ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Báo cáo Bài tập lớn Cuối kỳ 20232

Học phần: Hệ nhúng - IT4210
Đề tài: The Smart Green House System

NGUYỄN VĂN THÀNH ĐẠT 20225606
NGUYỄN MẠNH THÁI HÀ 20225621
NGUYỄN ĐỨC HOÀNG 20210370

Nhóm 2

Giảng viên hướng dẫn: TS.NGÔ LAM TRUNG

Hà Nội, 08/07/2024

Mục lục

1 Phân công

Họ và tên	MSSV	Phân công công việc
Nguyễn Văn Thành Đạt	20225606	Xử lý tín hiệu nút bấm và lắp mạch
Nguyễn Mạnh Thái Hà	20225621	Thiết kế phần mềm
Nguyễn Đức Hoàng	20210370	Kiểm thử project và làm báo cáo

2 Mô tả đề tài

2.1 Giới thiệu

Đề tài này tập trung vào việc xây dựng một hệ thống điều khiển và giám sát tự động sử dụng bộ vi điều khiển ESP32. Hệ thống này cho phép người dùng thiết lập các ngưỡng cho các thông số nhiệt độ, ánh sáng. Khi các thông số này vượt quá hoặc thấp hơn ngưỡng đã thiết lập, hệ thống sẽ tự động điều khiển các thiết bị như đèn, quạt để duy trì môi trường ổn định cho Smart Home.

2.2 Hướng dẫn sử dụng

Sau khi cấp nguồn 5V và 12V thì màn hình LCD sẽ hiển thị menu nhiệt độ, người dùng có 4 lựa chọn.

- Nhấn nút thứ nhất để tăng ngưỡng lên 0.5
- Nhấn nút thứ hai để giảm ngưỡng đi 0.5
- Nhấn nút thứ ba để chuyển sang menu tiếp theo
- Nhấn nút thứ tư để kết thúc quá trình chuyển ngưỡng bắt đầu hoạt động ,nếu muốn hệ thống quay lại chọn ngưỡng ấn lại nút 4 lần nữa.
- Hệ thống sẽ điều chỉnh thông số môi trường sao cho phù hợp với ngưỡng được chọn

3 Hệ thống phần cứng

3.1 Kiến trúc hệ thống

Hệ thống gồm 7 thành phần chính:

- Kit ESP32
- Mạch nút bấm
- Dây nối
- Hệ thống quạt

- Hệ thống cảm biến
- Relay

3.1.1 Kit ESP32

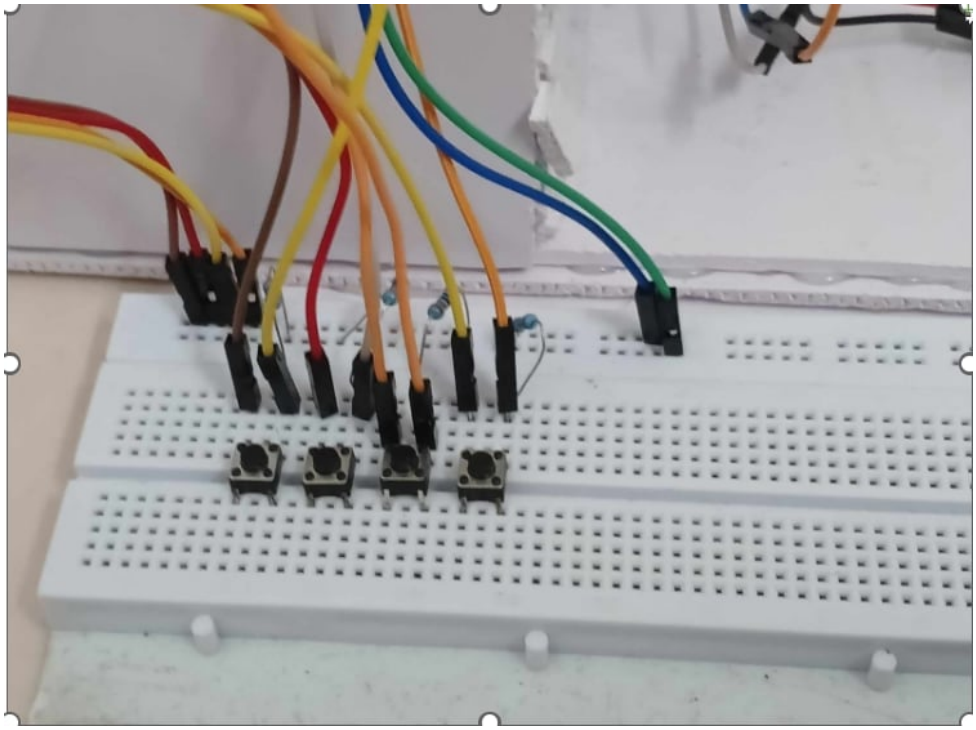
ESP32 là một bộ vi điều khiển thuộc danh mục vi điều khiển trên chip công suất thấp và tiết kiệm chi phí. Hầu hết tất cả các biến thể ESP32 đều tích hợp Bluetooth và Wi-Fi chế độ kép, làm cho nó có tính linh hoạt cao, mạnh mẽ và đáng tin cậy cho nhiều ứng dụng. ESP32 cũng được thiết kế để tiêu thụ điện năng thấp, lý tưởng cho các ứng dụng chạy bằng pin. Nó có hệ thống quản lý năng lượng cho phép nó hoạt động ở chế độ ngủ và chỉ thức dậy khi cần thiết, điều này có thể kéo dài tuổi thọ pin rất nhiều.

