|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  **VIỆN ĐIỆN TỬ - VIỄN THÔNG**  logo_128  **MÔN HỌC: Lập Trình Nâng Cao**  **Giảng Viên Hướng dẫn: Vũ Song Tùng**  **Đề Tài:**  **Máy soát vé lên, xuống tàu điện**  **Sinh viên thực hiện:**   |  |  | | --- | --- | | Nguyễn Nhật Bằng | 20172422 | | Bùi Trung Kiên | 20172638 |   Hà Nội, 12/2021 |

# Lời mở đầu

Trong nhiều năm trở lại đây, sự phát triển vượt bậc của khoa học kỹ thuật đặc biệt là trong các lĩnh vực truyền thông công nghiệp và công nghệ phần mềm đã mang lại rất nhiều giải pháp, khả năng mới trong nhiều lĩnh vực. Theo thời gian, các hệ thống tự động cũng lần lượt ra đời và mang lại rất nhiều tiện ích trong các lĩnh vực khác nhau như hàng không, sản xuất công nghiệp, vận tải… Với khả năng ứng dụng lớn trong nhiều lĩnh vực như vậy, việc xây dựng một hệ thống soát vé lên xuống tàu điện là hết sức cần thiết.

Từ vấn đề nêu trên, dựa vào việc tìm hiểu cở sở hạ tầng mạng tại Việt Nam cũng như trên thế giới, hiểu được nguyên lý soát vé lên xuống tàu điện, chúng em đã lên ý tưởng cho việc xây dựng một mô hình hệ thống tự động soát vé. Một hệ thống tự động soát vé lên xuống tàu điện với các chức năng đóng/mở barie tự động, trừ tiền, thu/trả vé, nối chuyến… chắc chắn sẽ rất hữu ích và phù hợp trong môi trường nghiên cứu và học tập của sinh viên. Hơn nữa, hiện nay mỗi cá nhân đều sử dụng máy tính có kết nối internet làm phương tiện làm việc, học tập, nghiên cứu do đó việc thiết lập được một hệ thống như trên là vô cùng đơn giản

[Lời mở đầu 2](#_Toc95852963)

[Chương I. Chỉ Tiêu Kỹ Thuật 6](#_Toc95852964)

[Chương II. Mô hình hóa hệ thống 7](#_Toc95852965)

[1. Sơ đồ Use Case 7](#_Toc95852966)

[2. Sơ đồ tuần tự 7](#_Toc95852967)

[2.1. Lên Tàu 7](#_Toc95852968)

[2.2. Xuống Tàu 8](#_Toc95852969)

[3. Scope Model 8](#_Toc95852970)

[Chương III. Đồng thiết kế cứng mềm 10](#_Toc95852971)

[1. Sơ đồ khối hệ thống 10](#_Toc95852972)

[2. Lựa chọn giải pháp phần cứng 10](#_Toc95852973)

[3. Tính toán thiết kế khối nguồn 13](#_Toc95852974)

[4. Lựa chọn giải pháp phần mềm 14](#_Toc95852975)

[4.1. Hệ Điều Hành: 14](#_Toc95852976)

[4.2. Web Client: 15](#_Toc95852977)

[Chương IV. Tổng hợp, cấu hình, biên dịch hệ điều hành 17](#_Toc95852978)

[Chương VI. Phát triển phần mềm 24](#_Toc95852979)

[1. Web-client mqtt 24](#_Toc95852980)

[2. Esp-client mqtt 27](#_Toc95852981)

[3. App hỗ trợ tra cứu thông tin người dùng 28](#_Toc95852982)

[Chương VII. Triển khai, mô phỏng hệ thống 31](#_Toc95852983)

[Chương VIII. Phân tích, đánh giá kết quả 34](#_Toc95852984)

[Hình 1: Sơ đồ Use Case hệ thống 7](#_Toc95852985)

[Hình 2: Sơ đồ trình tự xử lý khi khách lên tàu 8](#_Toc95852986)

[Hình 3: Sơ đồ trình tự xử lý khi khách xuống tàu 8](#_Toc95852987)

[Hình 4: Sơ đồ khối hệ thống soát vé 10](#_Toc95852988)

[Hình 5: Lựa chọn giải pháp phần cứng cho từng khối 10](#_Toc95852989)

[Hình 6: Adapter 5V 1A 13](#_Toc95852990)

[Hình 7: Khối nguồn nuôi Esp8266 12f 14](#_Toc95852991)

[Hình 8: Các mức độ ưu tiên trong phiên bản cmsis-v1 17](#_Toc95852992)

[Hình 9: Lịch trình của hệ thống 18](#_Toc95852993)

[Hình 10:Sơ đồ kết nối chân của GM65 và esp8266 12f 18](#_Toc95852994)

[Hình 11: Xử lý truyền dữ liệu và đọc dữ liệu để giao tiếp với GM65 19](#_Toc95852995)

[Hình 12: Kết quả 19](#_Toc95852996)

[Hình 13: Giao tiếp STM32 với ESP8266 12f 20](#_Toc95852997)

[Hình 14: Gửi mã tới Stm32f103 21](#_Toc95852998)

[Hình 15: Mã nhận được từ esp 21](#_Toc95852999)

[Hình 16: Kết quả mô phỏng 22](#_Toc95853000)

[Hình 17: Kết quả mô phỏng 23](#_Toc95853001)

[Hình 18: Kết quả mô phỏng 23](#_Toc95853002)

[Hình 19: Sơ đồ mô tả các tác vụ của web-client 24](#_Toc95853003)

[Hình 20: Giao diện web-client 25](#_Toc95853004)

[Hình 21: Tính toán chi phí hành trình 26](#_Toc95853005)

[Hình 22: Thông tin trao đổi giữa máy soát vé và web-client 26](#_Toc95853006)

[Hình 23: Gửi tín hiệu cần xử lý lên web-client 27](#_Toc95853007)

[Hình 24: Kết quả trả về 28](#_Toc95853008)

[Hình 25: Giao diện đăng nhập 28](#_Toc95853009)

[Hình 26: Màn hình chính 29](#_Toc95853010)

[Hình 27: Màn hình tra cứu thông tin 29](#_Toc95853011)

[Hình 28: Màn hình lịch sử giao dịch 30](#_Toc95853012)

[Hình 29: Sơ đồ mạch mô phỏng 31](#_Toc95853013)

[Hình 30: Mạch mô phỏng kia chua có tín hiệu vào 32](#_Toc95853014)

[Hình 31: Khi có tín hiệu vào 33](#_Toc95853015)

# Chương I. Chỉ Tiêu Kỹ Thuật

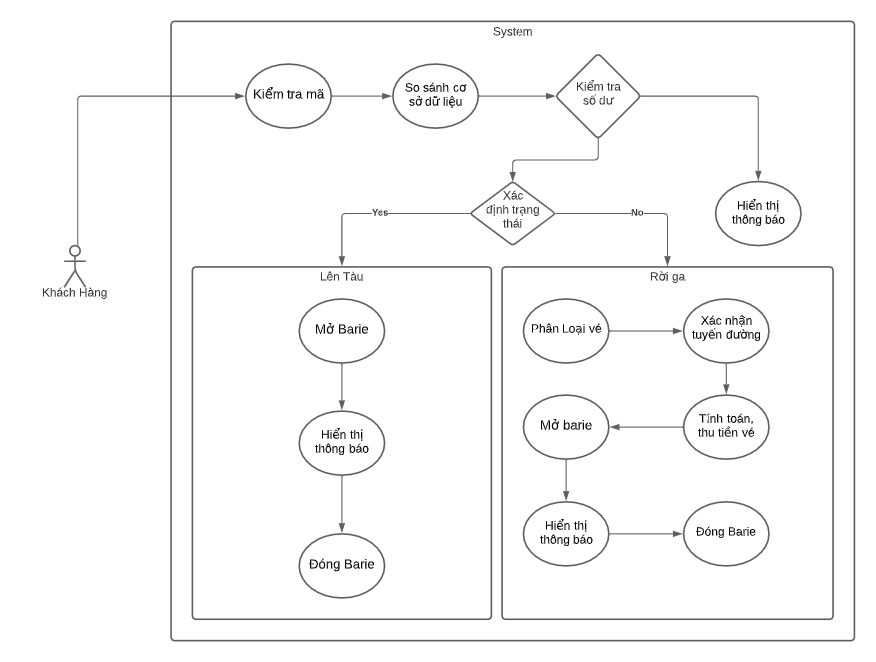
Với các yêu cầu Đóng/mở barie tự động, trừ tiền, thu/trả vé, nối chuyến, phân loại vé người lớn, trẻ nhỏ... Ta có thể xác định được các yếu tố sau đây về hệ thống khi áp dụng tại Việt Nam:

