**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**VÀ TRUYỀN THÔNG VIỆT HÀN**

**KHOA KỸ THUẬT MÁY TÍNH VÀ ĐIỆN TỬ**

**--------------**



**ĐỒ ÁN CHUYÊN NGÀNH I**

**ĐỀ TÀI: XÂY DỰNG HỆ THÔNG MỞ CỬA THÔNG MINH**

|  |  |
| --- | --- |
| Sinh viên thực hiện: | **Trần Lê Minh Quân - 22CE.B022** |
| Giảng viên hướng dẫn: | **ThS. Phan Thị Quỳnh Hương** |

*Đà Nẵng, tháng 5, năm 2025*

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**VÀ TRUYỀN THÔNG VIỆT HÀN**

**KHOA KỸ THUẬT MÁY TÍNH VÀ ĐIỆN TỬ**

**--------------**



**ĐỒ ÁN CHUYÊN NGÀNH I**

**ĐỀ TÀI: XÂY DỰNG HỆ THÔNG MỞ CỬA THÔNG MINH**

|  |  |
| --- | --- |
| Sinh viên thực hiện: | **Trần Lê Minh Quân - 22CE.B022** |
| Giảng viên hướng dẫn: | **ThS. Phan Thị Quỳnh Hương** |

*Đà Nẵng, tháng 5, năm 2025*

**Lời Cảm Ơn**

Trong suốt quá trình học tập và thực hiện đồ án chuyên ngành, em đã nhận được rất nhiều sự quan tâm, chỉ dạy và hỗ trợ quý báu từ thầy cô, bạn bè và gia đình.

Trước tiên, em xin bày tỏ lòng biết ơn sâu sắc đến ThS. Phan Thị Quỳnh Hương, người đã tận tình hướng dẫn, chỉ bảo em từ những bước đi đầu tiên cho đến khi hoàn thành đồ án. Sự giúp đỡ, góp ý chuyên môn và tinh thần trách nhiệm của thầy/cô đã giúp em hoàn thiện tốt đề tài này.

Em cũng xin chân thành cảm ơn các thầy cô trong khoa đã truyền đạt cho em những kiến thức quý báu trong suốt thời gian học tập tại trường – nền tảng để em có thể vận dụng vào thực tiễn và triển khai đồ án một cách hiệu quả.

Cuối cùng, em xin cảm ơn gia đình và bạn bè đã luôn động viên, tạo điều kiện thuận lợi để em tập trung nghiên cứu và hoàn thành tốt công việc của mình.

Do thời gian và kinh nghiệm còn hạn chế, đồ án không tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý từ thầy cô và các bạn để đề tài được hoàn thiện hơn.

Em xin chân thành cảm ơn!

**NHẬN XÉT**

**(Của giảng viên hướng dẫn)**

# Chương I: Giới Thiệu Đề Tài

## Đặt vấn đề:

* Ngày nay, với sự phát triển mạnh mẽ của nền công nghệ 4.0, theo đó là sự phát triển của các ngành điện tử tin học, các hệ thống thông minh và an toàn ra đời giúp đỡ con người trong rất nhiều công việc cũng như bảo mật.
* Điển hình trong đó là thiết lập một hệ thống bảo vệ cửa nhà trước sự xâm nhập của kẻ lạ cũng như vấn đề trộm cắp. Hệ thống bảo vệ đó có thể là một ổ khóa thông minh được người dùng cài đặt mật khẩu bằng các dãy số, hay hệ thống được tạo nên dựa trên cơ sở của công nghệ sinh trắc học như là nhận diện khuôn mặt hay cảm biến vân tay….
* Như đã nêu ở trên, hiện tại những nơi như nhà máy xí nghiệp, cơ quan, nhà cửa hay những nơi có sự bảo mật hàng đầu thì một hệ thống lối ra vào hay là phát hiện được sự xâm nhập của kẻ lạ, khả nghi giả mạo… là vô cùng cần thiết. Trong đề tài này, chúng em sẽ thiết kế một hệ thống bảo vệ đóng mở cửa bằng phương pháp cảm biển vân tay tích hợp mật khẩu. Module mật khẩu phòng khi người đến không có cảm biển vân tay hoặc là khi trời mưa cảm biến vân tay không nhận.

## Mục tiêu của đề tài:

* Thiết kế hệ thống đóng mở cửa bằng thẻ từ và mật khẩu.
* Cho phép người dùng lưu trữ nhiều thẻ và mật khẩu.
* Khi người dùng muốn mở cửa thì yêu cầu phải nhập chính xác mật khẩu hoặc sử dụng đúng thẻ từ trước đó, khi thẻ từ và mật khẩu được nhập đúng hoặc sai sẽ có thông báo cụ thể.
* Hệ thống chạy một cách ổn định, gọn gàng dễ lắp đặt.

## Phạm vi nghiên cứu:

### Nghiên cứu thực tế:

* Khảo sát nhu cầu sử dụng các hệ thống kiểm soát cửa thông minh trong môi trường dân dụng và doanh nghiệp.
* Đánh giá ưu, nhược điểm của các hệ thống hiện có để rút ra hướng phát triển phù hợp.

### Nghiên cứu lý thuyết:

* Tìm hiểu nguyên lý hoạt động của hệ thống nhúng và giao tiếp ngoại vi.
* Nghiên cứu các giao thức:
* GPIO: điều khiển thiết bị đầu ra như LED, relay.
* UART: giao tiếp với Bluetooth HC-05, ESP32.
* SPI: đọc dữ liệu từ RFID RC522.
* I2C: hiển thị thông tin trên LCD 16x2.
* Tìm hiểu cơ chế quản lý mật khẩu, mã hóa đơn giản, quản lý thời gian thực.

