. Hình ảnh mạch mô phỏng trên Tinkercad

3. Hệ thống đèn LED RGB

Link mô phỏng trên Tinkercad: RGB Led System | Tinkercad

Link video demo: Demo hệ thống đèn LED RGB

3.1. Mô tả

a) Tên dự án

Hệ thống đèn LED RGB điều khiển màu sắc.

b) Ứng dụng

- Nâng cấp của hệ thống đèn chiều sáng thường thấy trong đời sống.
- Điều chỉnh màu sắc dễ dàng theo sở thích.
- Thay đổi nhiệt độ màu đèn tránh dẫn đến đau mắt cho người dùng

c) Yêu cầu

- Lắp đặt thời gian thực.
- Điều chỉnh được màu sắc đèn, bật tắt thông qua điều khiển từ xa.

d) Cách sử dụng

- Sử dụng remote để bật tắt đèn và điều chỉnh màu sắc.
- Nút số 1 cho đèn màu đỏ, số 2 là đèn xanh lam và số 3 cho màu xanh lá.
- Nút sô 0 là đèn trắng hỗ trợ chức năng đổi nhiệt độ màu đèn.
  Hai nút << và >> dùng để điều chỉnh nhiệt độ màu.
- Nút nguồn để tắt đèn.

e) Ưu điểm

- Sử dụng điều khiển từ xa mang lại tính tiện lợi cho việc điều khiển đèn.
- Chính xác, độ tin cậy cao.
- Cá nhân hóa không gian tốt hơn các thiết bị chiếu sáng thông thường.

f) Nhược điểm

- Chưa có độ tự do 100% trong việc điều khiển.
- Chưa có khả năng lưu lại lựa chọn mỗi khi bật đèn.

3.2. Phần cứng

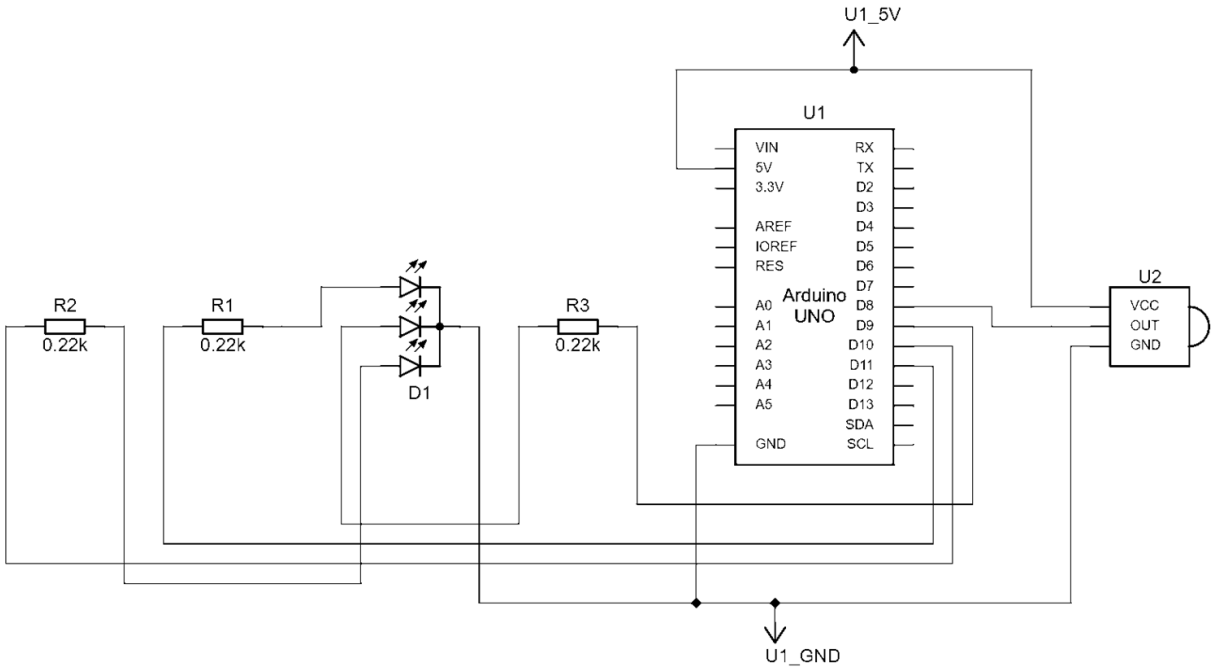
a) Các thành phần

- 01 Arduino Uno R3
- 01 IR Sensor (Cảm biến IR)
- 01 IR Remote (Điều khiển từ xa)
- 01 LED RGB
- 03 Resistor (Điện trở)

b) Đặc tính kỹ thuật

- Resistor : Điện trở 220 Ohm
- IR Sensor: Hiệu điện thế 5V

c) Schematic design



Hình 3-1. Schematic design của hệ thống đèn LED RGB

3.3. Phần mềm

a) Giải thích source code

Các thư viện sử dụng:

- Điều khiển IR: `IRremote.h`

Các hàm:

`setup()`: dùng để khởi đầu các hàm và thiết bị Arduino, ngoài ra con khởi động cảm biến IR để bắt đầu nhận tín hiệu.

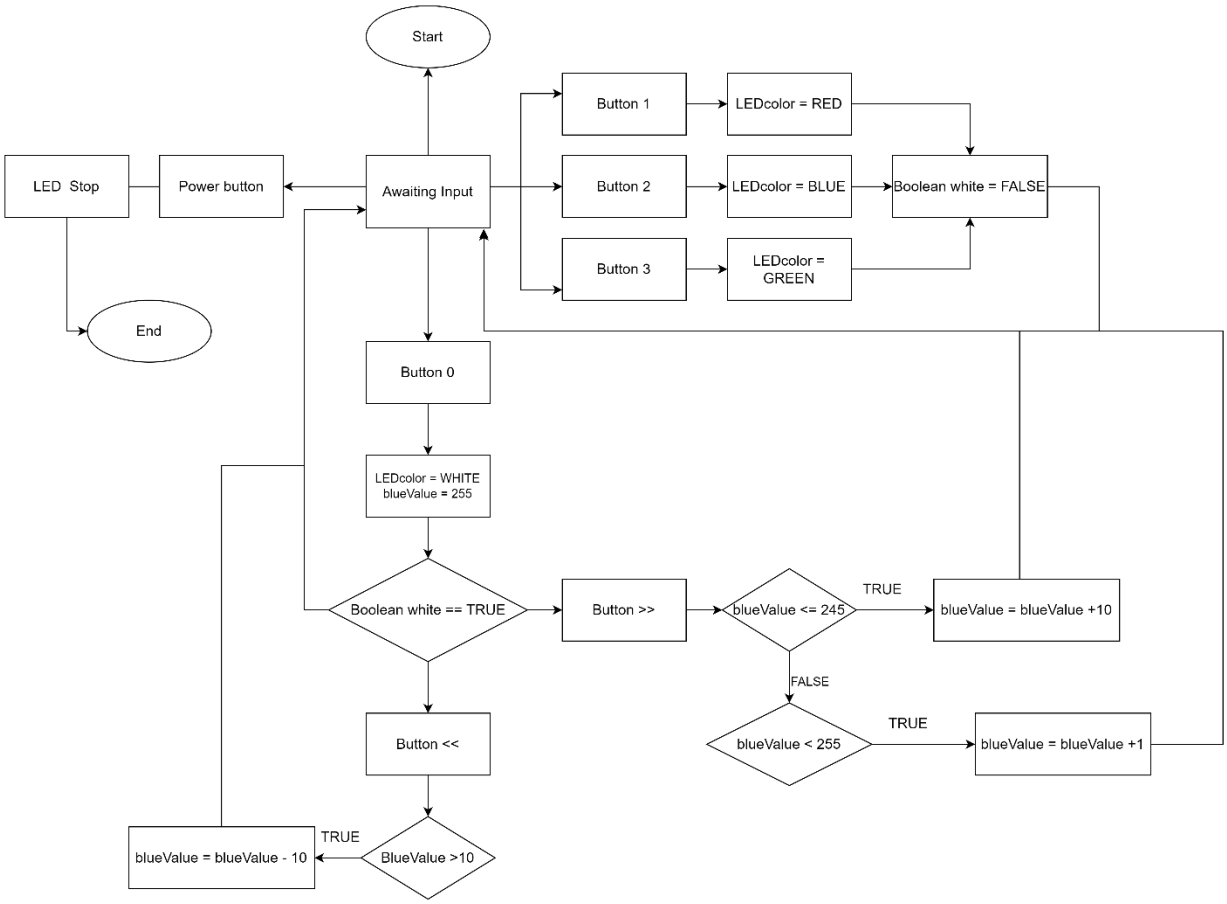
`loop()`: hàm chính để thực hiện việc xử lý thông tin từ thiết bị điều khiển từ xa và điều chỉnh đèn tương ứng:

irrecv.decode(&results):

- Được sử dụng để nhận tính hiệu từ điều khiển từ xa chuyển qua các dãy số và in ra console.
- Khi các switch case tương ứng các button 1, 2, 3 được gọi thì chỉnh đèn tương ứng và cho biến white = FALSE.
- Chỉ khi switch case của button 0 được gọi thì sẽ chỉnh đèn về màu trắng, biến white = TRUE và người dùng có thể điều chỉnh độ ấm của đèn.
- Switch case điều khiển độ ấm của đèn dùng để điều chỉnh giá trị màu xanh của đèn led. Gồm các hàm if dùng để ngăn giá trị màuvượt quá khoảng từ 0 đến 255 để hiển thị màu một cách chính xác. Đặc biệt trong Tinkercad có sự khác biệt khá lớn của đèn khi giá trị màu xanh đặt 254 và 255 nên phần mềm cần có thêm một hàm if đặc biệt để đảm bản giá trị đèn có thể đạt chính xác 255.

