**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**THÀNH PHỐ HỒ CHÍ MINH**

KHOA VIỄN THÔNG 2

****

**BÁO CÁO MÔN HỌC**

**THỰC TẬP CƠ SỞ**

NĂM HỌC 2024 - 2025

**ĐỀ TÀI: MÔ HÌNH ĐÈN CHIẾU SÁNG TỰ ĐỘNG**

**Giảng viên hướng dẫn** : ThS. Huỳnh Văn Hóa

**Sinh viên thực hiện: MSSV: Lớp:**

Nguyễn Chí Bảo N21DCVT007 D21CQVT01-N

Huỳnh Gia Bảo N21DCVT005 D21CQVT01-N

Nguyễn Trường Giang N21DCVT027 D21CQVT01-N

Nguyễn Hoàng Thắng N21DCVT097 D21CQVT01-N

Nguyễn Ngọc Trí N21DCVT107 D21CQVT01-N

Hoàng Lê Đức N21DCVT025 D21CQVT01-N

***Thành phố Hồ Chí Minh, 2024***

# **LỜI CẢM ƠN**

Để thực hiện và hoàn thành tốt báo cáo này, tôi chân thành cảm ơn các thành viên trong nhóm đã cống hiến thời gian nghiên cứu, kiến thức và nỗ lực của mình để hoàn thành bài báo cáo này. Sự lực không ngừng nghỉ đã giúp cho đề tài này hoàn thành tốt đẹp. Đặc biệt, tôi chân thành cảm ơn thầy Huỳnh Văn Hóa đã dành thời gian và kiến thức của mình để hỗ trợ trong quá trình thực hiện báo cáo này.

Một lần nữa, tôi xin chân thành cảm ơn sự đóng góp và sự hỗ trợ của tất cả mọi người. Chúng ta đã làm việc với nhau với tinh thần đoàn kết và sự chia sẻ, và kết quả cuối cùng không chỉ là thành quả cá nhân mà còn là sản phẩm của sự hợp tác của cả nhóm.

Chúng tôi xin chân thành cảm ơn!

**MỤC LỤC**

[**LỜI CẢM ƠN 2**](#_i8wvz366w1jg)

[**CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI 5**](#_wsmdb2sctok3)

[**“MÔ HÌNH ĐÈN CHIẾU SÁNG 5**](#_66bfbrplxbv6)

[**TỰ ĐỘNG” 5**](#_5ighprd2absk)

[1.1 Giới thiệu đề tài “Mô hình đèn chiếu sáng tự động” 5](#_94shxn61kkvo)

[1.2 Lý do chọn đề tài 5](#_dop794ir8kkk)

[1.3 Mục tiêu đề tài 6](#_wjwxqhqo253y)

[**­­CHƯƠNG 2. CÁC LINH KIỆN 6**](#_85wwnb44fzwr)

[2.1 Tìm hiểu về vi điều khiển STM32F401CDU6 7](#_v4k87o91x0js)

[2.1.1 Giới thiệu về vi điều khiển STM32F401CDU6 7](#_uiflm4p7y14a)

[2.1.2 Cấu tạo, thông số kỹ thuật 7](#_l7lbwt2o8ejc)

[2.1.3 Cấu tạo và nguyên lý hoạt động 8](#_khxslaa9dryc)

[2.1.4 Ngoại vi GPIOs 9](#_c4edr5olg51p)

[2.1.4 Bộ chuyển đổi tương tự sang số (ADC) 11](#_oatlz3z2847)

[2.1.5 Giao diện ngoại vi nối tiếp (SPI) 12](#_65adsoeqj56c)

[2.1.6 Memory Mapping 12](#_jtr611ox8l55)

[Các vùng bộ nhớ 13](#_98gf1snq1g16)

[Các bus ngoại vi 14](#_zh5pk6d6cby0)

[2.1.7 Ưu và nhược điểm khi dùng vi điều khiển STM32F401CDU6 cho hệ thống 15](#_cu8uhm504yv7)

[2.2 Màn hình LCD TFT Round 1.28 15](#_unfkn22pj3)

[2.2.1 Giới thiệu về màn hình LCD TFT IPS 1.28 Inch Round GC9A01 SPI Interface 15](#_koiartb32yiz)

[2.2.2 Thông số kỹ thuật 16](#_iyaa2fwavkk7)

[2.2.3 Kết nối giữa màn hình LCD với vi điều khiển 16](#_wkzojf9z7q9g)

[Các tín hiệu điều khiển: 17](#_dew9l6ugtkpc)

[2.3 Tìm hiểu về quang trở GL5516. 18](#_p999y17zxflc)

[2.3.1 Giới thiệu về Quang trở GL5516 18](#_cyyf5m3ywjyf)

[2.3.2 Thông số kỹ thuật 19](#_1a6at34vtxor)

[2.3.3 Cấu tạo và nguyên lý hoạt động 19](#_ehiy26n9ye37)

[2.3.4 Ưu và nhược điểm khi dùng quang trở cho hệ thống 20](#_unzdmjlwth73)

[2.3.5 Giao tiếp giữa quang trở GL5516 với STM32F401CDU6 20](#_9f1e9xrv9651)

[2.4 Tìm hiểu về cảm biến siêu âm HC-SR04 21](#_u8pwg79na0eo)

[2.4.1 Giới thiệu về cảm biến siêu âm HC-SR04 21](#_y00ovv2bv4zi)

[2.4.2 Thông số kỹ thuật 21](#_ae51k8blk3k6)

[2.4.3 Cấu tạo và nguyên lý hoạt động 22](#_lgg5n9rsn8xu)

[2.4.4 Giao tiếp giữa cảm biến siêu âm HC-SR04 với STM32F401CDU6 22](#_q437cax3k695)

[2.4.5 Ưu và nhược điểm 23](#_4u8iits47n3l)

[**CHƯƠNG 3. NGUYÊN LÝ HOẠT ĐỘNG CỦA MÔ HÌNH 23**](#_ohg1r6bmn7r2)

[3.1 Lưu đồ thuật toán 24](#_elscy4k9ulvn)

[3.1.1 Khởi tạo các biến có kiểu cấu trúc dành cho các chức năng riêng 25](#_br6ifuqdct8)

[3.1.2 Khối lấy thông tin cảm biến siêu âm HRSR04 25](#_foao2gq32l8w)

[3.1.3 Khối có nhiệm vụ xác định có người hay không 26](#_rlq43ij4xhia)

[3.1.4 Khối điều khiển đèn dựa trên biến human 28](#_whoa6pks3hyj)

[3.1.5.Khối đọc giá trị ánh sáng lux 30](#_3k1fma6erd0r)

[3.1.6 Hàm tiện ích 30](#_fbd2ouw3z0d)

[3.1.7. Hàm cấu hình và khởi tạo 30](#_km05m5d7thwh)

[3.1.8 Hàm xử lý lỗi 35](#_3g76ye9ehxdv)

[3.1.9 Các hàm HAL được ghi đè 35](#_5mpkl01ud2nx)

[3.2 Sơ đồ nguyên lý của mô hình đèn chiếu sáng tự động 36](#_2sb2dpktjmfb)

[**CHƯƠNG 4. KẾT QUẢ ĐỀ TÀI 38**](#_rp57easavcf6)

[4.1 Các tính năng và lợi ích nổi bật 39](#_rftcy7ryum61)

[4.2 Một số hạn chế 39](#_5w1m2fqmg6eh)

[**CHƯƠNG 5. ỨNG DỤNG VÀ PHÁT TRIỂN 40**](#_la232vvou5c7)

[5.1 Ứng dụng của đề tài “Mô hình đèn chiếu sáng tự động” 40](#_np5495svti0m)

[5.2 Hướng phát triển đề tài 40](#_acihamwd64w3)

[5.3 Kết luận 40](#_euz701anw9op)

[**PHỤ LỤC 41**](#_ehxdpyhzp4wm)

[PHỤ LỤC 1: Code của bài báo cáo 41](#_70qy0ii9pohq)

[PHỤ LỤC 2: Bảng thống kê đề tài và giá thành của đề tài 41](#_xnyb6f31wuqg)

[**TÀI LIỆU THAM KHẢO 42**](#_gz3ko71u31rh)

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI**

# **“MÔ HÌNH ĐÈN CHIẾU SÁNG**

# **TỰ ĐỘNG”**

# 

## **1.1 Giới thiệu đề tài “Mô hình đèn chiếu sáng tự động”**

Trong thời đại công nghệ tiên tiến, nhu cầu về chiếu sáng hiệu quả và tiết kiệm năng lượng ngày càng tăng cao. Mô hình đèn chiếu sáng tự động cho nhà ở là một giải pháp thông minh nhằm nâng cao tiện ích và tiết kiệm năng lượng trong sinh hoạt gia đình. Hệ thống này sử dụng các cảm biến chuyển động và ánh sáng để tự động điều chỉnh đèn theo nhu cầu thực tế của người sử dụng. Khi có người di chuyển vào khu vực được giám sát, đèn sẽ tự động bật và duy trì chiếu sáng cho đến khi không còn chuyển động trong khoảng thời gian cài đặt trước, sau đó sẽ tắt. Ngoài ra, hệ thống còn có thể điều chỉnh độ sáng tùy theo cường độ ánh sáng tự nhiên, đảm bảo không gian luôn được chiếu sáng hợp lý mà không lãng phí điện năng. Mô hình này không chỉ mang lại sự tiện nghi, an toàn cho ngôi nhà mà còn góp phần bảo vệ môi trường thông qua việc tiết kiệm năng lượng.

Mô hình đèn chiếu sáng tự động sử dụng vi xử lý STM32F401CDU6 với các cảm biến âm thanh để phát hiện người sử dụng cùng với cảm biến ánh sáng để điều khiển đèn sáng giúp tiết kiệm năng lượng giảm thiểu chi phí cho người dùng.

## **1.2 Lý do chọn đề tài**

**a. Tính độc đáo và chuyên biệt:**

Việc ứng dụng đèn chiếu sáng tự động cho nhà ở chưa được phổ biến rộng rãi, tạo nên sự độc đáo và tiềm năng khai thác mới cho đề tài.