### Nghiên cứu phần cứng:

* Vi điều khiển TM4C123GH6PM: làm trung tâm điều khiển toàn hệ thống.
* Keypad 4x4: nhập mật khẩu.
* Module RFID RC522: nhận diện thẻ từ hợp lệ.
* Bluetooth HC-05: điều khiển mở cửa từ xa bằng điện thoại.
* LCD 20x4 (I2C): hiển thị thông tin hoạt động.
* Relay + khóa điện: điều khiển việc đóng/mở cửa.
* ESP32: thu thập dữ liệu từ TM4C123 và gửi lên Firebase thông qua kết nối Wi-Fi.

### Nghiên cứu phần mềm:

* Lập trình TM4C123GH6PM bằng ngôn ngữ C, thao tác trực tiếp trên thanh ghi.
* Thiết kế chương trình điều khiển từng module và tích hợp hoạt động song song.
* Lập trình ESP32 để gửi/nhận dữ liệu lên Firebase (Arduino IDE).
* Quản lý các thông tin như: mật khẩu, UID thẻ từ, số lượt truy cập trong ngày.

## Phương pháp thực hiện:

* Thu thập tài liệu lý thuyết về các giao thức truyền thông và module phần cứng.
* Thiết kế mạch nguyên lý và mạch in (nếu cần), lắp ráp hệ thống hoàn chỉnh.
* Thử nghiệm thực tế, đo lường và kiểm tra tính ổn định hệ thống.
* Tổng hợp, viết báo cáo và trình bày kết quả đồ án.

# Chương II: Cơ Sở Lý Thuyết

## 2.1. Giới thiệu về hệ thống nhúng:

### 2.1.1. Khái niệm về hệ thống nhúng:

* Hệ thống nhúng (Embedded system) là cụm từ dùng để chỉ những hệ thống tích hợp cả phần mềm và phần cứng, có thể tự trị được các nhúng vào môi trường hay hệ thống mẹ. Nó là hệ thống phục vụ cho các bài toán chuyên dụng trong lĩnh vực quan trắc, truyền tin hay tự động hóa điều khiển.



Ảnh 1 - Hệ Thống Nhúng

* Về cơ bản, Embedded system gồm 3 thành phần là phần cứng, phần mềm ứng dụng và hiệu điều hành thời gian thực:
* Phần cứng
* Phần mềm ứng dụng
* Hệ điều hành thời gian thực (RTOS): RTOS giám sát phần mềm ứng dụng, cung cấp cơ chế cho phép bộ xử lý chạy theo lịch trình bằng cách tuân theo kế hoạch kiểm soát độ trễ. Hệ điều hành thời gian thực định nghĩa cách thức hoạt động của hệ thống, đặt ra quy tắc trong quá trình thực thi chương trình ứng dụng.
* Các hệ thống quy mô nhỏ có thể không có hệ điều hành thời gian thực.

### 2.1.2. Đặc điểm của hệ thống nhúng:

* Giao diện:

Hệ thống nhúng có thể có hoặc không có giao diện người dùng, giao diện có thể đơn giản (đèn LED và nút bấm) hoặc phức tạp (màn hình cảm ứng hoặc đồ họa), trong đó:

* Embedded system đơn nhiệm không sử dụng giao diện
* Hệ thống đa nhiệm có thể sử dụng giao diện màn hình cảm ứng, đồ họa, các nút bấm tương tác
* Kiến trúc CPU:

Hệ thống sử dụng các kiến trúc CPU như vi điều khiển (microcontroller) và vi xử lý (microprocessor). Vi điều khiển thường tích hợp các thiết bị ngoại vi trên cùng 1 chip để giảm chi phí và kích thước. Trong hệ thống các kiến trúc CPU phổ biến bao gồm ARM, PowerPC, 8051, x86, MIPS, PIC, và nhiều hơn nữa.

* Thiết bị ngoại vi:

Thiết bị ngoại vi sử dụng trong Embedded system để tương tác và giao tiếp với thế giới bên ngoài. Các thiết bị phổ biến bao gồm:

* Các giao diện giao tiếp tuần tự: SPI, I2C, SSC
* Giao diện nối tiếp: RS-422, RS-232, RS-485, USB
* Thiết bị định thời và xử lý tín hiệu: Compare, PLL, Time Processing Units…
* Công cụ phát triển:

Các phần mềm hệ thống được phát triển bằng các trình dịch hợp ngữ, trình biên dịch, các công cụ gỡ lỗi. Các công cụ phát triển bao gồm chương trình mô phỏng, bộ gỡ lỗi mạch, tiện ích checksum và CRC, các công cụ xử lý tín hiệu số.

* Độ tin cậy:

Embedded system cần đạt độ tin cậy cao, nhất là trong các ứng dụng yêu cầu hoạt động liên tục và không gặp lỗi. Các thiết bị không đáng tin cậy như công tắc, ổ đĩa, nút bấm thường được hạn chế sử dụng để đảm bảo độ tin cậy và tính ổn định.

### 2.1.3. Ứng dụng phổ biến trong đời sống và công nghiệp:

Embedded system được ứng dụng rất nhiều trong thực tế, sử dụng trong thiết bị gia dụng, thiết bị mạng, hệ thống định vị và dẫn đường không lưu, thiết bị văn phòng, thiết bị y tế, máy trả lời tự động, dây chuyền sản xuất tự động và robots.



Ảnh 2 - Ứng dụng của Hệ Thống Nhúng

* Thiết bị gia dụng

Hệ thống nhúng được tích hợp trong các thiết bị gia dụng thông thường như lò nước, lò vi sóng, tủ lạnh và nhiều sản phẩm khác. Ứng dụng này giúp tăng tính tiện ích, tự động hóa các hoạt động hàng ngày.

* Thiết bị mạng:

Embedded system được sử dụng trong các thiết bị mạng như gateway, router, hub. Ứng dụng hệ thống trong thiết bị mạng để điều phối, quản lý thông tin, truyền dữ liệu và tạo sự kết nối giữa các thiết bị.