b) Flowchart

Link: [Flowchart RGB Led System]



c) Source code

Ngôn ngữ: C++

```
#include <IRremote.h>

int greenLed = 9;
int redLed = 11;
int blueLed = 10;
int RECV_PIN = 8;
int blueValue = 255;
boolean white;
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup() {
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  Serial.begin(9600);
  Serial.println("Enabling IRin");
  irrecv.enableIRIn();
  Serial.println("Enabled IRin");
}

void loop() {
  if (irrecv.decode(&results)) {
    unsigned int value = results.value; // get value remote
    Serial.println(value);
    switch (value) {
      case 2295:  //button 1 -> red
        analogWrite(blueLed, 0 );
        analogWrite(redLed, 255);
        analogWrite(greenLed, 0);
        white = false;
        break;

      case 34935: //button 2 -> blue
        analogWrite(blueLed, 255 );
        analogWrite(redLed, 0);
        analogWrite(greenLed, 0);
        white = false;
        break;

      case 18615:   //button 3 -> green
        analogWrite(blueLed, 0 );
        analogWrite(redLed, 0);
        analogWrite(greenLed, 255);
        white = false;
        break;

      case 255: //power button -> off
        analogWrite(blueLed, 0 );
```

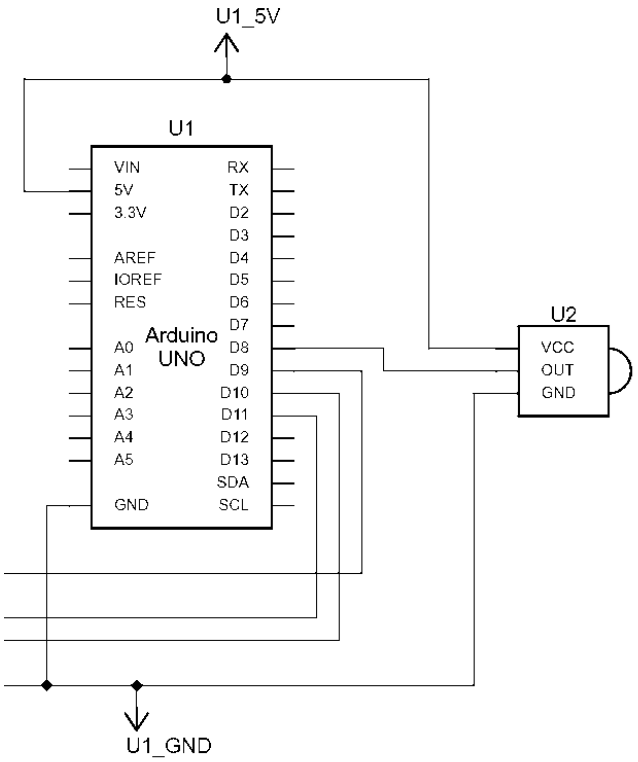
```
        analogWrite(redLed, 0);
        analogWrite(greenLed, 0);
        white = false;
        break;

      case 12495: //button 0 -> white
        analogWrite(blueLed, 255 );
        analogWrite(redLed, 255);
        analogWrite(greenLed, 255);
        blueValue = 255;
        white = true;
        break;

    }
    while (white == true) { //only change color temp when led is white
      switch (value) {
        case 24735: //decrease color temp
          if (blueValue < 240 ) {
            blueValue = blueValue + 20;
            analogWrite(blueLed, blueValue);
          }
          else {
            blueValue = 255; //for led to be white
            analogWrite(blueLed, blueValue);
          }
          analogWrite(blueLed, blueValue);
          break;

        case 8415 : //increase color temp
          if (blueValue > 21) {
            blueValue = blueValue - 20;
          }
          else {
            blueValue = 0;
            analogWrite(blueLed, blueValue);
          }
          analogWrite(blueLed, blueValue);
          break;

      }
    }
    irrecv.resume();
  }
}
```

3.4. Triển khai lắp đặt và chạy thử

a) Thiết kế lắp đặt

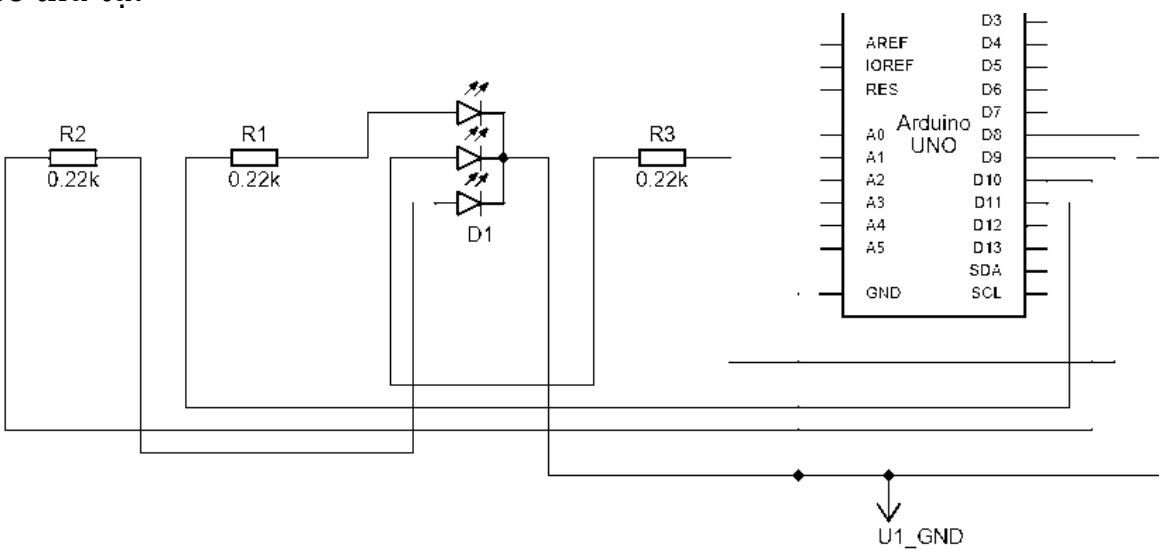
Hệ thống gồm 2 phần:

(1) Bộ phận xử lý: Là thiết bị Arduino và cảm biến IR lắp vào cổng 8 . Đây là bộ phận trung tâm, là nơi tiếp nhận thông tin của điều khiển từ xa. Lắp đặt nơi cao ráo, tránh ẩm ướt và nhiệt độ quá mức.



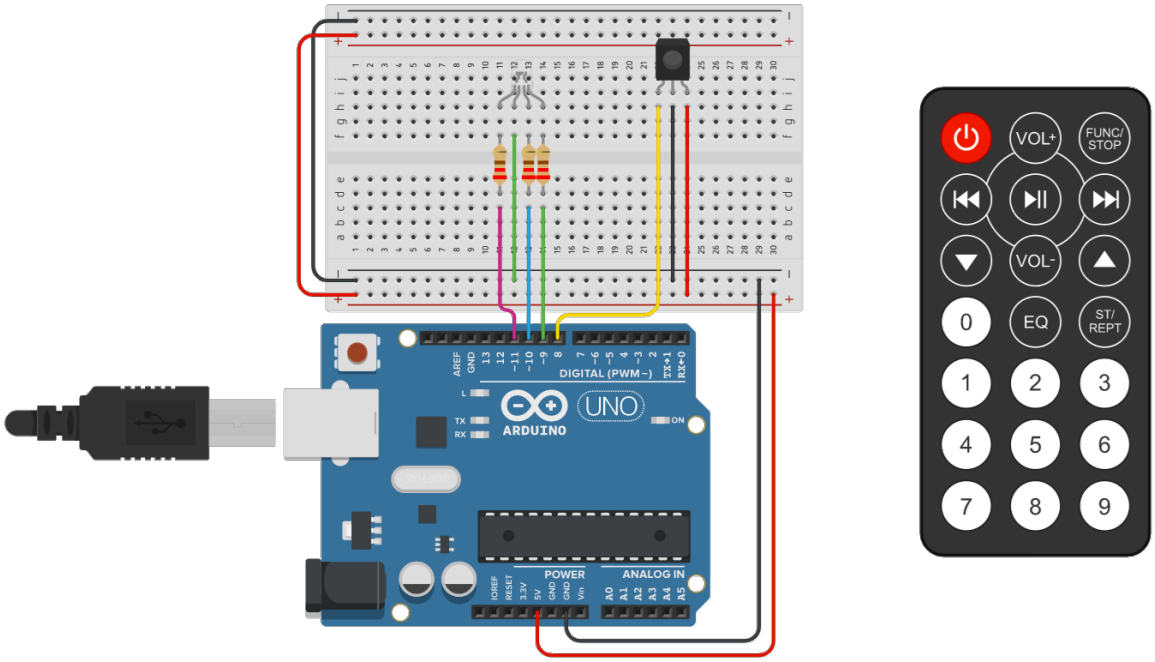
Hình 3-2. Bộ xử lý

(2) Hệ Thống đèn RGB LED: Gồm 1 hoặc nhiều đèn LED phụ thuộc và diện tích không gian. Dữ liệu sau khi được xử lý bởi hệ thống sẽ được truyền đến đèn. Lắp đặt ở nơi cao ráo. Lưu ý nối dây Red, Blue, Green vào các cổng 11, 10, 9 theo thứ tự.



Hình 3-3. Hệ thống đèn LED

Hình ảnh hệ thống mô phỏng trên Tinkercad



Hình 3-4. Hình ảnh mạch mô phỏng trên Tinkercad

b) Lỗi có thể phát sinh trong lắp đặt:

- Phần cứng, thiết bị bị hỏng hóc: Cần lưu ý việc lắp đặt để tránh bị cháy mạch.
- Cảm biến không nhận cảm biến của điều khiển: Cần kiểm tra lại xem điều khiển có tương thích với cảm biến .

c) Các bộ thử:

Nút bấm	Giá trị màu lục	Giá trị màu lam	Giá trị màu đỏ	Màu đèn
1	0	0	255	Đỏ
2	0	255	255	Lam
>>	0	255	255	Lam
0	255	255	255	Trắng
<<	255	245	255	Vàng nhạt
Nguồn	0	0	0	Không

d) Hướng phát triển

- Hệ thống có thể hỗ trợ việc chọn bất kì màu nào trong dải RGB.
- Người dùng có thể sử dụng điện thoại cá nhân để điều chỉnh nhiệt độ.
- Hệ thống hỗ trợ chức năng đi kèm với báo thức của điện thoại để việc thức dậy dễ dàng hơn.