**b. Tiết kiệm chi phí:**

Giảm thiểu nhu cầu sử dụng đèn chiếu sáng thủ công và chi phí vận hành liên quan.

Nâng cao hiệu quả sử dụng năng lượng, góp phần tiết kiệm chi phí điện năng.

**c. Góp phần bảo vệ môi trường:**

Sử dụng đèn chiếu sáng LED tiết kiệm năng lượng giúp giảm thiểu khí thải nhà kính và bảo vệ môi trường.

**Kết luận:**

Với những lý do trên, đề tài đèn chiếu sáng tự động là một lĩnh vực nghiên cứu khoa học và ứng dụng thực tiễn có giá trị cao, hứa hẹn mang lại nhiều lợi ích cho xã hội và góp phần bảo vệ môi trường.

## **1.3 Mục tiêu đề tài**

Mô hình đèn chiếu sáng tự động đề ra mục tiêu hoạt động dựa trên ánh sáng môi trường và kèm theo khả năng nhận biết con người hoạt động thông qua cảm biến để điều khiển đèn bật tắt tự động, hiển thị thông tin độ sáng lên màn hình LCD để người dùng theo dõi,kiểm soát, giúp tiết kiệm năng lương và giảm thiểu chi phí cho người dùng.

# 

# 

# 

# 

# 

# 

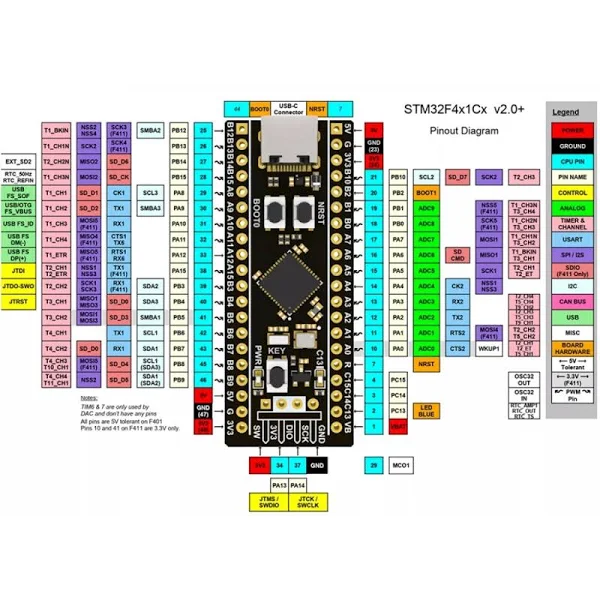
# 

# 

# **­­CHƯƠNG 2. CÁC LINH KIỆN**

## **2.1 Tìm hiểu về vi điều khiển STM32F401CDU6**

### **2.1.1 Giới thiệu về vi điều khiển STM32F401CDU6**



Hình 1.1 Vi điều khiển STM32F401CCU6

Vi điều khiển STM32F401CDU6 là một trong những dòng vi điều khiển 32-bit ARM Cortex-M4 được sản xuất bởi STMicroelectronics. Nó là một phần của dòng sản phẩm STM32F4, được thiết kế để cung cấp hiệu suất cao và tiêu thụ năng lượng thấp cho một loạt các ứng dụng nhúng.

### **2.1.2 Cấu tạo, thông số kỹ thuật**

* Lõi xử lý: ARM Cortex-M4F 32-bit, tốc độ xung nhịp lên đến 84 MHz.Dòng lệnh có thể thực hiện mỗi chu kỳ xung nhịp.
* Bộ nhớ: Flash: Lên đến 512 KB. RAM lên đến 96 KB.
* Nhiệt độ hoạt động: -40°C đến +85°C.
* Giao tiếp: UART, SPI, I2C, USB, CAN
* Ngõ vào ra: lên đến 80 chân GPIO.
* Bộ tính toán số: Bộ ADC, bộ DAC.
* Nguồn năng lượng: 1,7V đến 3,6V. Tiêu thụ năng lượng thấp.
* Bảo mật và an ninh: Các tính năng bảo mật phần cứng như phần cứng mã hóa, kiểm soát truy cập.

### **2.1.3 Cấu tạo và nguyên lý hoạt động**

**Cấu tạo:**

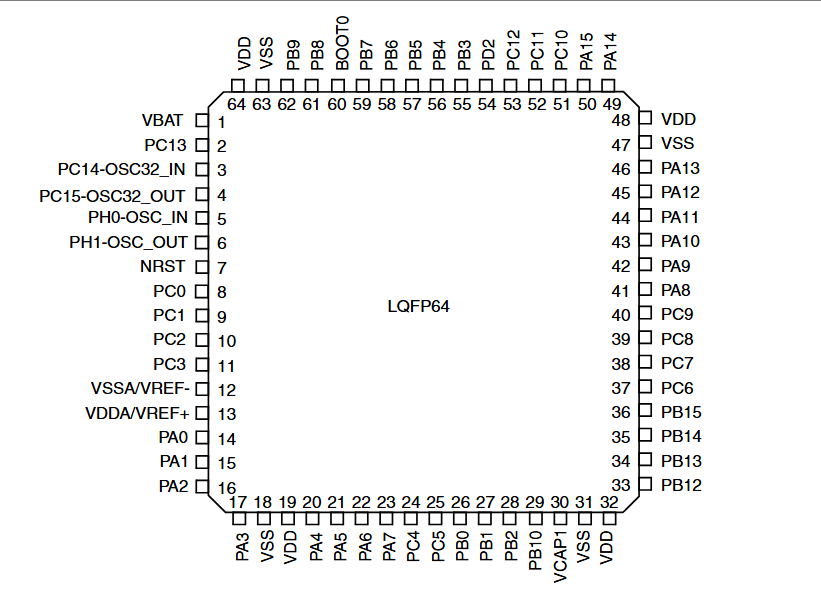
* STM32F401CDU6 sử dụng lõi ARM Cortex-M4F, đây là một lõi xử lý 32-bit có khả năng xử lý số, hỗ trợ các phép toán đơn giản và phức tạp.
* Bộ nhớ Flash tích hợp trong vi điều khiển cho phép lập trình chương trình ứng dụng. Ngoài ra, nó còn có bộ nhớ RAM để lưu trữ dữ liệu tạm thời.
* Chức năng I/O: STM32F401CDU6 có nhiều chân I/O (Input/Output) cho phép nó tương tác với các thiết bị ngoại vi như cảm biến, bộ điều khiển, và các thiết bị khác.

**Nguyên lý hoạt động:**

* Chu trình thực thi: Chương trình được lưu trữ trong bộ nhớ Flash được sao chép vào bộ nhớ RAM và được thực thi bởi lõi xử lý.
* Giao tiếp với các thiết bị ngoại vi: STM32F401CDU6 có nhiều giao tiếp ngoại vi như SPI, I2C, UART, ADC, PWM, để tương tác với các thiết bị ngoại vi và cảm biến.
* Ngắt (Interrupts): Vi điều khiển có khả năng phát hiện và xử lý ngắt từ các nguồn khác nhau như timer, GPIO, UART, I2C, SPI, v.v. để xử lý các sự kiện ngoại vi mà không cần phải kiểm tra liên tục.
* Clock: STM32F401CDU6 có bộ điều khiển Clock nội bộ cho phép lập trình chính xác tần số hoạt động của vi điều khiển.

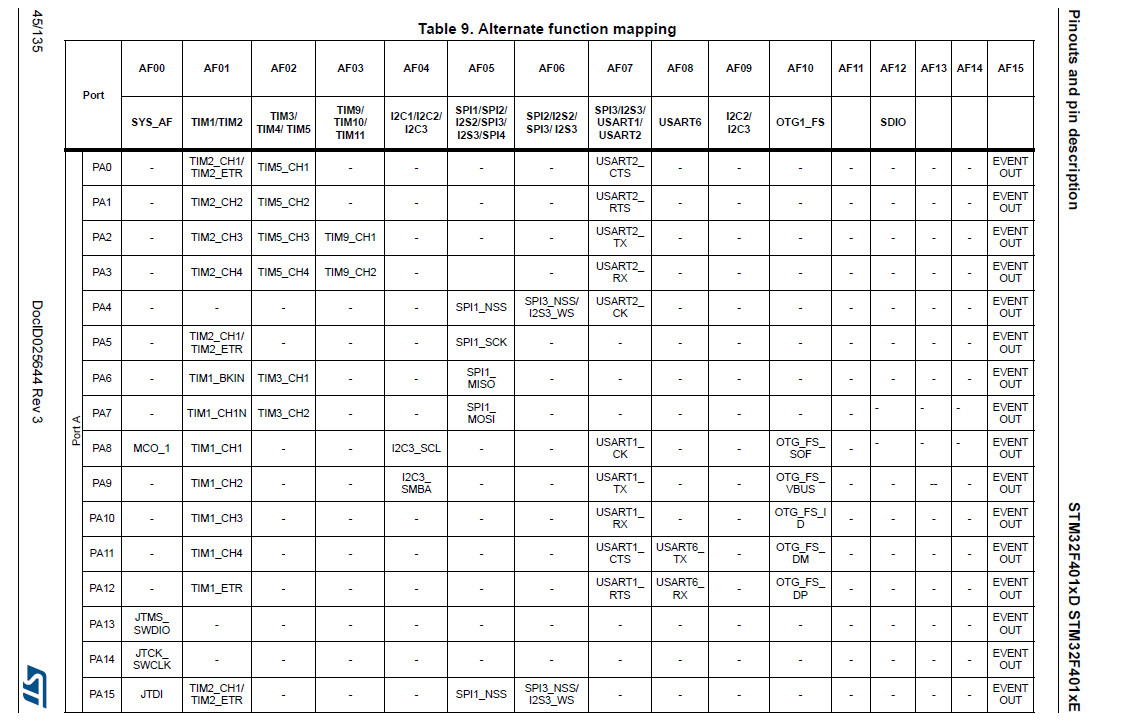
### **2.1.4 Ngoại vi GPIOs**

**Đặc điểm chính của GPIO trên STM32F401CDU6:** Mỗi chân GPIO có thể được cấu hình bằng phần mềm làm đầu ra (kéo đẩy hoặc thoát nước mở, có hoặc không kéo lên hoặc kéo xuống), làm đầu vào (nổi, có hoặc không kéo lên hoặc kéo xuống) hoặc như chức năng thay thế ngoại vi. Hầu hết các chân GPIO được chia sẻ với các chức năng thay thế kỹ thuật số hoặc tương tự. Tất cả GPIO đều có khả năng dòng điện cao và có lựa chọn tốc độ để quản lý tốt hơn tiếng ồn bên trong, mức tiêu thụ điện năng và phát xạ điện từ. Cấu hình I/O có thể bị khóa nếu cần bằng cách tuân theo một trình tự cụ thể để tránh việc ghi giả vào các thanh ghi I/O. Xử lý I/O nhanh cho phép chuyển đổi I/O tối đa lên tới 84 MHz.