Hình 1: Kit ESP32

3.1.2 Mạch Nút Bấm

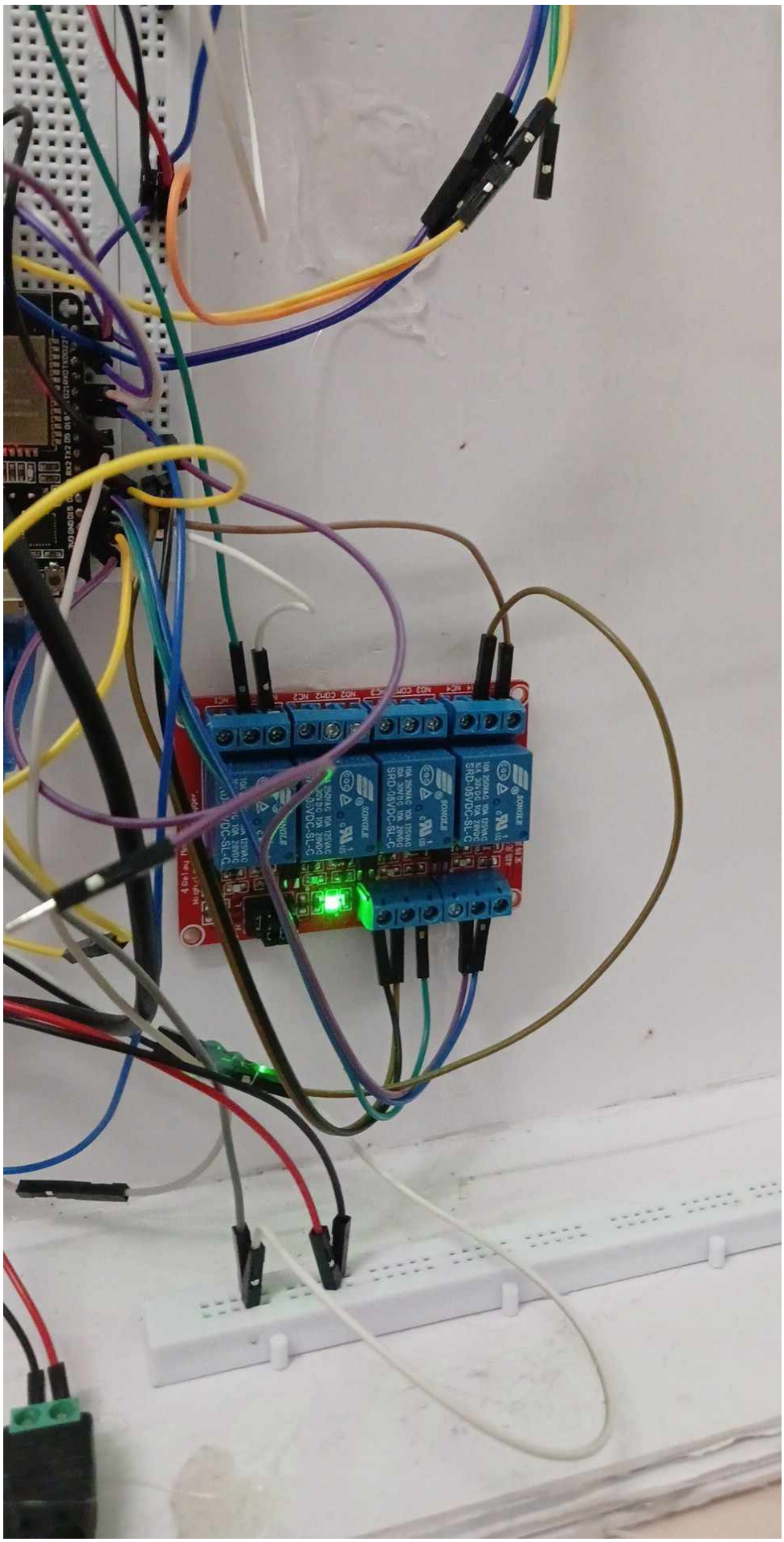
Nút 1: Tăng ngưỡng nhiệt độ và ánh sáng.
Nút 2: Giảm ngưỡng đi 0,5.
Nút 3: Chuyển sang menu tiếp theo chọn ngưỡng nhiệt độ.

