

HỌC VIỆN KỸ THUẬT MẬT MÃ
KHOA CÔNG NGHỆ THÔNG TIN



BÀI TẬP MÔN LẬP TRÌNH HỆ THỐNG NHÚNG LINUX
XÂY DỰNG ỨNG DỤNG VỚI GIAO DIỆN ĐỒ HỌA NHẬP VÀO DÒNG
KÝ TỰ, SAU ĐÓ HIỂN THỊ DÒNG KÝ TỰ ĐÓ LÊN MÀN HÌNH LCD
VÀ GIAO DIỆN ỨNG DỤNG

Khoa: Công nghệ thông tin

Chuyên ngành: Kỹ thuật phần mềm nhúng và di động

Người hướng dẫn

ThS. Lê Thị Hồng Vân

Khoa công nghệ thông tin - Học viện Kỹ thuật Mật mã

Nhóm sinh viên thực hiện:

Phạm Văn Dũng CT040308

Phạm Thị Phương Anh CT040401

Nhóm 01

Hà Nội - 2023

MỤC LỤC

| | |
|--|------------|
| MỤC LỤC | i |
| DANH SÁCH HÌNH VẼ | iii |
| DANH SÁCH BẢNG | iv |
| DANH SÁCH TỪ VIẾT TẮT | v |
| LỜI NÓI ĐẦU | vi |
| 1 CƠ SỞ LÝ THUYẾT | 1 |
| 1.1 Hệ thống nhúng | 1 |
| 1.1.1 Khái niệm hệ thống nhúng | 1 |
| 1.1.2 Cấu trúc hệ thống nhúng | 2 |
| 1.1.3 Phát triển phần mềm trong hệ thống nhúng | 2 |
| 1.2 Hệ thống nhúng sử dụng hệ điều hành linux | 3 |
| 1.2.1 Khái niệm hệ thống nhúng linux | 3 |
| 1.2.2 Các thành phần trong hệ thống nhúng linux | 4 |
| 1.2.3 Lập trình ngoại vi trong hệ thống nhúng linux | 5 |
| 1.2.4 Trình điều khiển thiết bị trong linux | 6 |
| 1.2.5 Trình điều khiển thiết bị I ² C | 6 |
| 1.2.6 Giao tiếp với trình điều khiển thiết bị | 7 |
| 1.2.7 Lập trình giao diện trong hệ thống nhúng linux | 8 |
| 1.3 Các thiết bị ngoại vi | 9 |
| 1.3.1 Máy tính nhúng Raspberry pi 3 | 9 |
| 1.3.2 Mô đun màn hình LCD 1602 HD44780 | 10 |
| 1.3.3 Mô đun I ² C PCF8574 | 11 |
| 2 PHÂN TÍCH - THIẾT KẾ | 13 |
| 2.1 Mô tả bài toán | 13 |

| | | |
|----------|--|-----------|
| 2.1.1 | Yêu cầu chức năng | 13 |
| 2.1.2 | Yêu cầu phi chức năng | 13 |
| 2.2 | Sơ đồ khối | 14 |
| 2.3 | Trình điều khiển thiết bị | 14 |
| 2.4 | Giao diện đồ họa | 17 |
| 3 | XÂY DỰNG HỆ THỐNG | 18 |
| 3.1 | Triển khai phần cứng | 18 |
| 3.2 | Triển khai driver | 19 |
| 3.3 | Triển khai giao diện đồ họa | 22 |
| 3.4 | Thử nghiệm thiết bị | 23 |
| 3.5 | Nhận xét | 25 |
| 3.6 | Hạn chế của thiết bị | 25 |
| 4 | KẾT LUẬN | 26 |
| 4.1 | Kết quả đạt được | 26 |
| 4.2 | Phương hướng phát triển và đề xuất | 26 |

DANH SÁCH HÌNH VẼ

| | | |
|-----|---|----|
| 1.1 | Cấu trúc hệ thống nhúng | 2 |
| 1.2 | Cấu trúc hệ điều hành | 4 |
| 1.3 | Khởi tạo HD44780 | 11 |
| 1.4 | I ² C bus | 11 |
| 2.1 | Biểu đồ usecase | 13 |
| 2.2 | Sơ đồ khối hệ thống | 14 |
| 2.3 | Sơ đồ khối của trình điều khiển thiết bị | 15 |
| 2.4 | Hoạt động của luồng điều khiển chính | 16 |
| 3.1 | Sơ đồ ra chân các ngoại vi của Raspberry Pi | 18 |
| 3.2 | Giao diện đồ họa | 23 |
| 3.3 | Thực hiện đổi nội dung hiển thị | 24 |
| 3.4 | Màn hình hiển thị sau khi đổi nội dung | 24 |

DANH SÁCH BẢNG

| | | |
|-----|---|----|
| 2.1 | Danh sách file sysfs | 17 |
| 3.1 | Sơ đồ nối dây giữa Raspberry Pi và mô đun PCF8574 | 18 |

DANH MỤC TỪ VIẾT TẮT

ADC Analog to Digital Converter. 2

I²C Inter-Integrated Circuit. i, iii, 5, 6, 11, 12, 26

SPI Serial Peripheral Interface. 6

UART Universal Asynchronous Receiver-Transmitter. 2

USB Universal Serial Bus. 5, 6, 9

LỜI NÓI ĐẦU

Các hệ thống nhúng đã hiện diện trong hầu hết các hoạt động của con người trong cuộc sống ngày nay, từ các thiết bị điện đơn giản đến các thiết bị phức tạp. Phát triển các hệ thống nhúng như vậy, yêu cầu lượng thời gian nghiên cứu cũng như lượng kiến thức lớn gấp bội khi độ phức tạp của hệ thống tăng lên. Để đơn giản hóa quy trình phát triển cũng như chi phí xây dựng hệ thống, các hệ thống nhúng nói trên có thể phát triển trên nền hệ điều hành linux.

Với mong muốn tìm hiểu sâu hơn về hệ sinh thái mã nguồn mở linux và quy trình phát triển hệ thống nhúng trên nền linux, nhóm chúng em quyết định thực hiện đề tài: **“Xây dựng ứng dụng với giao diện đồ họa nhập vào dòng ký tự, sau đó hiển thị dòng ký tự đó lên màn hình LCD và giao diện ứng dụng”**. Báo cáo bao gồm các chương và bố cục sau:

Chương 1: Cơ sở lý thuyết

Chương 2: Phân tích - Thiết kế

Chương 3: Xây dựng hệ thống

Chương 4: Kết luận

Hà Nội, ngày 06 tháng 6 năm 2022

Nhóm sinh viên thực hiện

Chương 1: CƠ SỞ LÝ THUYẾT

1.1 Hệ thống nhúng

1.1.1 Khái niệm hệ thống nhúng

Hệ thống nhúng (Embedded system) là một thuật ngữ để chỉ một hệ thống có khả năng tự trị được nhúng vào một môi trường hay một hệ thống mẹ. Đó là các hệ thống tích hợp cả phần cứng và phần mềm phục vụ các bài toán chuyên dụng trong nhiều lĩnh vực công nghiệp, tự động hoá điều khiển, quan trắc và truyền tin. Đặc điểm của các hệ thống nhúng là hoạt động ổn định và có tính năng tự động hoá cao. [3]

Hệ thống nhúng thường được thiết kế để thực hiện một chức năng chuyên biệt nào đó. Khác với các máy tính đa chức năng, chẳng hạn như máy tính cá nhân, một hệ thống nhúng chỉ thực hiện một hoặc một vài chức năng nhất định, thường đi kèm với những yêu cầu cụ thể và bao gồm một số thiết bị máy móc và phần cứng chuyên dụng mà ta không tìm thấy trong một máy tính đa năng nói chung. Vì hệ thống chỉ được xây dựng cho một số nhiệm vụ nhất định nên các nhà thiết kế có thể tối ưu hoá nó nhằm giảm thiểu kích thước và chi phí sản xuất. Các hệ thống nhúng thường được sản xuất hàng loạt với số lượng lớn. Hệ thống nhúng rất đa dạng, phong phú về chủng loại. Xét về độ phức tạp, hệ thống nhúng có thể rất đơn giản với một vi điều khiển hoặc rất phức tạp với nhiều đơn vị, các thiết bị ngoại vi và mạng lưới được nằm gọn trong một lớp vỏ máy lớn.