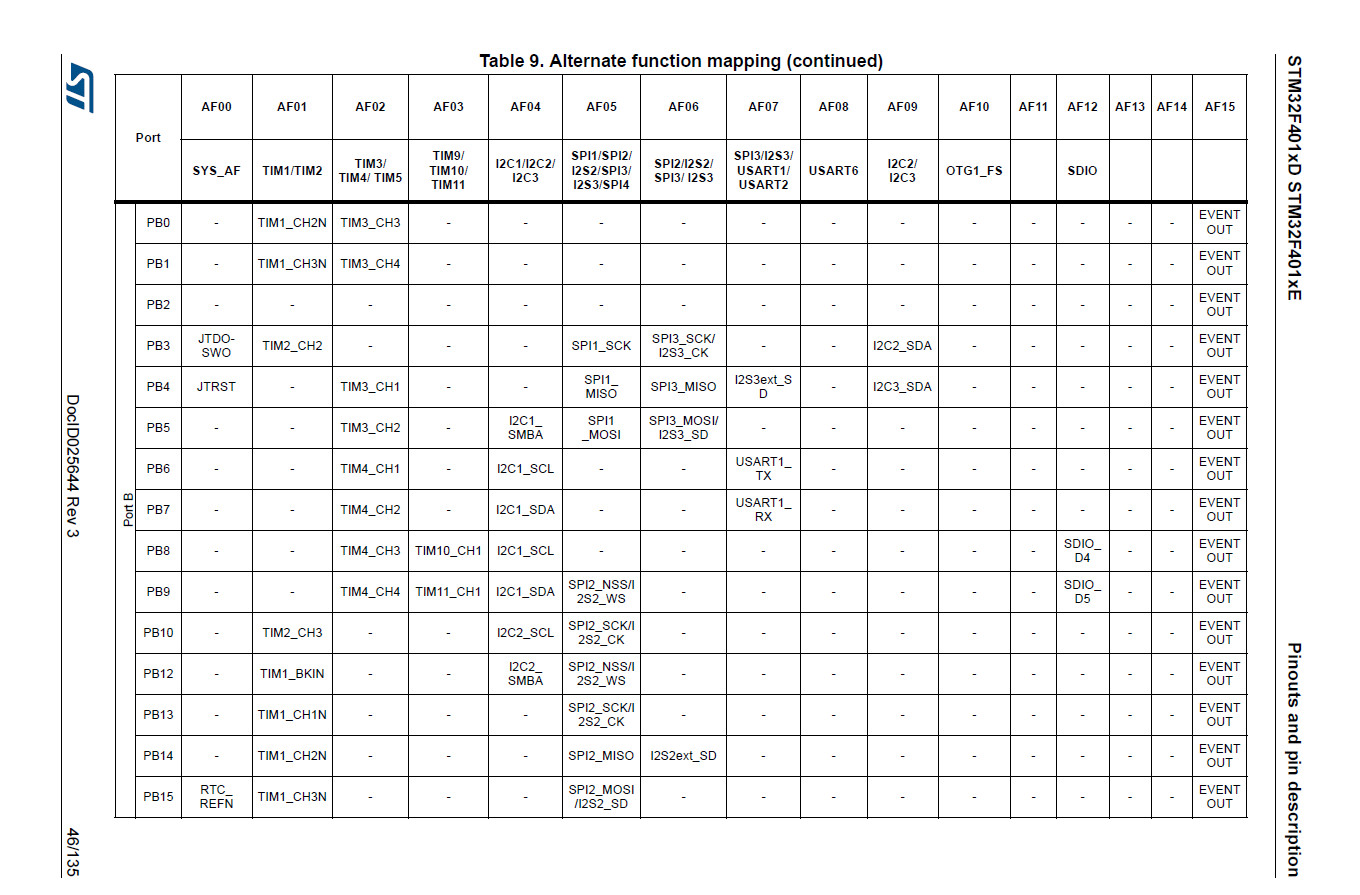


Hình 1.2 GPIOs VXL STM32F401CDU6

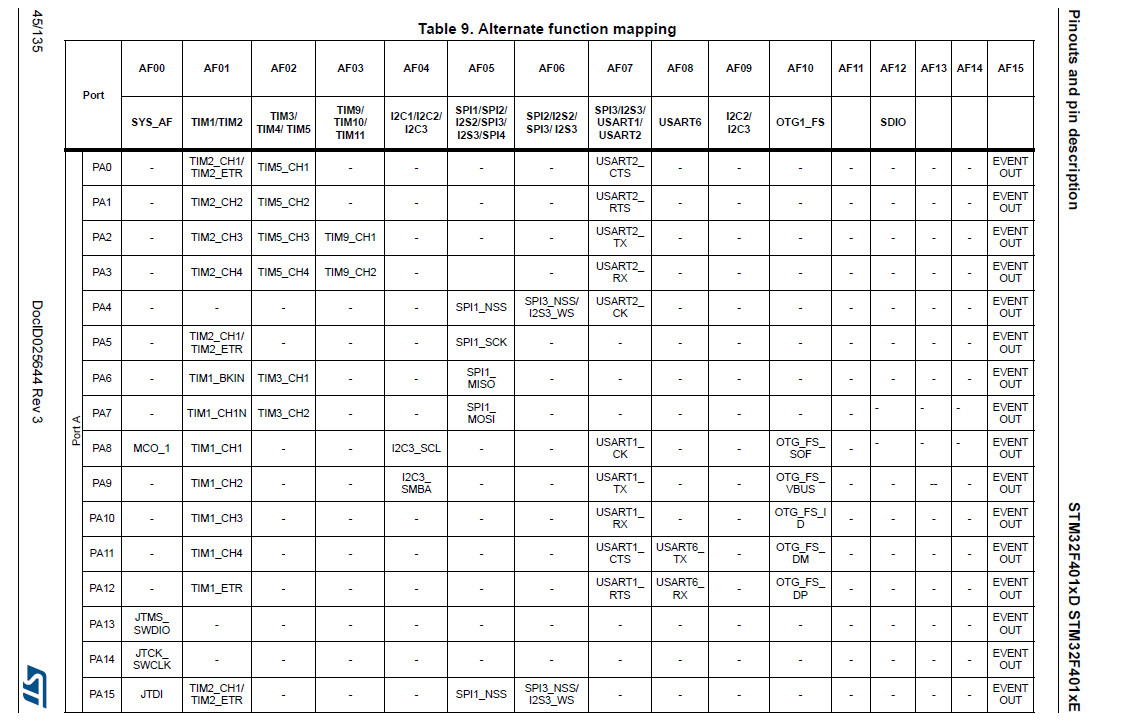
**Ánh xạ chức năng GPIOA**



**Ánh xạ chức năng GPIOB**



**Ánh xạ chức năng GPIOC**

****

### 

### 

### **2.1.4 Bộ chuyển đổi tương tự sang số (ADC)**

Một bộ chuyển đổi tương tự sang kỹ thuật số 12 bit được nhúng và chia sẻ tối đa 16 kênh bên ngoài, thực hiện chuyển đổi ở chế độ chụp một lần hoặc chế độ quét. Ở chế độ quét, việc chuyển đổi tự động được thực hiện trên một nhóm đầu vào analog đã chọn.

ADC có thể được phục vụ bởi bộ điều khiển DMA. Tính năng giám sát tương tự cho phép giám sát rất chính xác điện áp chuyển đổi của một, một số hoặc tất cả các kênh đã chọn. Một ngắt được tạo ra khi điện áp chuyển đổi nằm ngoài ngưỡng được lập trình. Để đồng bộ hóa bộ định thời và chuyển đổi A/D, ADC có thể được kích hoạt bởi bất kỳ bộ định thời TIM1, TIM2, TIM3, TIM4 hoặc TIM5 nào.

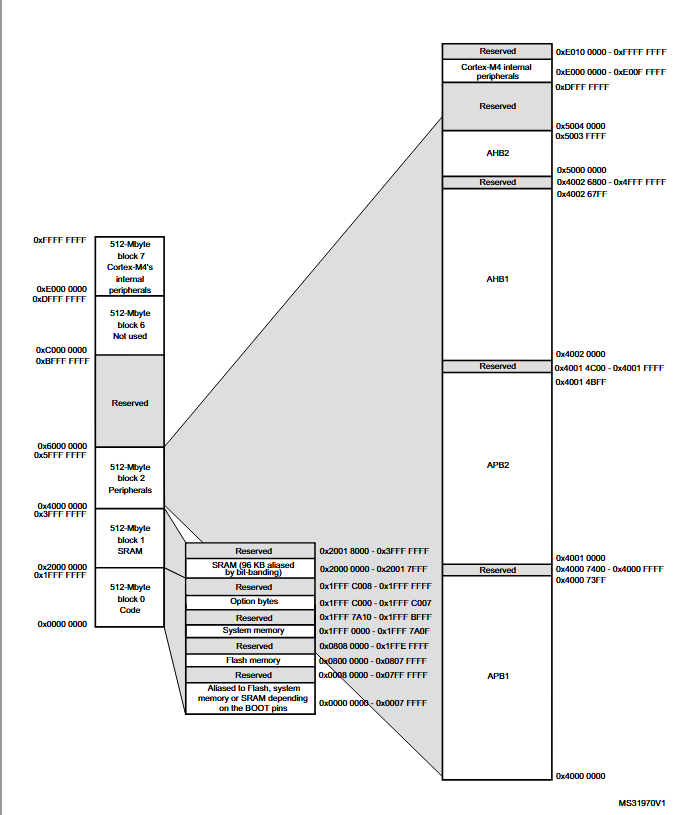
### **2.1.5 Giao diện ngoại vi nối tiếp (SPI)**

Các thiết bị có tối đa bốn SPI ở chế độ phụ và chế độ chính ở chế độ giao tiếp song công hoàn toàn và đơn giản. SPI1 và SPI4 có thể giao tiếp với tốc độ lên tới 42 Mbit/s, SPI2 và SPI3 có thể giao tiếp với tốc độ lên tới 21 Mbit/s. Bộ đếm gộp 3 bit cung cấp 8 tần số chế độ chính và khung có thể được cấu hình thành 8 bit hoặc 16 bit. Việc tạo/xác minh CRC phần cứng hỗ trợ các chế độ Thẻ SD/MMC cơ bản. Tất cả SPI có thể được phục vụ bởi bộ điều khiển DMA. Giao diện SPI có thể được cấu hình để hoạt động ở chế độ TI để liên lạc ở chế độ chính và chế độ phụ.