* Hệ thống dẫn đường không lưu, định vị:

Hệ thống nhúng được ứng dụng trong các hệ thống định vị toàn cầu, các vệ tinh, thiết bị dẫn đường để cung cấp thông tin vị trí và hướng dẫn cho người dùng.

* Thiết bị văn phòng:

Embedded system được tích hợp trong các thiết bị văn phòng như máy fax, máy scan, máy photo, máy in để thực hiện các chức năng sao chụp, in ấn, quét tự động và hiệu quả.

* Thiết bị y tế:

Các thiết bị y tế như máy điều hòa nhịp tim, máy thẩm thấu, các thiết bị giám sát và điều chỉnh thông số quan trọng của bệnh nhân được ứng dụng hệ thống nhúng.

* Máy trả lời tự động:

Các máy trả lời tự động ứng dụng Embedded system cho phép tự động xử lý cuộc gọi, cung cấp dịch vụ hoặc thông tin cho người dùng.

* Dây chuyền sản xuất tự động, robots:

Trong các dây chuyền sản xuất tự động, robots trong công nghiệp khi ứng dụng Embedded system giúp tăng năng xuất, hiệu quả, sự chính xác cho quá trình sản xuất.

## 2.2. Ngôn ngữ lập trình C:

### 2.2.1. Tổng quang về ngôn ngữ C:

* C là một ngôn ngữ lập trình cực kỳ phổ biến, đơn giản và linh hoạt, dễ sử dụng. Nó là một ngôn ngữ lập trình có cấu trúc độc lập với máy và được sử dụng rộng rãi để viết các ứng dụng khác nhau: Hệ điều hành như Windows và nhiều chương trình phức tạp khác như cơ sở dữ liệu Oracle, Git, trình thông dịch Python, v.v.
* C là ngôn ngữ lập trình bậc nhất, C là một cơ sở cho việc lập trình. Nếu bạn biết C, bạn có thể dễ dàng nắm bắt kiến thức của các ngôn ngữ lập trình khác dễ dàng hơn.

### 2.2.2. Ưu điểm của ngôn ngữ C trong lập trình vi điều khiển:

* Ngôn ngữ lập trình C thường được sử dụng cho vi điều khiển vì có những ưu điểm sau:
* Hiệu suất cao: C là một ngôn ngữ lập trình gần với ngôn ngữ máy nên có thể tối ưu hóa mã máy và thực thi nhanh chóng trên các vi điều khiển có tài nguyên hạn chế.
* Gần gũi với phần cứng: Cung cấp khả năng truy cập trực tiếp vào phần cứng của vi điều khiển, điều này rất quan trọng khi cần tương tác với các cổng I/O, bộ nhớ, và các thiết bị ngoại vi khác.
* Linh hoạt: Cung cấp các tính năng cần thiết để lập trình các ứng dụng nhúng phức tạp, bao gồm cấu trúc dữ liệu linh hoạt, con trỏ và hàm ngắn gọn.
* Hỗ trợ mạnh mẽ: Có nhiều thư viện hỗ trợ mạnh mẽ cho vi điều khiển, giúp việc phát triển ứng dụng nhúng trở nên dễ dàng hơn.
* Phổ biến: C là một ngôn ngữ lập trình phổ biến và có nhiều tài liệu, cộng đồng hỗ trợ lớn, giúp người lập trình dễ dàng tìm kiếm thông tin và giải quyết vấn đề.

Tóm lại, việc sử dụng ngôn ngữ C cho vi điều khiển mang lại hiệu suất cao, linh hoạt và hỗ trợ mạnh mẽ, giúp lập trình viên dễ dàng phát triển các ứng dụng nhúng phức tạp trên các vi điều khiển.

## 2.3. Vi điều khiển TM4C123GXL:

### 2.3.1. Tổng quan về dòng Tiva C Series:

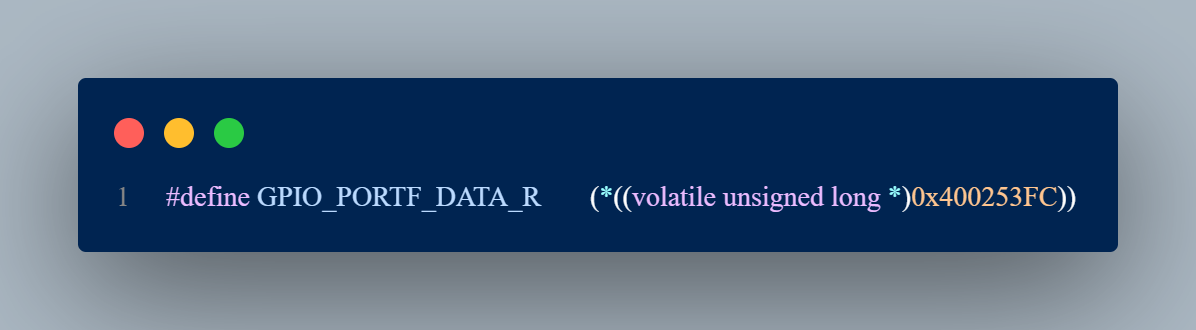


Ảnh 3 - Vi điều khiển TM4C123GXL

* Dòng vi điều khiển Tiva C Series được cung cấp bởi công ty Texas Instruments - một trong những tập đoàn công nghệ bán dẫn hàng đầu thế giới, được thành lập vào năm 1930 và tiến hành hoạt động chính trong lĩnh vực thiết kế và sản xuất vi mạch tích hợp.
* Tiva C Series là một trong những dòng vi điều khiển nổi trội được Texas Instrument sản xuất và bán ra thị trường, với một số thông số như:
* Lõi vi xử lý sử dụng ARM Cortex-M4F có tích hợp FPU (Floating Point Unit) và hỗ trợ DSP instructions (xử lý tín hiệu số). Tần số hoạt động lên đến 80 Hz.
* Bộ nhớ: Flash(32 - 256 KB), SRAM(6 - 32 KB), EEPROM(2 KB).
* Tích hợp nhiều cổng giao tiếp ngoại vi như UART, I2C, SPI, CAN, USB, . . .
* Hỗ trợ lập trình thông qua nhiều IDE như Keli, Code Composer Studio (CCS), . . .

