**ĐẠI HỌC BÁCH KHOA HÀ NỘI**



**TRƯỜNG ĐIỆN – ĐIỆN TỬ**



**THIẾT KẾ HỆ THỐNG NHÚNG EE4251 – 154748 – Nhóm 19**

**ĐỀ TÀI: THIẾT KẾ MÔ HÌNH ĐƠN NHIỆM, ĐA NHIỆM ĐO NHIỆT ĐỘ, ĐỘ ẨM**

**GVHD: TS. Lê Công Cường**

|  |  |
| --- | --- |
| **Họ và tên** | **MSSV** |
| Đặng Đình Hiển | 20210322 |
| Nguyễn Văn Doanh | 20212716 |
| Nguyễn Việt Anh | 20212690 |

**Hà Nội, 01/2025**

**MỤC LỤC**

[**CHƯƠNG 1: TỔNG QUAN** 4](#_Toc188479722)

[**1.1 Thông tin nhóm** 4](#_Toc188479723)

[**1.2 Yêu cầu bài tập lớn** 4](#_Toc188479724)

[**CHƯƠNG 2: GHÉP NỐI PHẦN CỨNG** 5](#_Toc188479725)

[**2.1. Nguyên lý làm việc của các module phần cứng** 5](#_Toc188479726)

[**2.2 Các ngoại vi STM32 sử dụng** 8](#_Toc188479727)

[**2.3 Sơ đồ ghép nối giữa vi điều khiển và module phần cứng** 9](#_Toc188479728)

[**2.4 Sơ đồ khối, lưu đồ thuật toán** 9](#_Toc188479729)

[**CHƯƠNG 3: THIẾT KẾ PHẦN MỀM** 13](#_Toc188479730)

[**3.1 Phân tích yêu cầu xử lý để đảm bảo tính thời gian thực** 13](#_Toc188479731)

[**3.2 Lựa chọn và tính toán tham số cho các task** 16](#_Toc188479732)

[3.2.1 Phân tích và lập trình theo mô hình thời gian 17](#_Toc188479733)

[3.2.2 Phân tích và lập trình theo mô hình đa nhiệm 17](#_Toc188479734)

[**CHƯƠNG 4: ĐÁNH GIÁ KẾT QUẢ** 19](#_Toc188479735)

[**4.1 Kết quả đạt được** 19](#_Toc188479736)

[**4.2 Phân tích và đánh giá kết quả** 20](#_Toc188479737)

[**4.3 Đánh giá mức độ đóng góp của các thành viên** 20](#_Toc188479738)

**DANH LỤC HÌNH ẢNH**

[Hình 1 DS18B20 5](#_Toc188434045)

[Hình 2 Soil Moisture Sensor V1.2 6](#_Toc188434046)

[Hình 3 Màn hình LCD 1602 6](#_Toc188434047)

[Hình 4 PCF8574T 7](#_Toc188434048)

[Hình 5 TTL CP2102 7](#_Toc188434049)

[Hình 6 Kit vi điều khiển STM32F103C8T6 Blue Pill 8](#_Toc188434050)

[Hình 7 Sơ đồ chân của kit Bluepill 9](#_Toc188434051)

[Hình 8 Sơ đồ nguyên lý hệ thống 9](#_Toc188434052)

[Hình 9 Sơ đồ khối của hệ thống 10](#_Toc188434053)

[Hình 10 Lưu đồ VĐK giao tiếp với DS1820B 11](#_Toc188434054)

[Hình 11 Lưu đồ VĐK giao tiếp với cảm biến độ ẩm đất 11](#_Toc188434055)

[Hình 12 Lưu đồ VĐK giao tiếp với TTL CP2102 12](#_Toc188434056)

[Hình 13 Lưu đồ VĐK giao tiếp với LCD-I2C 13](#_Toc188434057)

[Hình 14 Trạng thái Task 15](#_Toc188434058)

[Hình 15 Lập lịch theo Round - Robin 18](#_Toc188434059)

[Hình 16 Mạch thực nghiệm 19](#_Toc188434059)

[Hình 17 Kết quả truyền nhận dữ liệu qua UART 19](#_Toc188434059)

# **CHƯƠNG 1: TỔNG QUAN**

## **1.1 Thông tin nhóm**

|  |  |
| --- | --- |
| **Thành viên nhóm** | **Phụ trách công việc** |
| Nguyễn Văn Doanh  20212716 | Lập trình đa nhiệm, truyền nhận dữ liệu qua UART, giao tiếp với LCD, viết báo cáo |
| Nguyễn Việt Anh  20212690 | Lập trình đơn nhiệm, hiển thị lên LCD, giao tiếp với DS18B20 |
| Đặng Đình Hiển  20210322 | Lập trình đa nhiệm, giao tiếp cảm biến Soil Moisture, viết báo cáo |

## **1.2 Yêu cầu bài tập lớn**

- Lập trình VĐK STM32 đọc cảm biến nhiệt độ, độ ẩm hiện thị thông số lên LCD đồng thời truyền thông số lên máy tính thông qua UART. Cài đặt được thông số lập lịch và nhận lệnh điều khiển từ máy tính theo mô hình đơn nhiệm và đa nhiệm

- Yêu cầu cụ thể:

* Cảm biến nhiệt độ: DS18B20.
* Cảm biến độ ẩm đất điện dung: Soil Moisture Sensor V1.2.
* Hiển thị thông số lên LCD I2C đồng thời truyền thông số lên máy tính thông qua UART.
* Lập trình theo một mô hình đơn nhiệm và hai mô hình đa nhiệm.
* Kiểm tra và đánh giá hệ thống.