### **2.1.6 Memory Mapping**

Memory mapping là một sơ đồ mô tả cách bộ nhớ được phân chia và sử dụng bởi vi xử lý. Nó cho biết vị trí của các thành phần hệ thống khác nhau, chẳng hạn như bộ nhớ flash, bộ nhớ SRAM và các ngoại vi. Memory mapping rất quan trọng đối với lập trình viên vì nó cho phép họ truy cập và điều khiển các thành phần hệ thống này.

Sơ đồ phân phối bộ nhớ của STM32F401CDU6



Sơ đồ phân bố bộ nhớ của vi điều khiển STM32F401CDU6 được chia thành nhiều vùng, mỗi vùng phục vụ một mục đích cụ thể. Các vùng này được phân loại thành Code (mã chương trình), SRAM (bộ nhớ tĩnh), Peripheral (ngoại vi), External RAM (bộ nhớ ngoài), External Devices (thiết bị ngoài), và System memory (bộ nhớ hệ thống). Địa chỉ được ghi dưới dạng hệ thập lục phân (hexadecimal).

### **Các vùng bộ nhớ**

1. 0x0000 0000 - 0x1FFF FFFF: Code (khối 512-Mbyte)
   * 0x0000 0000 - 0x0003 FFFF: Bộ nhớ Flash, nơi lưu trữ chương trình của người dùng.
   * 0x1FFF 0000 - 0x1FFF 77FF: Bộ nhớ hệ thống (dùng cho bootloader).
2. 0x2000 0000 - 0x3FFF FFFF: SRAM (khối 512-Mbyte)
   * 0x2000 0000 - 0x2001 7FFF: SRAM1 (96 KB)
   * 0x2001 8000 - 0x2001 FFFF: Vùng ánh xạ của SRAM1, thường dùng cho bit-banding.
3. 0x4000 0000 - 0x5FFF FFFF: Peripheral (khối 512-Mbyte)
   * 0x4002 1000 - 0x4002 13FF: Các thanh ghi GPIO, TIM, ADC, USART và các thanh ghi ngoại vi khác.
4. 0x6000 0000 - 0x9FFF FFFF: External RAM (khối 512-Mbyte, không sử dụng)
   * Dành riêng cho mở rộng bộ nhớ ngoài, không sử dụng trong STM32F401CDU6.
5. 0xA000 0000 - 0xDFFF FFFF: External Device (khối 512-Mbyte, không sử dụng)
   * Dành riêng cho các thiết bị ngoại vi, không sử dụng trong STM32F401CDU6.
6. 0xE000 0000 - 0xFFFF FFFF: System memory
   * 0xE000 0000 - 0xE00E 0FFF: Các ngoại vi nội bộ của Cortex-M4 như NVIC, SCB, SysTick.
   * 0xE010 0000 - 0xFFFF FFFF: Dành riêng cho sử dụng trong tương lai hoặc các ngoại vi đặc thù của nhà sản xuất.

### **Các bus ngoại vi**

* AHB1 (Advanced High-performance Bus 1)
  + 0x4002 0000 - 0x4002 1FFF: Các cổng GPIO.
  + 0x4002 2000 - 0x4002 23FF: RCC (Reset and Clock Control).
  + 0x4002 4000 - 0x4002 4BFF: Các thanh ghi giao diện Flash.
* APB1 (Advanced Peripheral Bus 1)
  + 0x4000 0000 - 0x4000 73FF: Timer, USART, I2C và các ngoại vi tốc độ thấp khác.
* APB2 (Advanced Peripheral Bus 2)
  + 0x4001 0000 - 0x4001 4BFF: Timer, ADC, SPI và các ngoại vi tốc độ cao khác.

### **2.1.7 Ưu và nhược điểm khi dùng vi điều khiển STM32F401CDU6 cho hệ thống**

**Ưu điểm:**

* Hiệu suất cao.
* Khả năng tích hợp nhiều giao tiếp.
* Tiêu thụ năng lượng thấp.
* Dễ dàng phát triển.

**Nhược điểm:**

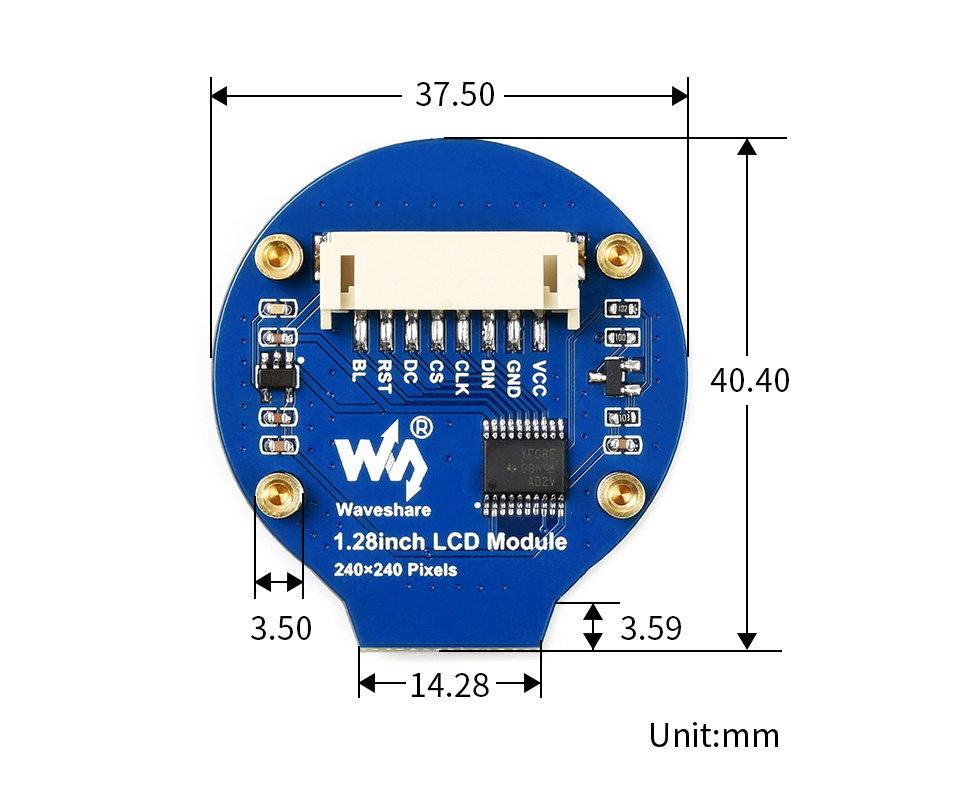
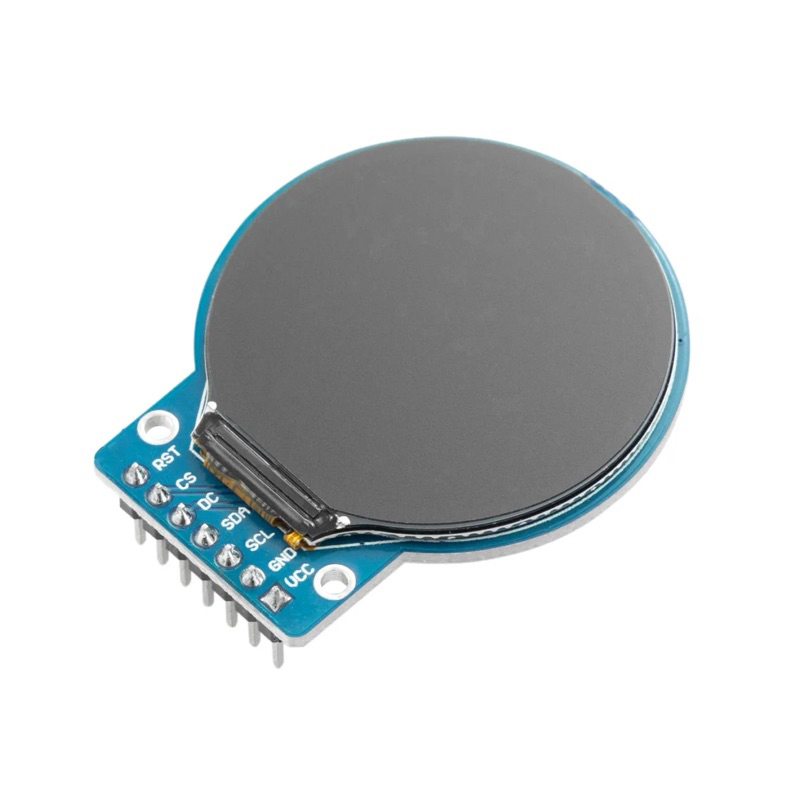
* Giá thành cao so với hiệu suất và tính năng của nó.
* Yêu cầu kiến thức kỹ thuật: Để tận dụng hết tiềm năng của STM32F401CDU6, người lập trình cần có kiến thức kỹ thuật vững vàng về vi điều khiển nhúng, lập trình nhúng, và các giao tiếp ngoại vi.