Nút 4: kết thúc quá trình chuyển ngưỡng bắt đầu hoạt động ,nếu muốn hệ thống quay lại chọn ngưỡng ấn lại nút 4 lần nữa.

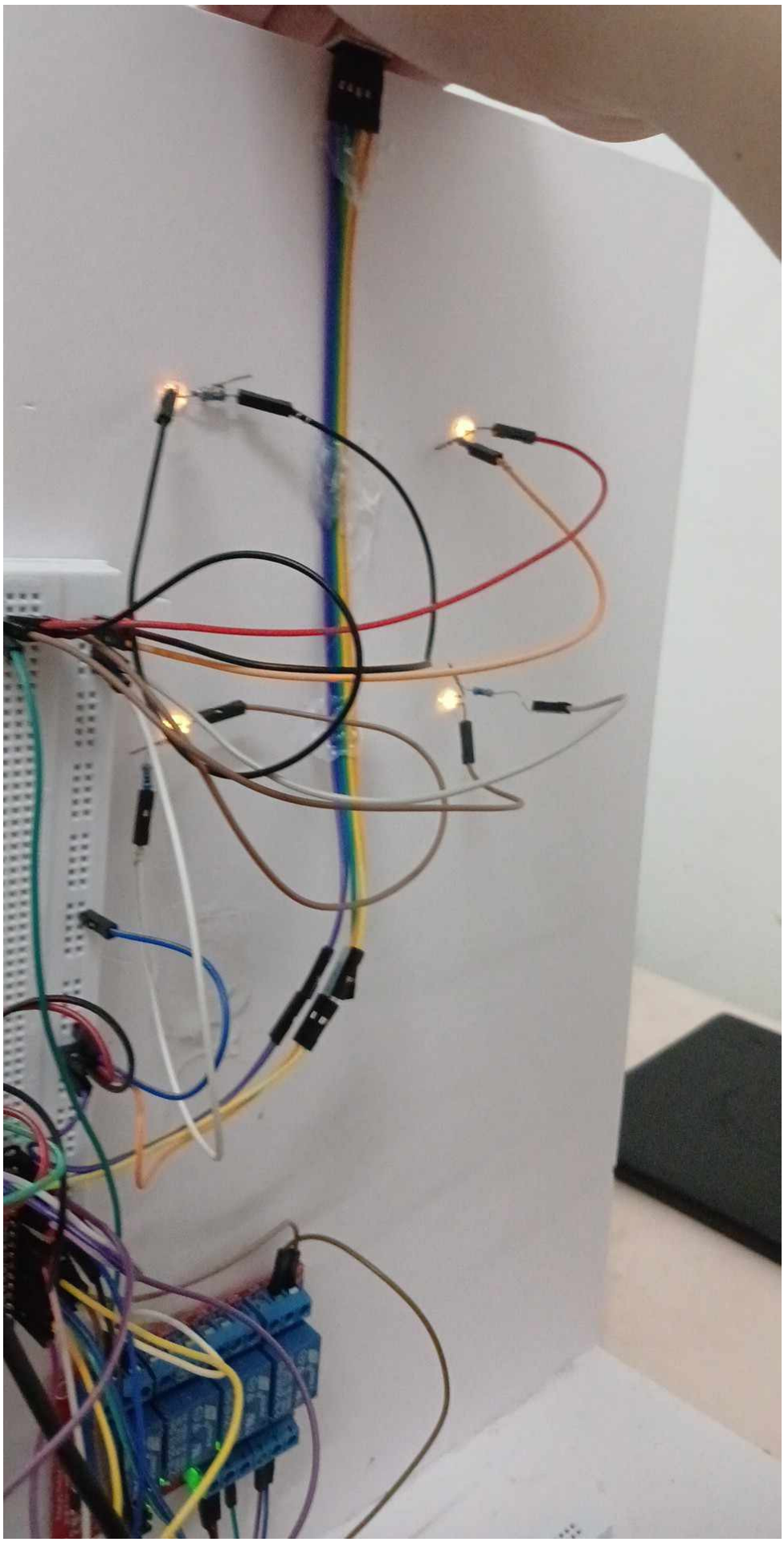


Hình 2: hệ thống nút bấm

Tất cả đèn, quạt đều được điều khiển qua relay để đảm