4. Hệ thống phơi đồ thông minh

Link file mô phỏng: Hệ thống phơi đồ thông minh | Proteus

Link video demo: Demo hoạt động hệ thống phơi đồ

4.1. Mô tả

a) Nội dung dự án

Xây dựng hệ thống phơi đồ thông minh, điều khiển hoạt động phơi đồ thông qua các chỉ số điều kiện được lấy từ các sensor và ngưỡng chỉ số điều khiển thiết lập bởi người dùng.

b) Ứng dụng

Nhu cầu giặt phơi đồ là nhu cầu thường ngày ở mỗi hộ gia đình, nhưng mùa mưa sẽ rất bất tiện nếu không kịp thu đồ khi trời mưa. Hệ thống phơi đồ thông minh tự động điều khiển cây phơi đồ tuỳ theo điều kiện thời tiết, đem lại tiện lợi cho người dùng khi không còn lo mình sẽ quên thu đồ khi mưa. Có thêm nút bấm để điều khiển thủ công. Sử dụng hiệu quả trong các căn hộ nhà ở, đặc biệt phù hợp với những khu vực có thời tiết thất thường.

c) Yêu cầu

- Nhận diện được các chỉ số: nhiệt độ, độ ẩm, thời gian, tình trạng thời tiết (mưa/không mưa) và xử lý.
- Tự động hoá quá trình thu vào, kéo ra của cây phơi đồ tuỳ theo điều kiện thời tiết.
- Các thao tác xử lý theo thời gian thực.
- Người dùng có thể tuỳ chỉnh ngưỡng các chỉ số cho hệ thống, có thể chủ động điều khiển cây phơi đồ bất kỳ thời điểm nào không phụ thuộc vào chế độ tự động hay các chỉ số được thiết lập.

d) Cách sử dụng

Ban đầu, khi bật hệ thống, mọi thông tin được thiết lập ở chế độ mặc định, bao gồm thời gian mở 7 giờ - đóng 17 giờ, chưa có thiết lập hẹn giờ (*timer*), nhiệt độ là $T = 30°C$, độ ẩm là $H = 30\%$.

Để thiết lập những thông số như thời gian mở – đóng, hẹn giờ kéo sào, tùy chỉnh ngưỡng chỉ số thời tiết. Người dùng sẽ nhấn chọn phím * trên keypad sau khi thiết bị được mở/khởi động lại.

Khi đó, màn hình LCD hiển thị 5 lựa chọn:

Mục 1. Weather setting

Cho người dùng đặt ngưỡng theo tình hình thời tiết hiện tại cho sào phơi đồ. Hiện tại, hệ thống gồm 2 chế độ:

- *Sunny (*30°*C* – 30% *độ ẩm)*
- *Cloudy(*25°*C* – 40% *độ ẩm)*

Sau khi chọn kiểu thời tiết, hệ thống sẽ thiết lập lại theo thông số của thời tiết đã chọn.

Mục 2. Time setting

Bao gồm hai chức năng: Change open-close time, Set timer

(1) Change open-close time:

- Khi người dùng chọn, hệ thống sẽ hiển thị thời gian mở - đóng cũ, và yêu cầu người dùng nhập thời gian mở - đóng mới.
  + *Nhập thời gian mở không hợp lệ: báo lỗi* $0 \leq$ *open time* ≤ 24.
  + *Nhập thời gian mở không hợp lệ: báo lỗi* $0 \leq$ *close time* ≤ 24.
  + *Nhập thời gian mở lớn hơn thời gian đóng, báo lỗi: "Close time should be after open time".*
- Các lỗi do hệ thống báo sẽ xuất hiện sau khi người dùng nhập thời gian mở và thời gian đóng. Nếu có lỗi xuất hiện, hệ thống sẽ quay về màn hình nhập và yêu cầu người dùng nhập lại.
- Nếu người dùng nhập đúng, quay về menu chính

(2) Set timer

- Hiển thị thời gian hẹn giờ cũ (nếu có), và yêu cầu người dùng nhập thời gian hẹn giờ mới.
- Nếu thời gian hẹn giờ mới không hợp lệ, báo lỗi $15 \leq timer \leq 1440$.

- Các lỗi do hệ thống báo sẽ xuất hiện sau khi người dùng nhập thời gian hẹn giờ. Nếu có lỗi xuất hiện, hệ thống sẽ quay về màn hình nhập và yêu cầu người dùng nhập lại.
- Nếu người dùng nhập đúng, quay về menu chính

Mục 3. Change auto mode

Chức năng này dùng để bật/tắt chế độ tự động của sào phơi đồ.

Trạng thái bật tắt sẽ dựa trên dòng chữ hiển thị "Change auto mode (on)" hoặc "Change auto mode (off)". Khi người dùng chọn, chế độ tự động sẽ chuyển trạng thái.

Ngoài ra, để thuận tiện, khi ở màn hình thiết lập, người dùng cũng có thể sử dụng chức năng này nếu hệ thống có vấn đề.

Mục 4. Current config

Chức năng này dùng để in ra cấu hình cũng như thời tiết hiện tại cho người dùng bao gồm:

- *Thời tiết đã chọn*
- *Nhiệt độ - độ ẩm*
- *Thời gian mở - đóng*
- *Hẹn giờ(nếu có)*

Mục 5. Exit

Chức năng này giúp người dùng quay về trạng thái ban đầu, LCD hiển thị màn hình khi mới khởi động thiết bị.

4.2. Phần cứng

a) Các thành phần

STT	Số lượng	Thiết bị	Tên trong proteus
01	01	Board Arduino Mega 2560	ARDUINO MEGA2560
02	01	Đồng hồ thời gian thực	DS107
03	01	Động cơ quay	MOTOR-PWMSERVO
04	01	Điện trở	RES

05	01	Nút bấm	SPST PUSH BUTTON
06	01	Bàn phím	4x3 KEYPAD-PHONE
07	01	Màn hình LCD	OLED 12864I2C (SSD1306 controller)
08	01	Cảm biến mưa	RAIN SENSOR
09	01	Logicstate	LOGICSTATE
10	01	Cảm biến nhiệt độ - độ ẩm	DHT11

b) Đặc tính kỹ thuật

Board Arduino Mega2560: là một vi điều khiển sử dụng ATmega2560. Arduino Mega2560 cơ bản vẫn giống Arduino Uno R3, chỉ khác số lượng chân và nhiều tính năng mạnh mẽ hơn. Bao gồm:

- *54 chân digital (15 có thể được sử dụng như các chân PWM)*
- *16 đầu vào analog,*
- *1 thạch anh 16 MHz,*
- *1 cổng kết nối USB,*
- *1 jack cắm điện,*
- *1 nút reset.*

Serial Real-time Clock DS1307: là IC thời gian thực giá rẻ, rất chính xác với thạch anh tích hợp có sẵn. IC có đầu vào cho pin riêng, tách biệt khỏi nguồn chính đảm bảo cho việc giữ thời gian chính xác.

Thời gian IC được giữ ở trạng thái giờ, phút, giây, ngày, thứ, tháng, năm. Tất cả được điều chỉnh cho phù hợp với hiện tại, có các chế độ 12h AmPm hoặc 24h. Trong chip có mạch điện áp chuẩn dùng để theo dõi trạng thái nguồn VCC, phát hiện lỗi nguồn, tự động chuyển nguồn khi có vấn đề. Thông số kĩ thuật :

- *Nguồn VCC : 3,5 – 5 V*
- *Clock : chip DS3231 (nâng cấp DS1307)*
- *Thông tin thời gian : thời gian đến 2100*

Motor-PWMServo: Động cơ servo có 3 chân:

- *GND – Màu đen hoặc nâu, chân MASS cho động cơ và mạch điều khiển.*

- *Tín hiệu – Màu trắng hoặc vàng, chân ngỏ vào của tín hiệu điều khiển*
- *VCC – Chân cấp nguồn cho động cơ và mạch điều khiển, thường là 5V.*

Resistor: 220Ω

SPST Push Button: Nút bấm nhấn thả - khi được nhấn sẽ đóng mạch điện, thả ra sẽ về trạng thái ngắt. Có 2 đầu ra vào, thường gọi là công tắc đơn cực.

4x3 Keypad: Gồm 12 phím (0 – 9, '*' và '#')

- *100M Ohm, 100V*
- *Contact bounce: 5ms*

128x64 Graphical LCD with SSD1306 controller

GLCD 128x64 có 128 cột và 64 hàng tương ứng có 128x64=8192 chấm (dot). Thông số kỹ thuật:

- *Driver: SSD1306*
- *Tiêu thụ điện năng thấp: 0.08W (fullscreen)*
- *Có thể điều chỉnh độ sáng, độ tương phản*
- *Chuẩn giao tiếp: I2C (qua 2 chân SCL, SDA)*
- *Điện áp hoạt động: 3V - 5V DC*

Rain sensor

- *Điện áp hoạt động: 3.3V - 5V DC*
- *Dòng điện: 15 mA*
- *Comparator chip: LM393*
- *Output types:*
  + *AO (Analog voltage output)*
  + *DO (Digital switching output 0/1)*
- *Độ nhạy cảm biến có thể điều chỉnh qua Trimpot*

Logicstate: Dùng mô phỏng trong proteus cùng với cảm biến mưa, trạng thái 1 thể hiện có mưa, trạng thái 0 thể hiện không mưa.