## **2.2 Màn hình LCD TFT Round 1.28**

### **2.2.1 Giới thiệu về màn hình LCD TFT IPS 1.28 Inch Round GC9A01 SPI Interface**

Màn hình LCD TFT IPS 1.28 Inch Round GC9A01 SPI Interface là một loại màn hình hiển thị TFT (Thin Film Transistor) có kích thước 1.28 inch, được thiết kế với công nghệ IPS (In-Plane Switching) để cung cấp góc nhìn rộng và màu sắc chất lượng. Giao diện SPI (Serial Peripheral Interface) cho phép việc truyền dữ liệu giữa màn hình và vi điều khiển thông qua các tín hiệu truyền dẫn đồng bộ.

Màn hình này thường được sử dụng trong các ứng dụng như đồng hồ thông minh, thiết bị đeo thông minh, hoặc các dự án điện tử khác đòi hỏi màn hình nhỏ gọn và hiển thị màu sắc đẹp. Với kích thước nhỏ gọn và góc nhìn rộng, nó có thể được tích hợp vào các thiết bị di động mà không tốn nhiều không gian.



Hình 1.3 Màn hình LCD TFT Round 1.28

Màn hình sử dụng driver GC9A01, một loại driver phổ biến trong các màn hình TFT, để điều khiển hiển thị và tương tác với dữ liệu đầu vào từ vi điều khiển. Giao diện SPI cung cấp tốc độ truyền dữ liệu nhanh và linh hoạt, làm cho nó trở thành lựa chọn phổ biến cho việc kết nối với các vi điều khiển nhúng và các hệ thống nhúng khác.

### **2.2.2 Thông số kỹ thuật**

* Model: LCD TFT IPS 1.28 Inch Round GGC9A01 SPI Interface.
* Điện áp sử dụng: 3.3~5VDC.
* Điện áp giao tiếp: TTL 3.3~5VDC.
* IC Driver hiển thị: GC9A01 giao tiếp SPI.
* Cỡ màn hình: 1.28 inch tròn.
* Độ phân giải: 240 x 240 pixels.
* Đường kính màn hình: 32.4mm.

### **2.2.3 Kết nối giữa màn hình LCD với vi điều khiển**

Màn hình LCD được kết nối với vi điều khiển thông qua giao thức SPI

**Giao thức SPI (Serial Peripheral Interface)**:

SPI là giao thức nối tiếp được sử dụng rộng rãi trong các hệ thống nhúng, bao gồm kết nối giữa vi điều khiển và màn hình LCD. SPI có các đặc điểm chính sau:

* Chế độ Master-Slave: Một thiết bị Master điều khiển một hoặc nhiều thiết bị Slave.
* Đồng bộ hóa: Sử dụng tín hiệu clock chung (SCK).
* Tốc độ cao: Truyền dữ liệu với tốc độ cao, phù hợp với các ứng dụng yêu cầu băng thông lớn.
* Dây dẫn: Sử dụng 4 dây chính:
  + MOSI (Master Out Slave In): Dữ liệu từ Master đến Slave.
  + MISO (Master In Slave Out): Dữ liệu từ Slave đến Master.
  + SCK (Serial Clock): Tín hiệu clock do Master cung cấp.
  + SS (Slave Select): Chọn thiết bị Slave.

**Kết nối vật lý giữa LCD với vi điều khiển thông qua giao thức SPI:**

1. SCK (Serial Clock):
   * Tạo xung clock để đồng bộ dữ liệu giữa master (vi điều khiển STM32F401CDU6) và slave (màn hình LCD).
   * Trong sơ đồ này, SCK được kết nối với chân A4 của STM32.
2. MOSI (Master Out Slave In):
   * Dữ liệu được gửi từ master tới slave.
   * Trong sơ đồ này, MOSI tương ứng với SDA (Serial Data) của LCD và được kết nối với chân A5 của STM32.
3. MISO (Master In Slave Out):
   * Dữ liệu được gửi từ slave về master.
   * Không được sử dụng trong sơ đồ này vì màn hình LCD chỉ nhận dữ liệu từ vi điều khiển.
4. SS (Slave Select):
   * Chọn thiết bị slave để giao tiếp.
   * Trong sơ đồ này, SS tương ứng với CS (Chip Select) của LCD và được kết nối với chân A2 của STM32.

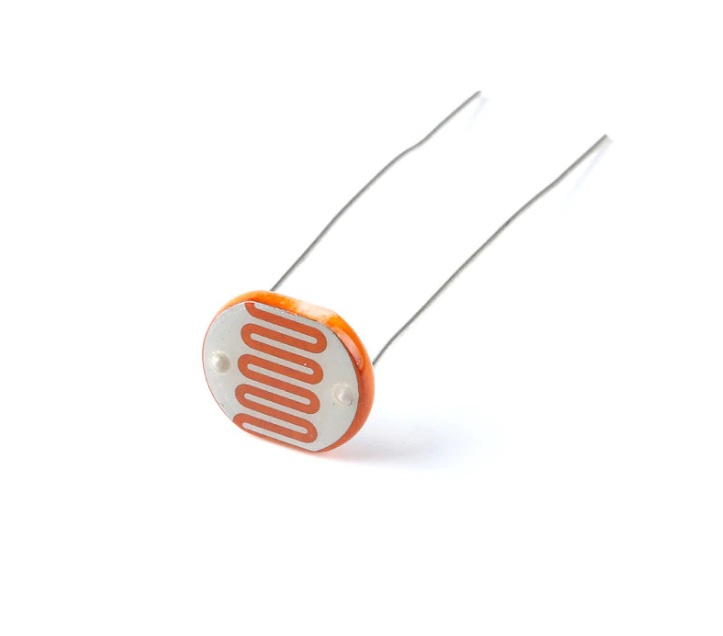
### **Các tín hiệu điều khiển:**

* DC (Data/Command):
  + Chân này được sử dụng để phân biệt dữ liệu và lệnh điều khiển được gửi đến màn hình.
  + Được kết nối với chân A3 của STM32.
* RST (Reset):
  + Chân này được sử dụng để khởi động lại màn hình.
  + Được kết nối với chân A1 của STM32.
* GND (Ground):
  + Kết nối đất chung giữa màn hình LCD và vi điều khiển.
  + Được kết nối với chân GND của STM32.
* VCC (Nguồn điện):
  + Cung cấp nguồn điện cho màn hình LCD.
  + Được kết nối với chân 5V của STM32.

## **2.3 Tìm hiểu về quang trở GL5516.**

### **2.3.1 Giới thiệu về Quang trở GL5516**

Quang trở GL5516 là một linh kiện điện tử có điện trở thay đổi giảm theo ánh sáng chiếu vào. Quang trở được dùng làm cảm biến nhạy sáng trong các mạch dò, như trong mạch đóng ngắt đèn chiếu bằng kích hoạt của sáng tối.



Hình 1.3 GL5516

### **2.3.2 Thông số kỹ thuật**

* Điện áp tối đa: 150 Volt.
* Công suất (mW): 100mW.
* Nhiệt độ môi trường: -30°C đến 70°C.
* Đỉnh phổ (nm): 560.
* Trở kháng khi có ánh sáng (10 lux): 5-10 .
* Trở kháng khi tối: 0.8 M.
* Thời gian đáp ứng: 30ms.
* Các đặc tính kháng illumination: 2.

### **2.3.3 Cấu tạo và nguyên lý hoạt động**

**Cấu tạo:** Quang trở được làm bằng chất bán dẫn có trở kháng cao và không có mặt tiếp giáp nào. Các điện trở quang thường có điện trở vài M trong bóng tối. Ngoài ra, giá trị điện trở có thể có được hạ từ 1 đến vài khi thắp sáng.

**Nguyên lý hoạt động:** dựa trên nguyên lý của hiệu ứng quang điện trong một khối vật chất. Khi một photon va chạm với nguồn năng lượng đủ lớn, electron bật ra khỏi phân tử và trở thành những electron tự do ở dạng khối và chất bán dẫn và các chất bán dẫn này sẽ chuyển thành các chất dẫn điện. Độ dẫn của điện trở quang phụ thuộc phần lớn vào các photon được hấp thụ. Khi ánh sáng chiếu vào, các electron sẽ được giải phóng và độ dẫn điện của nó sẽ được tăng lên. Tùy thuộc vào chất bán dẫn mà các điện trở quang sẽ có những phản ứng khác nhau với từng loại sóng photon khác nhau.

### **2.3.4 Ưu và nhược điểm khi dùng quang trở cho hệ thống**

**Ưu điểm:**

* Giá thành hợp lý.
* Kích thước nhỏ gọn.
* Dễ dàng tích hợp vào nhiều loại mạch.
* Có độ chính xác cao trong các ứng dụng đo lường và điều khiển ánh sáng.

**Nhược điểm:**

* Có độ trễ lớn gây ảnh hưởng đến việc gửi dữ liệu vào vi điều khiển, điều này không phù hợp cho các ứng dụng cần phản hồi tức thì.
* Ngoài ra, quang trở đòi hỏi nhiệt độ phải ở mức ổn định để có thể đưa ra được kết quả chính xác.

### **2.3.5 Giao tiếp giữa quang trở GL5516 với STM32F401CDU6**

Kết nối vật lý

**Các bước giao tiếp:**

* Cấu hình nguồn clock cho MCU.
* Khởi tạo GPIO dùng cho ADC.
* **Cấu hình ADC:**

§ Prescaler chia 4.

§ Độ phân giải 12 bit.

§ Chỉ đọc từ 1 kênh cấu hình sẵn.

§ Thực hiện một chuyển đổi duy nhất mỗi khi được kích hoạt.

§ ADC\_EXTERNALTRIGCONVEDGE\_NONE: có nghĩa là không có kích hoạt nào được sử dụng.

§ Một chuyển đổi được thực hiện mỗi khi ADC được kích hoạt.

§ Không liên tục yêu cầu DMA và chỉ được thực hiện cho mỗi chuyển đổi.