Các thiết bị PDA hoặc máy tính cầm tay cũng có một số đặc điểm tương tự với hệ thống nhúng như các hệ điều hành hoặc vi xử lý điều khiển chúng nhưng các thiết bị này không phải là hệ thống nhúng thật sự bởi chúng là các thiết bị đa năng, cho phép sử dụng nhiều ứng dụng và kết nối đến nhiều thiết bị ngoại vi.

Có rất nhiều hãng sản xuất bộ vi xử lý và phần cứng khác nhau: Texas Instrument, Freescale, ARM, Intel, Motorola, Atmel, AVR, ...

Những hệ điều hành khác nhau: QNX, ulTRON, Windows CE/XP Embedded, Embedded Linux, ...

Những ngôn ngữ lập trình khác nhau: C/C++, Assembly, PLC, ...

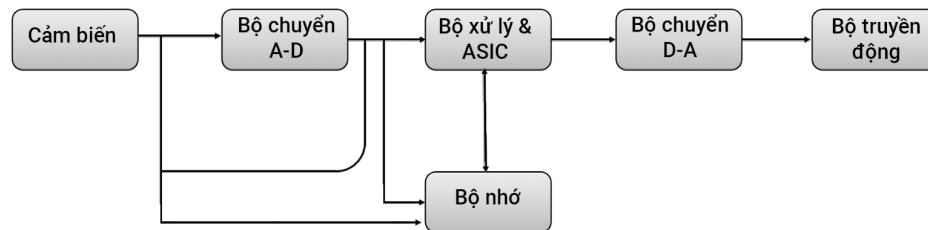
1.1.2 Cấu trúc hệ thống nhúng

Vơ bản thì cấu trúc của hệ thống nhúng sẽ bao gồm 3 phần: phần mềm hệ thống, phần mềm ứng dụng, bộ vi xử lý.

Phần mềm hệ thống: phần mềm này không bắt buộc phải có và có thể tái sử dụng trên một hệ thống nhúng khác. Chúng có chức năng quản lý bộ nhớ, quản lý tiến trình, quản lý chia sẻ tài nguyên. Trình điều khiển thiết bị: UART, Ethernet, ADC... Hệ điều hành nhúng: eCos, ucLinux, VxWorks, Monta Vista Linux, BIOS, QNX...

Phần mềm ứng dụng: không bắt buộc phải có và khó tái sử dụng trên một hệ thống nhúng khác. Chúng có chức năng quyết định hành vi cho một hệ thống nhúng.

Bộ vi xử lý (Hardware): đây là các thành phần bắt buộc trong một hệ thống nhúng, gồm có một số thiết bị như vi xử lý, bộ nhớ, tụ điện, điện trở, mạch tích hợp, bản in mạch,...



Hình 1.1: Cấu trúc hệ thống nhúng

1.1.3 Phát triển phần mềm trong hệ thống nhúng

Một hệ thống nhúng luôn có 3 thành phần:

- Hardware (Phần cứng): Là phần quan trọng nhất của hệ thống nhúng, nó quyết định chính đến chất lượng, giá thành của hệ thống nhúng.
- Operation System (Hệ điều hành): Là phần mềm chạy đầu tiên trong hệ thống cho chức năng điều khiển, quản lý tài nguyên của hệ thống và cung cấp một giao diện phát triển chung.

- Applications (Phần mềm ứng dụng): Quyết định chức năng của hệ thống và là thành phần tương tác với người dùng. Trong Linux, nó được gọi là phần mềm nhúng Linux

Hệ điều hành nhúng Linux (Linux Embedded Software) là những hệ điều hành nhúng sử dụng nhân Linux, có chức năng điều khiển phần cứng hoặc thiết bị kết nối với hệ thống. Không giống với những phần mềm ứng dụng (Application Software) có thể chạy nhiều hệ thống máy tính, phần mềm nhúng Linux chỉ có thể chạy được trên những phần cứng cố định. Phần mềm nhúng Linux phải được thiết kế để chạy được trên những hệ thống hạn chế về tốc độ xử lý, bộ nhớ... mà vẫn đảm bảo hiệu năng.

1.2 Hệ thống nhúng sử dụng hệ điều hành linux

1.2.1 Khái niệm hệ thống nhúng linux

Linux là một hệ điều hành mã nguồn mở (Open-source OS) chạy trên hầu hết các kiến trúc vi xử lý, bao gồm dòng vi xử lý ARM. Linux được hỗ trợ bởi một cộng đồng mã nguồn mở (GNU), chính điều này làm cho Linux rất linh hoạt và phát triển rất nhanh với nhiều tính năng không thua kém các hệ điều hành khác hiện nay. Tất cả các ứng dụng chạy trên hệ điều hành UNIX đều tương thích với Linux.

Hầu hết các bản Linux đều hỗ trợ rất nhiều ngôn ngữ lập trình, đặc biệt công cụ GCC cho phép người lập trình có thể biên dịch và thực thi ứng dụng viết bằng nhiều ngôn ngữ lập trình khác nhau: C/C++, Java, .v.v... Ngoài ra, Linux còn hỗ trợ ngôn ngữ lập trình để phát triển các ứng dụng đồ họa như: JTK+, Qt, .v.v...

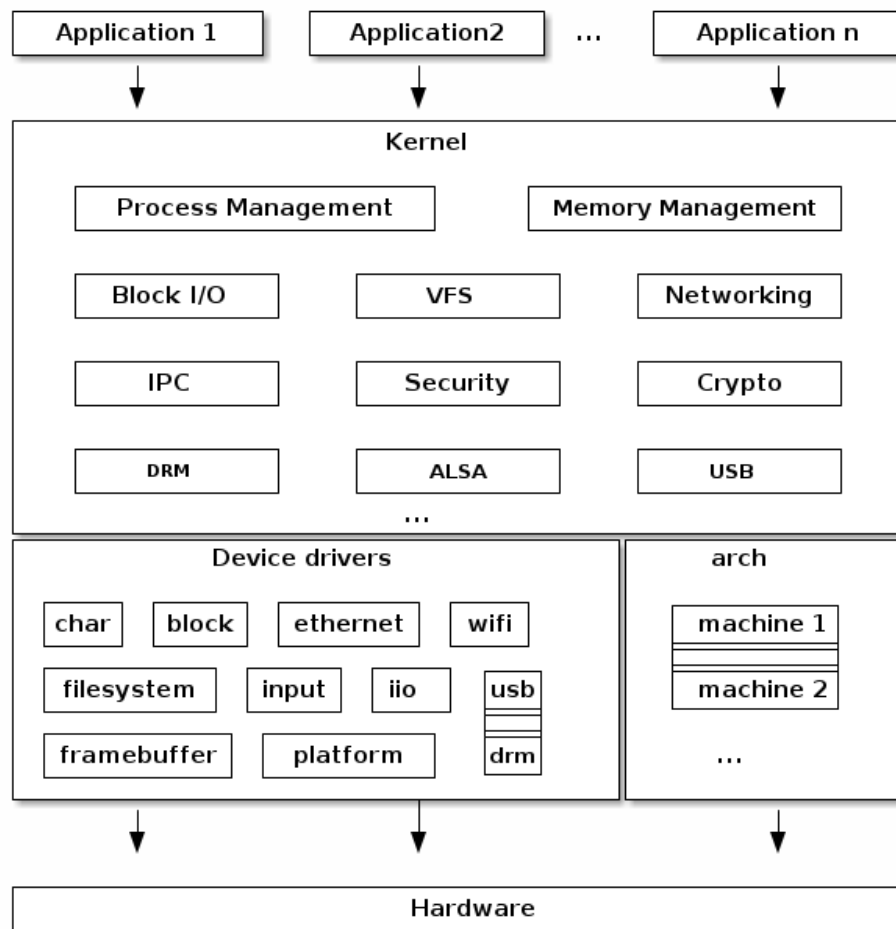
Do chi phí thấp, khả năng thay đổi tương thích với phần cứng dễ dàng nên nhúng Linux thường được sử dụng rất nhiều trong các hệ thống nhúng (embedded system). Ngày nay, Linux đã trở thành một đối thủ cạnh tranh lớn về lĩnh vực hệ điều hành trong các dòng điện thoại thông minh (smartphone), các thiết bị PDA (personal digital assistant) và là một lựa chọn thay thế cho sự độc quyền của Windows CE và Palm OS.

Theo định nghĩa của IEEE (Hiệp hội kỹ sư điện và điện tử Hoa kỳ): Hệ thống nhúng là một phần của hệ thống lớn hơn và thực hiện một số chức năng của hệ thống đó.