* Khi có khách hàng sử dụng hệ thống:
* Khi khách hàng quét vé lên tàu, Hệ thống sẽ quét mã -> Kiểm tra số dư khả dụng -> mở barie -> xác nhận điểm đi -> Một màn LCD hiển thị thông báo -> đóng barie.
* Khi khách hàng quét vé rời ga, Hệ thống sẽ quét mã -> Phân loại vé người lớn/ trẻ em -> kiểm tra điểm đi -> Xác nhận điểm đến -> Tính toán giá vé -> Thu tiền vé -> Một màn LCD hiển thị thông báo -> đóng barie.
* Phía server sẽ tiến hành nhận và xử lý mã các client gửi về và gửi trả tín hiệu phản hồi cho các client.
* Đầu vào:
* Mã vé (Một định dạng mã, kí hiệu dùng để nhận diện).
* Một ngắt được tạo ra ở tất cả các cổng khi có mã được quét.
* Một thông báo/ lệnh được gửi tới từng cổng.
* Đầu ra:
* Barie mở.
* Thông báo được hiển thị trên màn hình LCD.
* Barie đóng.
* Chức năng hệ thống:
* Hệ thống sử dụng module quét mã khi khách hàng quét thẻ, vé.
* Hệ thống mở barie khi đã kiểm tra số dư khả dụng hoặc xác nhận thanh toán từ khách hàng, màn hình LCD hiển thị thống báo, và barie đóng khi khách đã đi qua barie.
* Sử dụng CSDL chung cho toàn bộ hệ thống.
* Chỉ số thiết kế:
* Sự thất thoát năng lượng: Đảm bảo duy trì mức thấp.
* Thời hạn xử lý: Bộ đếm thời gian cần phải được đặt để bất cứ khi nào khách hàng sử dụng hệ thống, hệ thống cần phải đưa ra phản hồi cho khách hàng trong vòng vài giây để đóng/ mở barie, thu và hủy vé tuyến và giao tiếp qua màn hình LCD với khách hàng.
* Chi phí kỹ thuật: Chi phí cho việc kiểm tra - chỉnh sửa - gỡ lỗi phần cứng và phần mềm
* Sự an toàn: Các đảm bảo an toàn cho hệ thống cần phải được đảm bảo, tránh gây ra các sự cố cho người dùng. Có tác động xử lý kịp thời khi sự cố xảy ra.

# Chương II. Mô hình hóa hệ thống

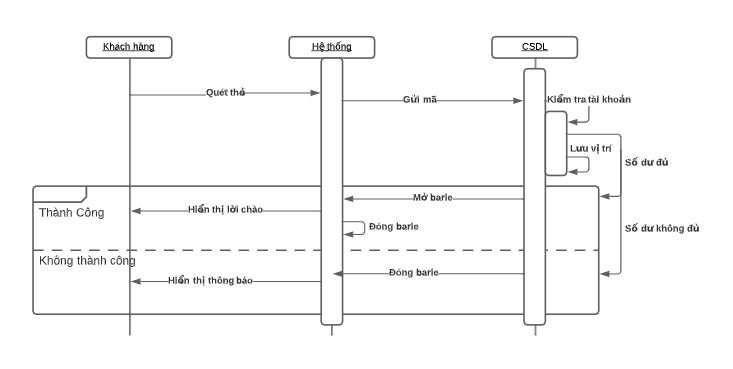
Từ các chỉ tiêu kỹ thuật đã được nêu ra ở Chương I của báo cáo này, sử dụng công cụ thiết kế hệ thống UML để mô hình hóa 1 cách khái quát lại các nhiệm vụ, chức năng và trình tự tiến hành mà hệ thống cần thực hiện.

1. Sơ đồ Use Case



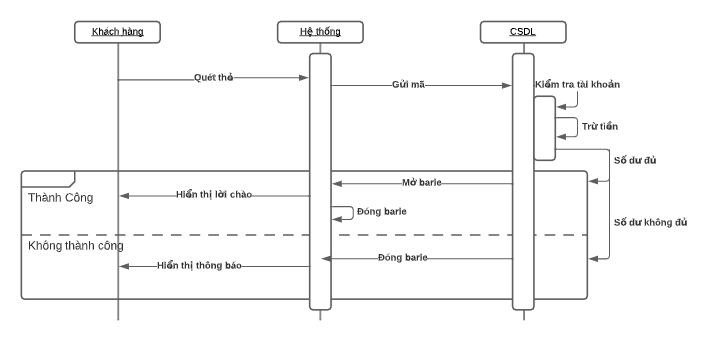
Hình 1: Sơ đồ Use Case hệ thống

1. Sơ đồ tuần tự
   1. Lên Tàu



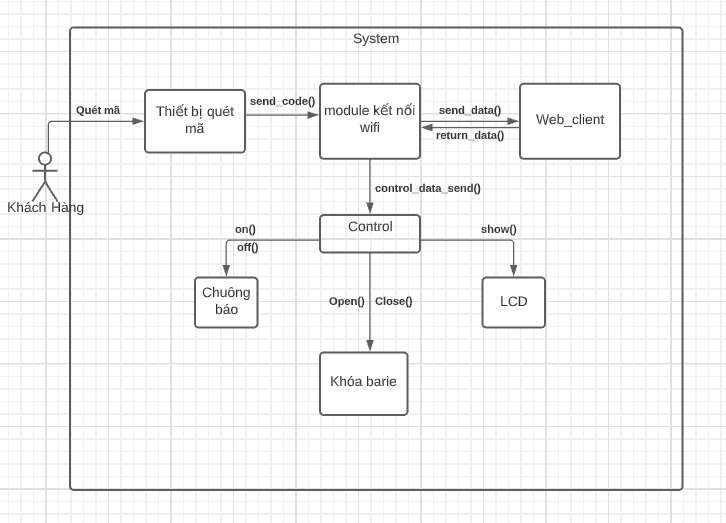
Hình 2: Sơ đồ trình tự xử lý khi khách lên tàu

* 1. Xuống Tàu



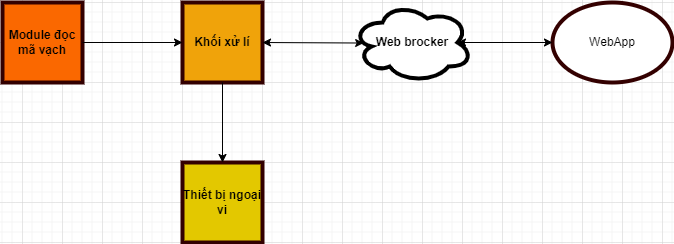
Hình 3: Sơ đồ trình tự xử lý khi khách xuống tàu

1. Scope Model



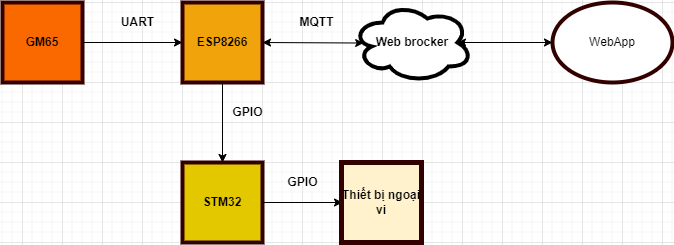
# Chương III. Đồng thiết kế cứng mềm

1. Sơ đồ khối hệ thống



Hình 4: Sơ đồ khối hệ thống soát vé

1. Lựa chọn giải pháp phần cứng



Hình 5: Lựa chọn giải pháp phần cứng cho từng khối

Chi tiết các linh kiện:

**Module đọc mã vạch GM65**: là module đọc mã vạch 2 chiều hiệu suất cao. Nó có thể đọc mã vạch 1D, 2D dễ dàng với tốc độ cao. Module cũng quét nhanh đối với các mã vạch tuyến tính, ngay cả các mã vạch trên nhãn dán, màn hình.