**Các bước hoạt động của ADC:**

* Lấy mẫu.
* Giữ mẫu.
* Chuyển đổi analog sang digital.
* Đọc và xử lý tín hiệu.

## **2.4 Tìm hiểu về cảm biến siêu âm HC-SR04**

### **2.4.1 Giới thiệu về cảm biến siêu âm HC-SR04**

Cảm biến siêu âm HC-SR04 là một thiết bị được sử dụng rộng rãi trong lĩnh vực điện tử và tự động hóa để đo khoảng cách đến một vật thể bằng sóng siêu âm.



Hình 1.4 Cảm biến siêu âm HC-SR04

### **2.4.2 Thông số kỹ thuật**

* **Điện áp hoạt động:** 5V DC.
* **Dòng tiêu thụ:** 15 mA.
* **Phạm vi đo:** 2 cm đến 400 cm.
* **Độ chính xác:** 3 mm.
* **Góc phát hiện:** 15 độ.
* **Tần số hoạt động:** 40 kHz.
* **Kích thước:** 45 mm x 20 mm x 15 mm.

### **2.4.3 Cấu tạo và nguyên lý hoạt động**

**Cấu tạo:**

* Bộ phát siêu âm (trigger).
* Bộ thu siêu âm (echo).

### **2.4.4 Giao tiếp giữa cảm biến siêu âm HC-SR04 với STM32F401CDU6**

**Các bước giao tiếp:**

* Cấu hình nguồn clock cho MCU.
* **Khởi tạo GPIO:**

§ **Cấu hình chân Trigger:**

· Chế độ output push pull.

· Tần số hoạt động 2 MHz.

§ **Cấu hình chân Echo:**

· Chế độ input (no push).

**Nguyên lý hoạt động:**

* Để đo khoảng cách, ta sẽ phát 1 xung rất ngắn (5 us) từ Trig. Sau đó, cảm biến sẽ tạo ra 1 xung mức cao ở chân Echo cho đến khi nhận lại được sóng phản xạ ở chân Echo. Độ rộng xung sẽ bằng thời gian sóng siêu âm được phát từ cảm biến và quay trở lại.
* Tốc độ của âm thanh trong không khí là 340m/s tương đương 29.412 us/cm nên ta có thể tính được khoảng cách là: độ dài xung mức cao ở chân Echo (us) / 2 / 29.412.

### **2.4.5 Ưu và nhược điểm**

**Ưu điểm:**

* HC-SR04 có giá thành rẻ.
* Độ chính xác cao.

**Nhược điểm:**

* Góc phát hiện hẹp.
* Bị ảnh hưởng bởi các yếu tố môi trường như: nhiệt độ, độ ẩm.

# 

# 

# 

# 

# 

# 

# 

# 

# 

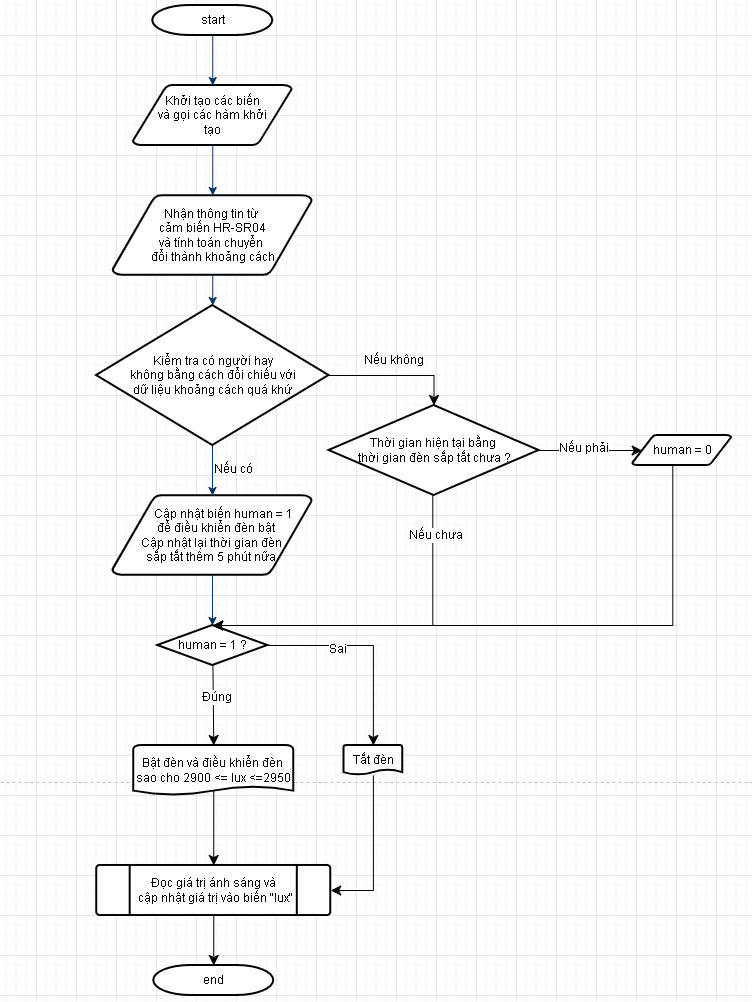
# 

# 

# 

# **CHƯƠNG 3. NGUYÊN LÝ HOẠT ĐỘNG CỦA MÔ HÌNH**

## **3.1 Lưu đồ thuật toán**



Hình 1.5 Lưu đồ thuật toán

### 

### **3.1.1 Khởi tạo các biến có kiểu cấu trúc dành cho các chức năng riêng**

RTC\_HandleTypeDef hrtc;

ADC\_HandleTypeDef hadc1;

RTC\_HandleTypeDef hrtc;

SPI\_HandleTypeDef hspi1;

DMA\_HandleTypeDef hdma\_spi1\_tx;

DMA\_HandleTypeDef hdma\_spi1\_rx;

RTC\_TimeTypeDef sTime;

RTC\_DateTypeDef sDate;

TIM\_HandleTypeDef htim1;

TIM\_HandleTypeDef htim4;

DMA\_HandleTypeDef hdma\_tim1\_ch3;

**char** time[15];

**char** date[15];

**char** sensor[5]; //Các biến mảng để chứa các giá trị string đề suât ra LCD

**char** restminute[2];

**char** latelyoff[2];

**float** distance = 26; //Thời gian hiện tại khởi tạo bằng một giá trị tùy vào môi trường

**float** distance1; //lưu thời gian quá khứ

**float** distance2 = 0; //lưu thời gian hiện tại

### **3.1.2 Khối lấy thông tin cảm biến siêu âm HRSR04**

Thuật toán trong khối này cốt lõi sử dụng một biến numTicks để đếm thời gian mà một chu kì sóng siêu âm được gửi đi và quay về sau đó tính khoảng cách bằng công thức distance = (numTicks + 0.0f)\*2.8\*Speedofsound; có thể hiểu đơn giản là quãng đường bằng vận tốc \*thời gian (1 numTicks = 2.8s)

Cách để lấy dữ liệu là: gửi một xung 10us vào chân trigger của module cảm biến để kích hoạt cảm biến siêu âm:

HAL\_GPIO\_WritePin(Trigger\_GPIO\_Port, Trigger\_Pin, *GPIO\_PIN\_RESET*);

usDelay(3);

HAL\_GPIO\_WritePin(Trigger\_GPIO\_Port, Trigger\_Pin, *GPIO\_PIN\_SET*);

usDelay(10);

HAL\_GPIO\_WritePin(Trigger\_GPIO\_Port, Trigger\_Pin, *GPIO\_PIN\_RESET*);

Cùng lúc đó chúng ta xem xét tín hiệu chân Echo có bằng 0 hay không, nếu có thì đặt biến đếm numTicks = 0, còn nếu không thì biến numsTicks tăng lên 1 và chờ 2us

Sau đó biến distance sẽ được tính bằng công thức:

distance = (numTicks + 0.0f)\*2.8\*Speedofsound;

Cuối cùng chúng ta đã có được thông tin của cảm biến siêu âm

### **3.1.3 Khối có nhiệm vụ xác định có người hay không**

-Khối này hoạt động bằng các xét xem giá trị cảm biến hiện tại có bằng giá trị cảm biến quá khứ không, nếu bằng thì xem như không có vật nào chuyển động từ đó suy ra không có người, trường hợp nếu có sự thay đổi thì ta xét xem giá trị thay đổi này có lớn hơn mức cho phép hay không, nếu sự thay đổi nhỏ thì vẫn bỏ qua và xem như chưa có người, nếu sự thay đổi này lớn(đáng kể) thì kết luận là có người hoạt động trong phòng

Code:

Ban đầu ta gán biến chứa giá trị khoảng cách hiện tại cho biến distance2, biến này cũng có cùng ý nghĩa với biến distance, mục đích là sử dụng biến distance2 để tránh ảnh hưởng tới biến distance phòng trường hợp sử dung biến distance cho mục đích khác

distance2 = distance;

Tiếp theo là một hàm if, else để thiết lập cho nhiệm vụ xác định có người hay không

**if**(((distance2-distance1)>=5)|(distance2<20))

{

**if** (nextFiveMinutes >= 60){

nextFiveMinutes -= 55 ;

};

human = 1;

nextFiveMinutes = sTime.Minutes;

nextFiveMinutes += 0b0101;

**sprintf**(restminute, "%02d'",nextFiveMinutes);

showzifustr(60, 135, restminute , BLACK, WHITE);

}

Hàm này xác định có người khi giá trị khoảng cách hiện tại khi distance2 – distance1 >=5 và khi phát hiện có vật thể quá gần cụ thể là < 20 (giá trị này có thể thay đổi tùy vào môi trường cụ thể khác nhau.