Đặc trưng của hệ điều hành nhúng:

- Tăng tính tin cậy (reliability)

- Tăng tính khả chuyển (portability)
- Khả năng tương thích mềm: dễ dàng nâng cấp hay thu gọn để tương thích với nền tảng hệ thống
- Cung cấp các cơ chế lập lịch (scheduler) hỗ trợ thời gian thực (Realtime OS – RTOS)



Hình 1.2: Cấu trúc hệ điều hành

1.2.2 Các thành phần trong hệ thống nhúng linux

Nhân Linux (Linux Kernel): Nhân Linux là thành phần cốt lõi của hệ điều hành Linux. Nó quản lý tài nguyên phần cứng, quản lý bộ nhớ, giao tiếp với thiết bị và cung cấp các dịch vụ hệ thống cần thiết. Nhân Linux có thể được tùy chỉnh để phù hợp với các yêu cầu của hệ thống nhúng và được cung cấp bởi dự án Linux Kernel.

Bootloader: Bootloader là chương trình đầu tiên được chạy khi hệ thống được khởi động. Nó có nhiệm vụ khởi động và tải nhân Linux vào bộ nhớ và sau đó chuyển quyền kiểm soát cho nhân Linux. Các bootloader phổ biến cho hệ thống nhúng Linux bao gồm U-Boot và GRUB.

Thư viện hỗ trợ: Hệ thống nhúng Linux cung cấp các thư viện hỗ trợ như libc, libpthread, libm và nhiều thư viện khác. Những thư viện này cung cấp các hàm, giao diện và tài nguyên hỗ trợ cho việc phát triển ứng dụng.

Công cụ phát triển: Hệ thống nhúng Linux cung cấp các công cụ phát triển mạnh mẽ. Đây là thành phần quan trọng giúp biên dịch các chương trình C tới đúng định dạng nhị phân của hệ thống đích bao gồm:

- **Compiler:** GNU C Compiler.
- **C library:** Bao gồm các header file cho phép giao tiếp với hệ điều hành.
- **Gdb:** Hỗ trợ gỡ lỗi.
- **Hệ thống Makefile**

Giao diện người dùng: Giao diện người dùng trong hệ thống nhúng Linux có thể được phát triển bằng sử dụng các framework như Qt hay GTK+. Các giao diện người dùng có thể được hiển thị trên màn hình, giao tiếp thông qua các thiết bị nhập liệu và hiển thị kết quả của ứng dụng.

1.2.3 Lập trình ngoại vi trong hệ thống nhúng linux

Trong Linux, việc lập trình các thiết bị ngoại vi có thể được thực hiện ở hai nơi khác nhau: trong không gian người dùng (user space) hoặc trong không gian nhân (kernel space).

Viết chương trình trong không gian người dùng không đòi hỏi kiến thức sâu về hệ điều hành Linux và kernel. Chương trình trong không gian người dùng sử dụng các API, thư viện hoặc giao thức cung cấp bởi kernel để tương tác với thiết bị thông qua các giao tiếp như USB, I²C, GPIO. Chương trình dạng này không có quyền truy cập trực tiếp vào phần cứng và cần sự hỗ trợ từ driver đã được cài đặt trong kernel.

Viết driver trong không gian nhân yêu cầu kiến thức về lõi hệ điều hành Linux và cách thức hoạt động của thiết bị. Driver trong không gian nhân được biên dịch thành mô-đun nhân và chạy trong không gian nhân. Chúng có quyền truy cập trực tiếp vào phần cứng và cung cấp các API để cho phép các ứng dụng trong không gian người dùng tương tác với thiết bị. Viết driver trong không gian nhân có thể đòi hỏi quyền root (hoặc sử dụng sudo) để tải và cài đặt driver vào hệ thống.

Trong nhiều trường hợp, viết chương trình trong không gian người dùng là đủ để tương tác với các thiết bị ngoại vi trong Linux. Tuy nhiên, trong một số trường hợp đặc biệt hoặc khi yêu cầu tương tác phức tạp hơn, viết driver trong không gian nhân có thể là lựa chọn tốt hơn.

1.2.4 Trình điều khiển thiết bị trong linux

Linux kernel driver là một module được nạp vào nhân linux và chạy trên không gian địa chỉ của nhân linux giúp cung cấp các chức năng mở rộng cho nhân hệ điều hành.

Device driver trong linux là một kernel module cung cấp giao diện giao tiếp với các thiết bị phần cứng như USB, PCI, I²C, SPI, ... Device driver thực hiện việc giao tiếp như gửi và nhận dữ liệu, từ thiết bị, đảm bảo cho thiết bị được hoạt động đúng mục đích đặt ra. [2]

1.2.5 Trình điều khiển thiết bị I²C

Khác với các thiết bị như PCI hay USB, thiết bị sử dụng giao tiếp I²C mặc định không được nhận biết bởi phần cứng mà thay vào đó phần mềm phải nhận biết những thiết bị nào được kết nối ở địa chỉ nào. Có một số cách để thực hiện việc này nhưng một phương pháp hiệu quả là sử dụng Device tree. [1]

Device tree là một cấu trúc dữ liệu và một hệ thống mô tả phần cứng được sử dụng trong hệ điều hành linux. Trong Device tree, các thiết bị I²C được khai báo tại bus và địa chỉ xác định. Cùng với thông tin bus và địa chỉ, Device tree còn mô tả loại driver tương thích với thiết bị. Khi trình điều khiển thiết bị được nạp, hàm probe sẽ được gọi khi trình điều khiển tương thích với thiết bị được khai báo trong Device tree.

Một Device tree cho thiết bị slave I2C có cấu trúc dạng như sau: thiết bị i2c-device được kết nối tới i2c-bus có địa chỉ 0x50. Driver tương thích là “vendor,i2c-device”:

| | |
|---|-------|
| 1 | i2c { |
|---|-------|

```
2         compatible = "i2c-bus";
3         #address-cells = <1>;
4         #size-cells = <0>;
5         i2c-device@50 {
6             compatible = "vendor,i2c-
              device";
7             reg = <0x50>;
8         }
9     }
```

Bên trong driver, sau khi hàm probe được gọi với tham số struct i2c_client tương ứng, thực hiện lưu lại biến này để thực hiện giao tiếp với thiết bị. Để đọc từ i2c device, sử dụng hàm i2c_smbus_read*, và i2c_smbus_write* cho việc ghi.

1.2.6 Giao tiếp với trình điều khiển thiết bị

Một số lựa chọn để giao tiếp với device driver của thiết bị nhúng bao gồm:

- Sử dụng Character device driver
- Sử dụng sysfs
- Kết hợp các phương pháp

Với character device driver thiết bị được xuất hiện trên không gian người dùng với một file duy nhất bên trong /dev. Các thao tác có thể được thực hiện thông qua các lời gọi từ struct file_operations như open, close, read, write, ioctl. Thông qua device file, thiết bị dạng vào/ra sẽ đơn giản hơn thông qua các lời gọi hệ thống. Một số ứng dụng sử dụng device file bao gồm: Giao tiếp với cổng serial, đọc từ ổ đĩa, ...

Khác với character device driver, ứng dụng có thể sử dụng sysfs cho việc thiết lập và quản lý cấu hình. Các file trong không gian người dùng xuất hiện dưới dạng thư mục trong /sys/kernel/. Bên trong thư mục này là các file cấu hình được tạo ra với device driver giúp chương trình trong không gian người dùng có thể tương tác với driver thông qua việc đọc ghi các file. Giao tiếp với thiết bị phần cứng sẽ dễ dàng hơn thông qua sysfs do phần cứng có nhiều tham số cần đọc và ghi và chúng được tách riêng chúng ra từng file.

Với phần cứng hỗ trợ cả cấu hình các tham số và vào ra lượng thông tin lớn, kết hợp cả character device driver và sysfs sẽ tận dụng được lợi thế của cả hai phương pháp. Các thao tác vào ra sẽ được thực hiện qua character device driver, các thao tác cấu hình sẽ được thực hiện thông qua hệ thống sysfs và thao tác ioctl của character device driver. Khi đó người dùng có thể làm quen với thiết bị thông qua sysfs và thực hiện các thao tác qua chúng. Khi lập trình, việc sử dụng character device driver sẽ đem lại hiệu quả cao hơn.

1.2.7 Lập trình giao diện trong hệ thống nhúng linux

Khi lập trình giao diện đồ họa trong hệ thống nhúng linux, việc lựa chọn công nghệ phổ biến sẽ giúp quá trình phát triển trở nên dễ dàng hơn với bộ thư viện hỗ trợ rộng lớn, hỗ trợ từ cộng đồng cũng nhiều hơn.