# **CHƯƠNG 2: GHÉP NỐI PHẦN CỨNG**

## **2.1. Nguyên lý làm việc của các module phần cứng**

**2.1.1. Cảm biến nhiệt độ DS18B20**

- Cảm biến nhiệt độ DS18B20 là một loại cảm biến nhiệt độ kỹ thuật số chính xác cao được sản xuất bởi Maxim Integrated. Nó là một loại cảm biến nhiệt độ kỹ thuật số 1 dây, rất phổ biến và thường được sử dụng trong các ứng dụng IoT (Internet of Things).

- Thông số kỹ thuật:

* Nguồn cung cấp: 3.0V – 5.5V.
* Dải đo: -55°C đến +125°C.
* Độ chính xác: ±0.5°C.
* Độ phân giải: lập trình 9 bit – 12 bit.



Hình 1. DS18B20

- Một trong những đặc điểm nổi bật của cảm biến nhiệt độ DS18B20 là cảm biến này chỉ sử dụng 1 dây để kết nối với vi điều khiển, làm cho việc kết nối trở nên đơn giản và tiết kiệm chi phí.

**2.1.2. Cảm biến độ ẩm đất điện dung**

- Cảm biến độ ẩm đất Soil Moisture Sensor V1.2 có thiết kế đơn giản, dễ sử dụng, đặc biệt phần điện cực cắm xuống đất được phủ lớp sơn chống ăn mòn nên cảm biến có độ bền và độ ổn định cao hơn các loại mạ kẽm thông thường, cảm biến trả giá trị độ ẩm đất tương ứng với điện áp qua chân Analog, thích hợp cho các ứng dụng tưới cây tự động, vườn thông minh.

- Thông số kỹ thuật:

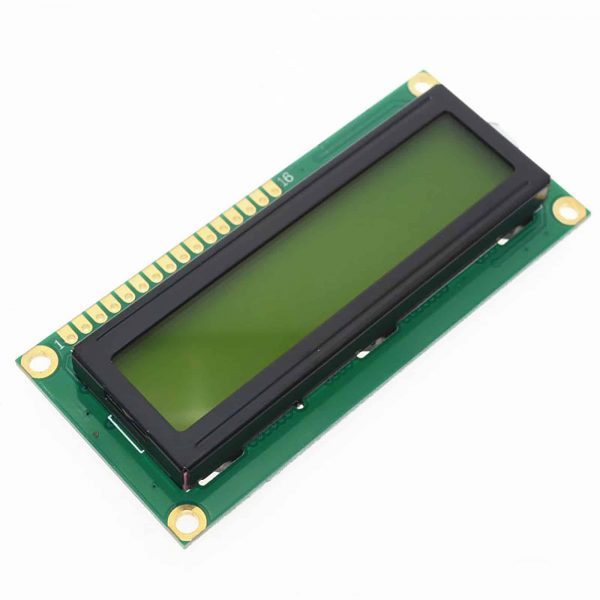
* Điện áp hoạt động: 3.3-5VDC
* Điện cục phủ sơn chống ăn mòn cho độ bền và độ ổn định cao
* Điện áp xuất ra chân analog: 0-3VDC
* Chuẩn jack cắm: PH2.5454-3P
* Kích thước PCB: 98x23mm

****

Hình 2. Soil Moisture Sensor V1.2

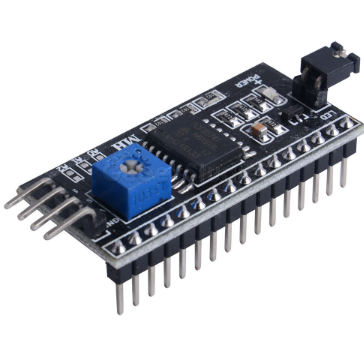
**2.1.3. Màn hình LCD**

- Màn hình LCD 1602 xanh lá sử dụng driver HD44780, có khả năng hiển thị 2 dòng với mỗi dòng 16 ký tự, màn hình có độ bền cao, rất phổ biến, nhiều code mẫu và dễ dàng sử dụng hơn nếu đi kèm mạch chuyển tiếp I2C.



Hình 3. Màn hình LCD 1602

- Mạch chuyển tiếp I2C sử dụng module PCF8574T có thể tăng số lượng chân GPIO trên vi điều khiển thông qua giao tiếp I2C. Nó đặc biệt hữu ích trong các ứng dụng yêu cầu nhiều chân I/O hơn khả năng cung cấp của vi điều khiển.