Thông số kĩ thuật:

|  |
| --- |
| Điện áp sử dụng: 5VDC |
| Điện áp giap tiếp: TTL 3.3~5VDC. |
| Current: 120mA during scanning / 30mA during standby |
| Interface: USB, UART (HID or VSP) |
| Optical system: CMOS |
| Capture light source: 617nm LED |
| Lighting source: 6500K LED |
| Reading angle:   * rotating 360 °   + Deflection ± 60 °   + Tilt ± 65° |
| Scanning angle: 34 ° (horizontal) 26 ° (vertical) |
| Minimum contrast: 30% |
| Resolution: ≥ 0. 1mm (4mil) |
| Ambient light: Ambient light: 0~86,000 lux |

Bảng 1: Thông số kỹ thuật GM65

**STM32f103**: là dòng chip 32bit của hãng STMicrochip sử dụng công nghệ lõi ARM Cortex mạnh mẽ, hiệu năng tốt nhưng vẫn giữ được giá thành rẻ. Phù hợp với đa số các công ty hiện nay

Các lý do nên chọn STM32f103 đó là:

* Tốc độ xử lý cao, ngoại vi hỗ trợ rất nhiều, dòng chip phân khúc thâp là STM32F0x cũng có thể hoạt động lên tới 48Mhz, 64kB Flash, 16kB RAM, 8 bộ Timer 16 bit, 1 bộ Timer 32 bit, 10 bộ ADC 12 bit, 8 bộ USART, 2 bộ SPI, 2 bộ I2C.
* Giá thành rẻ nhưng hiệu quả đem lại lớn.
* Học lập trình STM32 rất dễ dàng do cộng đồng hỗ trợ nhiều.
* Công cụ lập trình đều Free và đầy đủ tài liệu hỗ trợ

Thông số kĩ thuật:

|  |
| --- |
| Vi điều khiển chính: STM32F103RCT6 ARM Cortex-M3 |
| Nguồn sử dụng: 5VDC từ cổng Mini USB hoặc chân GPIO. |
| Tích hợp Led, Button. |
| Tích hợp cổng USB. |
| Tích hợp bộ nhớ Flash. |
| Tích hợp cổng nạp chuẩn Jtag. |
| Tích hợp khe cắm mạch RF NRF24L01+, UART, LCD,... |

Bảng 2: Thông số kỹ thuật STM32f103

**ESP8266 12f**: Mạch thu phát Wifi SoC ESP8266 ESP-12F Ai-Thinker có kích thước nhỏ gọn, ra chân đầy đủ của IC ESP8266, mạch được thiết kế và gia công chất lượng tốt với vỏ bọc kim loại chống nhiễu và anten Wifi PCB tích hợp cho khoảng các truyền xa và ổn định.

Thông số kĩ thuật:

|  |
| --- |
| Điện áp sử dụng: 3.0V~3.6V(Optimal 3.3V) |
| Working current: ≈70mA(170mA MAX), standby＜200uA |
| Deep sleep holding current 10 uA, shutdown current of less than 5 uA |
| 30 pins (10 GPIO, every GPIO can be PWM, I2C, 1-wire) |
| SRAM size: 36 KB |
| ROM size: 32Mbit (SPI External Flash) |
| Antena on PCB |
| Transmission data rate: 110-460800bps |
| 10 bit precision ADC pinout on board (0~1V) |
| WiFi @ 2.4 GHz, supports WPA / WPA2 security mode |
| Wi-Fi Connectivity (802.11 b/ g/ n) |
| Support UART/GPIO data communication interface |
| Support STA/AP/STA+AP 3 working modes |
| Built-in TCP/IP protocol stack, maximum 5 clients |
| Working temperature: -40℃～＋125℃ |
| Dimensions: lenght: 24mm, width: 16mm, height: 3mm |

Bảng 3: Thông số kỹ thuật Esp8266 12f

1. Tính toán thiết kế khối nguồn

Nguồn là 1 vấn đề quan trọng trong bất kỳ hệ thống nào. Trong hệ thống cần cung cấp nguồn cho module GM65, khối xử lý và thiết bị ngoại vi với các yêu cầu cụ thể là:

* Module GM65 và STM32: +5V
* ESP8266 12f: 3÷3.6V

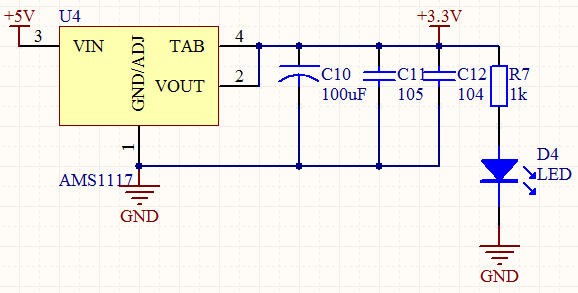
Sử dụng adapter đầu ra 5V 1000mA để cung cấp nguồn cho hệ thống.



Hình 6: Adapter 5V 1A

Tuy nhiên hệ thông sử dụng điện áp vào khác nhau tại mỗi module, do đó cần phải sử dụng mạch nguồn 3.3V để tạo ra điện áp phù hợp cho ESP8266 12f hoạt động.

Ở đây, AMS là dòng IC nguồn ổn áp tuyến tính hiệu suất cao của *Advanced Monolithic Systems, Inc*. Với các ưu điểm như nhiễu thấp, dòng rò nhỏ, tích hợp mạch chống quá tải... rất thích hợp cho nhiều ứng dụng như các mạch Op-amp, sạc pin hay MCU. Với những ưu điểm như vậy, ở đây AMS1117 được sử dụng cho mạch nguồn +3.3V.



Hình 7: Khối nguồn nuôi Esp8266 12f

|  |  |
| --- | --- |
| **Thông số** | **Giá trị** |
| IC | AMS1117 |
| Điện áp đầu vào | +5V |
| Điện áp đầu ra | +3.3V |
| Dòng | 1A |
| Dropout Voltage | 1V |

Bảng 4: Thông số khối nguồn nuôi Esp8266 12f

1. Lựa chọn giải pháp phần mềm
   1. Hệ Điều Hành:

Dựa trên nền tảng STM, có năm phương pháp truyền để lựa chọn nhằm đáp ứng các yêu cầu điều khiển thời gian thực của hệ điều hành. Chúng là μClinux, μC / OS-II, eCos, FreeRTOS và Hệ điều hành Dujiangyan (djyos).

* μClinux: Đặc điểm lớn nhất của μClinux là nó được thiết kế cho các bộ vi xử lý không có MMU. Điều này áp dụng cho stm32f103 không có chức năng MMU, nhưng hệ thống này yêu cầu ít nhất 512KB dung lượng RAM, 1MB dung lượng ROM / FLASH và stmf103 có 256K. FLASH yêu cầu bộ nhớ ngoài, điều này làm tăng chi phí thiết kế phần cứng. μClinux có cấu trúc phức tạp, khó cấy ghép và nhân lớn. Hiệu suất thời gian thực của nó cũng rất kém. Nếu trọng tâm phát triển của các sản phẩm nhúng là hệ thống tệp và các ứng dụng mạng, μClinux là một lựa chọn tốt
* μC / OS-II: Nó là một nhân hệ điều hành nhúng với cấu trúc đơn giản, đầy đủ các chức năng và hiệu suất thời gian thực mạnh mẽ. Rất thích hợp cho các CPU không có chức năng MMU. Nó yêu cầu ít không gian nhân và không gian lưu trữ dữ liệu, hiệu suất thời gian thực tốt, khả năng mở rộng tốt và mã nguồn mở. Có rất nhiều tài liệu và ví dụ trên Internet, vì vậy chuyển sang sử dụng CPU stm32f103 sẽ thích hợp hơn.
* ECos: eCos được đặc trưng bởi cấu hình linh hoạt, hỗ trợ di chuyển CPU, không yêu cầu MMU, là mã nguồn mở, có tính di động tốt và phù hợp hơn để di chuyển sang CPU trên nền tảng stm32. Nhưng ứng dụng của eCOS không rộng rãi, không rộng rãi như μC / OS-II và dữ liệu cũng không rộng rãi như μC/OS-II. eCos phù hợp với một số hệ thống nhúng công nghiệp hoặc thương mại nhạy cảm về chi phí, chẳng hạn như một số ứng dụng trong lĩnh vực điện tử tiêu dùng...
* FreeRTOS: Là một hệ điều hành nhẹ, FreeRTOS cung cấp các chức năng như quản lý tác vụ, quản lý thời gian, semaphores, hàng đợi tin nhắn, quản lý bộ nhớ và chức năng ghi nhật ký để đáp ứng nhu cầu của các hệ thống nhỏ hơn. Hạt nhân FreeRTOS hỗ trợ thuật toán lập lịch ưu tiên. Mỗi nhiệm vụ có thể được ưu tiên nhất định theo mức độ quan trọng. CPU luôn chạy tác vụ với mức ưu tiên cao nhất ở trạng thái sẵn sàng. Nhân FreeRT0S hỗ trợ thuật toán lập lịch quay cùng một lúc. Hệ thống cho phép các tác vụ khác nhau sử dụng cùng một mức độ ưu tiên. Khi không có tác vụ ưu tiên cao hơn nào sẵn sàng, các tác vụ ưu tiên giống nhau sẽ chia sẻ thời gian sử dụng CPU.
* hệ điều hành Dujiangyan (djyos): Hệ điều hành Dujiangyan, được gọi là djyos, được đặt theo tên của một dự án bảo tồn nước vĩ đại Dujiangyan. Không giống như các hệ điều hành truyền thống, djyos không phải là một luồng mà là một lõi lập lịch dựa trên sự kiện. Thuật toán lập lịch trình này giải phóng lập trình viên khỏi lối suy nghĩ về việc mô phỏng quá trình thực thi của máy tính mà viết ứng dụng theo cách của thế giới nhận thức của con người. Như VC được giới thiệu trong lập trình nhúng. Thuật toán lập lịch của Djyos cho phép các lập trình viên thoát khỏi những ràng buộc của các luồng và quy trình. DJyos không có API apis và một lập trình viên không biết kiến thức về luồng có thể viết ứng dụng dưới djyos một cách trôi chảy.