bảo hoạt động chính xác và hiệu quả.



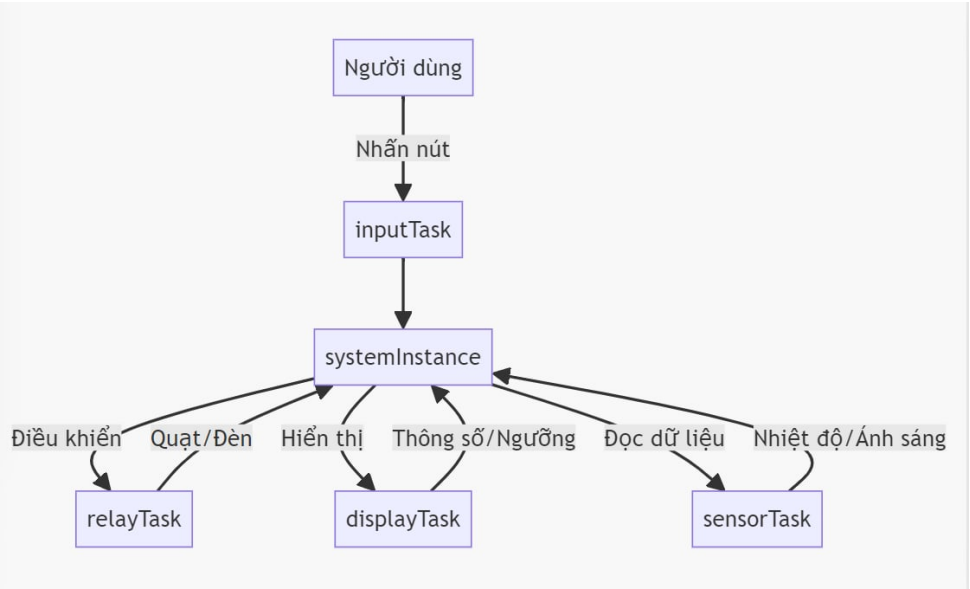
Hình 3: hệ thống relay



Hình 4: hệ thống đèn

4 Thông tin phần mềm

4.1 Kiến trúc phần mềm



Hình 5: sơ đồ hệ thống

Kiến trúc phần mềm có các thành phần chính:

- **inputTask**: Nhận dữ liệu từ người dùng thông qua các nút bấm và gửi tín hiệu tới systemInstance để xử lý. Mỗi nút bấm đại diện cho một hành động cụ thể như tăng/giảm ngưỡng nhiệt độ, ánh sáng, chuyển đổi giữa các menu hoặc thực thi ngưỡng hiện tại.
- **systemInstance**: Thành phần trung tâm của hệ thống, chịu trách nhiệm điều khiển toàn bộ hoạt động của hệ thống. Nó bao gồm các chức năng chính như điều khiển relay, hiển thị thông tin lên màn hình LCD, và đọc dữ liệu từ các cảm biến. systemInstance xử lý tất cả các tín hiệu từ inputTask và cập nhật trạng thái của hệ thống dựa trên các dữ liệu cảm biến và các ngưỡng đã thiết lập.
- **relayTask**: Điều khiển các relay để bật/tắt quạt, đèn dựa trên các ngưỡng đã thiết lập và dữ liệu cảm biến. relayTask đảm bảo rằng các thiết bị hoạt động đúng theo yêu cầu và điều kiện môi trường hiện tại.
- **displayTask**: Hiển thị các thông số môi trường (nhiệt độ, ánh sáng) và các ngưỡng đã thiết lập lên màn hình LCD. displayTask cập nhật thông tin dựa trên trạng thái hiện tại của hệ thống và các dữ liệu cảm biến được thu thập.

- **sensorTask**: Đọc dữ liệu từ các cảm biến nhiệt độ (DS18B20), ánh sáng (BH1750) . sensorTask cung cấp các thông tin môi trường cần thiết cho systemInstance để quyết định cách điều khiển các thiết bị và cập nhật thông tin hiển thị.

4.2 Hoạt động của phần mềm

4.2.1 Nhận dữ liệu từ người dùng

inputTask: Xử lý việc nhấn nút của người dùng. Đọc dữ liệu từ các nút bấm (`BUTTON1_PIN`, `BUTTON2_PIN`, `BUTTON3_PIN`, `BUTTON4_PIN`) và điều chỉnh ngưỡng hoặc chuyển đổi trạng thái hiển thị dựa trên đầu vào của người dùng.

```
  // Khởi tạo các nút bấm
  pinMode(BUTTON1_PIN, INPUT_PULLUP);
  pinMode(BUTTON2_PIN, INPUT_PULLUP);
  pinMode(BUTTON3_PIN, INPUT_PULLUP);
  pinMode(BUTTON4_PIN, INPUT_PULLUP);
}

void loop() {
  // Kiểm tra nút bấm và điều chỉnh ngưỡng
  static unsigned long lastButtonPress = 0;
  unsigned long debounceDelay = 200; // Thời gian debounce cho nút bấm

  if (millis() - lastButtonPress > debounceDelay) {
    if (digitalRead(BUTTON1_PIN) == LOW) {
      switch (currentState) {
        case STATE_SET_TEMP_THRESHOLD:
          tempThreshold += 0.5;
          break;
        case STATE_SET_LUX_THRESHOLD:
          luxThreshold += 0.5;
          break;
        default:
          break;
      }
      lastButtonPress = millis();
    }
    if (digitalRead(BUTTON2_PIN) == LOW) {
      switch (currentState) {
        case STATE_SET_TEMP_THRESHOLD:
          tempThreshold -= 0.5;
          break;
        case STATE_SET_LUX_THRESHOLD:
          luxThreshold -= 0.5;
          break;
```

```
        default:
          break;
      }
      lastButtonPress = millis();
    }
    if (digitalRead(BUTTON3_PIN) == LOW) {
      switch (currentState) {
        case STATE_SET_TEMP_THRESHOLD:
          currentState = STATE_SET_LUX_THRESHOLD;
          break;
        case STATE_SET_LUX_THRESHOLD:
          currentState = STATE_DISPLAY;
          break;
      }
      lastButtonPress = millis();
    }
    if (digitalRead(BUTTON4_PIN) == LOW) {

      if (currentState == STATE_DISPLAY) {
        currentState = STATE_SET_TEMP_THRESHOLD;
        systemPaused = true; // Tạm dừng hệ thống
      } else {
        currentState = STATE_DISPLAY;
        systemPaused = false; // Khởi động lại hệ thống
      }
      lastButtonPress = millis();
    }
  }

  if (systemPaused) {
    // Hiển thị thông tin cài đặt ngưỡng lên LCD
    lcd.clear();
    switch (currentState) {
      case STATE_SET_TEMP_THRESHOLD:
        lcd.setCursor(0, 0);
        lcd.print("Set Temp Thresh:");
        lcd.setCursor(0, 1);
        lcd.print(tempThreshold);
        lcd.print(" C");
        break;
      case STATE_SET_LUX_THRESHOLD:
        lcd.setCursor(0, 0);
        lcd.print("Set Lux Thresh:");
        lcd.setCursor(0, 1);
        lcd.print(luxThreshold);
        lcd.print(" Lux");
```

```
        break;
      default:
        break;
    }
  }

  delay(500); // Chờ trước khi cập nhật lại
}