****

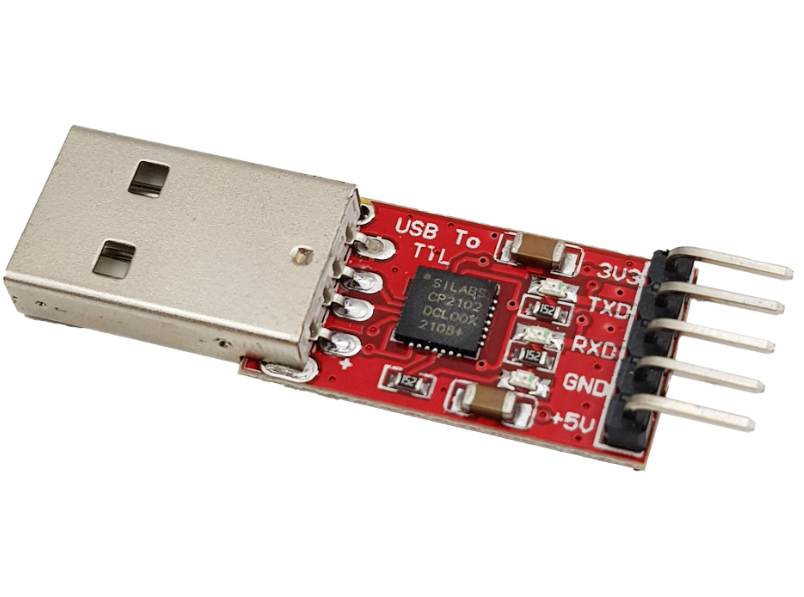
Hình 4. PCF8574T

- Thông số kỹ thuật:

* **Nguồn cấp (Vcc):** 2.5V đến 6V.
* **Số chân I/O:** 8 chân GPIO (P0 đến P7)
* Địa chỉ I2C: Có thể cấu hình từ 0x20 đến 0x27 (3bit cấu hình A0, A1, A2).
* **Tần số giao tiếp:** Lên đến 100 kHz (I2C chuẩn) hoặc 400 kHz (I2C tốc độ cao).

**2.1.4. Mạch chuyển đổi USB UART**

**-** TTL CP2102 là một module chuyển đổi tín hiệu **USB sang UART (TTL)**. Nó thường được sử dụng để giao tiếp giữa máy tính và các thiết bị sử dụng giao tiếp UART.



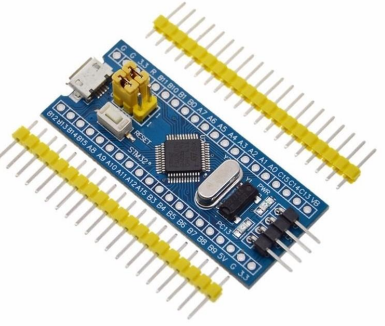
Hình 5. TTL CP2102

- Thông số kỹ thuật:

* Sử dụng điện áp 5VDC cấp trực tiếp từ cổng USB.
* Các chân tín hiệu ngõ ra:
  + 5V: Chân cấp nguồn 5VDC từ cổng USB, tối đa 500mA.
  + VCC: Chân chọn mức tín hiệu TTL cho TX/RX, nếu nối với chân 5V sẽ là 5VDC, nối với 3V3 sẽ là 3.3VDC.
  + 3V3: Chân nguồn 3.3VDC (dòng cấp rất nhỏ tối đa 100mA), không sử dụng để cấp nguồn, thường chỉ sử dụng để thiết đặt mức tín hiệu Logic.
  + TXD: chân truyền dữ liệu UART TTL, dùng kết nối đến chân nhận RX của các module sử dụng mức tín hiệu TTL.
  + RXD: chân nhận dữ liệu UART TTL, dùng kết nối đến chân nhận TX của các module sử dụng mức tín hiệu TTL.
* GND: chân cấp nguồn 0VDC.
* Tích hợp led tín hiệu TX, RX.
* Tương thích với hầu hết các hệ điều hành hiện nay: Windows, Mac, Linux

**2.1.5 Kit nạp**

- Kit STM32F103C8T6 Mini là một dòng chip phổ thông nhất của STM, có đầy đủ các ngoại vi cơ bản, bộ nhớ vừa đủ và giá thành rẻ, rất phù hợp với những người mới bắt đầu học và nghiên cứu.

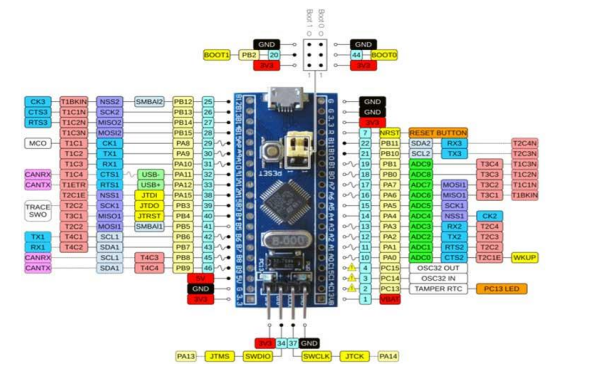


Hình 6. Kit vi điều khiển STM32F103C8T6 Blue Pill

- Dưới đây là một số thông số kỹ thuật của kit:

* Điện áp hoạt động: 3.3V → 5V
* Tích hợp thạch anh ngoại 8 Mhz
* Tích hợp cổng USB để nạp Bootloader hoặc sử dụng ngoại vi USB
* Đèn led tích hợp ở chân PC13 của vi điều khiển
* Sử dụng chuẩn mạch SWD, thích hợp với các loại mạch nạp ST-Link, JLink.