Qt là một framework phát triển ứng dụng đa nền tảng rất mạnh mẽ và phổ biến. Nó cung cấp các công cụ và thư viện để phát triển giao diện người dùng, bao gồm cả hỗ trợ cho các thiết bị nhúng. Lập trình viên có thể sử dụng Qt để xây dựng giao diện người dùng đẹp và tương tác trong hệ thống nhúng Linux.

Qt được sử dụng để phát triển giao diện người dùng đồ họa (GUI) và các ứng dụng đa nền tảng chạy trên tất cả các nền tảng máy tính để bàn lớn và hầu hết các nền tảng di động hoặc nhúng. Hầu hết các chương trình GUI được tạo bằng Qt đều có giao diện tự nhiên, trong trường hợp này Qt được phân loại là widget toolkit. Ngoài ra các chương trình không phải GUI cũng có thể được phát triển, chẳng hạn như các công cụ dòng lệnh và consoles cho server. Một ví dụ về một chương trình không phải GUI sử dụng Qt là khung công tác web Cutelyst.

Qt hỗ trợ các trình biên dịch khác nhau, bao gồm trình biên dịch GCC C++ và bộ Visual Studio và có hỗ trợ quốc tế hóa rộng rãi. Qt cũng cung cấp Qt Quick, bao gồm một ngôn ngữ kịch bản lệnh được gọi là QML cho phép sử dụng JavaScript hoặc nhiều ngôn ngữ khác để cung cấp logic. Với Qt Quick, việc phát triển ứng dụng nhanh chóng cho các thiết bị di động trở nên khả thi, trong khi logic vẫn có thể được viết bằng mã gốc để đạt được hiệu suất tốt nhất có thể.

Các tính năng khác bao gồm truy cập cơ sở dữ liệu SQL, phân tích cú pháp XML, phân tích cú pháp JSON, quản lý luồng và hỗ trợ mạng.

Một số lợi thế của bộ công cụ QT:

- Phát triển đa nền tảng: Qt cho phép nhà phát triển viết mã một lần và triển khai trên nhiều nền tảng, bao gồm Windows, macOS, Linux, Android và iOS.

- Qt Widgets và Qt Quick: Qt cung cấp hai phương pháp chính để xây dựng giao diện người dùng. Qt Widgets là một framework trưởng thành và mạnh mẽ để tạo ra các ứng dụng trên máy tính để bàn, trong khi Qt Quick là một framework hiện đại và khai báo để tạo ra giao diện người dùng mượt mà và linh hoạt, phù hợp cho các nền tảng di động và nhúng.
- Tín hiệu và khe cắm: Cơ chế tín hiệu và khe cắm của Qt cho phép lập trình dựa trên sự kiện, giúp kết nối các phần khác nhau của ứng dụng và phản ứng với hành động người dùng hoặc sự kiện hệ thống.
- Thư viện phong phú: Qt cung cấp một loạt thư viện và mô-đun, bao gồm hỗ trợ cho giao tiếp mạng, phân tích XML, tích hợp cơ sở dữ liệu (Qt SQL), đa phương tiện (Qt Multimedia), đồ họa 2D/3D (Qt Graphics View) và nhiều hơn nữa.
- Môi trường phát triển tích hợp (IDE): Qt cung cấp một môi trường phát triển tích hợp được gọi là Qt Creator, cung cấp các tính năng như chỉnh sửa mã, gỡ lỗi, quản lý dự án và một trình thiết kế giao diện người dùng đồ.

1.3 Các thiết bị ngoại vi

1.3.1 Máy tính nhúng Raspberry pi 3

Raspberry Pi 3 là một bo mạch nhúng mạnh mẽ và phổ biến được sử dụng rộng rãi trong các ứng dụng nhúng. Một số lợi thế khi sử dụng Raspberry Pi 3 cho các ứng dụng nhúng:

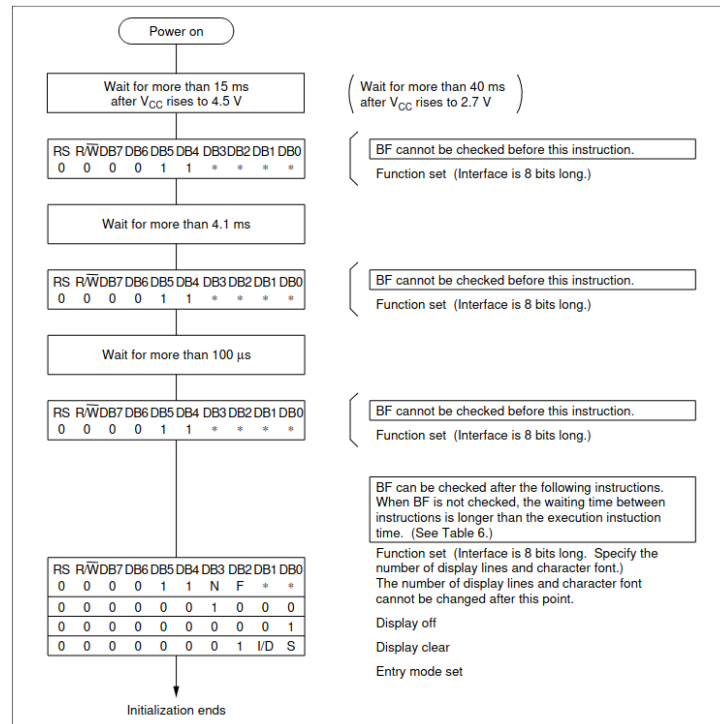
1. Hiệu suất mạnh mẽ: Raspberry Pi 3 được trang bị bộ vi xử lý ARM Cortex-A53 64-bit tốc độ 1.2GHz và RAM 1GB, mang lại khả năng xử lý nhanh và hiệu suất cao cho các ứng dụng đòi hỏi.
2. Kết nối đa dạng: Raspberry Pi 3 có sẵn các cổng kết nối như HDMI, Ethernet, USB, Bluetooth và Wi-Fi, cho phép kết nối dễ dàng với các thiết bị ngoại vi và mạng.
3. GPIO (General Purpose Input/Output): Raspberry Pi 3 có giao diện GPIO cho phép tương tác với các linh kiện ngoại vi như cảm biến, động cơ và đèn LED. Điều này cho phép bạn tạo ra các ứng dụng nhúng tùy chỉnh và điều khiển các thiết bị ngoại vi khác.

4. Hỗ trợ hệ điều hành Linux: Raspberry Pi 3 hỗ trợ nhiều hệ điều hành Linux như Raspbian, Ubuntu và các biến thể khác. Điều này giúp dễ dàng phát triển và triển khai ứng dụng nhúng dựa trên các công cụ và thư viện mạnh mẽ của Linux.
5. Cộng đồng phát triển mạnh mẽ: Raspberry Pi có một cộng đồng phát triển đông đảo và nhiều tài liệu hướng dẫn, ví dụ như tài liệu phần cứng, tài liệu lập trình và diễn đàn. Điều này giúp giảm thời gian phát triển và tìm kiếm giải pháp cho các vấn đề gặp phải.
6. Giá cả phải chăng: Raspberry Pi 3 có giá thành khá thấp so với nhiều bo mạch nhúng khác trên thị trường. Điều này giúp nó trở thành lựa chọn phổ biến cho các dự án nhúng có ngân sách hạn chế.

Như vậy, có thể thấy rằng Raspberry pi là một bo mạch nhúng mạnh mẽ và đa năng, thích hợp cho nhiều ứng dụng nhúng như Internet of Things (IoT), hệ thống điều khiển nhúng và các dự án cá nhân.

1.3.2 Mô đun màn hình LCD 1602 HD44780

Màn hình LCD1602 là màn hình ma trận điểm với 2 hàng, mỗi hàng chứa 16 ký tự kích thước 5x8 điểm. HD44780 là một IC điều khiển màn hình ma trận 5x8 hoặc 5x10 với tối đa 80 ký tự, phù hợp với màn hình LCD1602. Việc điều khiển yêu cầu tối thiểu 6 chân GPIO. Các bước khởi tạo màn hình.