Thiết lâp biến human = 1, biến này có chức năng đại diện cho trạng thái có người (1) hoặc không có người(0)

Biến nextFiveMinutes này lưu giá trị của 5 phút sau kể từ phút hiện tại, cứ mỗi khi phát hiện có người thì biến này sẽ được cập nhật lên bằng câu lệnh:

nextFiveMinutes = sTime.Minutes;

nextFiveMinutes += 0b0101;

sau đó in giá trị ra màn hình LCD bằng câu lệnh:

**sprintf**(restminute, "%02d'",nextFiveMinutes);

showzifustr(60, 135, restminute , BLACK, WHITE);

hàm:

**if** (nextFiveMinutes >= 60) {

nextFiveMinutes -= 55 ;

};

có chức năng cài đặt lại giá trị phút về 5 khi nextFiveMinutes vượt qua ngưỡng 60 phút

Tiếp theo là hàm xác định không có người

**else**{

**if**(sTime.Minutes == nextFiveMinutes){

human = 0;

**sprintf**(timelately, "%02dh'",sTime.Hours);

showzifustr(80, 150, timelately , BLACK, WHITE);

**sprintf**(latelyoff, "%02d'",nextFiveMinutes);

showzifustr(100, 150, latelyoff , BLACK, WHITE);

}

};

Nếu không có người thì thì sẽ kiểm tra thời gian hiện tại có bằng thời gian nextFiveMinutes không, nếu không thì vẫn chưa đến lúc tắt đèn nên không có hành động gì diễn ra trong hàm, ngược lại nếu bằng **if**(sTime.Minutes == nextFiveMinutes) thì trang thái lúc này sẽ là không có người human = 0 sau đó là xuất trạng thái ra màn hình LCD.

### **3.1.4 Khối điều khiển đèn dựa trên biến human**

Ở đây có hai trạng thái, có người và không có người. Như đã cài đặt thì thanh ghi TIM-CCR3 chính là thanh ghi điều chỉnh độ rộng/hẹp của xung điều khiển đèn từ đó tăng hoặc giảm cường độ sáng của đèn cùng với đó ta có biến lux để lưu giá trị cường độ ánh sáng được đọc từ chân ADC giá trị lux này sẽ dao động từ 0-4000 dựa theo độ sáng của môi trường thông thường độ sáng khoảng 2900 thì căn phòng sẽ sáng ở mức tốt nên em sẽ lấy giá trị trong khoảng 2900-2950 để làm mốc

**if**(human == 1){

**if**(lux < 2900){

**if**(TIM1->CCR3 <= 100){

TIM1->CCR3 += 5;

}

}

**if**(lux>2950){

**if**(TIM1->CCR3 > 0){

TIM1->CCR3 -= 5;

}

**if**(TIM1->CCR3 == 0){

TIM1->CCR3 = 0;

}

}

**if**((lux>=2900)&&(lux<=2950))

{

TIM1->CCR3 = TIM1->CCR3;

}

HAL\_TIM\_PWM\_Start(&htim1, TIM\_CHANNEL\_3);

Nếu trạng thái là có người thì thanh ghi xung này sẽ điều chỉnh độ sáng lên tư từ cho đến khi lux trong căn phòng đạt đến ngưỡng 2900(lux), nếu độ sáng trong căn phòng vượt quá 2950(lux) thì sẽ tự điều chỉnh giảm độ sáng đèn lại, nếu như đèn nằm trong khoảng 2900-2950 thì sẽ không tăng không giảm đọ sáng của đèn

**else**{

TIM1->CCR3 = 0;

HAL\_TIM\_PWM\_Start(&htim1, TIM\_CHANNEL\_3);

}

Ngược lại nếu là trạng thái không có người thì đèn sẽ tắt ngay lập tức

### **3.1.5.Khối đọc giá trị ánh sáng lux**

HAL\_ADC\_Start(&hadc1);

HAL\_ADC\_PollForConversion(&hadc1, 20);

lux = HAL\_ADC\_GetValue(&hadc1);

**sprintf**(light, "%d (lux)", lux);

showzifustr(150, 140,light, BLACK, WHITE);

**sprintf**(control, "%d", TIM1->CCR3);

showzifustr(150, 155, control, BLACK, WHITE);

HAL\_Delay(1000);

KHởi động ADC và thiết lập chân adc ở chức năng PollForConversion sau đó dùng hàm HAL\_ADC\_GetValue(&hadc1) để đọc và lưu giá trị vào biến lux

### **3.1.6 Hàm tiện ích**

**Hàm** ‘usDelay’

void usDelay(uint32\_t uSec)

{

if(uSec < 2) uSec = 2;

usTIM->ARR = uSec - 1; /\*sets the value in the auto-reload register\*/

usTIM->EGR = 1; /\*Re-initialises the timer\*/

usTIM->SR &= ~1; //Resets the flag

usTIM->CR1 |= 1; //Enables the counter

while((usTIM->SR&0x0001) != 1);

usTIM->SR &= ~(0x0001);

}

Hàm này tạo ra một độ trễ (delay) micro giây. Nó thiết lập bộ đếm của bộ định thời (usTIM), khởi động lại bộ đếm, và đợi cho đến khi thời gian trễ kết thúc.

### **3.1.7. Hàm cấu hình và khởi tạo**

**Hàm ‘**SystemClock\_Config**’**

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE2);

RCC\_OscInitStruct.OscillatorType=

RCC\_OSCILLATORTYPE\_HSI|RCC\_OSCILLATORTYPE\_LSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.LSIState = RCC\_LSI\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSI;

RCC\_OscInitStruct.PLL.PLLM = 8;

RCC\_OscInitStruct.PLL.PLLN = 84;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 4;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK |RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV2;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2) != HAL\_OK)

{

Error\_Handler();

}

}

Hàm này cấu hình hệ thống đồng hồ. Nó thiết lập các bộ dao động (oscillators) và bộ chia tần số (clock dividers) cho các bus (AHB, APB1, APB2) và hệ thống đồng hồ chính (SYSCLK).

**Hàm** ‘MX\_RTC\_Init**’**

static void MX\_RTC\_Init(void)

{

RTC\_TimeTypeDef sTime = {0};

RTC\_DateTypeDef sDate = {0};

hrtc.Instance = RTC;

hrtc.Init.HourFormat = RTC\_HOURFORMAT\_24;

hrtc.Init.AsynchPrediv = 127;

hrtc.Init.SynchPrediv = 255;

hrtc.Init.OutPut = RTC\_OUTPUT\_DISABLE;

hrtc.Init.OutPutPolarity = RTC\_OUTPUT\_POLARITY\_HIGH;

hrtc.Init.OutPutType = RTC\_OUTPUT\_TYPE\_OPENDRAIN;

if (HAL\_RTC\_Init(&hrtc) != HAL\_OK)

{

Error\_Handler();

}

sTime.Hours = 0x4;

sTime.Minutes = 0x11;

sTime.Seconds = 0x0;

sTime.DayLightSaving = RTC\_DAYLIGHTSAVING\_NONE;

sTime.StoreOperation = RTC\_STOREOPERATION\_RESET;

if (HAL\_RTC\_SetTime(&hrtc, &sTime, RTC\_FORMAT\_BCD) != HAL\_OK)

{

Error\_Handler();

}

sDate.WeekDay = RTC\_WEEKDAY\_TUESDAY;

sDate.Month = RTC\_MONTH\_APRIL;

sDate.Date = 0x23;

sDate.Year = 0x24;

if (HAL\_RTC\_SetDate(&hrtc, &sDate, RTC\_FORMAT\_BCD) != HAL\_OK)

{

Error\_Handler();

}

}

Hàm này khởi tạo RTC và thiết lập thời gian và ngày tháng ban đầu cho RTC.

**Hàm ‘**MX\_ADC1\_Init**’**

static void MX\_ADC1\_Init(void)

{

ADC\_ChannelConfTypeDef sConfig = {0};

hadc1.Instance = ADC1;

hadc1.Init.ClockPrescaler = ADC\_CLOCK\_SYNC\_PCLK\_DIV4;

hadc1.Init.Resolution = ADC\_RESOLUTION\_12B;

hadc1.Init.ScanConvMode = DISABLE;

hadc1.Init.ContinuousConvMode = DISABLE;

hadc1.Init.DiscontinuousConvMode = DISABLE;

hadc1.Init.ExternalTrigConvEdge = ADC\_EXTERNALTRIGCONVEDGE\_NONE;

hadc1.Init.ExternalTrigConv = ADC\_SOFTWARE\_START;

hadc1.Init.DataAlign = ADC\_DATAALIGN\_RIGHT;

hadc1.Init.NbrOfConversion = 1;

hadc1.Init.DMAContinuousRequests = DISABLE;

hadc1.Init.EOCSelection = ADC\_EOC\_SINGLE\_CONV;

if (HAL\_ADC\_Init(&hadc1) != HAL\_OK)

{

Error\_Handler();

}

sConfig.Channel = ADC\_CHANNEL\_0;

sConfig.Rank = 1;

sConfig.SamplingTime = ADC\_SAMPLETIME\_15CYCLES;

if (HAL\_ADC\_ConfigChannel(&hadc1, &sConfig) != HAL\_OK)

{

Error\_Handler();

}

}

Hàm này khởi tạo ADC1 với các thông số cấu hình khác nhau như độ phân giải, chế độ quét, và kênh ADC.

**Các hàm ‘**MX\_SPI1\_Init, MX\_TIM1\_Init’, ‘MX\_TIM4\_Init’, và ‘MX\_DMA\_Init**’**

Các hàm này tương tự nhau trong cách thiết lập cấu hình cho các phần cứng tương ứng: SPI1, Timer 1, Timer 4, và DMA. Chúng khởi tạo và cấu hình các thông số cần thiết cho từng phần cứng.

**Hàm** ‘MX\_GPIO\_Init’

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

HAL\_GPIO\_WritePin(GPIOA, Trigger\_Pin, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, LCD\_CS\_Pin|LCD\_RST\_Pin|LCD\_DC\_Pin, GPIO\_PIN\_RESET);

GPIO\_InitStruct.Pin = Trigger\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(Trigger\_GPIO\_Port, &GPIO\_InitStruct);

GPIO\_InitStruct.Pin = Echo\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(Echo\_GPIO\_Port, &GPIO\_InitStruct);

}

Hàm này khởi tạo các chân GPIO. Nó bật các clock cho các port GPIO và cấu hình các chân như Trigger\_Pin và Echo\_Pin với các chế độ và tốc độ phù hợp.