## **2.2 Các ngoại vi STM32 sử dụng**

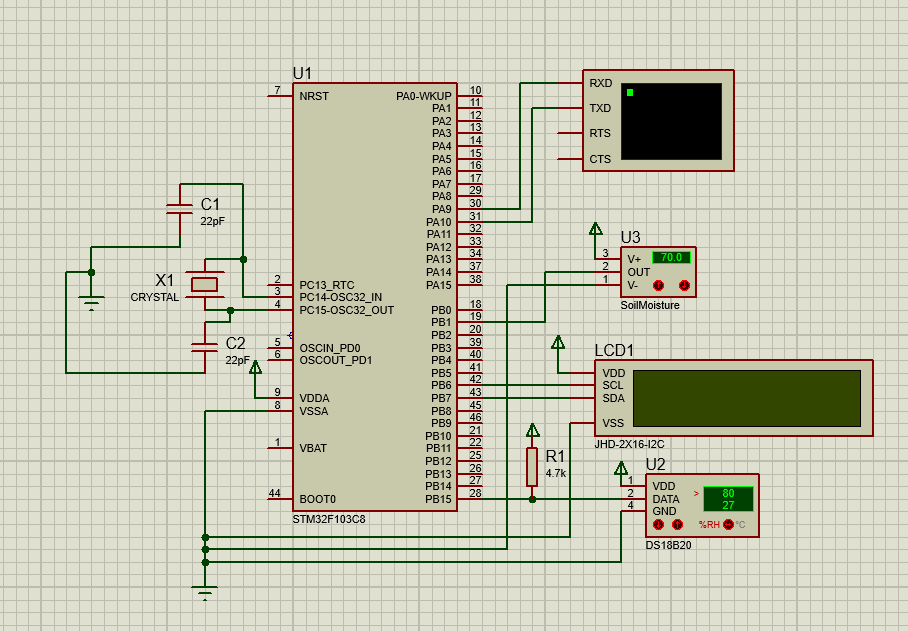
****

Hình 7. Sơ đồ chân của kit Bluepill

Với yêu cầu đề tài, các ngoại vi cần sử dụng bao gồm:

* GPIO\_Output: cảm biến DS18B20
* ADC: đọc giá trị cảm biến Soil Moisture
* UART: giao tiếp và nhận lệnh điều khiển từ máy tính
* I2C: hiển thị dữ liệu lên LCD

## **2.3 Sơ đồ ghép nối giữa vi điều khiển và module phần cứng**

****

Hình 8. Sơ đồ nguyên lý hệ thống

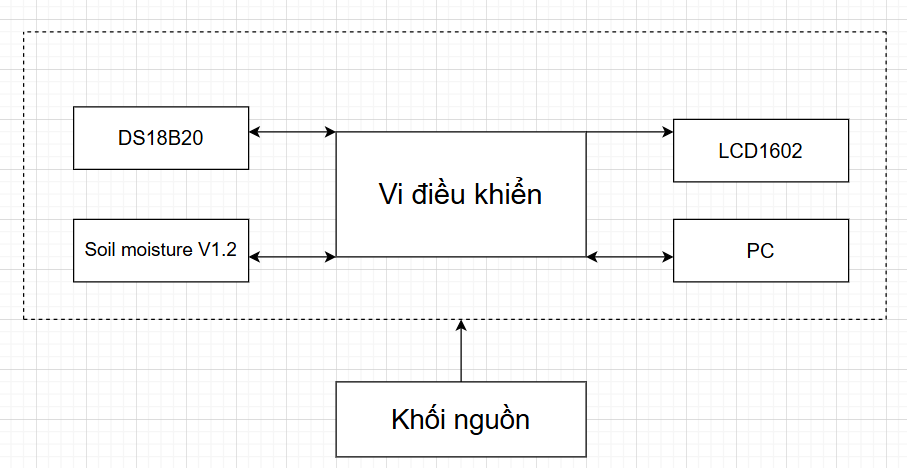
- Trên đây là sơ đồ mạch nguyên lý ghép nối vi điều khiển với các module phần cứng bao gồm:

* Mạch nạp
* Nút reset vi điều khiển
* Module LCD1602-I2C
* Cảm biến nhiệt độ DS18B20
* Cảm biến độ ẩm Soil Moisture V1.2

- Ngoài ra còn có 2 thạch anh (32768 Hz và 8 MHz) cung cấp xung cho vi điều khiển hoạt động.

## **2.4 Sơ đồ khối, lưu đồ thuật toán**

**2.4.1 Sơ đồ khối của hệ thống**

****

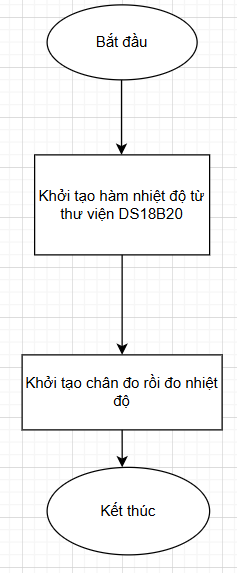
Hình 9. Sơ đồ khối của hệ thống

- Với các yêu cầu từ bài tập và các module phần cứng đã có, chúng em đã thiết kế hệ thống gồm các khối:

* Khối nguồn: cung cấp năng lượng cho toàn bộ hệ thống hoạt động
* Khối vi điều khiển: điều khiển, tính toán các hoạt động của hệ thống
* Cảm biến Soil moisture V1.2: thu thập dữ liệu độ ẩm truyền về vi điều khiển
* Cảm biến nhiệt độ DS18B20: Thu thập dữ liệu nhiệt độ gửi về vi điều khiển
* Khối module LCD1602: Hiển thị thông tin lên màn hình
* PC: Cài đặt thông số, hiển thị thông tin từ vi điều khiển qua module TTL CP2102

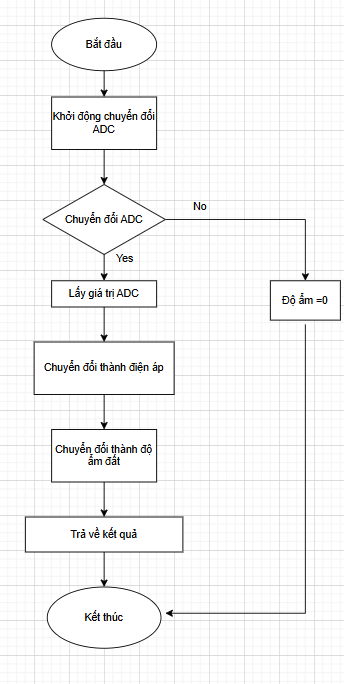
**2.4.2 Lưu đồ thuật toán giao tiếp vi điều khiển với module phần cứng**

**a. Vi điều khiển với cảm biến nhiệt độ DS18B20**

****

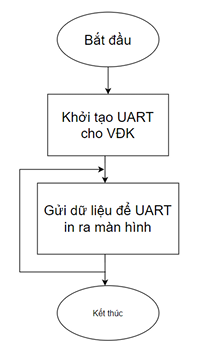
Hình 10. Lưu đồ VĐK giao tiếp với DS18B20

**b. Vi điều khiển với cảm biến độ ẩm Soil Moisture V1.2**

****

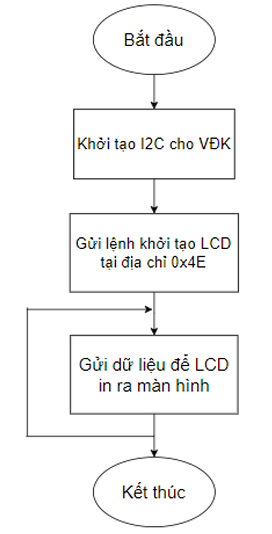
Hình 11. Lưu đồ VĐK giao tiếp với cảm biến độ ẩm đất

**c. Vi điều khiển với module TTL CP2102**

****

Hình 12. Lưu đồ VĐK giao tiếp với TTL CP2102

**d. Vi điều khiển với Module I2C – LCD1602**

****

Hình 13. Lưu đồ VĐK giao tiếp với LCD-I2C

# **CHƯƠNG 3: THIẾT KẾ PHẦN MỀM**

## **3.1 Phân tích yêu cầu xử lý để đảm bảo tính thời gian thực**

**3.1.1 Hệ điều hành thời gian thực RTOS**

- Hệ thống thời gian thực là hệ thống mà sự hoạt động tin cậy của nó không chỉ phụ thuộc vào sự chính xác của kết quả mà còn phụ thuộc vào thời điểm đưa ra kết quả, hệ thống có lỗi khi yêu cầu về thời gian không được đáp ứng. Bảng dưới đây so sánh hệ điều hành thông thường và hệ điều hành thời gian thực:

|  |  |
| --- | --- |
| **Hệ điều hành thông thường** | **Hệ điều hành thời gian thực** |
| Có thể kể đến Window, Linux, Android, Ios, … Khi mở một phần mềm trên đó, có thể chúng ta phải chờ đợi thời gian tải dữ liệu khá lâu, việc chờ đợi này cũng không ảnh hưởng gì bởi vì đa số phần mềm tương tác với con người chứ ít tương tác với các phần mềm hay thiết bị khác. | Các tác vụ cần sự phản hồi nhanh của hệ thống, thường được nhúng trong các dòng vi điều khiển và không có giao diện tương tác với người dùng. Chúng cần thời gian phản hồi nhanh bởi vì đa số tác vụ tương tác với thiết bị, máy móc khác chứ không phải con người. Các tài nguyên bên trong rất hữu hạn nên chỉ một sự trễ rất nhỏ cũng có thể làm hệ thống sai lệch hoàn toàn. |

- Chúng ta cần sử dụng RTOS khi thiết kế ứng dụng lớn, nhiều trạng thái máy hoặc liên quan tới các tác vụ xử lý nhanh, xử lý ảnh, âm thanh. RTOS sẽ chia các ứng dụng phức tạp thành các phần nhỏ hơn và dễ quản lý hơn. Ta có thể kể đến một số ưu điểm của RTOS:

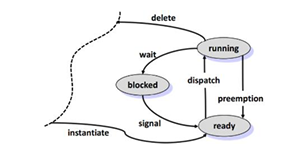
* Chia sẻ tài nguyên một cách đơn giản: Cung cấp cơ chế phân chia các yêu cầu về bộ nhớ và ngoại vi của MCU
* Dễ debug và phát triển: mọi người trong nhóm có thể làm việc một cách độc lập, tránh các tương tác với ngắt, timer, với phần cứng.
* Tăng tính linh động và dễ dàng bảo trì: thông qua API của RTOS.