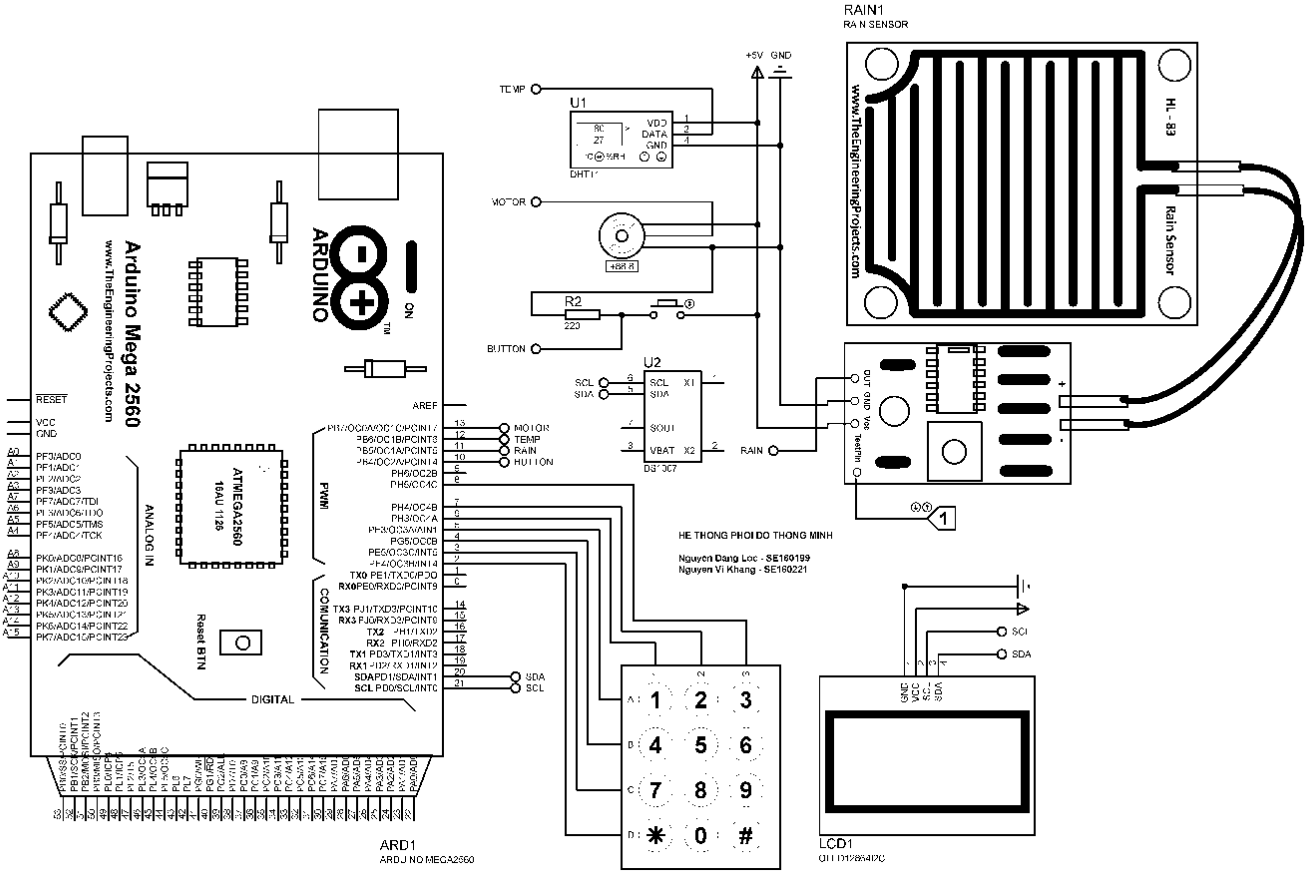
Humidity & Temperature sensor DHT11

Đi kèm với một điện trở nhiệt NTC chuyên dụng để đo nhiệt độ và vi điều khiển 8-bit để xuất các giá trị nhiệt độ và độ ẩm.

- *Operating Voltage: 3.5V to 5.5V*
- *Điện áp hoạt động: 3.3V-5V DC*
- *Dòng điện: 0.3mA (measuring) 60uA (standby)*
- *Output: Serial data*
- *Phạm vi nhiệt độ: 0°C to 50°C*
- *Phạm vi độ ẩm: 20% to 90%*
- *Độ chính xác: ±1°C and ±1%*

c) Schematic design

Link: Schematic design hệ thống phơi đồ



Hình 4-1. Schematic design của hệ thống phơi đồ

Nguyên lý hoạt động

Người dùng được lựa chọn tuỳ chỉnh giữa 2 chế độ: tự động và thủ công.

Chế độ thủ công: Người dùng sử dụng nút bấm để điều khiển thu vào hoặc đưa cây phơi đồ ra ngoài. Hệ thống không hoạt động theo các số liệu của cảm biến hay đồng hồ hẹn giờ.

Chế độ tự động: Quyết định kéo vào hay đưa ra phụ thuộc vào chỉ số nhiệt độ, độ ẩm và thời gian hẹn giờ.

- Trong điều kiện thời tiết đáp ứng:
  + *Thời gian mặc định đưa ra: 7 giờ sáng*
  + *Thời gian mặc định thu vào: 17 giờ (5 giờ chiều)*
  + *Người dùng có thể tuỳ chỉnh thời gian đưa ra và thu vào theo ý định (nhưng hệ thống sẽ chỉ hoạt động nếu điều kiện thời tiết đáp ứng)*
- Trong điều kiện thời tiết không đáp ứng (trời mưa, nhiệt độ-độ ẩm không đạt ngưỡng): Hệ thống sẽ thu vào trong
- Người dùng có thể tuỳ chỉnh các mức độ thiết lập ngưỡng điều kiện thời tiết.

4.3. Phần mềm

a) Giải thích source code

Các thư viện sử dụng:

- Hiển thị màn hình LCD: `Adafruit_GFX.h`, `Adafruit_SSD1306.h`
- Cảm biến nhiệt độ - độ ẩm: `DHT.h`
- Đồng hồ thời gian thực: `RTClib.h`
- Bàn phím: `Keypad.h`
- Động cơ motor: `Servo.h`
- Dây: `Wire.h` (cần cho RTC)

Các biến toàn cục (global variables):

- Các hằng số:

  `BUTTON_PIN`: Cổng tín hiệu nút bấm

  `RAIN_PIN`: Cổng tín hiệu cảm biến mưa

`DHTPIN`: Cổng tín hiệu cảm biến nhiệt độ - độ ẩm

`DHTTYPE`: Loại cảm biến nhiệt độ - độ ẩm

`MOTOR_PIN`: Cổng tín hiệu motor

- Các biến dùng cho keypad

  `ROWS`: Số hàng, `COLS`: số cột

  `keys[ROWS][COLS]`: Gán các giá trị của keypad vào mảng 2 chiều

  `rowPins[ROWS]`: Cổng pinout của các hàng keypad

  `colPins[COLS]`: Cổng pinout của các cột keypad

  `keypad`: Khai báo đối tượng keypad

- `myservo`: Khai báo đối tượng servo motor
- `rtc`: Khai báo đối tượng của đồng hồ thời gian thực
- `pos`: Lưu vị trí góc quay của servo motor
- `buttonState`: Lưu trạng thái hiện tại của nút bấm (đóng/mở)
- `lastButtonState`: Lưu trạng thái của nút bấm ở vòng `loop()` trước
- `servoState`: Lưu trạng thái của servo (thu vào/đưa ra)
- `autoMode`: Quyết định chế độ tự động hay không
- `lastMillis`: Lưu lại thời gian lần gần nhất người dùng bấm nút
- `userButton`: xem người dùng có thao tác bấm nút không
- `tempLimit`: Khi nhiệt độ bé hơn tempLimit, sào phơi đồ sẽ đóng
- `humidLimit`: Khi độ ẩm bé hơn humidLimit, sào phơi đồ sẽ đóng
- `weatherState`: Ngưỡng thời tiết hiện tại (1 - sunny, 2 - cloudy)
- `openTime`: Là giờ mở của sào phơi đồ
- `closeTime`: Là giờ đóng của sào phơi đồ
- `timer`: Là giờ hẹn đóng của sào phơi đồ
- `lastTimerMillis`: Là bộ đếm giờ hẹn đóng của sào phơi đồ
- `estimatedHour`: Thời gian sào phơi đồ kéo vào (đơn vị: giờ)
- `estimatedMinute`: Thời gian sào phơi đồ kéo vào (đơn vị: phút)

Các hàm:

- `testing()`: Hàm dùng để thử nghiệm tính năng của sào phơi đồ.
- `setup()`: Hàm khai báo, khởi tạo các cảm biến:

+ Khởi tạo rtc: đồng hồ thời gian thực
+ Khởi tạo dht: cảm biến đo nhiệt độ - độ ẩm
+ Khởi tạo Rain sensor
+ Khởi tạo servo và đưa về vị trí ban đầu

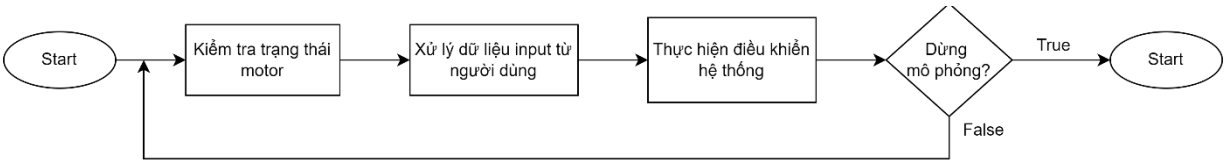
- `resetOLED()`: Xóa màn hình, đồng thời đặt con trỏ trên màn hình về 0
- `resetServo()`: Khởi động lại servo và đưa về vị trí ban đầu
- `getCurrentDate()`: Lấy ngày – tháng – năm hiện tại theo rtc
- `getCurrentTime()`: Lấy giờ:phút:giây hiện tại theo rtc
- `getCurrentHour()`: Lấy giờ hiện tại theo rtc
- `getCurrentMinute()`: Lấy phút hiện tại theo rtc
- `getCurrentSecond()`: Lấy giây hiện tại theo rtc
- `printDate()`: In ra thời gian hiện tại theo rtc, bao gồm ngày-tháng-năm và giờ:phút:giây
- `getInt(String msg = "Enter number: ")`: Lấy về số nguyên mà người dùng nhập vào lưu vào kiểu dữ liệu String, cho tới khi người dùng nhập phím * hoặc #, sau đó trả về số nguyên đó theo kiểu int
- `changeWeatherLimit(int t, int h)`: Thay đổi giới hạn nhiệt độ và độ ẩm tùy theo kiểu thời tiết mà người dùng chọn
- `weatherSetting()`: In ra các lựa chọn về thời tiết đi kèm với các thông số, và cho phép người dùng chọn kiểu phù hợp
- `changeOpenCloseTime()`: In ra thời gian mở - đóng cũ, sau đó yêu cầu người dùng nhập thời gian mở - đóng mới
- `updateTimer(int newTimer)`: Cập nhật bộ hẹn giờ mới, cũng như tính toán lại thời gian hẹn đóng sào phơi đồ
- `setTimer()`: In ra thời gian hẹn cũ(nếu có), và yêu cầu người dùng nhập thời gian hẹn giờ mới
- `timeSetting()`: In ra các lựa chọn thiết lập về thời gian, bao gồm thay đổi thời gian mở-đóng và hẹn giờ
- `changeAutoModeState()`: Thay đổi trạng thái của chế độ tự động(đóng -> mở hoặc mở -> đóng)