Qua các giới thiệu khái quát trên, có thể đưa ra lựa chọn ứng dụng hệ điều hành FreeRtos vào dự án do hệ điều hành FreeRTOS là hệ điều hành hoàn toàn miễn phí với mã nguồn mở, khả năng di động, khả năng mở rộng và chiến lược lập lịch linh hoạt, có thể dễ dàng chuyển sang các bộ vi điều khiển khác nhau.

* 1. Web Client:

MQTT là một giao thức truyền thông điệp nhỏ gọn dựa trên mô hình publish/subcsribe. Nó sử dụng băng thông thấp, độ tin cậy cao và có thể hoạt động trong điều kiện truyền không ổn định. MQTT được thiết kế cho các mạng SCADA và các tình huống sử dụng băng thông thấp và gần đây nó đã được chứng minh là rất mạnh mẽ trong việc hỗ trợ các dự án IoT. Trong dự án này, giao thức MQTT sẽ được sử dụng để trao đổi dữ liệu giữa hệ thống bán vé và khách hàng mạng tập trung dữ liệu.

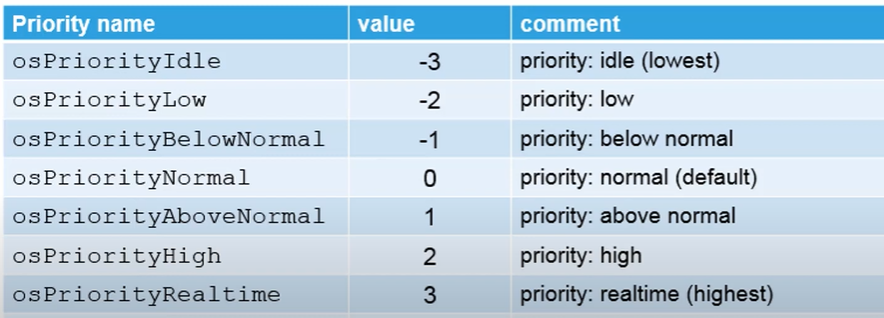
Ưu điểm của MQTT:

* Chuyển thông tin hiệu quả hơn
* Tăng khả năng mở rộng
* Giảm đáng kể tiêu thụ băng thông mạng
* Giảm tốc độ cập nhật xuống giây
* Rất phù hợp cho điều khiển và do thám
* Tối đa hóa băng thông có sẵn
* Chi phí cực nhẹ
* Rất an toàn với bảo mật dựa trên sự cho phép
* Được sử dụng bởi ngành công nghiệp dầu khí, Amazon, Facebook và các doanh nghiệp lớn khác
* Tiết kiệm thời gian phát triển
* Giao thức publish/subscribe thu thập nhiều dữ liệu hơn với ít băng thông hơn so với giao thức cũ.

# Chương IV. Tổng hợp, cấu hình, biên dịch hệ điều hành

* 1. Tổng quan về FreeRTos
* Khái niệm cơ bản về Rtos: RTOS (Real-Time operating system) hay được gọi là hệ điều hành thời gian thực mà cho phép ứng dụng có thể chạy đa tác vụ và có thể đáp ứng được “deadline” theo thời gian thực. Lưu ý rằng việc đáp ứng được “deadline” không nhất nhiết có nghĩa là phải nhanh mà ở đây là mang tính “đúng thời điểm” và chính xác.
* Ứng dụng của RTOS:
  + Chạy các dự án lớn đòi hỏi xử lý nhiều công việc nhưng vẫn phải đáp ứng được về mặt thời gian.
  + Các ứng dụng về viễn thông và IOT, các thiết bị liên quan đến y tế…
* Xác định mức độ ưu tiên giữa các nhiệm vụ:

Đặt task có nhiệm vụ nhận mã từ esp ở mức cao nhất và các task có nhiệm vụ thực hiện đóng/ mở barie, chuông báo, màn hình hiển thị sẽ ở mức ưu tiên thấp hơn.



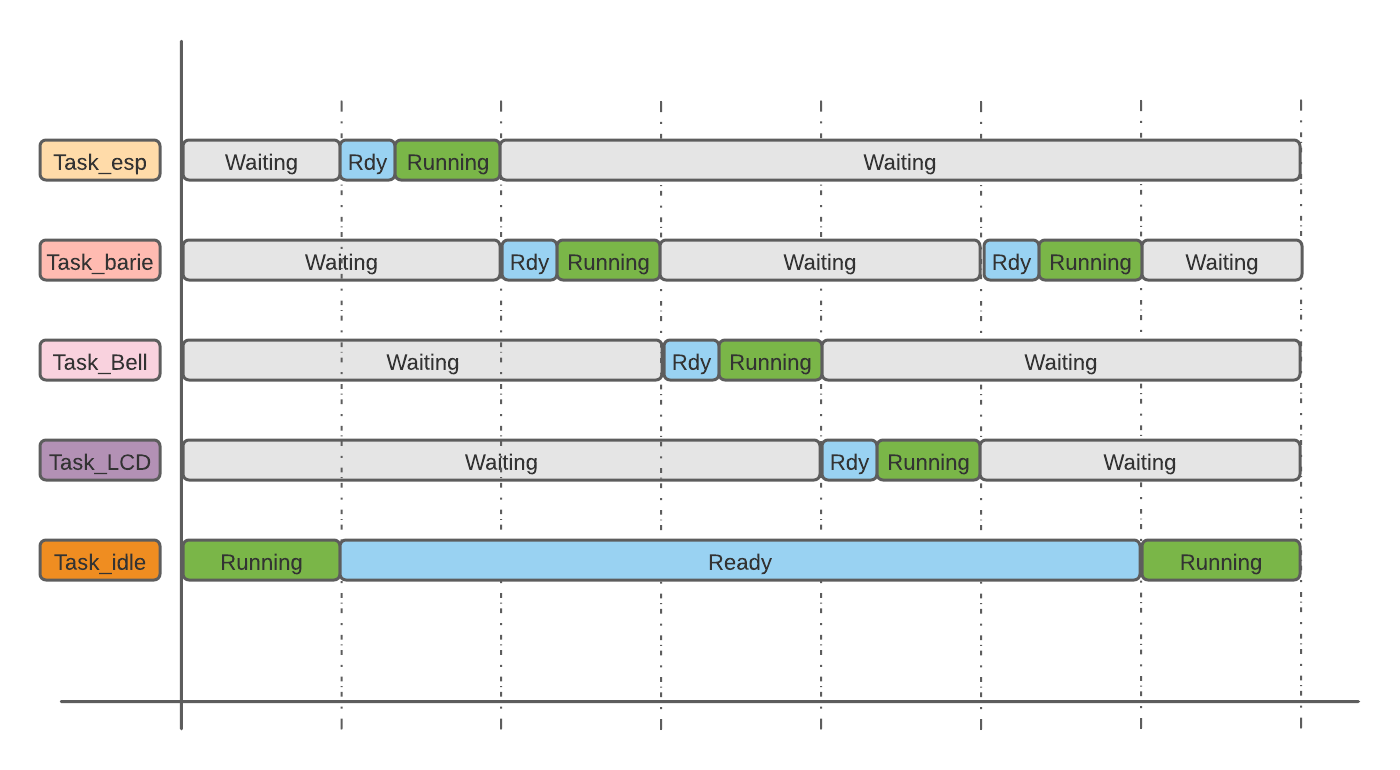
Hình 8: Các mức độ ưu tiên trong phiên bản cmsis-v1