### 2.3.2. Cách cấu hình và thao tác thanh ghi:

* Tổng quan về thanh ghi và bộ nhớ ánh xạ:
* TM4C123GH6PM sử dụng bộ nhớ ánh xạ (Memory-mapped I/O), nghĩa là mỗi thanh ghi điều khiển đều được gán một địa chỉ cố định trong bộ nhớ.
* Truy cập thanh ghi đồng nghĩa với việc truy cập một vùng nhớ cụ thể qua con trỏ hoặc macro.
* Các thanh ghi được định nghĩa sẵn trong file tiêu đề *tm4c123gh6pm.h* (nếu có sử dụng). Nếu không, ta phải tự định nghĩa địa chỉ thanh ghi bằng macro.
* Cách truy cập thanh ghi:

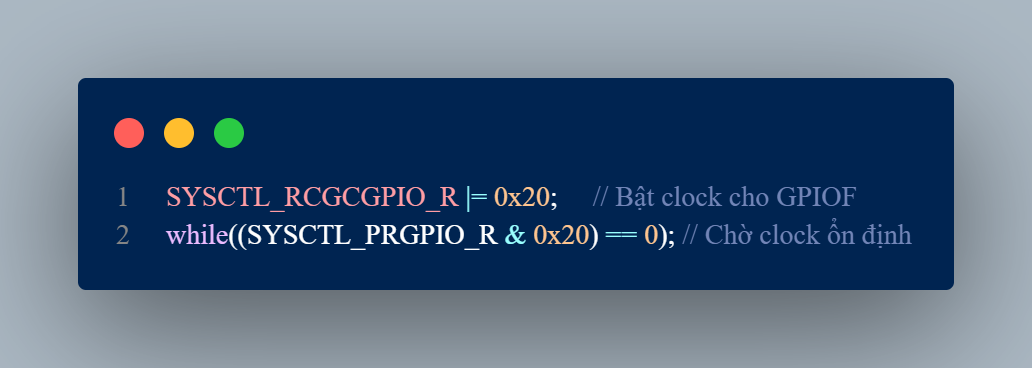


Ảnh 4 - Truy cập thanh ghi

* GPIO\_PORTF\_DATA\_R: đặt tên biến trùng với tên thanh ghi tương ứng để thuận tiện trong quá trình code và debug.
* (\*((volatile . . . \*). . .)): thiết lập biến có thể thay đổi bởi phần cứng.
* (0x400253FC) là địa chỉ vùng nhớ của thanh ghi tương ứng muốn truy cập (như trong ví dụ là thanh ghi GPIODATA của PORT F).
* Các bước cấu hình cho một ngoại vi cơ bản (ví dụ: cấu hình GPIO)

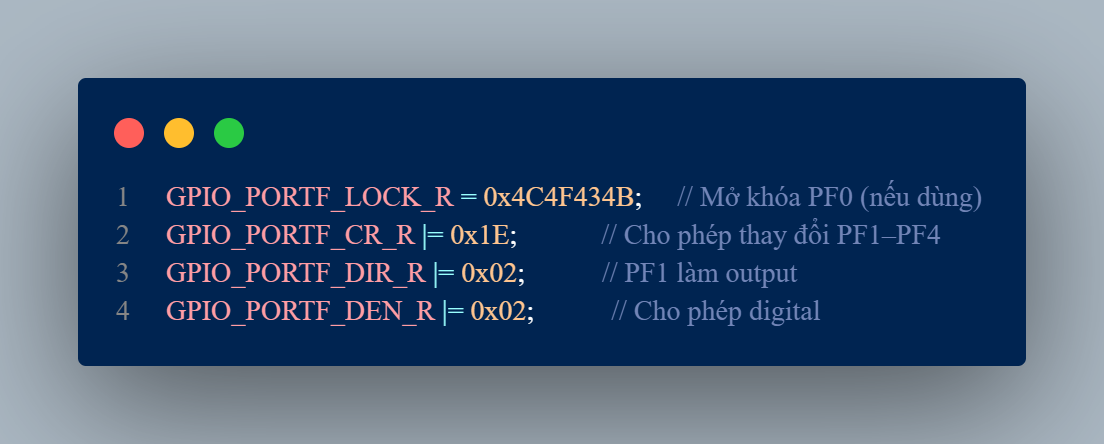
Chúng ta sẽ tiến hành cấu hình chân PF1 làm output để điều khiển LED.