- `printInfo()`: In ra các thông tin, thông số hiện tại của hệ thống
- `servoMotor()`: Quay servo khi người dùng bấm nút
- `timerAlarm()`: Kiểm tra xem đã đến giờ hẹn, theo thời gian của rtc
- `getServoStatus()`: Kiểm tra tính trạng thời tiết hiện tại để thực hiện mở/đóng sào phơi đồ một cách hợp lý
- `displayMenu()`: In ra menu lựa chọn thiết lập thời tiết, thời gian, chế độ tự động và xem thông tin hệ thống
- `loop()`: Hàm lặp của arduino, dùng để in thời gian, trạng thái hiện tại của sào phơi đồ, và cho người dùng truy cập tùy chỉnh hệ thống

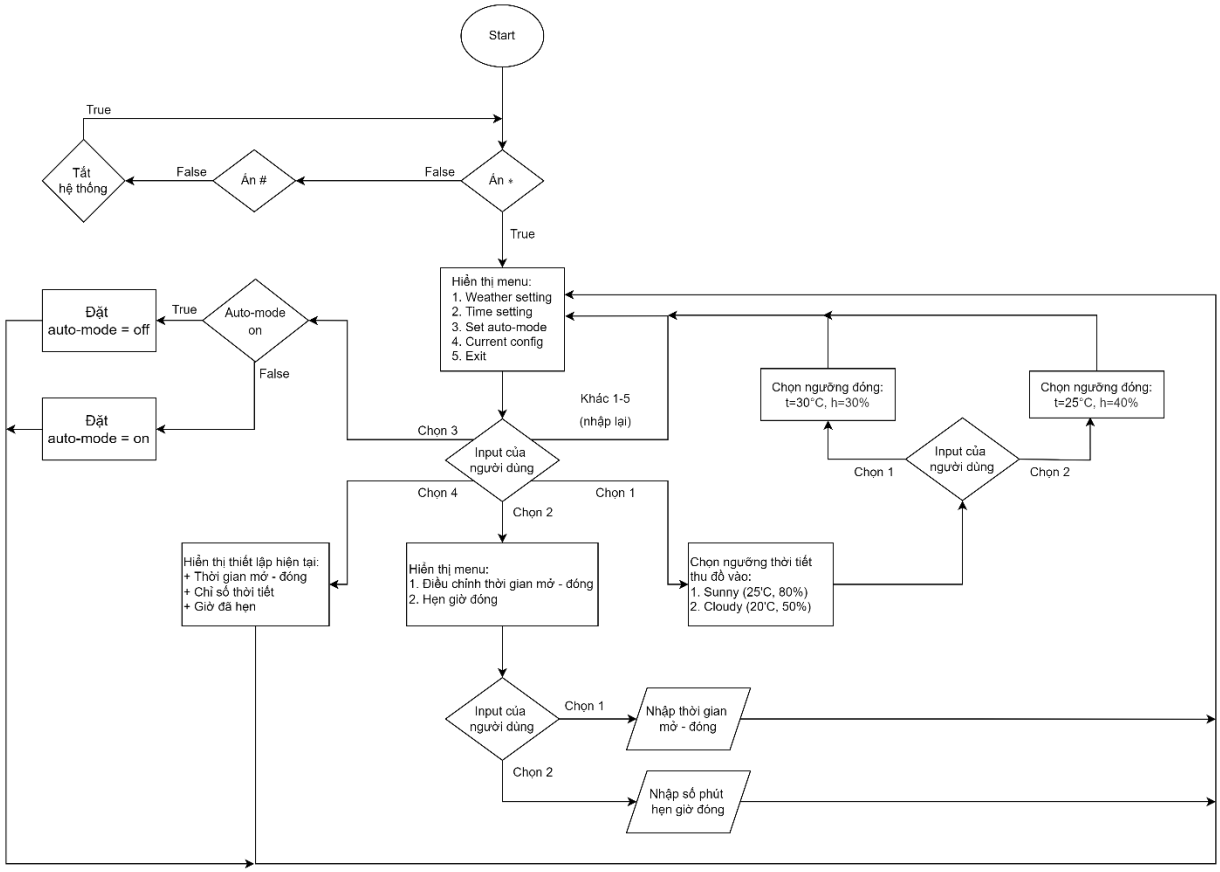
b) Flowchart

Link: [Flowchart Hệ thông phơi đồ thông minh]

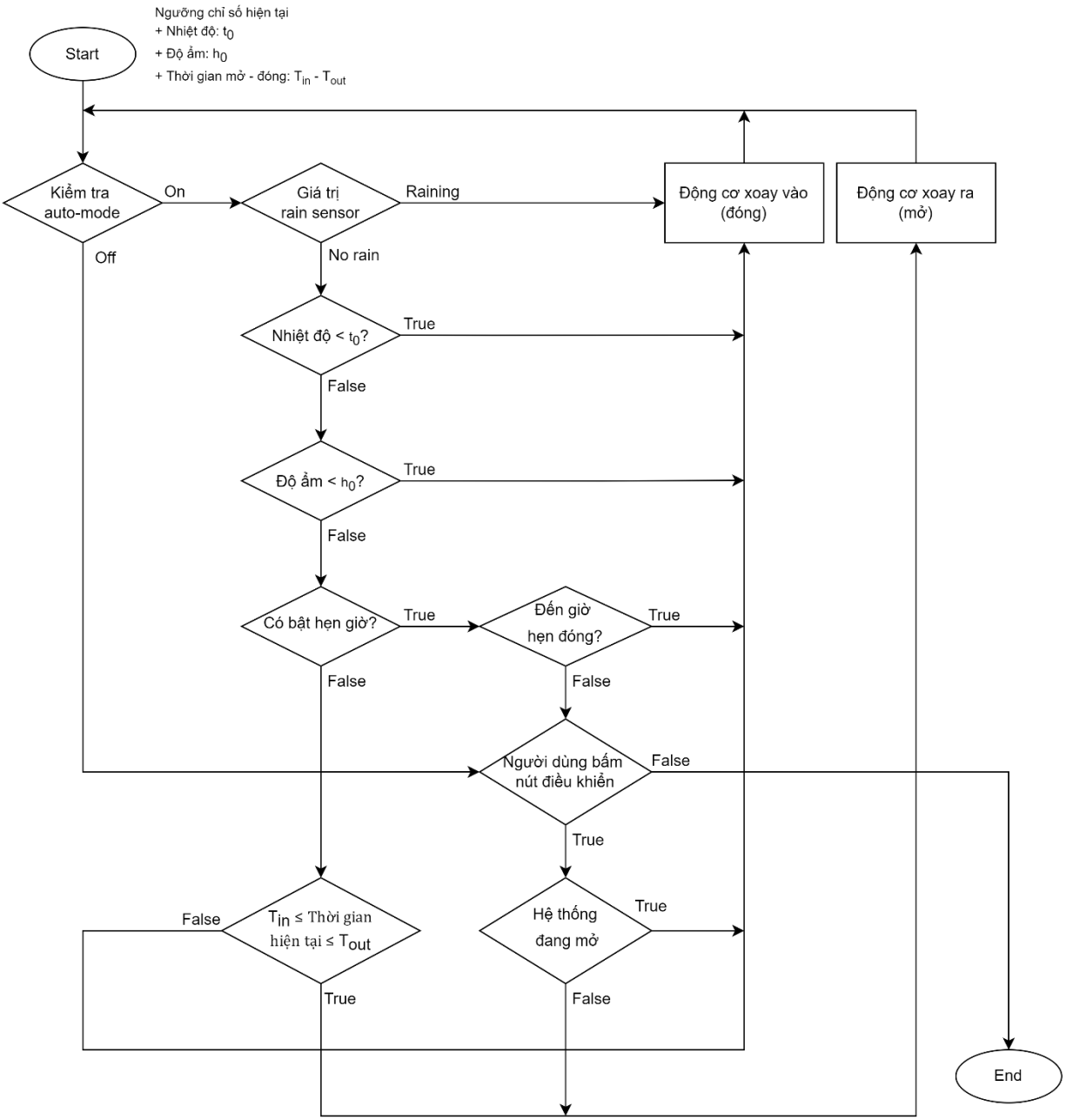
Flowchart cho hàm loop()



Flowchart xử lý dữ liệu đầu vào (nhập từ keypad hiển thị qua LCD)