```

4.2.2 Điều khiển các thiết bị như quạt, đèn.

relayTask: Ba dòng lệnh này thiết lập các chân điều khiển relay (RELAY_LIGHT_PIN, RELAY_FAN_PIN, RELAY_PUMP_PIN) thành các chân xuất (OUTPUT). Điều này có nghĩa là các chân này sẽ gửi tín hiệu điều khiển tới các relay. Đoạn code này điều khiển các thiết bị như quạt và đèn dựa trên các giá trị cảm biến đo được về nhiệt độ và độ sáng. Các điều kiện if-else được sử dụng để quyết định khi nào các thiết bị nên được bật hoặc tắt.

```


void loop() {
  // Điều khiển relay
  if (!systemPaused) {
    // Đọc nhiệt độ từ DS18B20
    sensors.requestTemperatures();
    float temperature = sensors.getTempCByIndex(0);
    // Đọc ánh sáng từ BH1750
    float lux = lightMeter.readLightLevel();

    if (temperature > tempThreshold) {
      digitalWrite(RELAY_FAN_PIN, HIGH); // Bật quạt
    } else {
      digitalWrite(RELAY_FAN_PIN, LOW); // Tắt quạt
    }

    if (lux < luxThreshold) {
      digitalWrite(RELAY_LIGHT_PIN, LOW); // Bật đèn
    } else {
      digitalWrite(RELAY_LIGHT_PIN, HIGH); // Tắt đèn
    }
  }

  delay(500); // Chờ trước khi cập nhật lại
```

```
}

```

4.2.3 Cảm biến

sensorTask: Đọc dữ liệu từ các cảm biến nhiệt độ, ánh sáng.
lcd.begin(16, 2);: Khởi tạo màn hình LCD với kích thước 16 cột và 2 hàng.
lcd.setBacklight(255);: Thiết lập độ sáng nền của màn hình LCD (255 là mức sáng tối đa).
lcd.setCursor(0, 0);: Đặt con trỏ ở vị trí cột 0, hàng 0.D.

```


void setup() {
  Serial.begin(115200);

  // Khởi tạo cảm biến nhiệt độ DS18B20
  sensors.begin();

  // Khởi tạo cảm biến ánh sáng BH1750
  Wire.begin();
  lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE);

  // Khởi tạo các relay
  pinMode(RELAY_LIGHT_PIN, OUTPUT);
  pinMode(RELAY_FAN_PIN, OUTPUT);

  // Tắt các relay ban đầu
  digitalWrite(RELAY_LIGHT_PIN, HIGH);
  digitalWrite(RELAY_FAN_PIN, LOW);
}
```

5 Kết quả thực hiện

5.1 Tự đánh giá

- Tất cả các thành phần của hệ thống đã được xây dựng và kiểm tra thành công.
- Hệ thống hoạt động ổn định và đáp ứng tốt các yêu cầu đề ra.
- Môi trường hệ thống kiểm soát được duy trì ổn định thông qua các thiết bị điều khiển tự động.