* Bật xung clock cho cổng GPIO\_F



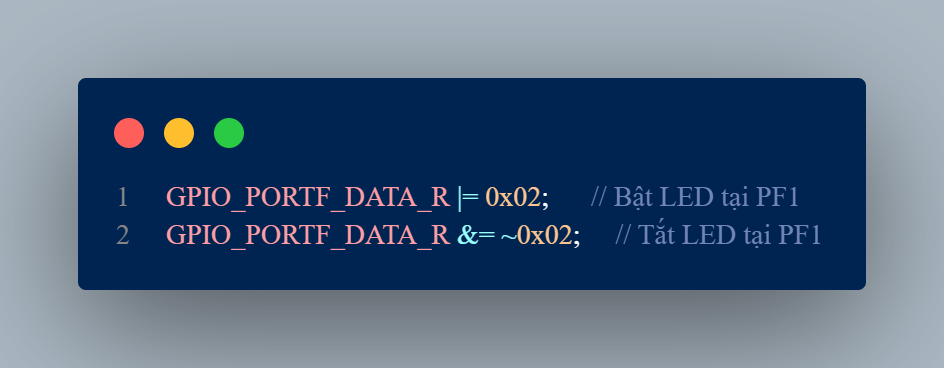
Ảnh 5 - Bật xung clock

* Cấu hình chức năng cho chân



Ảnh 6 - Cấu hình chức năng cho chân

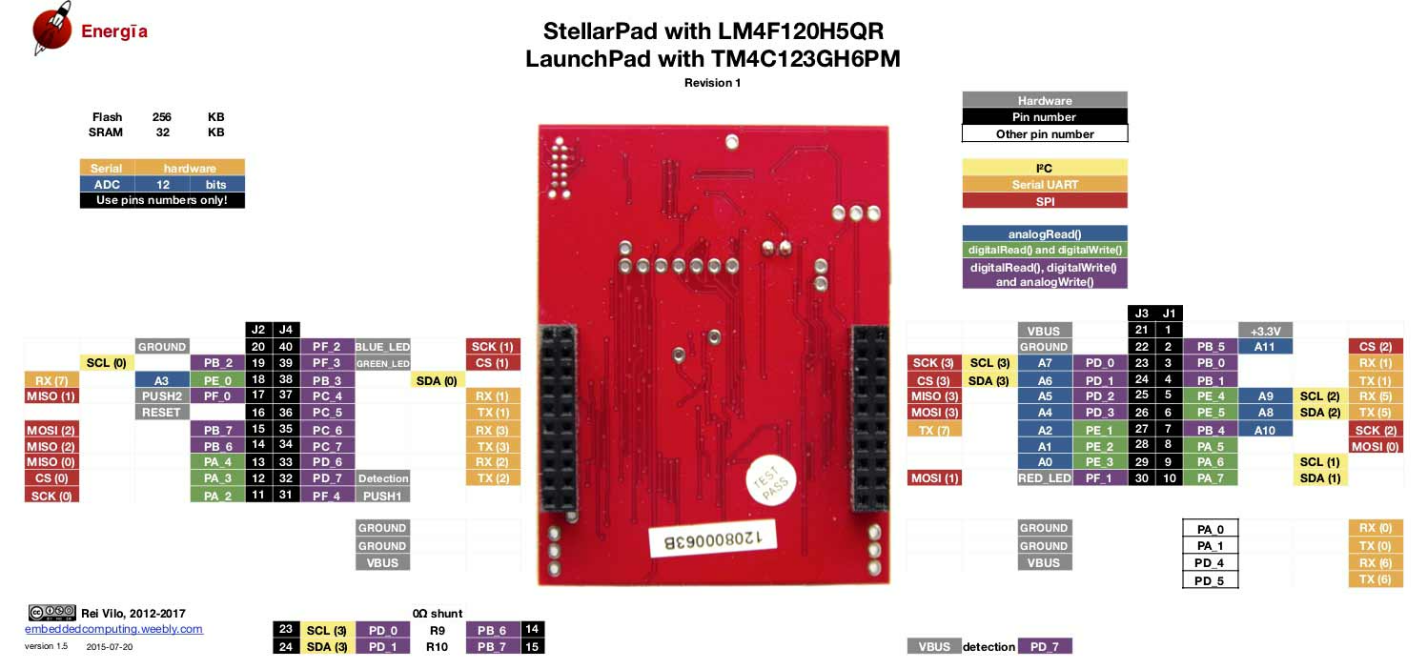
* Điều khiển chân



Ảnh 7 - Điều khiển chân

## 2.4. Các giao tiếp truyền dữ liệu:

Giao tiếp truyền dữ liệu là những cách thức để vi điều khiển có thể truyền hoặc tiếp nhận dữ liệu từ module hay một vi điều khiển khác.



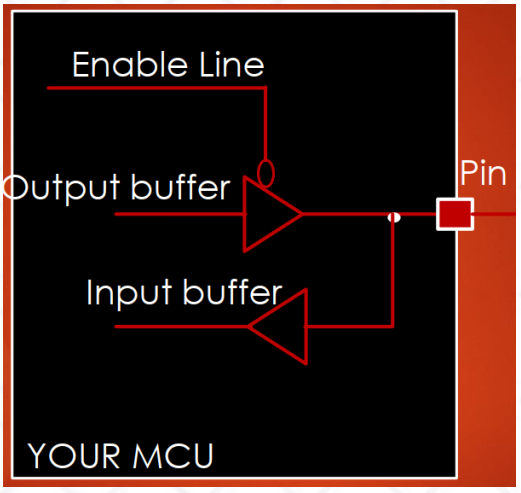
Ảnh 8 - Sơ đồ chân giao tiếp của TM4C123GXL

### 2.4.1. Giao tiếp GPIO:

* General Purpose Input Output (GPIO) về cơ bản là một chân có thể được cấu hình làm đầu vào hoặc đầu ra. Nếu chúng ta cấu hình chân như một đầu ra, chúng ta có thể ghi 0 (LOW) hoặc 3,3 / 5 V (VDD) vào chân đó. Khi được cấu hình làm đầu vào, chúng ta có thể đọc tín hiệu trên chân đó. GPIO là giao diện tiêu chuẩn mà qua đó vi điều khiển có thể giao tiếp với thế giới bên ngoài.
* Có thể được sử dụng để đọc các giá trị từ cảm biến analog hoặc kỹ thuật số, điều khiển đèn LED, điều khiển đồng hồ cho giao tiếp I2C,…

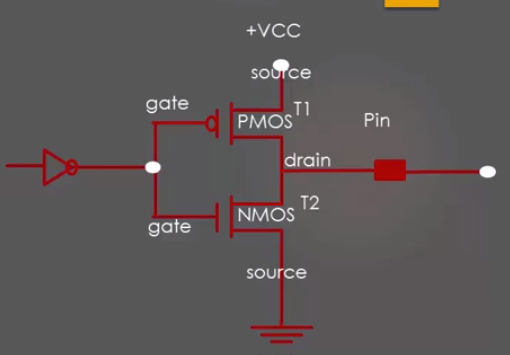
Cách GPIO hoạt động:

* Mỗi chân bao gồm hai bộ đệm - bộ đệm đầu vào và đầu ra cùng với một dòng enable. Khi dòng enable là 0, bộ đệm đầu ra được kích hoạt và bộ đệm đầu vào sẽ bị vô hiệu hóa. Nếu chúng ta đặt dòng enable là 1, thì chân sẽ được cấu hình ở chế độ đầu vào bằng cách kích hoạt bộ đệm đầu vào.