- Hệ điều hành thời gian thực được chia làm 2 loại:

* Hệ điều hành thời gian thực cứng (Hard Real Time Operating System): là hệ điều hành thời gian thực mà các tác vụ không chỉ đúng về thực thi mà còn phải đúng về thời gian, không cho phép sai lệch về thời gian. Nó 14 thường được sử dụng ở các lấp thấp với tầng vật lý. Ví dụ như hệ thống điều khiển năng lượng ô tô, máy điều hòa nhịp tim, hệ thống điều khiển các quá trình công nghiệp, …
* Hệ điều hành thời gian thực mềm (Soft Real Time Operating System): là hệ điều hành thời gian thực cho phép sai lệch về thời gian và dung sai lỗi ở một mức độ nhất định. Nó thường được sử dụng ở những chỗ xảy ra sự tranh chấp và cần giữ số kết nối hệ thống phản ứng với sự thay đổi hoàn cảnh. Ví dụ: phần mềm duy trì và cập nhật kế hoạch bay hãng hàng không, hệ thống truyền hình, nghe nhạc trực tuyến, bộ điều khiển máy giặt, …

- Một số khái niệm quan trọng trong hệ điều hành thời gian thực RTOS:

* Kernel (nhân): có nhiệm vụ quản lý và điều phối các task. Mọi sự kiện (event) như ngắt, timer, data truyền tới, … đều qua kernel xử lý để quyết định xem nên làm gì tiếp theo. Thời gian xử lý của kernel thường rất nhanh nên độ trễ rất thấp.
* Task (tác vụ): là một đoạn chương trình thực thi một hay nhiều nhiệm vụ, được Kernel quản lý. Kernel sẽ quản lý việc chuyển đổi giữa các task, nó sẽ lưu lại ngữ cảnh của task sắp bị hủy và khôi phục lại ngữ cảnh của task tiếp theo bằng cách:
  + Kiểm tra thời gian thực thi đã được định nghĩa trước (time slice được tạo ra bởi ngắt systick).
  + Khi có các sự kiện unblocking một task có quyền cao hơn xảy ra (signal, queue, semaphore, …)
  + Khi task gọi hàm Yield () để ép Kernel chuyển sang các task khác mà không phải chờ cho kết time slice
  + Khi khởi động thì kernel sẽ tạo ra một task mặc định gọi là Ilde Task.
* Trạng thái Task:



Hình 14. Trạng thái Task

* + Running: một task đang được thực thi bởi vi xử lý (trong time slice của task). Tại một thời điểm chỉ có duy nhất trong trạng thái running.
  + Ready: trạng thái chờ của task để vào trạng thái running.
  + Blocked: task không sẵn sàng chạy (vào trạng thái running) kể cả khi bộ vi xử lý trong trạng thái nghỉ. Task ở trong trạng thái này vì chúng đợi một sự kiện bên ngoài tác động để kích hoạt nó trở lại trạng thái Ready (ví dụ task khác đang dùng UART, task này cần chưa được giải phóng).

**3.1.2 Hệ điều hành FreeRTOS**

**a) Khái niệm**

* FreeRTOS là lõi của hệ điều hành thời gian thực miễn phí. Hệ điều hành này được Richard Barry công bố rộng rãi từ năm 2003, phát triển mạnh đến nay và được cộng động mạng mã nguồn mở ủng hộ.
* FreeRTOS có tính khả chuyển, mã nguồn mở, lõi có thể tải miễn phí và nó có thể dùng cho các ứng dụng thương mại. Nó có nhiều ưu điểm nỗi bật so với các hệ điều hành nhúng thời gian thực khác như có kích thước nhỏ gọn nên rất phù hợp với các hệ thống nhúng thời gian thực nhỏ, được viết bằng ngôn ngữ C nên có độ phù hợp cao với nền phần cứng khác nhau.
* Ngoài ra FreeTOS còn hỗ trợ các cơ chế như cho phép tạo cả task và coroutie với số lượng task không giới hạn phụ thuộc vào tài nguyên của phần cứng chip, hỗ trợ cơ chế truyền thông đồng bộ giữa các task hoặc giữa task và ngắt bằng cách sử dụng hàng đợi hoặc semaphore đếm và các mutex, cho phép nhận biết khi ngăn xếp bị tràn.
* Ngay cả trong các hệ thống nhúng lớn người ta vẫn có thể sử dụng FreeRTOS để tiết kiệm được dung lượng bộ nhớ và làm cho hệ thống ít bị quá tải.