Flowchart điều khiển hệ thống



c) Source code

Ngôn ngữ: C++

```
#include <Keypad.h>
#include <Servo.h>
#include <Adafruit_GFX.h>       // thư viện cho LCD
#include <Adafruit_SSD1306.h>   // thư viện cho LCD
#include <Wire.h>               // thư viện cần cho RTC
#include "DHT.h"                // thư viện cho cảm biến NĐ-ĐA
#include "RTClib.h"             // thư viện cho đồng hồ thời gian thực

const int BUTTON_PIN = 10;
const int RAIN_PIN = 11;
const int DHTPIN = 12;
const int DHTTYPE = DHT11;
const int MOTOR_PIN = 13;

Servo myservo;
RTC_DS1307 rtc;
DHT dht(DHTPIN, DHTTYPE);

// Khai báo cho màn hình LCD tích hợp I2C
#define OLED_RESET -1
Adafruit_SSD1306 display(128, 64, &Wire, -1);

// Khai báo cho KEYPAD
const byte ROWS = 4;
const byte COLS = 3;
char keys[ROWS][COLS] = {
  {'1', '2', '3'},
  {'4', '5', '6'},
  {'7', '8', '9'},
  {'*', '0', '#'}
};
byte rowPins[ROWS] = {5, 4, 3, 2};  // row pinout của keypad
byte colPins[COLS] = {6, 7, 8};     //column pinouts của keypad
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

int pos = 0;                    // góc quay của servo motor
bool servoState = false;        // lưu trạng thái đóng / mở của servo motor
int buttonState = 0, lastButtonState = 0;

bool autoMode = true;

int lastMillis = -1;
bool userButton = false;

int tempLimit = 30, humidLimit = 30; // ngưỡng nhiệt độ - độ ẩm mặc định
int weatherState = 1; //1 = sunny, 2 = cloudy

int openTime = 7;               //thời gian mở mặc định: 7
int closeTime = 17;             //thời gian đóng mặc định: 17 (5pm)
int timer = 0;                  // đếm theo phút
int lastTimerMillis = 0;

int estimatedMinute = -1, estimatedHour = -1;

void testing() {
  //This is a testing sub-code for this board. Please hide it when you want to
use our project in a proper way! Thank you
  //rtc.adjust(DateTime(2022, 3, 13, 6, 59 , 55)); // test open time
  //rtc.adjust(DateTime(2022, 3, 13, 16, 59 , 50)); // test close time
```

```
  rtc.adjust(DateTime(2022, 3, 13, 15, 0, 0));
  //timer = 1;
  lastTimerMillis = millis();
}

void setup() {

  Serial.begin(9600);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);

  dht.begin();              // khởi chạy cảm biến nhiệt độ - độ ẩm

  pinMode(RAIN_PIN, INPUT); // Đặt chân cảm biến mưa là INPUT, vì tín hiệu sẽ
được truyền đến cho Arduino

  rtc.begin();              // Khởi chạy đồng hồ thời gian thực
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

  //testing();
  resetServo();
}

void resetOLED() {
  display.clearDisplay();
  display.setCursor(0, 0);
}

void resetServo() {
  myservo.attach(MOTOR_PIN);

  for (pos = 90; pos >= 0; pos -= 1)   {
    myservo.write(pos);
    delay(5);
  }
}

/**
 * @brief Lấy ngày hiện tại
 * format: dd/MM/yyyy
 * @return String
 */
String getCurrentDate() {
  DateTime currentTime = rtc.now();
  int d = currentTime.day();
  int m = currentTime.month();
  int y = currentTime.year();
  String date = "Date: " + String(d);
  date += "/" + String(m);
  date += "/" + String(y);

  return date;
}

/**
 * @brief Lấy giờ hiện tại
 * format: hh:mm:ss
 * @return String
 */
String getCurrentTime() {
  DateTime currentTime = rtc.now();
  int h = currentTime.hour();
```

```
  int m = currentTime.minute();
  int s = currentTime.second();
  String time = "Time: " + String(h);
  time += ":" + String(m);
  time += ":" + String(s);

  return time;
}

int getCurrentHour() {
  DateTime ct = rtc.now();
  return ct.hour();
}

int getCurrentMinute() {
  DateTime ct = rtc.now();
  return ct.minute();
}

int getCurrentSecond() {
  DateTime ct = rtc.now();
  return ct.second();
}

void printDate() {
  String currentDate = getCurrentDate();
  String currentTime = getCurrentTime();

  display.println(currentDate);
  display.println(currentTime);
}

/**
 * @brief Get the Int object
 * Đọc input nhập từ keypad
 * @param msg
 * @return int
 */
int getInt(String msg = "Enter number: ") {
  display.print(msg);
  display.display();
  char inputKey = keypad.getKey();
  String str = "";
  while (str == "") {
    while (inputKey != '#' && inputKey != '*') {
      if ((int)keypad.getState() == PRESSED) {
        str = str + String(inputKey);
        display.print(String(inputKey));
        display.display();
      }
      inputKey = keypad.getKey();
    }
    if (str == "") inputKey = keypad.getKey();
  }
  display.println();
  return str.toInt();
}

/**
 * @brief  Thiết lập lại ngưỡng nhiệt độ- độ ẩm
 * @param t
 * @param h
 * @return int
 */
```

```
int changeWeatherLimit(int t, int h) {
  tempLimit = t, humidLimit = h;

  resetOLED();

  display.println("Successfully changed!");
  display.display();
  delay(500);

  displayMenu();
}

/**
 * @brief Menu chọn ngưỡng thời tiết để đóng thiết bị
 * (kéo cây phơi đồ vào)
 */
void weatherSetting() {
  resetOLED();

  if (weatherState == 1) display.println("Current weather: Sunny");
  else display.println("Current weather: Cloudy");

  display.println("1. Sunny weather(30'C-30%)");
  display.println("2. Cloudy weather(25'C-40%)");
  display.println("3. Back");

  char key = 'x';
  do {
    display.display();
    key = keypad.getKey();
    if (key) {
      switch (key) {
        case '1':
          weatherState = 1;
          changeWeatherLimit(30, 30);
          //weatherState = 1;
          break;
        case '2':
          weatherState = 2;
          changeWeatherLimit(25, 40);
          //weatherState = 2;
          break;
        case '3':
          displayMenu();
          return;
        default:
          resetOLED();
          display.println("Number [1-3] only!");
          display.display();
          timeSetting();
          break;
      }
    }
  } while (true);

  resetOLED();
  displayMenu();
}

/**
 * @brief Cài đặt thời điểm mở - đóng thiết bị
 * Mặc định: 7 giờ - mở, 17 giờ - đóng
 */
void changeOpenCloseTime() {
```

```
  int newOpenTime = -1;
  int newCloseTime = -2;

  while (true) {
    resetOLED();
    display.println("Old: " + (String)openTime + "h-" + (String)closeTime +
"h");
    display.println("Press '#' to enter");
    //display.display();

    newOpenTime = getInt("Input open time: ");
    newCloseTime = getInt("Input close time: ");

    bool error = false;

    if (newOpenTime < 0 || newOpenTime > 23) display.println("0 <= Open time <
24!"), error = true;
    if (newCloseTime < 0 || newCloseTime > 23) display.println("0 <= Close time
< 24!"), error = true;
    if (newOpenTime >= newCloseTime) display.println("Close time should be
after open time"), error = true;

    display.display();

    if (!error) break;
    else delay(500);
  }
  resetOLED();

  display.println("Successfully changed!");
  display.display();

  openTime = newOpenTime;
  closeTime = newCloseTime;

  delay(500);
  resetOLED();

  displayMenu();
}

/**
 * @brief Cập nhật thời gian đóng theo giờ hẹn
 * @param newTimer
 */
void updateTimer(int newTimer) {
  timer = newTimer;

  int currentHour = getCurrentHour();
  int currentMinute = getCurrentMinute();

  estimatedMinute = currentMinute + newTimer;
  estimatedHour = (estimatedMinute / 60) + currentHour;

  estimatedMinute = estimatedMinute % 60;
  //  delay(1000);
}

/**
 * @brief Set the Timer object
 * Cài đặt hẹn giờ cho thiết bị
 */
void setTimer() {
  int newTimer = -1;
```

```
  while (true) {
    resetOLED();
    if (timer == 0)
      display.println("Old timer: none");
    else display.println("Old timer: " + (String)timer + "(minutes)");
    display.println("Press '#' to enter");
    //display.display();

    newTimer = getInt("Input new timer(minutes): ");
    lastTimerMillis = millis();
    bool error = false;

    if (newTimer < 15 || newTimer > 1440) display.println("15 <= Open time <=
1440!"), error = true;

    display.display();

    if (!error) break;
    else delay(500);
  }

  resetOLED();

  display.println("Successfully changed!");
  display.display();

  updateTimer(newTimer);

  delay(500);
  resetOLED();

  displayMenu();
}

/**
 * @brief Menu cài đặt giờ cho thiết bị
 */
void timeSetting() {
  resetOLED();

  display.println("1. Change open-close time");
  if (timer == 0)
    display.println("2. Set timer (Current: none)");
  else display.println("2. Set timer (Close  at: " + (String)estimatedHour +
":" +
                          (String)estimatedMinute);
  display.println("3. Back");

  char key = 'x';
  do {
    display.display();
    key = keypad.getKey();
    if (key) {
      switch (key) {
        case '1':
          changeOpenCloseTime();
          break;
        case '2':
          setTimer();
          break;
        case '3':
          displayMenu();
          return;
```

```
        default:
          resetOLED();
          display.println("Number [1-3] only!");
          display.display();
          timeSetting();
          break;
      }
    }
  } while (true);
}

void changeAutoModeState(int place = 0) {
  resetOLED();
  autoMode = autoMode ? false : true;

  display.setCursor(0, 0);
  display.println("Successfully changed!");
  display.display();

  delay(500);
  resetOLED();

  if(place)
    displayMenu();

  else return;
}

void printInfo() {
  resetOLED();

  if (weatherState == 1) display.println("Weather: Sunny(30'C-30%)");
  else display.println("Weather: Cloudy(25'C-40%)");

  float temp = dht.readTemperature();
  float humi = dht.readHumidity();

  display.print("T: " + (String)(int)temp + "'C    ");
  display.println("H: " + (String)(int)humi + "%");

  display.println("Open time: " + (String)openTime + "h");
  display.println("Close time: " + (String)closeTime + "h");
  if (timer != 0)
    display.println("Timer: " + (String)estimatedHour + ":" +
(String)estimatedMinute);
  else display.println("Timer: none");
  display.display();

  delay(1000);

  resetOLED();
  displayMenu();
}

void printCredit() {
  resetOLED();

  display.println("Smart clothes pole,  made by Iostream and Dolphin!");
  display.display();
  delay(1000);

  resetOLED();
```

```
  display.println("Special thanks to An,Trieu, Phu and Cong  for your
contribution!");
  display.display();
  delay(1000);

  resetOLED();

  display.println("Last but not least,  thank you Mr.ThongNT for teaching
us!");
  display.display();
  delay(1000);

  resetOLED();
  displayMenu();
}

/**
 * @brief Thực hiện thao tác xoay motor
 */
void servoMotor() {
  if (pos <= 0) {
    for (pos = 0; pos <= 180; pos += 5) {
      myservo.write(pos);
      delay(15);
    }
    servoState = true;
  } else {
    for (pos = 180; pos >= 0; pos -= 5) {
      myservo.write(pos);
      delay(15);
    }
    servoState = false;
  }
}

/**
 * @brief Xoay đóng hệ thống
 */
void closeServo() {
  if (pos > 0)
    for (pos = 180; pos >= 0; pos -= 5) {
      myservo.write(pos);
      delay(15);
    }
}

/**
 * @brief Xoay mở hệ thống
 */
void openServo() {
  if (pos <= 0) {
    for (pos = 0; pos <= 180; pos += 5) {
      myservo.write(pos);
      delay(15);
    }
  }
}

/**
 * @brief Kiểm tra xem đã đến giờ hẹn, theo thời gian của rtc
 * @return true / false
 */
bool timerAlarm() {
  if (timer == 0) return false;
```

```
  if ((millis() - lastTimerMillis) / 1000 >= timer * 60) {

    estimatedHour = -1;
    estimatedMinute = -1;
    timer = 0;
    return true;
  }
  return false;
}

bool compare(int h1, int h2) {
  h1 = h1 == 0 ? 24 : h1;
  h2 = h2 == 0 ? 24 : h2;
  return h1 == h2;
}

/**
 * @brief
 * Kiểm tra tình trạng thời tiết hiện tại để thực hiện mở/đóng sào phơi đồ một
cách hợp lí
 */
void getServoStatus() {
  String reasonDebug = "NO";
  if (autoMode) { // neu auto mode dang bat thi mach se chay tu dong
    int openState = 0; // 1 = open, -1 = close, 0 = stay;
    float temp = dht.readTemperature();
    float humi = dht.readHumidity();
    int rainValue = digitalRead(RAIN_PIN);

    if (rainValue == LOW) {
      //display.println("RAINING");
      reasonDebug = "RAINING";
      openState = -1;
    }
    else {
      if (temp < tempLimit || humi <  humidLimit) {
        //display.println("TEMP");
        reasonDebug = "TEMP";
        openState = -1;
      }
      else {
        if (timerAlarm()) {
          reasonDebug = "TIMER";
          //display.println("timer");
          openState = -1;
        }
        else {
          buttonState = digitalRead(BUTTON_PIN);
          if (buttonState == HIGH && lastButtonState == LOW) {
            reasonDebug = "USER";
            userButton = true; // will turn false after 5 minutes
            lastMillis = millis();
            servoMotor();
            openState = 0;

            timer = 0;
            lastTimerMillis = 0;
          }
          else {
            if (timer >= 0) { //timer activated but no user button pressed
              if (openState == 0 && openTime <= getCurrentHour() &&
getCurrentHour() < closeTime && !userButton) { //when in middle of open-close
time
```

```
                if (servoState == false) servoState = true;
                openState = 1;
                reasonDebug = "AUTO";
              }
              if (openState == 0 && (getCurrentHour() < openTime || closeTime
<= getCurrentHour()) && !userButton) { //when in middle of open-close time
                if (servoState == true) servoState = false;
                openState = -1;
                reasonDebug = "AUTO";
              }
            }
            else openState = 0; // neu timer con ton tai thi khong su dung che
do tu dong
          }
          lastButtonState = buttonState;
        }
      }
    }

    if (openState == 1) servoState = true, openServo();
    else if (openState == -1) servoState = false, closeServo();
  }
  else { // neu auto mode dang tat thi mach se do user dieu khien
    buttonState = digitalRead(BUTTON_PIN);
    if (buttonState == HIGH && lastButtonState == LOW) {
      reasonDebug = "USER";
      userButton = true; // will turn false after 5 minutes
      lastMillis = millis();
      servoMotor();
    }

    lastButtonState = buttonState;
  }

  //display.println(reasonDebug);
  //display.display();
}

/**
 * @brief Hiển thị menu chính của thiết bị
 */
void displayMenu() {
  resetOLED();

  display.println("1. Weather setting");
  display.println("2. Time setting");
  if (autoMode) display.println("3. Auto mode(on)");
  else display.println("3. Auto mode (off)");
  display.println("4. Current config");
  display.println("5. Exit");

  char key = 'x';
  do {
    display.display();
    key = keypad.getKey();

    if (key) {
      switch (key) {
        case '1':
          weatherSetting();
          break;
        case '2':
          timeSetting();
          break;
```