* Task giao tiếp giữa stm32 với esp8266v1 sẽ nhận mức ưu tiên cao nhất do tính quan trọng của nhiệm vụ (mức hightest).
* Các task mở/đóng barie, giao tiếp với LCD, chuông báo ở mức ưu tiên thấp hơn (mức high).
* Lựa chọn thuật toán cho lập lịch

Có 3 kỹ thuật được áp dụng là:

* **Round Robin Scheduling**: Các task sẽ được chia 1 khe thời gian cố định, nếu chưa thực thi xong sẽ tạm dừng và đợi đến lượt.
* **Co-operative scheduling**: Các task được thực thi tới khi kết thúc, các task tiếp theo mới được thực thi.
* **Priority base**: Task được phân quyền cao nhất sẽ được thực hiện trước, nếu các task có cùng quyền như nhau thì sẽ giống với round-robin, các task có mức ưu tiên thấp hơn sẽ được thực hiện cho đến cuối time slice.

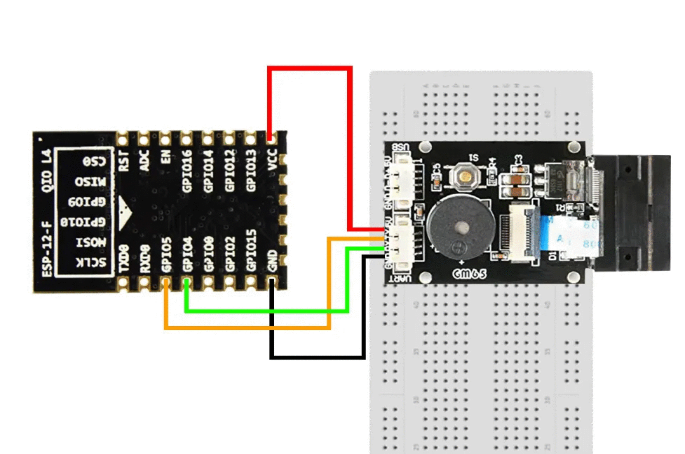
Có thể thấy, với các nhiệm vụ mà hệ thống nhúng cần thực hiện, kỹ thuật priority base phù hợp hơn cả do các nhiệm vụ có mức độ ưu tiên khác nhau.



Hình 9: Lịch trình của hệ thống

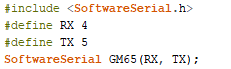
* 1. Các giao tiếp ngoại vi
* ESP8266 12f với module đọc mã vạch:

Sơ đồ kết nối chân của GM65 và esp8266 12f

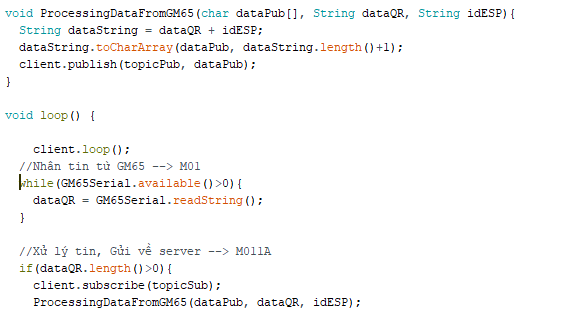


Hình 10:Sơ đồ kết nối chân của GM65 và esp8266 12f

Việc kết nối GM65 với ESP8266 12f ta sử dụng giao tiếp UART. Giao tiếp UART chỉ cần 2 dây để truyền dữ liệu. Do đó, ta kết nối chân Tx và Rx của GM65 với GPIO5 và GPIO4 trên ESP8266. Ta triển khai 2 chân GPIO5 và GPIO4 trong phần mềm như sau:



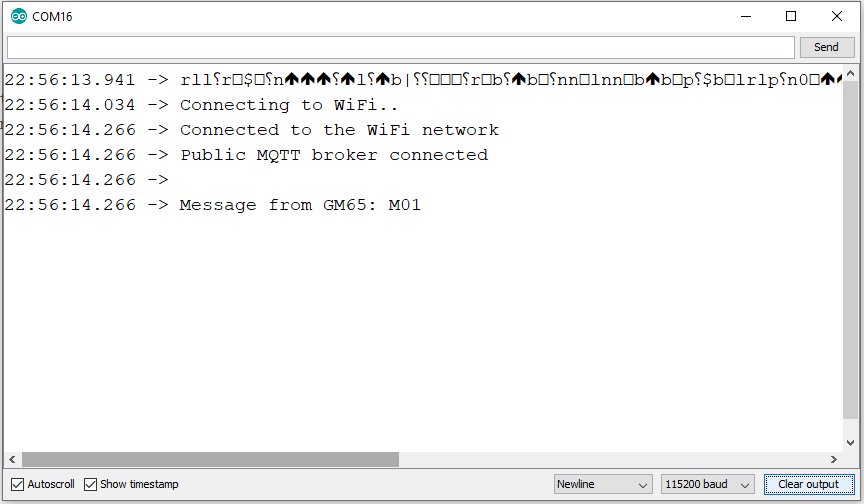
Tại đây, ta sẽ truyền dữ liệu và đọc dữ liệu để giao tiếp với GM65.



Hình 11: Xử lý truyền dữ liệu và đọc dữ liệu để giao tiếp với GM65

Sau khi nhận được tin “M01” từ GM65, ESP8266 xử lý tin rồi gửi về server mã có nội dung “M011A”.

Ta sẽ nhận được kết quả như dưới hình:



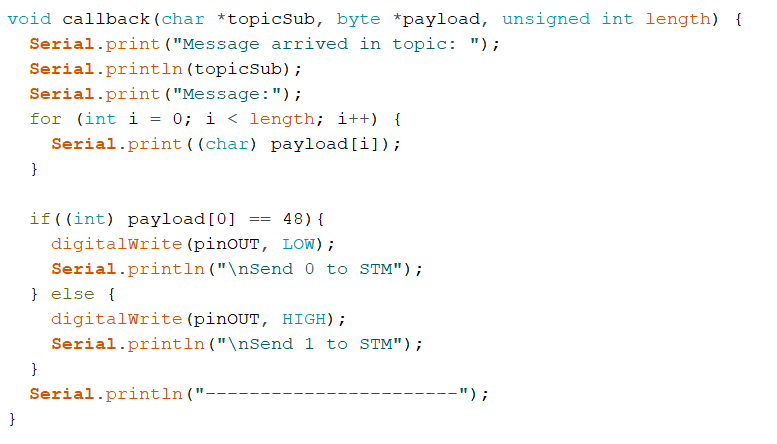
Hình 12: Kết quả

* STM32 với ESP8266 12f:

Sau khi ESP8266 nhận được thông tin đã qua sử lý từ server, ESP8266 tiếp tục đẩy thông tin về STM32 để tiến hành mở cửa (hoặc đóng cửa do khách hàng không đủ tiền)

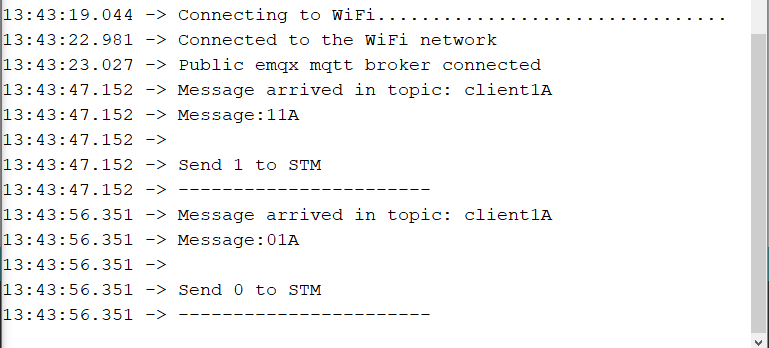
Việc kết nối STM32 với ESP8266 12f ta sử dụng GPIO. Giao tiếp này chỉ cần một dây để truyền dữ liệu là 0 và 1. Do nhu cầu truyền dữ liệu đơn giản là bit 0 và 1 để quyết định việc đóng hay mở cửa nên giao tiếp GPIO là lựa chọn tối ưu. Ta sẽ để đầu ra của ESP là cổng GPIO 1 và đầu vào của STM32 là cổng A4.