**b) Quản lý hoạt động trong FreeRTOS**

* Quản lý các task: đây là nhiệm vụ quan trọng trong FreeRTOS. Các nhiệm vụ mà người lập trình muốn hệ thống thực hiện sẽ được viết trong nội dung các task. Mỗi task sẽ được gán một độ ưu tiên phù hợp và được bộ lập lịch sắp xếp thời gian hoạt động.
* Quản lý hàng đợi: FreeRTOS cung cấp cơ chế hàng đợi (Queue) hoạt động theo nguyên tắc FIFO (vào trước ra trước). Hàng đợi là nơi lưu trữ dữ liệu của các task. Khi một task chờ để ghi dữ liệu vào hàng đợi hoặc là đọc dữ liệu ra từ hàng đợi thì nó sẽ bị rơi vào trạng thái block.
* Quản lý sự kiện ngắt: FreeRTOS cho phép quản lý hiệu quả các ngắt. Khi ngắt xảy ra CPU sẽ chuyển vào hàm thủ tục ngắt, hàm thủ tục ngắt phát đi một tín hiệu để hàm thực hiện chức năng của ngắt thực hiện, hàm này có độ ưu tiên cao hơn tất cả các task khác nên nó sẽ được thực hiện ngay lập tức. Tín hiệu ở đây chính là Semaphore, trong FreeRTOS có hai loại Semaphore là Binary Semaphore và Counting Semaphore với hai tác động chính vào Semaphore là “Take” và “Give”. "Take" là dùng cho hàm thực hiện chức năng của ngắt, khi chưa có ngắt hàm này sẽ ở trạng thái khóa (Blocking) và chờ đợi sự kiện ngắt xảy ra. Tác động "Give" được thực hiện trong hàm thủ tục ngắt, nó sẽ phát ra tín hiệu là có ngắt xảy ra (Semaphore ở trạng thái Full), khi đó hàm thực hiện chức năng ngắt sẽ ngay lập tức được chuyển sang trạng thái sẵn sàng (Semaphore ở trạng thái Empty). Sau khi thực hiện xong nó lại trở lại trạng thái khóa và chờ đợi cho sự kiện ngắt tiếp theo xảy ra.
* Đồng bộ dữ liệu trong FreeRTOS: Nếu Semaphore được sử dụng trong FreeRTOS để đồng bộ các sự kiện thì Mutex được sử dụng để đồng bộ dữ liệu giữa các task khi chúng cần truy xuất đến một vùng nhớ chứa dữ liệu chung. Mutex khác semaphore ở chỗ là nó bắt buộc phải trả về vùng đệm dữ liệu sau khi dùng. Để tạo ra một Mutex sử dụng phương thức xSemaphoreCreateMutex (void). Hai hàm tác động vào Mutex là "Take" và "Give" được định nghĩa để bảo vệ vùng mã không cho phép task khác truy xuất khi một task đang sử dụng nó.

**c) Phân tích yêu cầu xử lý thời gian để đảm bảo tính thời gian thực**

Chương trình chúng em gồm hoạt động sau:

* Vi điều khiển giao tiếp với DS18B20 để đọc giá trị nhiệt độ từ môi trường
* Vi điều khiển giao tiếp với Soil Moisture để đọc giá trị độ ẩm từ môi trường
* Hiển thị giá trị nhiệt độ, độ ẩm lên màn hình LCD
* Truyền giá trị nhiệt độ, độ ẩm đó từ vi điều khiển để hiển thị trên máy tính thông qua UART CP2102

- Để xây dựng chương trình tham số đảm bảo tính thời gian thực thì ta cần đánh giá, xác định các tham số sau của mỗi task:

* Period (Chu kỳ): Khoảng thời gian giữa các lần kích hoạt tiến trình.
* Initiation interval (Khoảng cách khởi tạo): Nghịch đảo của chu kỳ (1/chu kỳ).
* Initiation time (Thời gian khởi tạo): Thời điểm mà tiến trình trở nên sẵn sàng để thực thi.
* Deadline (Thời hạn): Thời điểm mà tiến trình phải hoàn thành trước đó.
* Response time (Thời gian phản hồi): Khoảng thời gian từ khi một “sự kiện” xảy ra cho đến khi CPU phản hồi lại.

- Những yếu tố cần cân nhắc trong quá trình lập lịch:

* Response time to an event: Thời gian từ khi một sự kiện xảy ra đến khi hệ thống phản hồi lại.
* Turnaround time: Tổng thời gian cần thiết để một tiến trình được hoàn thành từ khi bắt đầu đến khi kết thúc.
* Overhead: Tài nguyên hoặc thời gian bổ sung cần thiết để thực hiện quản lý hệ thống, như chuyển ngữ cảnh hoặc lên lịch.
* Fairness: Ai sẽ được thực thi tiếp theo, đảm bảo tài nguyên được phân chia hợp lý giữa các tiến trình.
* Throughput: Số lượng tác vụ hoàn thành mỗi giây.
* Starvation: Khi một tác vụ không bao giờ được thực thi do bị các tác vụ khác ưu tiên hơn.
* Preemptive vs. non-preemptive scheduling:
* Preemptive: Cho phép ngắt một tiến trình đang chạy để thực thi tiến trình khác.
* Non-preemptive: Một tiến trình chỉ bị thay thế khi nó hoàn thành.
* Deterministic scheduling: Đảm bảo thời gian thực hiện của các tiến trình.
* Static, dynamic scheduling:
  + Static scheduling: Quyết định trình tự thực thi trước khi chương trình chạy.
  + Dynamic scheduling: Quyết định trình tự thực thi trong thời gian chạy, dựa trên trạng thái hiện tại của hệ thống.

## **3.2 Lựa chọn và tính toán tham số cho các task**

Với yêu cầu đề tài tiến hành chia thành 3 task như sau:

* Task 1: Đọc giá trị nhiệt độ từ DS18B20
* Task 2: Đọc giá trị độ ẩm từ Soil Moisture V1.2
* Task 3: Hiển thị thông số lên màn hình LCD và truyền nhận dữ liệu lên máy tính qua UART

### **3.2.1 Phân tích và lập trình theo mô hình thời gian**

Các task chạy tuần tự theo chu kỳ định sẵn, Task 3 (hiển thị và giao tiếp UART) yêu cầu ưu tiên cao hơn vì nó phải xử lý dữ liệu đầu ra thường xuyên.Các task có thể thêm một khoảng trễ nhỏ để giảm tải CPU.