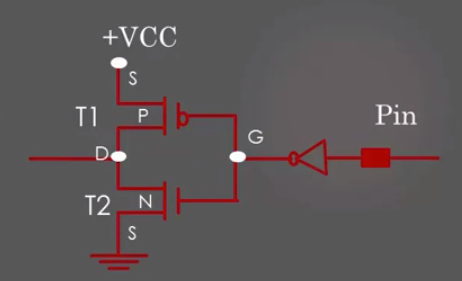
Ảnh 9 - Cách GPIO hoạt động

* Bộ đệm đầu ra về cơ bản là hai transistor CMOS được kết nối theo kiểu bên dưới. Khi chúng ta ghi 1 từ phần mềm, inverter sẽ đặt nó là 0 và mạch PMOS (T1) sẽ hoạt động do đó chân sẽ được kéo lên VCC. Khi chúng ta ghi 0 thì ngược lại. Đây là cấu hình mặc định của chân GPIO và còn được gọi là đầu ra push-pull



Ảnh 10 - Bộ đệm đầu ra (Output Buffer)

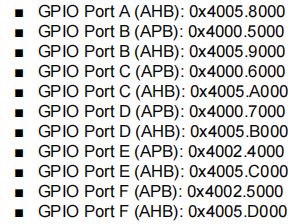
* Khi chân đọc 1 (HIGH), inverter sẽ đảo mức logic cao này về 0 và transistor T1 PMOS sẽ BẬT. Vì nó được kéo lên VCC, phần mềm đọc logic là 1. Khi chân đọc 0 (LOW), inverter đảo mức logic thấp này thành 1 và T2 NMOS sẽ được kích hoạt và nó sẽ được kéo xuống 0, do đó phần mềm đọc 0. Dòng kích hoạt này được cấu hình bởi phần mềm.



Ảnh 11 - Bộ đệm đầu vào (Input Buffer)

GPIO trong TM4C123GXL:

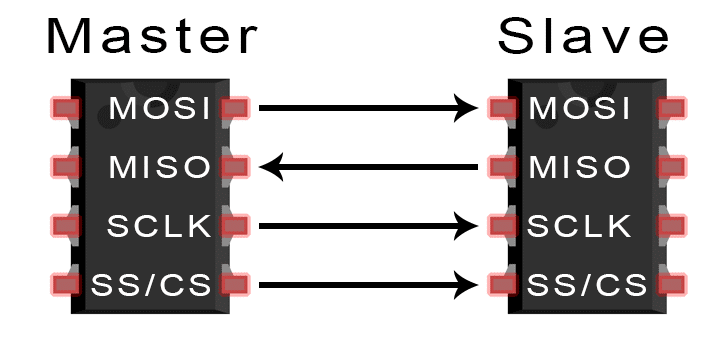
* Trong vi điều khiển TM4C123GXL, GPIO sẽ được chia làm 8 cổng (port) từ A - F. Mỗi cổng sẽ có 8 chân từ pin 0 - 7. Mỗi cổng sẽ có địa chỉ dưới dạng HEX tương ứng để người dùng có thể thiết lập khi cần sử dụng đến.



Ảnh 12 - Địa chỉ các cổng GPIO trong vi điều khiển TM4C123GXL

### 2.4.2. Giao tiếp SPI:

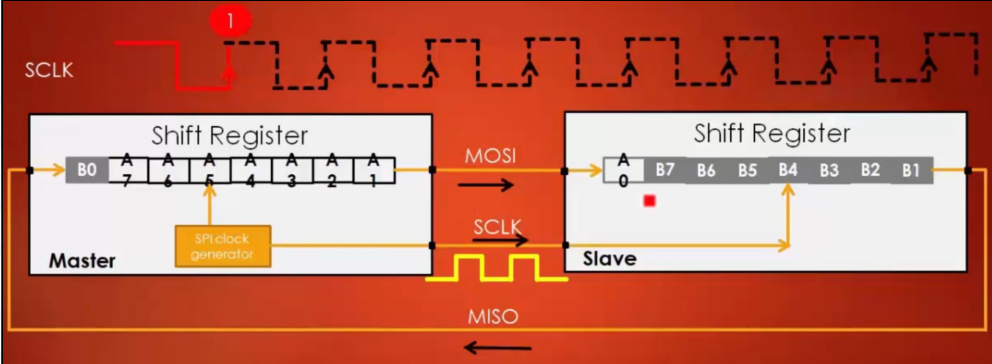
* Giao tiếp ngoại vi nối tiếp hoặc Serial Peripheral Interface (SPI) là một chuẩn đồng bộ nối tiếp để truyền dữ liệu ở chế độ song công toàn phần (full – duplex) tức trong cùng một thời điểm có thể xảy ra đồng thời quá trình truyền và nhận.
* SPI là một loại giao thức kiểu Master – Slave cung cấp một giao diện chi phí đơn giản và chi phí thấp giữa vi điều khiển và các thiết bị ngoại vi của nó.