### **3.1.8 Hàm xử lý lỗi**

void Error\_Handler(void)

{

\_\_disable\_irq();

while (1)

{

}

}

Hàm này được gọi khi có lỗi xảy ra. Nó vô hiệu hóa ngắt và bước vào vòng lặp vô hạn.

### **3.1.9 Các hàm HAL được ghi đè**

#ifdef USE\_FULL\_ASSERT

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

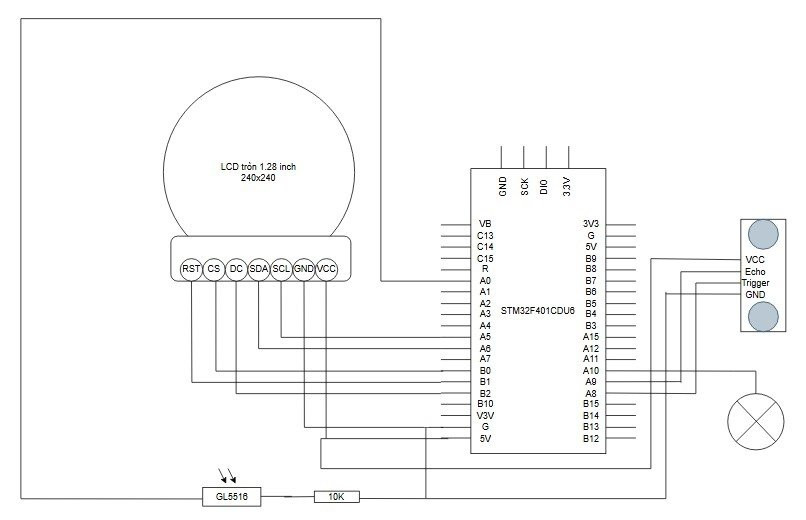
/\* User can add his own implementation to report the file name and line number \*/

}

#endif

Hàm assert\_failed này là tùy chọn, chỉ được sử dụng khi tính năng assert đầy đủ (USE\_FULL\_ASSERT) được bật. Nó báo cáo tên file và số dòng khi một assert thất bại.

## **3.2 Sơ đồ nguyên lý của mô hình đèn chiếu sáng tự động**



Hình 1.6 Sơ đồ nguyên lý của mô hình

**Các kết nối giữa vi điều khiển và linh kiện:**

1. Vi điều khiển STM32F401CDU6:

o Đây là vi điều khiển trung tâm, được kết nối với các thiết bị ngoại vi khác trong hệ thống.

2. Màn hình LCD tròn 1.28 inch, độ phân giải 240x240:

o Kết nối với vi điều khiển thông qua các chân sau:

RST (Reset) -> B1

CS (Chip Select) -> B0

DC (Data/Command) -> B2

SDA (Serial Data) -> A6 (SPI\_MOSI)

SCL (Serial Clock) -> A5 (SPI\_SCK)

GND (Ground) -> GND

VCC (Power) -> 5V

3. Cảm biến siêu âm HC-SR04:

o Được sử dụng để đo khoảng cách.

o Kết nối với vi điều khiển qua các chân:

VCC -> 5V

Echo -> A9

Trigger -> A8

GND -> GND

4. Quang trở GL5516:

o Được sử dụng để đo ánh sáng.

o Kết nối với vi điều khiển thông qua một mạch chia điện áp với điện trở 10K ohm.

o Chân giữa của bộ chia điện áp này được kết nối với A2 của vi điều khiển để đọc giá trị điện áp tương ứng với cường độ ánh sáng.

5. Đèn LED:

o Được kết nối với chân A10 của vi điều khiển.

o Đèn LED sẽ được điều khiển để bật/tắt dựa trên tín hiệu từ vi điều khiển.

**Phân tích chi tiết kết nối**

1 Màn hình LCD:

o Màn hình LCD được kết nối qua giao thức SPI với các chân MOSI (A6), SCK (A5). Chân CS (B0) và DC (B2) được sử dụng để điều khiển chế độ và chọn chip của màn hình.

o Chân RST (B1) để reset màn hình.

2 Cảm biến siêu âm HC-SR04:

o Sử dụng chân A9 để gửi tín hiệu Trigger từ vi điều khiển.

o Chân Echo được kết nối với A8 để nhận tín hiệu phản hồi từ cảm biến.

3 Quang trở GL5516:

o Kết nối với một điện trở 10K ohm để tạo thành mạch chia điện áp.

o Tín hiệu từ quang trở được đưa vào chân A10 của vi điều khiển để đọc giá trị ADC tương ứng với cường độ ánh sáng.

4 Đèn LED:

o Kết nối với chân B1 để điều khiển bật/tắt đèn.

# 

# 

# **CHƯƠNG 4. KẾT QUẢ ĐỀ TÀI**

## **4.1 Các tính năng và lợi ích nổi bật**

**Tự điều chỉnh ánh sáng:** Đèn tự động có thể được lập trình để điều chỉnh độ sáng đến mức mong muốn, phù hợp với nhu cầu sử dụng của từng khu vực. Ví dụ, bạn có thể cài đặt mức sáng 70% cho phòng a, 50% cho phòng b,…

**Cảm biến chuyển động:** Đèn tự động được trang bị cảm biến siêu âm có khả năng phát hiện chuyển động của con người hoặc động vật. Khi có người di chuyển trong khu vực được cài đặt, đèn sẽ tự động bật sáng, cung cấp ánh sáng đầy đủ cho người sử dụng. Sau 2 phút không phát hiện chuyển động, đèn sẽ tự động tắt, giúp tiết kiệm điện năng và tăng tuổi thọ cho bóng đèn.

**Tiết kiệm điện năng:** Đèn tự động giúp tiết kiệm điện năng hiệu quả nhờ khả năng tự động tắt khi không có người sử dụng.

**Tăng tuổi thọ bóng đèn:** Do được sử dụng một cách hợp lý, bóng đèn của đèn tự động có tuổi thọ cao hơn so với bóng đèn thông thường.

**Tạo sự tiện lợi:** Đèn tự động mang đến sự tiện lợi cho người sử dụng khi không cần phải bật tắt đèn thủ công.

**Nâng cao tính an toàn:** Đèn tự động giúp đảm bảo an ninh cho khu vực sử dụng bằng cách tự động bật sáng khi có người di chuyển.

## **4.2 Một số hạn chế**

**Phạm vi sử dụng hạn chế:** vì phạm vi hoạt động của đèn là dựa vào cảm biến nên bị hạn chế về khoảng cách.

**Về việc phân biệt giữa người và động vật:** tuy là có thể điều chỉnh góc phát hiện của cảm biến nhưng vẫn khó tránh khỏi việc nhầm lẫn giữa người và động vật.

# **CHƯƠNG 5. ỨNG DỤNG VÀ PHÁT TRIỂN**

## **5.1 Ứng dụng của đề tài “Mô hình đèn chiếu sáng tự động”**

Với các tính năng nổi trội có thể thấy ứng dụng khá đa dạng:

**Nhà ở:** Đèn tự động được có thể sử dụng phổ biến trong nhà ở, đặc biệt là cho các khu vực như cầu thang, hành lang, ban công, phòng kho, sân vườn,... Nhờ có cảm biến chuyển động, đèn sẽ tự động bật sáng khi bạn đi qua, giúp bạn di chuyển an toàn và tránh va chạm trong điều kiện thiếu sáng.

**Văn phòng:** Đèn tự động cũng được sử dụng trong văn phòng, đặc biệt là cho các khu vực như phòng họp, nhà vệ sinh, hành lang,... Đèn sẽ tự động bật sáng khi có người di chuyển, giúp tiết kiệm điện năng và tạo môi trường làm việc tiện lợi.

**Khu vực công cộng:** Đèn tự động được sử dụng trong các khu vực công cộng như bệnh viện, trường học, trung tâm thương mại,... Đèn sẽ tự động bật sáng khi có người di chuyển, giúp đảm bảo an ninh và tạo môi trường an toàn cho người sử dụng.

## **5.2 Hướng phát triển đề tài**

Kết hợp với các cảm biến khác để nâng cao tiện ích như cảm biến giọng nói để điều khiển bằng giọng nói khi muốn chủ động bật tắt,…

Ứng dụng công nghệ IoT: tích hợp vào hệ thống nhà thông minh cho phép điều khiển từ xa qua thiết bị như điện thoại thông minh,…

Nghiên cứu phát triển để ứng dụng cho lĩnh vực giao thông, an ninh…

## **5.3 Kết luận**

Đèn tự động có khả năng cân bằng ánh sáng sẽ là giải pháp chiếu sáng hữu ích cho một cuộc sống hiện đại và tiết kiệm. Nó mang đến nhiều tiện ích và lợi ích cho người sử dụng.

# 

# 

# 

# 

# **PHỤ LỤC**

## **PHỤ LỤC 1: Code của bài báo cáo**

Github: [Light\_Control\_Project/myprojeck1/Core/Src/main.c at master · BaoNguyen247/Light\_Control\_Project (github.com)](https://github.com/BaoNguyen247/Light_Control_Project/blob/master/myprojeck1/Core/Src/main.c)

## **PHỤ LỤC 2: Bảng thống kê đề tài và giá thành của đề tài**

**Bảng giá linh kiện**

|  |  |  |
| --- | --- | --- |
| Tên linh kiện | Số lượng | Thành tiền |
| STM32F401CDU6 | 1 | 100.000 VND |
| Cảm biến siêu âm HR-SR04 | 1 | 26.000 VND |
| LCD TFT 1.28 Round | 1 | 69.000 VND |
| Trở quang GL5516 | 1 | 2.000 VND |
| Trở, dây điện… | 1 | 20.000 VND |

|  |  |
| --- | --- |
| Tổng tiền | 217.000 VND |

# **TÀI LIỆU THAM KHẢO**

Datasheet STM32F401xD STM32F401xE ARM ® Cortex ® -M4 32b MCU+FPU, 105 DMIPS, 512KB Flash/RAM 96KB, 11 TIM, 1 ADC, 11 comm. <https://www.st.com/resource/en/datasheet/stm32f401re.pdf>

# 