Sau khi nhận thông tin từ server ví dụ như 11A bit, với đầu tiên là ‘1’ có nghĩa là cho phép mở cửa để khách qua (ngược lại nếu bit đầu tiên là 0 thì đóng cửa không để khách qua), 2 bit tiếp theo là ‘1A’ là mã của máy soát vé; ta tiến hành tách chuỗi để lấy ra bit đầu tiên, gửi về STM32 với dòng hàm callback như sau:

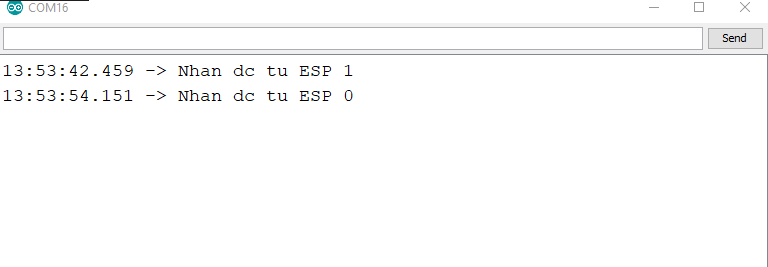


Hình 13: Giao tiếp STM32 với ESP8266 12f

Và ta được kết quả như hình dưới:



Hình 14: Gửi mã tới Stm32f103

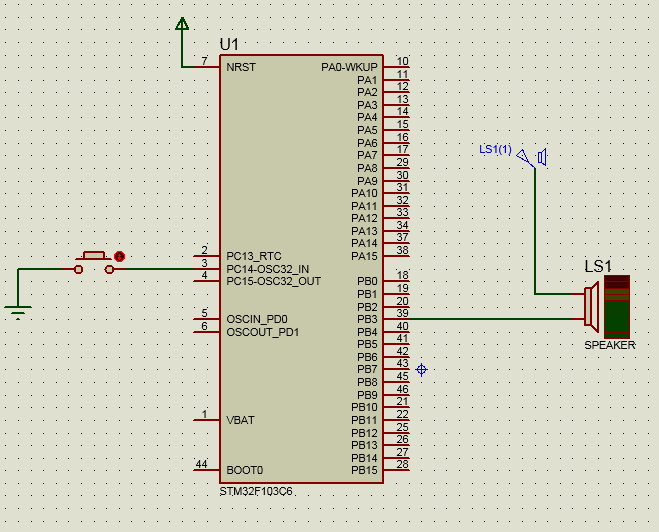


Hình 15: Mã nhận được từ esp

* STM32 với các thiết bị ngoại vi khác:
  + STM32 kết nối chuông báo

+) Input: PC14 (GPIO Pull-up)

+) Output: PB3 (GPIO output level: High, Output push pull, Pull-up, Speed: High)



Hình 16: Kết quả mô phỏng

* + Kết nối song song với LCD 16x02

Sử dụng 5 chân GPIO\_Output giao tiếp với LCD:

+) Nhóm chân dữ liệu cao của LCD:

+) PB10-D7

+) PB11-D6

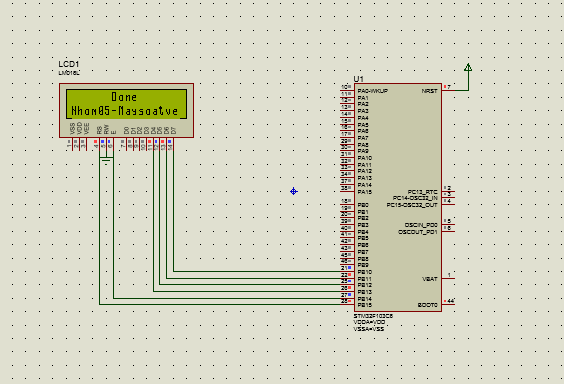
+) PB12-D5

+) PB13-D4

+) Nhóm chân điều khiển của LCD (R/W nối đất):

+) PB14-E

+) PB15-RS



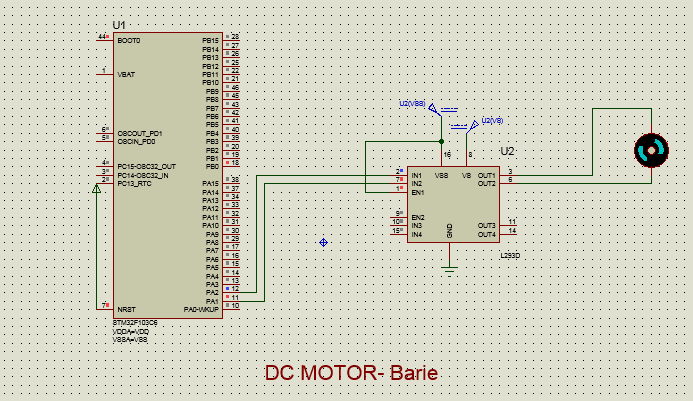
Hình 17: Kết quả mô phỏng

* + STM32 kết nối khóa barie thông qua Driver Motor Shield L293D (Barie được coi là DC Motor EL2030)

+) Chân OUT1, OUT2 của L293D kết nối với Motor EL2030.

+) VSS được cấp nguồn DC 5 V, và được nối chung với EN1.

+) VS được cấp nguồn DC 12V.



Hình 18: Kết quả mô phỏng

# Chương VI. Phát triển phần mềm

Việc phát triển phần mềm đảm bảo quá trình làm việc xuyên suốt của hệ thống. Các giá trị trao đổi giữa các module phải được thống nhất để thuận tiện cho việc lập trình cũng như làm việc nhóm.

Ta có thể quy định các tham số xuyên suốt dự án này như sau:

* Mã định danh thẻ (ID): Mxx - với xx là các số từ 00 tới 99 (giả sử có 100 thẻ)

VD: M01,M02,..M99.

Mã này sẽ được thu từ module quét mã vạch.

* Vị trí các máy quét (ADD): XY - với X là các số từ 1-9 và Y là các chữ cái in hoa

VD: thiết bị quét mã client1A sẽ có vị trí 1A, client9C sẽ có vị trí 9C.

* Tín hiệu gửi từ ESP8266 tới web-client : ID+ADD.

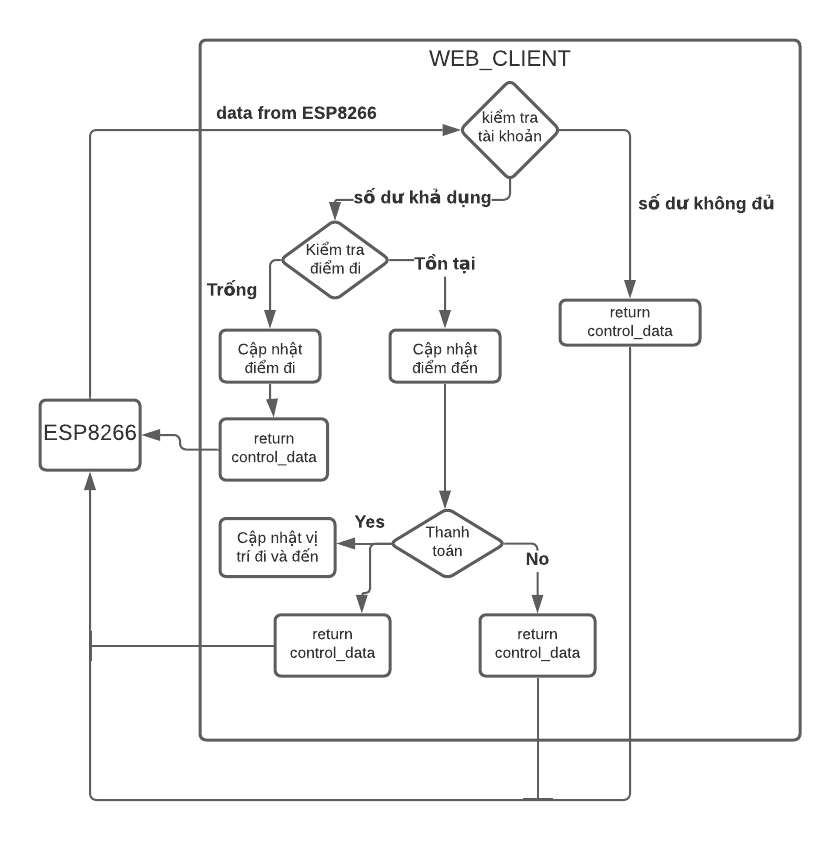
VD: M011A sẽ cho web-client biết phải xử lý thẻ có ID M01 và gửi trả tín hiệu phản hồi cho thiết bị quét mã vạch có vị trí 1A

* Tín hiệu phản hồi từ web-client: CS(control signal) sẽ có giá trị là 0 và 1. Tín hiệu có vai trò giúp vi điều khiển điều khiển chính xác hoạt động của các thiết bị ngoại vi.