Ảnh 13 - Giao tiếp SPI

Cách SPI hoạt động:

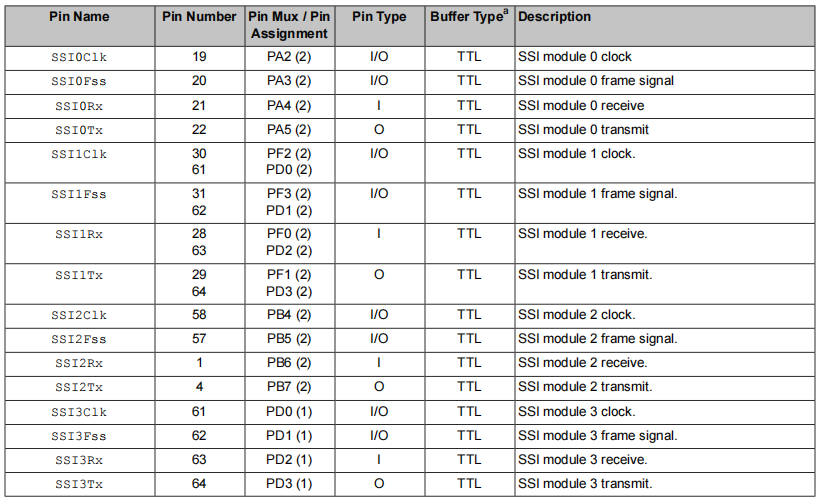
* Bus SPI bao gồm 4 tín hiệu hoặc chân. Chúng là:
* Master – Out / Slave – In (MOSI hay SI): cổng ra của bên Master, cổng vào của bên Slave, dành cho việc truyền dữ liệu từ thiết bị Master đến thiết bị Slave .
* Master – In / Slave – Out (MISO hay SO): cổng vào của bên Master, cổng ra của bên Slave, dành cho việc truyền dữ liệu từ thiết Slave đến thiết bị Master.
* Serial Clock (SCK hay SCLK): xung giữ nhịp cho giao tiếp SPI
* Chip Select (CS) hay Slave Select (SS): chọn chip
* Một Master có thể cùng lúc kết nối với nhiều Slave khác nhau, nhưng chỉ có thể truyền/nhận dữ liệu cho 1 Slave trong 1 phiên làm việc.
* Một phiên truyền nhận dữ liệu giữa Master và Slave sẽ diễn ra như sau:
* Master kéo chân SS nối với Slave tương ứng xuống LOW
* Lần lượt động thời Master và Slave sẽ trao đổi từng bit cho đến khi dữ liệu được truyền/ nhận hoàn tất. Đối với mỗi bit được truyền bởi một thiết bị, thiết bị kia phải gửi dữ liệu một bit, tức là Master truyền dữ liệu đồng thời trên MOSI và nhận dữ liệu từ Slave trên đường MISO.
* Dữ liệu được truyền/nhận sẽ được xác thực thông qua tín liệu clock được gửi đi từ Master.
* Khi kết thúc phiên truyền/nhận dữ liệu thì chân SS của Slave tương ứng được kéo lên HIGH.



Ảnh 14 - Cách SPI hoạt động

SPI trong TM4C123GXL:

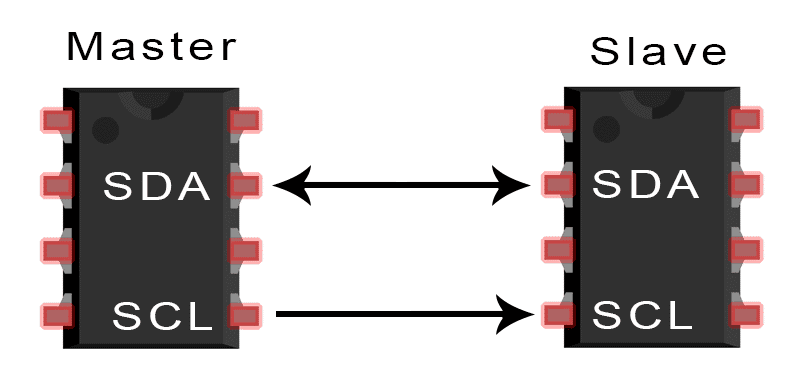
* Trong vi điều khiển TM4C123GXL, SPI được chia thành 4 module SPI 0 - 3. Mỗi chân SS, SCK, MOSI và MISO của từng module sẽ được tích hợp vào những chân GPIO có sẵn để dễ dàng thao tác kết nối.



Ảnh 15 - Giao tiếp SPI trong vi điều khiển TM4C123GXL

### 2.4.3. Giao tiếp I2C:

* Inter-Integrated Circuit (I2C) là một giao thức giao tiếp được dùng để truyền dữ liệu giữa một bộ xử lý trung tâm với nhiều IC trên cùng một board mạch chỉ sử dụng hai đường truyền tín hiệu.

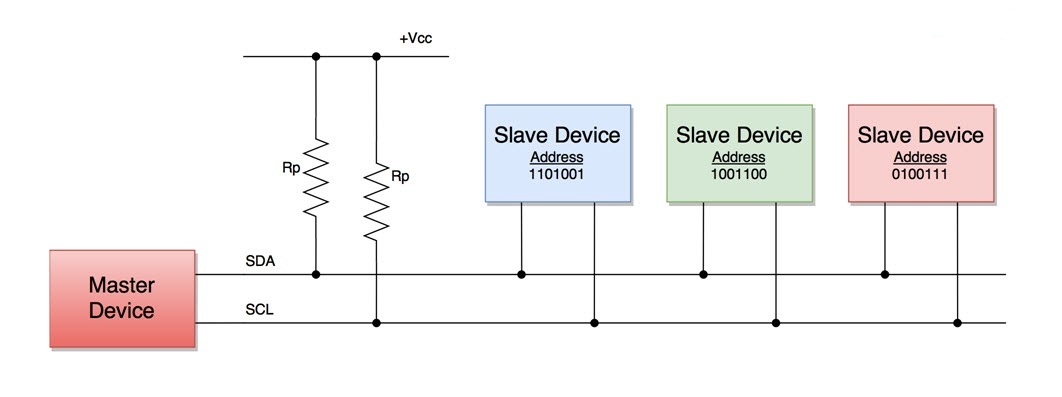


Ảnh 16 - Giao tiếp I2C

* Đặc điểm của giao tiếp I2C:
* Chỉ cần có hai đường bus (dây) chung để điều khiển bất kỳ thiết bị / IC nào trên mạng I2C
* Không cần thỏa thuận trước về tốc độ truyền dữ liệu như trong giao tiếp UART. Vì vậy, tốc độ truyền dữ liệu có thể được điều chỉnh bất cứ khi nào cần thiết
* Cơ chế đơn giản để xác thực dữ liệu được truyền
* Sử dụng hệ thống địa chỉ 7 bit để xác định một thiết bị / IC cụ thể trên bus I2C
* Các mạng I2C dễ dàng mở rộng. Các thiết bị mới có thể được kết nối đơn giản với hai đường bus chung I2C