* Task 1 (đọc nhiệt độ): Thời gian thực hiện ≈ 10ms.
* Task 2 (đọc độ ẩm): Thời gian thực hiện ≈ 10ms.
* Task 3 (hiển thị và giao tiếp UART): Thời gian thực hiện ≈ 50ms.
* Hệ điều hành thực hiện việc chuyển ngữ cảnh (context switch): Chi phí context switch ≈ 1ms.

Chu kỳ thực hiện của các task:

* Task 1 và Task 2: mỗi 1000 ms (1s).
* Task 3: mỗi 500ms (0.5s).

### **3.2.2 Phân tích và lập trình theo mô hình đa nhiệm**

* Ci​: Thời gian thực hiện của task thứ iii (execution time).
* Ti: Chu kỳ của task thứ iii (task period).
* n: Số lượng task.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Execution Time (Ci)** | **Period (Ti​)** | **Utilization (Ui)** |
| **Task 1**: Đọc nhiệt độ | 10ms | 1000ms | 0.01 |
| **Task 2**: Đọc độ ẩm | 10ms | 1000ms | 0.01 |
| **Task 3**: Hiển thị + UART | 50ms | 500ms | 0.1 |

U=U1​ + U2 ​+ U3​ = 0.01 + 0.01 + 0.1 = 0.12

Tổng mức sử dụng CPU là 12%. Điều này cho thấy hệ thống vẫn còn đủ khả năng để thực hiện các tác vụ khác hoặc xử lý ngắt.

Với n = 3: > 0.12 Hệ thống đạt yêu cầu về tính khả thi và ổn định.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Execution Time (Ci​)** | **Period (Ti​)** | **Priority** | **Deadline** |
| **Task 1**: Đọc nhiệt độ | 10ms | 1000ms | Thấp | 1000ms |
| **Task 2**: Đọc độ ẩm | 10ms | 1000ms | Thấp | 1000ms |
| **Task 3**: Hiển thị + UART | 50ms | 500ms | Cao | 500ms |

- Lập lịch theo non - preemptive, priority-based: thứ tự thực thi các task được quyết định dựa trên mức ưu tiên được định nghĩa trước.

Độ ưu tiên của task:

* Task 3: Hiển thị và UART, ưu tiên cao nhất (3).
* Task 1: Đọc nhiệt độ, ưu tiên trung bình (2).
* Task 2: Đọc độ ẩm, ưu tiên thấp nhất (1).

Quy tắc Non-Preemptive:

* Khi một task được CPU chọn, task đó sẽ hoàn thành trước khi chuyển sang task tiếp theo.
* CPU chọn task mới dựa trên mức ưu tiên khi task hiện tại hoàn thành.

- Lập lịch theo Round – robin: mỗi task sẽ có một thời gian thực thi nhất định, gọi là Time Slice, hết khoảng thời gian này thì sẽ phải nhường CPU cho task khác thực hiện. Cách thưc hiện dùng Timer để đếm thời gian Time Slide, sau đó sinh ra một ngắt để gọi từng Task thực thi.

A diagram of a project

Description automatically generated with medium confidence

Hình 15 Lập lịch theo Round - Robin

# **CHƯƠNG 4: ĐÁNH GIÁ KẾT QUẢ**

## **4.1 Kết quả đạt được**

**A close up of a circuit board

Description automatically generated**

Hình 16 Mạch thực nghiệm

A screenshot of a computer

Description automatically generated

Hình 17 Kết quả truyền nhận dữ liệu qua UART

## **4.2 Phân tích và đánh giá kết quả**

Thông qua quá trình làm bài tập lớn học phần “Thiết kế hệ thống nhúng” nhóm chúng em đã làm được các công việc:

- Lập trình STM32 đo nhiệt độ từ cảm biến DS18B20, đo độ ẩm từ cảm biến Soil Moisture V1.2 và truyền những thông số lên máy tính qua UART, tuy nhiên chưa hiển thị được lên LCD.

- Phân tích và lập trình các công việc trên trong hệ điều hành thời gian thực (RTOS) với các loại mô hình cụ thể là:

* Mô hình đơn nhiệm
* Hai mô hình đa nhiệm: chưa hoàn thành hết

## **4.3 Đánh giá mức độ đóng góp của các thành viên**

|  |  |  |
| --- | --- | --- |
| **Thành viên** | **Phụ trách công việc** | **Đánh giá mức độ đóng góp (thang 10)** |
| Nguyễn Văn Doanh  20212716 | Lập trình đa nhiệm, truyền nhận dữ liệu qua UART, giao tiếp với LCD, viết báo cáo | 9,5 |
| Nguyễn Việt Anh  20212690 | Lập trình đơn nhiệm, hiển thị lên LCD, giao tiếp với DS18B20 | 9 |
| Đặng Đình Hiển  20210322 | Lập trình đa nhiệm, giao tiếp cảm biến Soil Moisture, viết báo cáo | 9 |