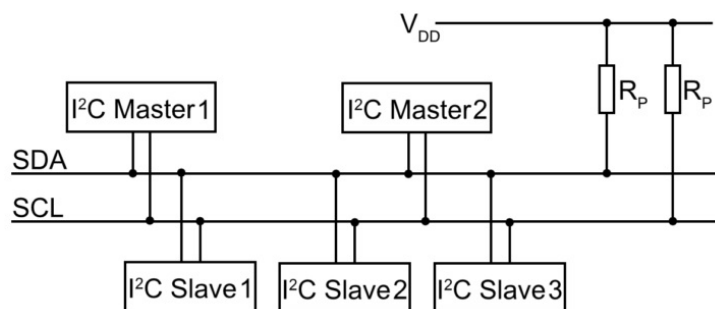
Hình 1.3: Khởi tạo HD44780

1.3.3 Mô đun I²C PCF8574

I²C là một chuẩn được phát triển bởi Philips. Nó là một giao thức 2 dây chậm với tốc độ có thể lên đến 400 kHz. Nó được sử dụng rộng rãi trong hệ thống nhúng. Một số thiết bị sử dụng biến thể không đạt được yêu cầu của I²C nên có thể được gọi bằng tên khác như TWI hoặc IIC.

SMBus (System Management Bus) dựa trên giao thức I²C được sử dụng rộng rãi trong hệ thống máy tính. Rất nhiều thiết bị I²C có thể hoạt động trên SMBus như I²C EEPROMs và một số chip theo dõi phần cứng.

Trong một bus I²C, có thể có một hay nhiều master chip được kết nối tới một hay nhiều slave chip.

Hình 1.4: I²C bus

Master chip là thiết bị kết nối tới slave. Trong nhân linux, master chip được gọi là một adapter hoặc bus. Các trình điều khiển của adapter nằm trong thư mục `con drivers/i2c/busses`.

Một algorithm chứa mã có thể sử dụng để cài đặt nhiều loại I2C adapter. Mỗi adapter driver có thể phụ thuộc vào một algorithm driver trong thư mục `con driver-s/i2c/algos` hoặc chứa mã cho riêng nó.

Một slave chip là nút thực hiện giao tiếp khi có yêu cầu của master. trong Linux, nó được gọi là client. Client driver được lưu trong thư mục nào phụ thuộc vào các chức năng mà nó cung cấp, ví dụ như trong `drivers/media/gpio` cho thiết bị mở rộng GPIO.

Để đơn giản hóa việc điều khiển một thiết bị sử dụng nhiều GPIO, module mở rộng chân sử dụng kết nối nối tiếp sẽ giúp đơn giản hóa việc kết nối và tiết kiệm số chân GPIO. PCF8574 sử dụng giao tiếp I²C có 8 địa chỉ có thể thay đổi cứng. Địa chỉ mặc định là 0x27 và có 8 chân GPIO song song.

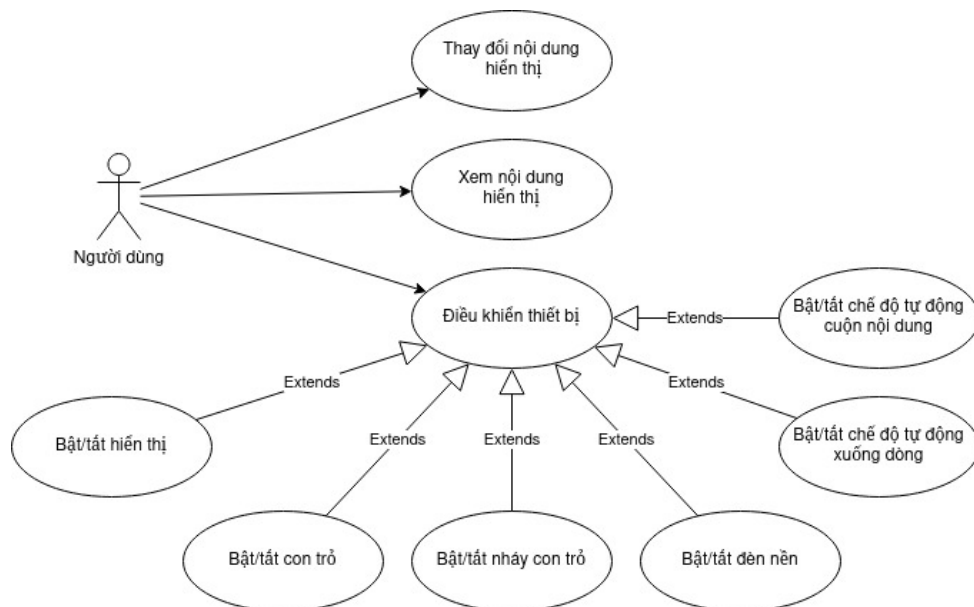
Việc đọc 8 chân GPIO có thể thực hiện bằng việc đọc 8 bit từ module PCF8574, tương tự, ghi vào 8 chân GPIO được thực hiện bằng việc ghi 8 bit tương ứng.

Chương 2: PHÂN TÍCH - THIẾT KẾ

2.1 Mô tả bài toán

2.1.1 Yêu cầu chức năng

Màn hình LCD1602 có nhiều chức năng phục vụ việc hiển thị văn bản. Vì vậy, thiết bị có một số yêu cầu chức năng trong hình 2.1 bao gồm các chức năng điều khiển con trỏ, chức năng điều khiển các chức năng hiển thị và nội dung hiển thị.



Hình 2.1: Biểu đồ usecase

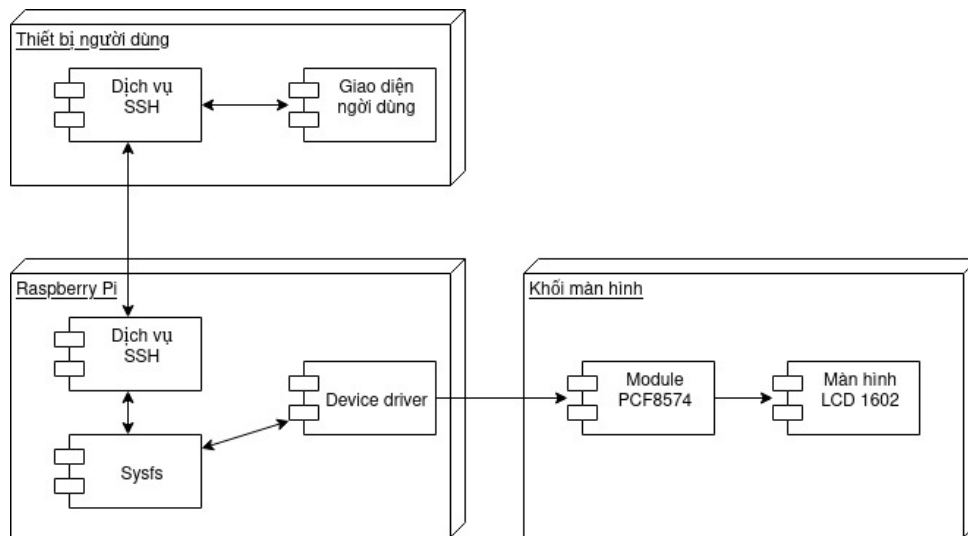
2.1.2 Yêu cầu phi chức năng

Hệ thống cần đạt được một số yêu cầu phi chức năng:

- Hệ thống cần đáp ứng nhanh, xử lý nhanh tránh trường hợp người dùng chờ quá lâu sau khi nhập.
- Số ký tự tối đa được nhập dưới 100 ký tự, nếu vượt quá khoảng hiển thị của màn hình LCD1602, thực hiện thuật toán cuộn trang để hiển thị các ký tự.
- Thông tin hiển thị rõ ràng, chính xác.

2.2 Sơ đồ khối

Phần cứng bao gồm thiết bị và máy tính điều khiển. Trong đó thiết bị bao gồm 2 khối màn hình và mạch Raspberry Pi 3. Trong mỗi thành phần lại bao gồm các mô đun phần mềm và phần cứng khác nhau (Hình 2.2). Khối máy tính điều khiển của người dùng thực hiện chạy chương trình điều khiển và giao tiếp với mạch Raspberry Pi qua dịch vụ SSH. Tại Raspberry Pi, dịch vụ SSH chạy lệnh tương tác với hệ thống Sysfs của trình điều khiển thiết bị màn hình để điều khiển hoạt động của màn hình. Khối màn hình bao gồm mô đun PCF8574 và màn hình LCD1602 sẽ hoạt động theo sự điều khiển của trình điều khiển trên mạch Raspberry Pi.