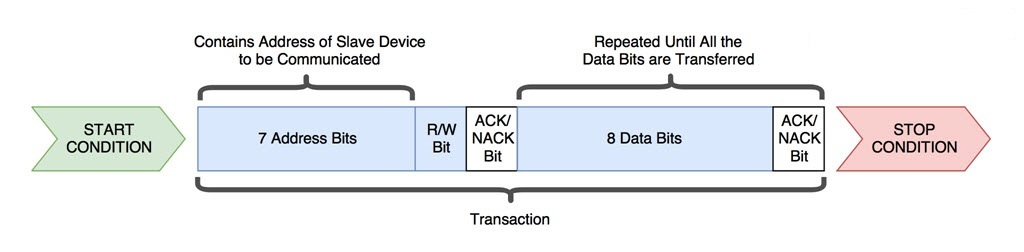
Cách I2C hoạt động:

* Bus I2C (dây giao tiếp) chỉ gồm hai dây và được đặt tên là Serial Clock Line (SCL) và Serial Data Line (SDA). Dữ liệu được truyền đi được gửi qua dây SDA và được đồng bộ với tín hiệu đồng hồ (clock) từ SCL. Tất cả các thiết bị / IC trên mạng I2C được kết nối với cùng đường SCL và SDA như sau:



Ảnh 17 - Kết nối giữa Master và Slave trong giao tiếp I2C

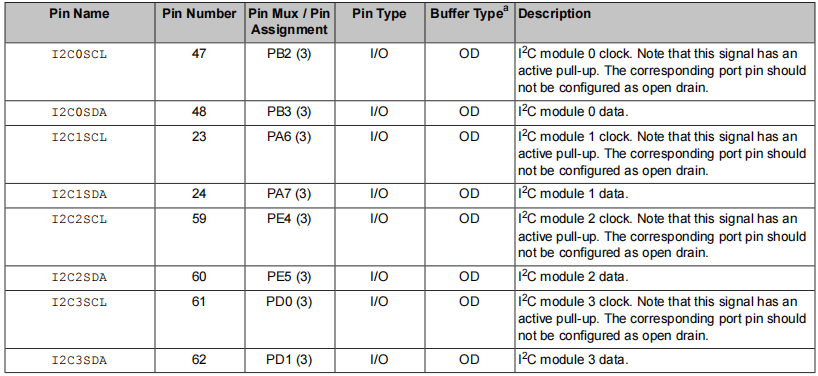
* Dữ liệu được truyền giữa thiết bị Master và các thiết bị Slave thông qua một đường dữ liệu SDA duy nhất, thông qua các chuỗi có cấu trúc gồm các số 0 và 1 (bit). Mỗi chuỗi số 0 và 1 được gọi là giao dịch (transaction) và dữ liệu trong mỗi giao dịch có cấu trúc như sau:



Ảnh 18 - Cấu trúc truyền/nhận dữ liệu trong giao tiếp I2C

I2C trong TM4C123GXL:

* Trong vi điều khiển TM4C123GXL, I2C được chia thành 4 module I2C 0 - 3. Mỗi chân SDA và SCL của từng module sẽ được tích hợp vào những chân GPIO có sẵn để dễ dàng thao tác kết nối.

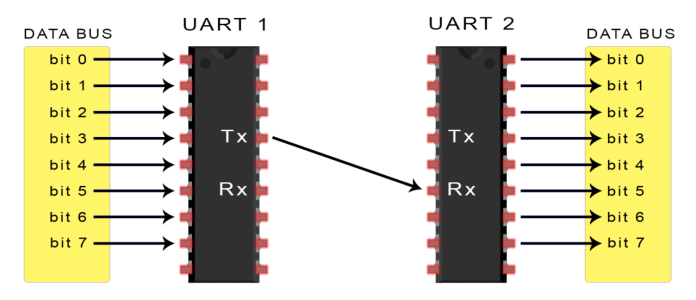


Ảnh 19 - Giao tiếp I2C trong vi điều khiển TM4C123GXL

### 2.4.4. Giao tiếp UART:

Universal Asynchronous Receiver-Transmitter (UART) là một trong những giao thức truyền thông giữa thiết bị với thiết bị được sử dụng nhiều nhất.

Bạn có thể thấy giao tiếp UART được sử dụng nhiều trong các ứng dụng để giao tiếp với các module như: Wifi, Bluetooth, Xbee, module đầu đọc thẻ RFID với Raspberry Pi, Arduino hoặc vi điều khiển khác. Đây cũng là chuẩn giao tiếp thông dụng và phổ biến trong công nghiệp từ trước đến nay.



Ảnh 20 - Giao tiếp UART

Cách UART hoạt động:

UART trong TM4C123GXL:

## 2.4. Giới thiệu về các module phần cứng:

### 2.4.1. Keypad 4x4:

### 2.4.2. Module RFID RC522:

### 2.4.3. Module Bluetooth HC-05:

### 2.4.4. Màn hình LCD 20x4 (I2C):

### 2.4.5. Relay và khóa điện:

### 2.4.6. ESP32:

## 2.5. Tổng quan về Firebase:

### 2.5.1. Firebase là gì?

### 2.5.2. Realtime Database / Firestore:

### 2.5.3. Ứng dụng trong lưu trữ dữ liệu IoT:

# Chương III: Phân Tích Và Thiết Kế Hệ Thống

# Chương IV: Xây Dựng Hệ Thống

# Chương V: Kết Luật Và Hướng Phát Triển