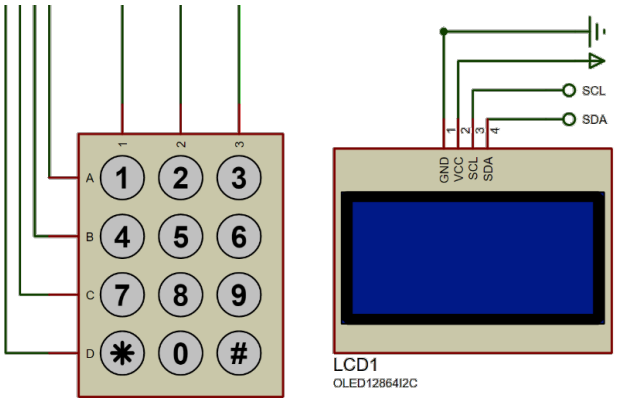
```
        case '3':
          changeAutoModeState(1);
          break;
        case '4':
          printInfo();
          break;
        case '5':
          return;
        default:
          resetOLED();
          display.println("Number [1-5] only!");
          display.display();
          delay(1000);
          displayMenu();
          break;
      }
    }
  } while (true);
}

void loop() {
  //cnt++;
  resetOLED();
  getServoStatus();

  printDate();

  char key = keypad.getKey();

  if (key) {
    display.println(key);
    switch (key) {
      case '*':
        displayMenu();
        break;
      case '#':
        changeAutoModeState(0);
        break;
    }
  }
  if (servoState) display.println("Pole: OPEN");
  else display.println("Pole: CLOSE");

  display.println("Press * to set up");
  if (autoMode) display.println("Press # to switch auto mode(on)");
  else display.println("Press # to switch auto mode(off)");

  if (userButton) {
    if ((millis() - lastMillis) / 1000 >= 15 * 60) { //15 minutes
      userButton = false,
      lastMillis = -1;
    }
  }
  display.display();
  //delay(500);
}
```

4.4. Triển khai lắp đặt và chạy thử

a) Thiết kế lắp đặt

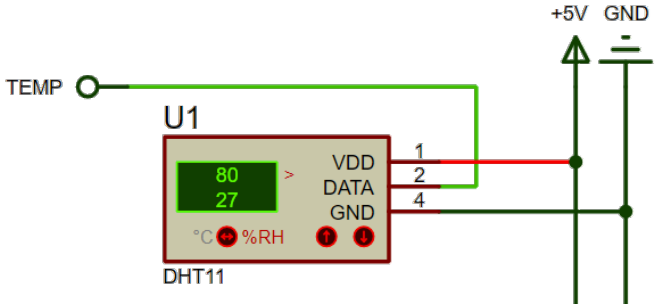
Hệ thống gồm các phần chính:

(1) Giao diện người dùng: gồm 1 keypad để nhập dữ liệu điều khiển cho hệ thống và các thông tin được hiển thị qua màn hình LCD. Tại keypad có thể ấn # để bật/tắt auto-mode.

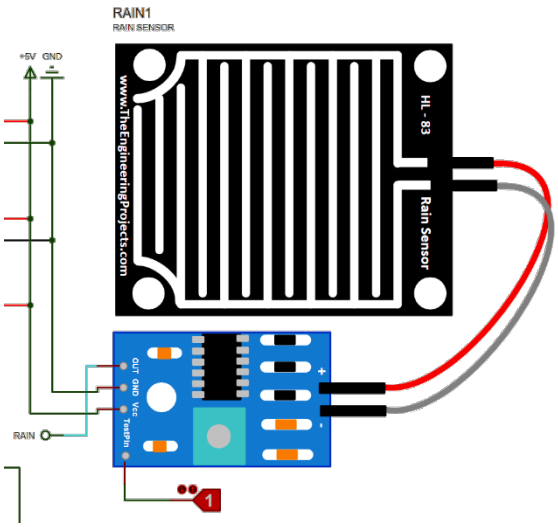


Hình 4-2. Keypad và LCD

(2) Cảm biến: Gồm cảm biến nhiệt độ - độ ẩm và cảm biến mưa được đặt ở môi trường bên ngoài để đọc các chỉ số thời tiết.

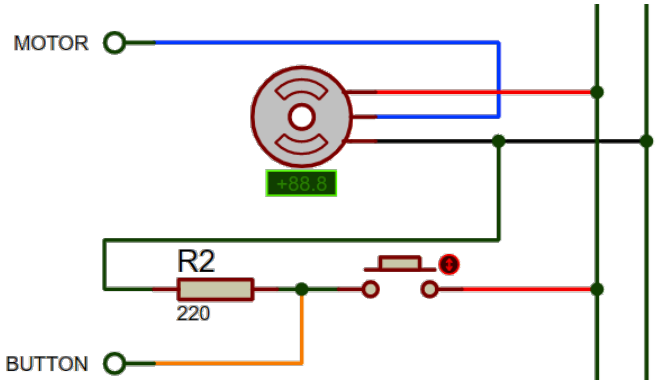


Hình 4-3. Cảm biến nhiệt độ - độ ẩm



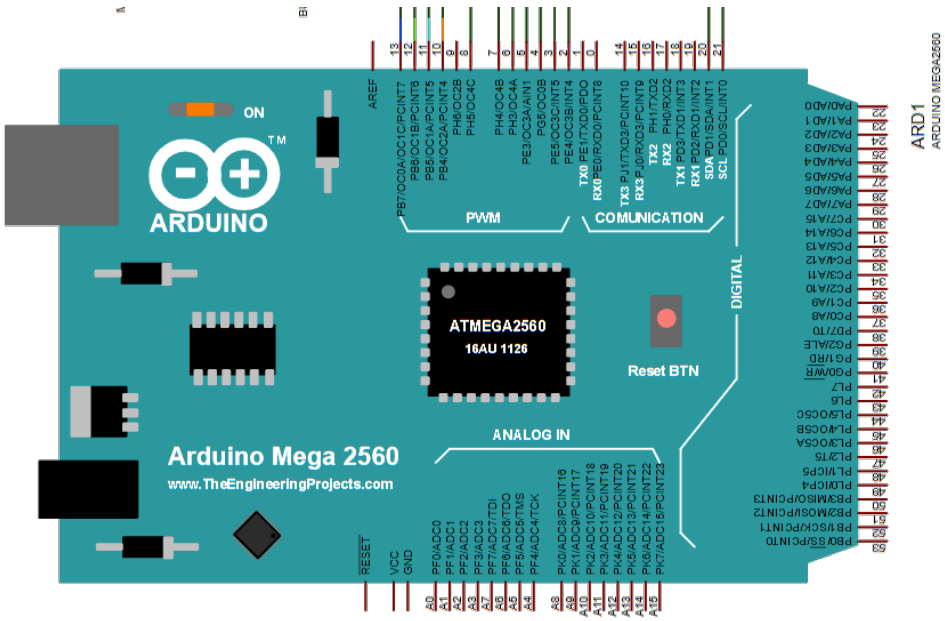
Hình 4-4. Cảm biến mưa

(3) Phần động cơ quay: thực hiện thao tác kéo thu sào đồ vào và thao tác đưa sào đồ ra bên ngoài. Đi kèm một nút bấm để có thể điều khiển thủ công bất cứ lúc nào.