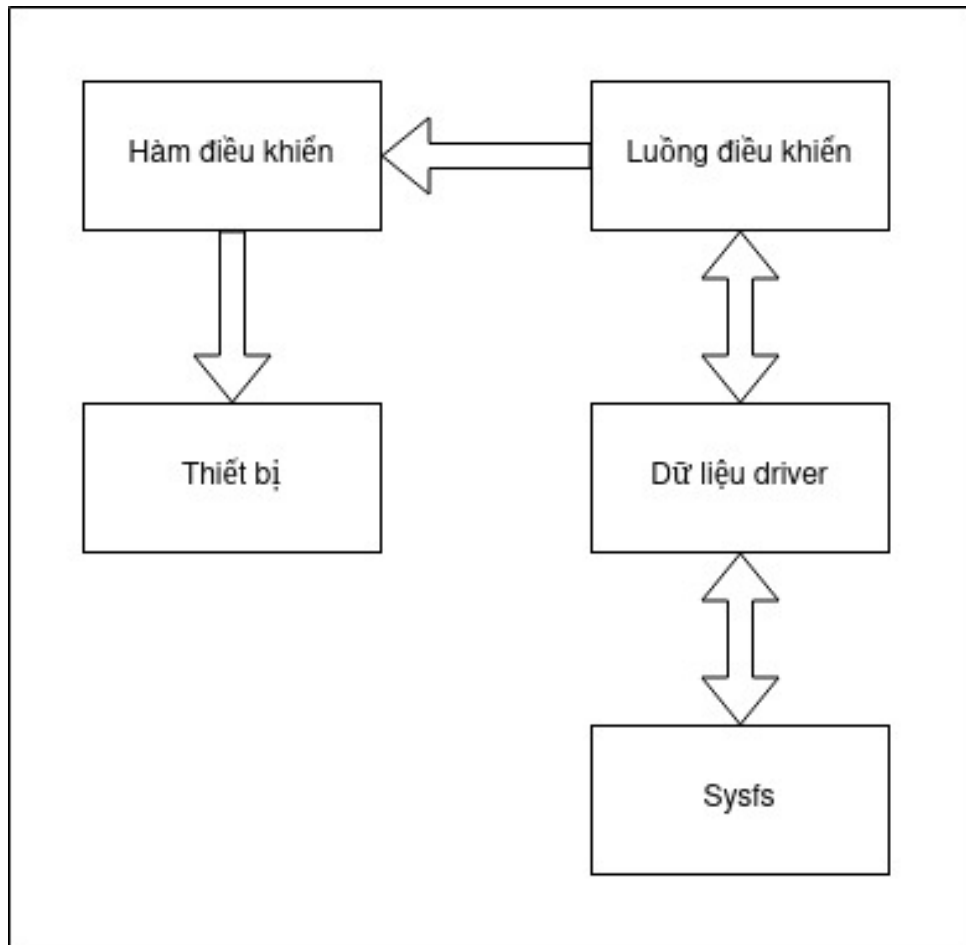


Hình 2.2: Sơ đồ khối hệ thống

2.3 Trình điều khiển thiết bị

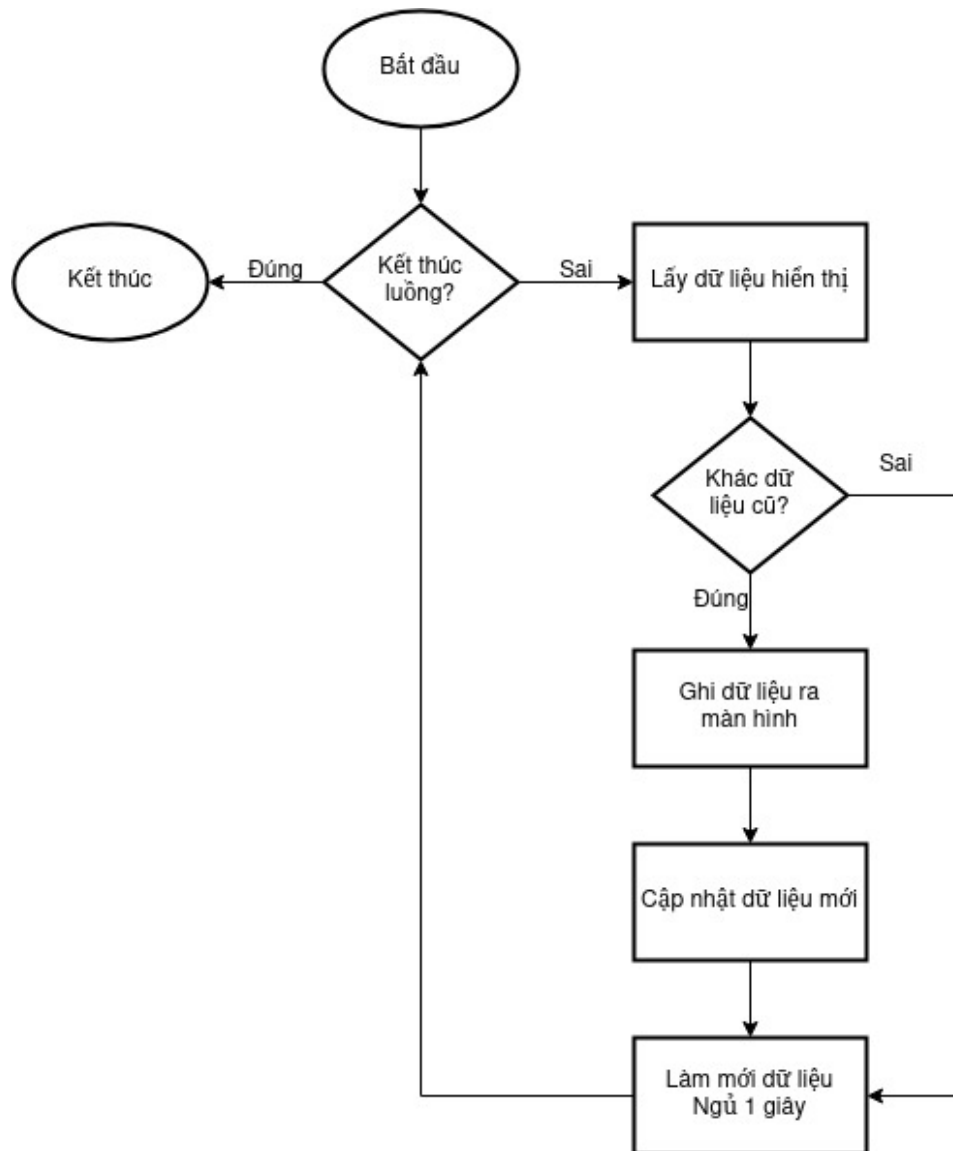
Trình điều khiển thiết bị (Driver) là thành phần phần mềm chính của thiết bị. Hoạt động của mô đun này được mô tả trong hình 2.3. Luồng điều khiển liên tục chạy và thay đổi trạng thái màn hình tùy theo dữ liệu người dùng nhập như khi người dùng thay đổi nội dung, thay đổi trạng thái màn hình hay khi thực hiện cuộn văn bản.

Dữ liệu driver được cập nhật bởi người dùng thông qua hệ thống sysfs. Dữ liệu trạng thái sau khi cập nhật sẽ được cập nhật bởi luồng điều khiển và hiển thị trên màn hình LCD1602.



Hình 2.3: Sơ đồ khối của trình điều khiển thiết bị

Luồng điều khiển chính được mô tả trong hình 2.4. Do luồng trong không gian nhân không thể kết thúc một cách tự động hoặc kết thúc từ luồng khác mà chỉ có thể gửi tín hiệu kết thúc đến nó nên các câu lệnh phải được đặt bên trong vòng lặp kiểm tra trạng thái kết thúc. Bên trong vòng lặp, thực hiện lấy dữ liệu hiển thị và gọi hàm hiển thị tương ứng nếu dữ liệu có sự thay đổi. Sau đó gọi hàm làm mới dữ liệu và tạm dừng luồng trong 1 giây để chuyển sang vòng lặp tiếp theo.



Hình 2.4: Hoạt động của luồng điều khiển chính

Người dùng thực hiện thao tác với màn hình qua hệ thống file ảo của sysfs bằng việc đọc hoặc ghi vào chúng. Thao tác này có thể thực hiện trực tiếp trên shell nên có thể lập trình giao tiếp qua các dịch vụ từ xa như Secure Shell. Các file nằm trong đường dẫn `/sys/kernel/pcf_lcd`.