1. Web-client mqtt

Với yêu cầu 1 vị trí để lưu trữ thông tin của các thẻ cũng như trạng thái của các thẻ vào từng thời điểm, dựa vào các thông tin đó để thực hiện các bước xử lý và tính toán đơn giản như tính chi phí đi lại.

* Yêu cầu chức năng đối với web:

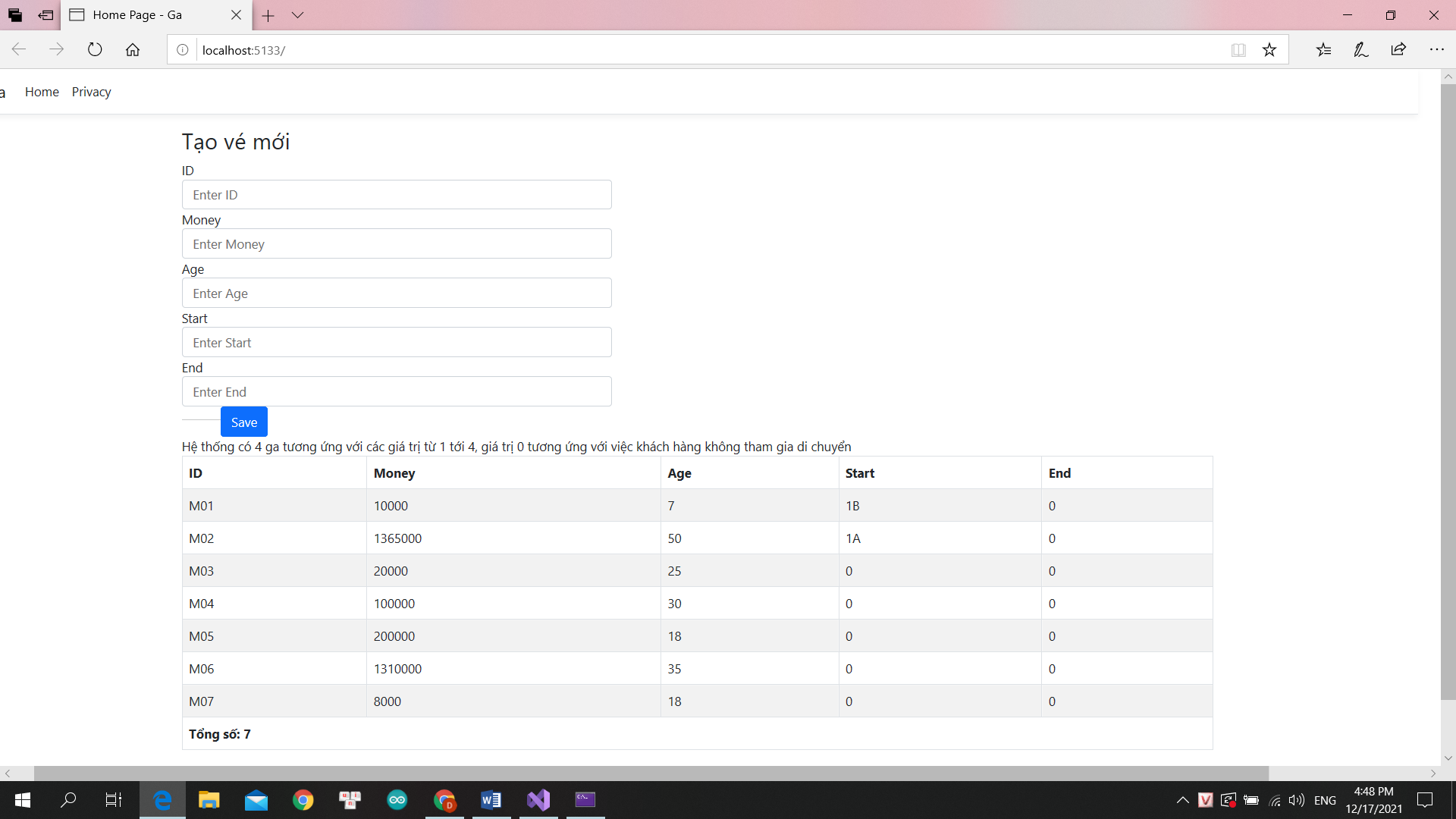


Hình 19: Sơ đồ mô tả các tác vụ của web-client

Với các yêu cầu như đã nêu các phần trước, có thể đưa ra một sơ đồ tổng quát mô tả lại quá trình hoạt động của đơn vị web-client.

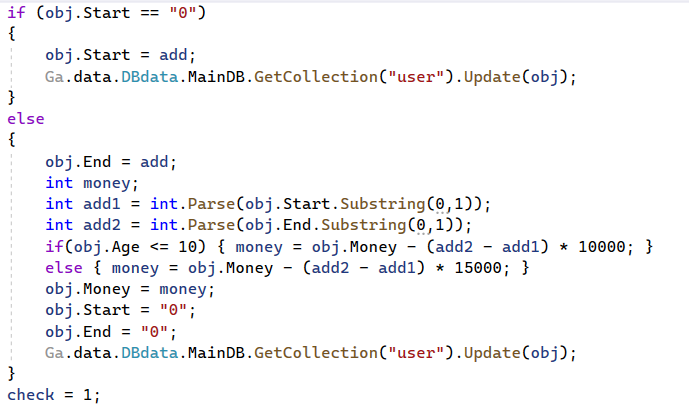
Về cơ bản, web-client cần phải đáp ứng được các nhiệm vụ cơ bản như: đọc mã được gửi từ các máy soát vé (ID+ADD), Kiểm tra số dư tài khoản thông qua ID, cập nhật vi trí xuất phát và kết thúc của hành khách, Thanh toán tiền theo lộ trình khách đã di chuyển, phản hồi tín hiệu điều khiển về các máy soát vé 1 cách chính xác.

* + Kết quả thu được:



Hình 20: Giao diện web-client

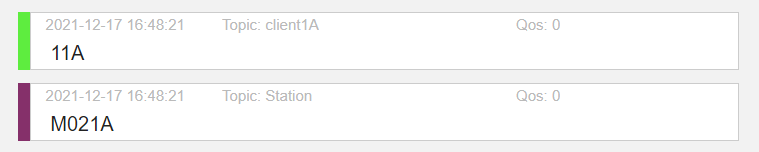
Việc xây dựng web-client sử dụng mô hình web api mvc với ngôn ngữ C#, web sẽ lưu và hiển thị các thông tin quan trọng về từ ID như số dư, tuổi chủ sở hữu, diểm bắt đầu và điểm kết thúc trong lộ trình. Các giá trị về Lộ trình của hành khách sẽ được để mặc định là 0, có nghĩa khách hàng không tham gia di chuyển nếu giá trị cột Start = 0 và khách hàng chưa hoàn thành thanh toán hoặc đang trong quá trình di chuyển nếu cột End = 0. Khi khách hàng thanh toán xong, các giá trị sẽ trở về 0 như hình dưới.

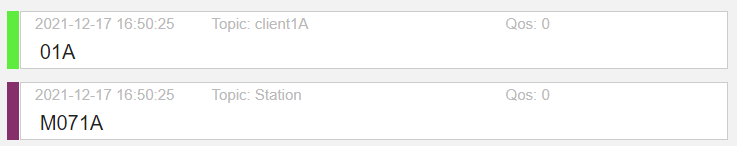


Hình 21: Tính toán chi phí hành trình

Với mô hình thanh toán đang áp dụng hiện nay tại Việt Nam là sử dung vé tuyến sẽ gây thua lỗ cho các công ty dịch vụ vận tải khách. Chính vì vậy, cần phải xác định chính xác quảng đường khách hàng đã di chuyển để tính toán chi phí cần chi trả.

Trong dự án này, tạm thời giả thuyết giá vé di chuyển giữa các ga cạnh nhau là bằng nhau. Do đó, giá vé bằng được tính bằng số chặng \* giá từng chặng. Giả sử vé cho trẻ em (<10 tuổi ) là 10000đ và người lớn (> 10 tuổi) là 15000đ.