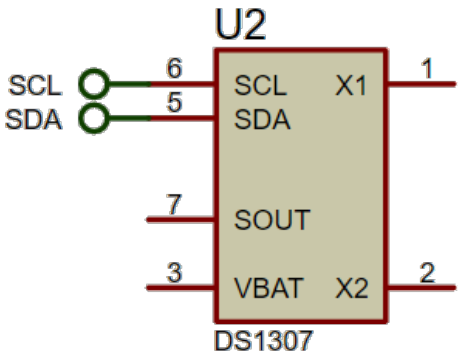


Hình 4-5. Động cơ quay servo motor và nút bấm

(4) Phần cứng: gồm board Arduino Mega 2560 và đồng hồ thời gian thực DS1307. Bộ phận này được nạp code xử lý logic, điều khiển thiết bị. Được lắp đặt cẩn thận, tránh tiếp xúc môi trường ngoài.



Hình 4-6. Board Arduino Mega 2560



Hình 4-7. Đồng hồ thời gian thực DS1307

b) Chạy thử

Sai số cho phép: ± 1 phút

STT	Thời gian đóng mở	Kiểu thời tiết	Hẹn giờ	Thời tiết thực (mưa, nhiệt, ẩm)	Chênh lệch thời gian quay servo	Hiển thị trên màn hình	Chênh lệch giá trị cảm biến nhiệt-ẩm	Chênh lệch cảm biến mưa	Tổng điểm
1	7-17	Sunny	Không	RAINING 28°C, 0%	1	1	1	1	4
2	7-17	Sunny	120p	NO RAIN 30°C, 5%	1	0	1	1	3
3	7-17	Cloudy	60p	NO RAIN 26°C, 2%	1	1	1	1	4
4	6-18	Sunny	120p	RAINING 32°C, 8%	1	1	1	1	4
5	6-18	Cloudy	60p	NO RAIN 24°C, 9%	1	1	0	1	3

Chênh lệch về nhiệt độ không quá 1.5°C.

Chênh lệch về độ ẩm không quá 2%.

Thời gian hiển thị trên màn hình chênh lệch không quá 1.5s.

Thời gian servo quay chênh lệch không quá 2.5s.

4.5. Tổng kết

a) Kết luận

Hệ thống phơi đồ thông minh của nhóm đã hoạt động đúng như dự định, đáp ứng những yêu cầu đã đề ra như sử dụng theo thời gian thực, tuỳ chỉnh ngưỡng chỉ số, có thể can thiệp điều khiển thủ công không phụ thuộc cảm biến. Giải quyết được hầu hết các tình huống, đôi khi vì yếu tố ngoại cảnh (cảm biến, phần cứng bị lỗi, hỏng hóc, tác động mạnh,..) có thể dẫn đến hoạt động của hệ thống sai sót.

Ưu điểm của sản phẩm:

- Sản phẩm nhìn chung hoàn thành được mục tiêu đề ra của đề bài.

- Sản phẩm có thể ứng dụng rộng rãi trong thực tiễn.
- Cách sử dụng đơn giản, 2 cơ chế điều khiển tự động và thủ công hoạt động theo thời gian thực.
- Thông số mặc định phù hợp với đa số vùng miền địa lý, có thể tuỳ chỉnh thông số của thiết bị.

Nhược điểm của sản phẩm

- Người dùng chỉ được chọn các ngưỡng thời tiết có sẵn trong tuỳ chọn do khả năng nhập liệu và hiển thị hạn chế của keypad và LCD.
- Hạn chế khả năng mô phỏng, có thể có sai số về thời gian thực do phụ thuộc vào hiệu năng của máy tính (CPU load).
- Chưa có chức năng lưu trữ các số liệu thu thập được, các dữ liệu đầu vào mới chỉ được xử lý ở mức thô sơ.

Thuận lợi trong quá trình thực hiện:

- Có sẵn các cảm biến cùng với các thư viện hỗ trợ cho việc lập trình được thuận tiện, ngắn gọn, nhanh chóng.

Khó khăn trong quá trình thực hiện:

- Quá trình xây dựng mối quan hệ giữa các tính năng.
- Xử lý logic, các trường hợp ngoại lệ trong quá trình hoạt động của 2 cơ chế tự động và thủ công.

b) Hướng phát triển

- Sử dụng thiết bị nhập liệu và hiển thị tối ưu hơn, bổ sung các tính năng nâng cao về quản lý, tuỳ chỉnh, nhận đánh giá từ người dùng…
- Tích hợp điều khiển thiết bị vào một hệ thống chung trong tổng thể hệ thống smarthome. Người dùng có thể nắm bắt các thông số, điều khiển và theo dõi các thiết bị thông qua 1 ứng dụng đa nền tảng.
- Thu thập và xử lý dữ liệu, ứng dụng các mô hình học máy, phân tích dữ liệu để hệ thống hoạt động thông minh hơn như đưa ra gợi ý ngưỡng thời tiết, cảm nhận và xử lý chỉ số điều khiển chính xác hơn, người dùng có thể xem được những thống kê, phân tích,…

PHỤ LỤC

Tổng hợp file dự án

Link google drive: Hệ thống Smarthome trên Arduino

Link video demo sản phẩm

Hệ thống quạt tự động

Hệ thống đèn tự động

Hệ thống phơi đồ tự động

Link flowchart

Hệ thống quạt tự động

Hệ thống đèn tự động

Hệ thống phơi đồ tự động

Link code Arduino của dự án

Hệ thống quạt tự động

Hệ thống đèn tự động

Hệ thống phơi đồ tự động

Link mạch arduino mô phỏng

Hệ thống quạt tự động (trên Tinkercad)

Hệ thống đèn tự động (trên Tinkercad)

Hệ thống phơi đồ tự động (trong phần mềm Proteus 8.13)

TÀI LIỆU THAM KHẢO

[1] A. Chalimov, "***IoT in Agriculture: 5 Technology Use Cases for Smart Farming (and 4 Challenges to Consider)***", *Eastern Peak - Technology Consulting & Development Company*, 2022. [Online]. Available: https://easternpeak.com/blog/iot-in-agriculture-technology-use-cases-for-smart-farming-and-challenges-to-consider/. [Accessed: 12- Mar- 2022].

[2] M. Ayaz, M. Ammad-Uddin, Z. Sharif, A. Mansour and E. Aggoune, "***Internet-of-Things (IoT)-Based Smart Agriculture: Toward Making the Fields Talk***", IEEE Access, vol. 7, pp. 129551-129583, 2019. Available: 10.1109/access.2019.2932609.

[3] "***Serial | print() | Cộng đồng Arduino Việt Nam***", Arduino.vn, 2022. [Online]. Available: http://arduino.vn/reference/library/serial/1/huong-dan-ham/print. [Accessed: 29- Feb- 2022].

[4] "***Hiển thị nhiệt độ, độ ẩm lên LCD 16×2 giao tiếp bằng I2C sử dụng Arduino***", Học Điện Tử, 2022. [Online]. Available: https://mobitool.net/hien-thi-nhiet-do-do-am-len-lcd-162-giao-tiep-bang-i2c-su-dung-arduino.html. [Accessed: 29- Feb- 2022].

[5] "***PWM là gì? Cách điều chế độ rộng xung***", Uniduc, 2022. [Online]. Available: https://uniduc.com/vi/blog/pwm. [Accessed: 02- Mar- 2022].

[6] "***Giới thiệu phần cứng Arduino Uno R3 | Các chân chức năng cơ bản***", Youtube.com, 2022. [Online]. Available: https://youtube.com/watch?v=k6BBEwRP4JE. [Accessed: 03- Mar- 2022].

[7] "***Cảm biến mưa với Arduino | Cộng đồng Arduino Việt Nam***", Arduino.vn, 2022. [Online]. Available: http://arduino.vn/bai-viet/274-cam-bien-mua-voi-arduino. [Accessed: 05- Mar- 2022].

[8] ***"[Khám phá thế giới IoT với bSmart] Bài 2 - Theo dõi nhiệt độ, độ ẩm và tạo báo động | Cộng đồng Arduino Việt Nam***", Arduino.vn, 2022. [Online]. Available: http://arduino.vn/bai-viet/7621-kham-pha-gioi-iot-voi-bsmart-bai-2-theo-doi-nhiet-do-do-am-va-tao-bao-dong. [Accessed: 01- Mar- 2022].

[9] ***"Mô phỏng cảm biến thời gian thực DS1307 hiển thị LCD.***", Youtube.com, 2022. [Online]. Available: https://www.youtube.com/watch?v=SXElPQIS-lg. [Accessed: 01- Mar- 2022].

[10] W.(c) 2022, "***0.96Inch OLED I2C Display 128x64***", Startingelectronics.org, 2022. [Online]. Available: https://startingelectronics.org/tutorials/arduino/modules/OLED-128x64-I2C-display/. [Accessed: 07- Mar- 2022].