Bảng 2.1: Danh sách file sysfs

| Tên file | Chức năng |
|--------------|-----------------------------------|
| content | đọc hoặc ghi nội dung hiển thị |
| line1 | đọc nội dung dòng 1 |
| line2 | đọc nội dung dòng 2 |
| scroll | bật/tắt chế độ cuộn văn bản |
| auto_newline | bật/tắt chế độ tự động xuống dòng |
| backlight | bật/tắt đèn nền |
| cursor | bật/tắt con trỏ |
| blink | bật/tắt nháy con trỏ |
| display | bật/tắt hiển thị |

2.4 Giao diện đồ họa

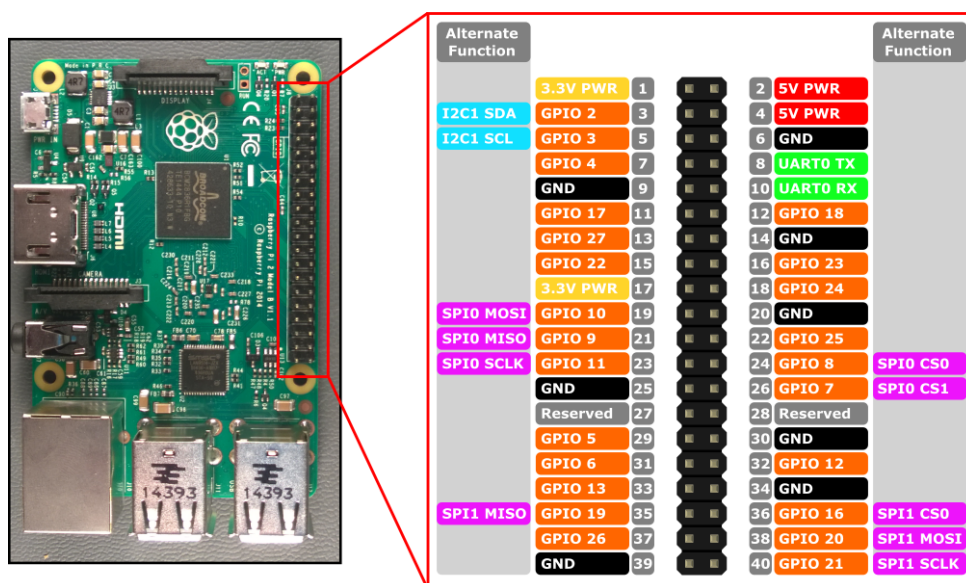
Giao diện đồ họa được sử dụng nhằm điều khiển hoạt động của màn hình LCD1602. Các thành phần trong giao diện tương ứng với các chức năng của driver bao gồm:

- Các chức năng bật/tắt: Sử dụng checkbox.
- Thay đổi nội dung: Sử dụng một khung nhập văn bản và nút nhấn Submit.
- Thực hiện chức năng kết nối: Sử dụng một dòng input để nhập Hostname và nút nhấn Login, thực hiện kiểm tra đăng nhập vào thiết bị với người dùng Root qua kết nối SSH.

Chương 3: XÂY DỰNG HỆ THỐNG

3.1 Triển khai phần cứng

Phần cứng máy tính nhúng được sử dụng là mạch phát triển Raspberry Pi 3 với sơ đồ ra chân như hình 3.1.



Hình 3.1: Sơ đồ ra chân các ngoại vi của Raspberry Pi

Mô đun PCF8574 cho màn hình LCD1602 được kết nối tới Raspberry Pi qua ngoại vi I2C1 duy nhất với 4 dây nối tương ứng trong bảng 3.1.

Bảng 3.1: Sơ đồ nối dây giữa Raspberry Pi và mô đun PCF8574

| Chân Raspberyr Pi | Chân PCF8574 |
|-------------------|--------------|
| 5V | VCC |
| GND | GND |
| GPIO3 | SCL |
| GPIO2 | SDA |

3.2 Triển khai driver

Driver được triển khai trên nhiều file. File `lcd_data.h` chứa dữ liệu cho hoạt động điều khiển của driver (nội dung hiển thị, các dữ liệu điều khiển hiển thị, cuộn trang, ...). File `lcd_sysfs.h` giúp xuất dữ liệu driver và thay đổi dữ liệu driver theo yêu cầu của người dùng sử dụng hệ thống `sysfs`. File `lcd_i2c_client.h` chứa các hàm tương tác với thiết bị như điều khiển đèn nền, điều khiển cuộn trang hay thay đổi dữ liệu. File `pcf_lcd.c` là file chính của driver, thực hiện khai báo driver, khởi tạo luồng điều khiển, các hàm `probe`, `remove`.

Trong luồng điều khiển hoạt động của driver, thực hiện liên tục lấy dữ liệu hiển thị cho màn hình. Dữ liệu này có thể thay đổi sau mỗi lần lặp do dữ liệu mới được cập nhật hoặc sau khi làm mới dữ liệu, ở chế độ cuộn văn bản, dữ liệu hiển thị bị dịch chuyển sang ký tự khác. Sơ đồ luồng điều khiển được thể hiện trong hình 2.4.

Khi khai báo driver, thực hiện tạo đối tượng struct `i2c_driver` trong đó bao gồm các trường `probe_new` cho việc khởi tạo thiết bị, `.remove` cho việc gỡ bỏ thiết bị, `.id_table` là đối tượng struct `i2c_device_id` cho thiết bị, `.of_match_table` là struct `of_device_id` chứa thông tin tên thiết bị tương thích với driver được lưu trong device tree.

```
66 static int lcd_probe(struct i2c_client *client) {
67     pr_info("Probed %x", client->addr);
68     lcd_client = client;
69     lcd_init(lcd_client);
70     lcd_sysfs_init();
71     task_struct = kthread_run(work_loop, NULL, "pcf lcd
        work loop");
72     set_content("LCD\n Hello!");
73     return 0;
74     return 0;
75 }
76 static int lcd_remove(struct i2c_client *client) {
77     set_content("LCD\n Goodbye!");
78     refresh();
79     msleep(refresh_timeout * 2);
80     kthread_stop(task_struct);
```

```
81     lcd_sysfs_deinit();
82
83     pr_info("Removed\n");
84     return 0;
85 }
86
87 static const struct of_device_id my_driver_ids[] = {
88     {.compatible = "pcf,lcd"},
89     {} ,
90 };
91 MODULE_DEVICE_TABLE(of, my_driver_ids);
92
93 static const struct i2c_device_id lcd_device_id[] = {
94     {DEVICE_NAME, 0},
95     {} ,
96 };
97
98 MODULE_DEVICE_TABLE(i2c, lcd_device_id);
99
100 static struct i2c_driver lcd_driver = {
101     .probe_new = lcd_probe,
102     .remove = lcd_remove,
103     .id_table = lcd_device_id,
104     .driver = {.name = DEVICE_NAME,
105               .owner = THIS_MODULE,
106               .of_match_table = my_driver_ids},
107 };
108
109 module_i2c_driver(lcd_driver);
```

Việc làm mới dữ liệu được thực hiện liên tục để phục vụ mục đích hiển thị dữ liệu mới từ người dùng hoặc phục vụ mục đích cuộn nội dung. Hàm làm mới dữ liệu được khai báo bên trong file chứa dữ liệu.

```
8 char __content[CONTENT_MAX_LEN + 2] = {0};
9 char __display_lines[100][101];
```

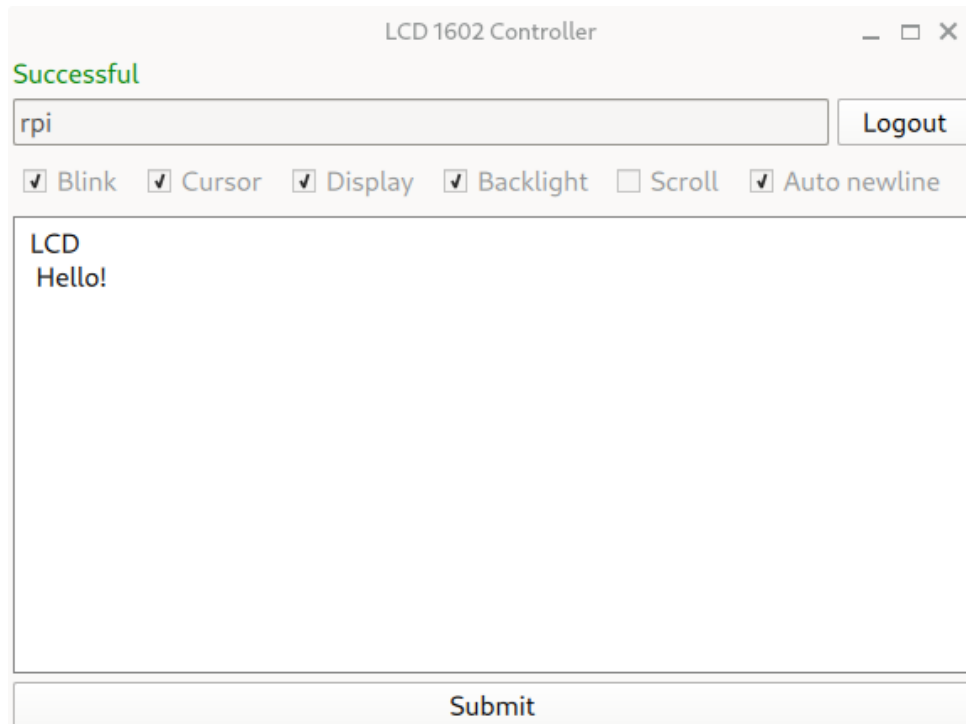
```
10 uint __nline;
11 uint __l1_idx = -1;
12 uint __l2_idx = -1;
13 uint __l1_off = -1;
14 uint __l2_off = -1;
15 uint __backlight = 1;
16 uint __cursor = 1;
17 uint __blink = 1;
18 uint __display = 1;
19 uint __auto_newline = 1;
20 uint __scroll = 0;
21 uint __scroll_start = 0;
22 uint __l1_scroll_done = 0;
23 uint __l2_scroll_done = 0;
```

```
143
144 void inc_offset(void) {
145     if (strlen(__display_lines[__l1_idx]) - __l1_off >
146         16) {
147         __l1_off += 1;
148     } else {
149         __l1_off = 0;
150         __l1_scroll_done = 1;
151     }
152     if (strlen(__display_lines[__l2_idx]) - __l2_off >
153         16) {
154         __l2_off += 1;
155     } else {
156         __l2_off = 0;
157         __l2_scroll_done = 1;
158     }
159 }
160
161 void refresh(void) {
162     if (get_scroll()) {
```

```
161     if (__scroll_start) {
162         if (__l1_scroll_done && __l2_scroll_done) {
163             __l1_idx = (__l1_idx + 2) % __nline;
164             __l2_idx = (__l2_idx + 2) % __nline;
165             __l1_off = 0;
166             __l2_off = 0;
167             __l1_scroll_done = 0;
168             __l2_scroll_done = 0;
169         } else {
170             inc_offset();
171         }
172     } else {
173         __scroll_start = 1;
174     }
175 }
176 }
```

3.3 Triển khai giao diện đồ họa

Giao diện đồ họa được thực hiện thông qua bộ công cụ QT sử dụng ngôn ngữ QML và Rust. Kết nối SSH được thực hiện thông qua phần mềm của hệ thống. Với mỗi thao tác như nhấn vào checkbox hay thay đổi nội dung hiển thị, màn hình sẽ thực hiện thao tác tương ứng.

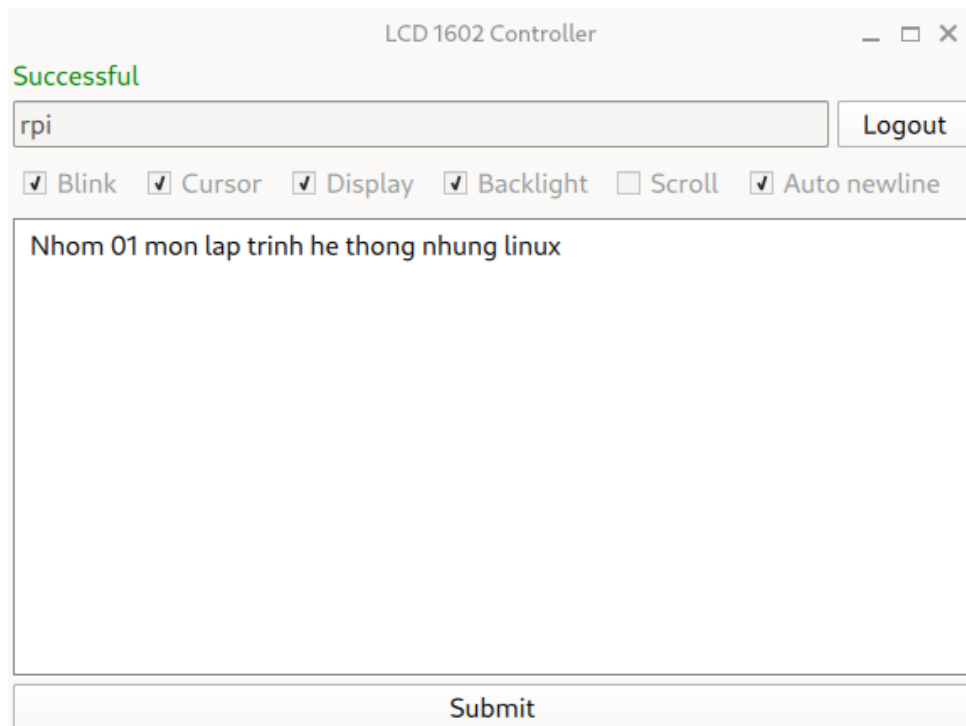


The screenshot shows a web browser window titled "LCD 1602 Controller". At the top left, the word "Successful" is displayed in green. Below it is a text input field containing the username "rpi". To the right of the input field is a "Logout" button. Below the input field is a row of checkboxes: ☒ Blink, ☒ Cursor, ☒ Display, ☒ Backlight, ☐ Scroll, and ☒ Auto newline. Below these checkboxes is a large rectangular area representing the LCD display, which shows the text "LCD" on the first line and "Hello!" on the second line. At the bottom of the interface is a "Submit" button.

Hình 3.2: Giao diện đồ họa

3.4 Thử nghiệm thiết bị

Sau khi thực hiện kiểm tra kết nối bằng cách nhấn login, thực hiện thay đổi nội dung hiển thị và nhấn submit trên giao diện (Hình 3.3), màn hình thiết bị sẽ thực hiện hiển thị nội dung với chế độ tương ứng (Hình 3.4).



Hình 3.3: Thực hiện đổi nội dung hiển thị



Hình 3.4: Màn hình hiển thị sau khi đổi nội dung

3.5 Nhận xét

Hệ thống đã đảm bảo đầy đủ theo các chức năng như đề ra trong đề tài. Hệ thống đã được xây dựng với các thiết bị phần cứng và phần mềm đúng như yêu cầu môn học đưa ra. Nội dung hiển thị chính xác và đầy đủ.

3.6 Hạn chế của thiết bị

Hệ thống đã đạt được mong muốn đề ra nhưng cũng không tránh khỏi những sai sót. Do hệ thống được phát triển ở mức thực nghiệm nên thiết kế sản phẩm có sơ sài về mặt kết nối các phần cứng. Đôi khi kết nối chập chờn dẫn đến sai sót trong hiển thị.

Chương 4: KẾT LUẬN

4.1 Kết quả đạt được

Qua quá trình nghiên cứu các thuật toán, kiến thức liên quan đến linux, trình điều khiển thiết bị, nhóm chúng em đã đạt được một số thành tựu:

- Tìm hiểu giao thức, điều khiển thành công việc đọc, ghi dữ liệu qua giao tiếp I²C.
- Tìm hiểu phương pháp phát triển trình điều khiển thiết bị cho thiết bị I²C trên môi trường linux.
- Hiểu phương pháp lập trình nhúng trên Linux, biết thêm phương pháp lập trình giao diện.

4.2 Phương hướng phát triển và đề xuất

Bên cạnh những công việc đã đạt được, nhóm cũng có các phương hướng mở rộng đề tài như sau:

- Thực hiện hiển thị nội dung thời gian thực trên giao diện đồ họa.
- Tối ưu thuật toán để nội dung hiển thị được dễ nhìn hơn.
- Thêm các chức năng hiển thị các thông số khác của kernel.
- Chỉnh sửa, thêm các thành phần tối ưu cho mục đích truyền nhận dữ liệu.

TÀI LIỆU THAM KHẢO

- [1] Linux kernel document. Instantiate i2c devices. Truy cập lần cuối 20/06/2023.
- [2] SLR. Linux device driver tutorial. Truy cập lần cuối 20/06/2023.
- [3] ThS. Lê Thị Hồng Vân and TS. Phạm Văn Hương. *Giáo trình lập trình hệ thống nhúng*.

PHỤ LỤC: MÃ NGUỒN CHƯƠNG TRÌNH

Mã nguồn chương trình được đăng tải trực tuyến tại đường dẫn: <https://github.com/dungph/driver-lab>