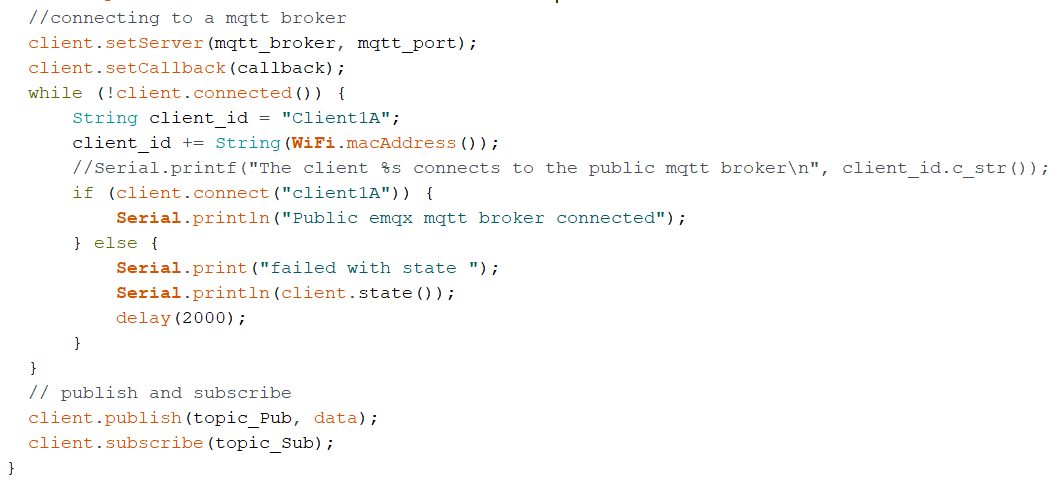
Hình 22: Thông tin trao đổi giữa máy soát vé và web-client

Trong hình trên, có thể thấy kết quả truyền tin giữa thiết bị soát vé (client1A) với web-client (Station). Khi thẻ có ID là M02 được quét, sẽ gửi mã M021A tới web-client. Mã này cho biết cần phải kiểm tra thẻ có ID là M02 và gửi trả tín hiệu về client1A, nếu thẻ hợp lệ trả CS là 1 về client1A. Tương tự mới thẻ có ID là M07 được quét tại client 1A, tuy nhiên, có thể thấy trong hình số x, số dư trong thẻ chỉ còn 8000đ, không đủ cho việc di chuyển giữa 1 chặng ( 10000đ với vé trẻ em). Do đó, tín hiệu trả về cho client1A là 0.

1. Esp-client mqtt

Với yêu cầu gửi tín hiệu cho web-client xử lý với giao thức MQTT đã được nêu ở trên, ta cần kết nối trạm soát vé với vai là một client với các chức năng:

Gửi tín hiệu cần xử lý lên web-client và nhận phản hồi từ nó.



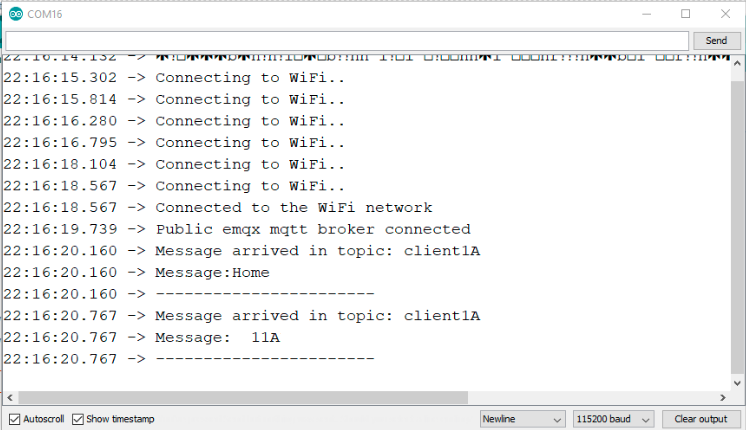
Hình 23: Gửi tín hiệu cần xử lý lên web-client

Ta kết nối với MQTT broker bằng câu lệnh client.setServer(mqtt\_broker, mqtt\_port) với mqtt\_broker là tên broker và mqtt\_port là cổng MQTT

Dùng vòng lặp while để kiểm tra kết nối giữa client với broker, sau khi kết nối thành công sẽ hiện thị thông báo đã kết nối thành công với câu lệnh Serial.println("Public emqx mqtt broker connected"), Nếu không thành công sẽ hiển thị trạng thái lỗi

Khi kết nối thành công, client sẽ gửi tín hiệu cần xử lý và nhận phản hồi theo 2 chủ đề đã được khai báo client.publish(topic\_Pub, data); và client.subscribe(topic\_Sub);

Ta sẽ nhận được kết quả như hình dưới:



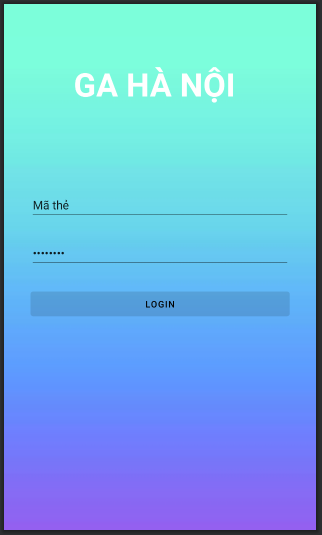
Hình 24: Kết quả trả về

Hình trên cho ta thấy client nhận được thông báo là Message: 11A và trùng khớp với tin nhắn trả về từ web-client đã nêu trong phần trên với thẻ có ID: M02.

Nhận thấy thời gian lúc truyền tín hiệu đi (22:16:20.16) và thời gian nhận được tín hiệu phản hồi (22:16:20.76) là vào khoảng 0.6s trong các điều kiện liên quan không mấy hoàn hảo.

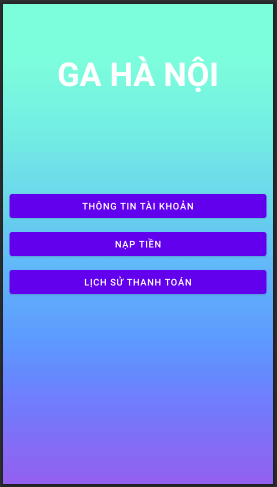
1. App hỗ trợ tra cứu thông tin người dùng

Giao diện màn hình đăng nhập của app. Người dùng tiến hành nhập các thông tin về mã thẻ và số chứng minh nhân dân làm mật khẩu để đăng nhập vào hệ thống.



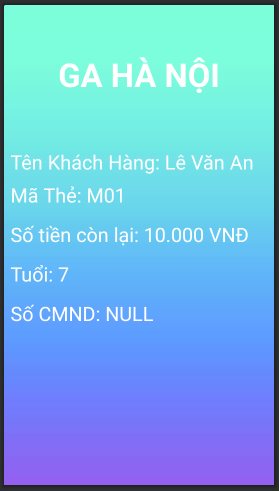
Hình 25: Giao diện đăng nhập

Sau khi đăng nhập thành công, ứng dụng sẽ hướng người dùng tới giao diện chính của app. Giao diện sẽ có các mục chính như: Kiểm tra thông tin tài khoản, nạp tiền (chưa khả dụng) và lịch sử thanh toán.



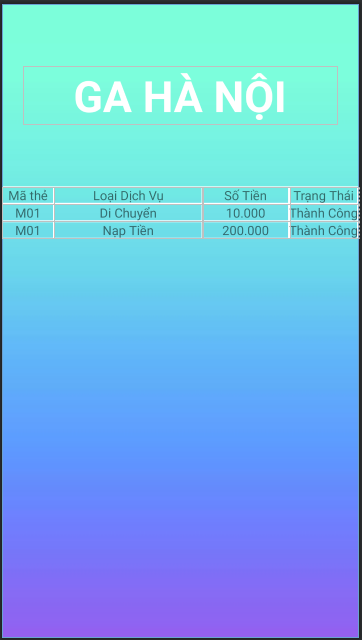
Hình 26: Màn hình chính

Khi nhấn vào mục Thông tin tài khoản, ứng dụng sẽ hiển thị các thông tin về Tên khách hàng, Mã thẻ mà khách hàng sở hữu, Số tiền còn lại, Tuổi và Số CMND. Các thông tin có được từ việc thu thập khi khách đăng ký làm thẻ.



Hình 27: Màn hình tra cứu thông tin

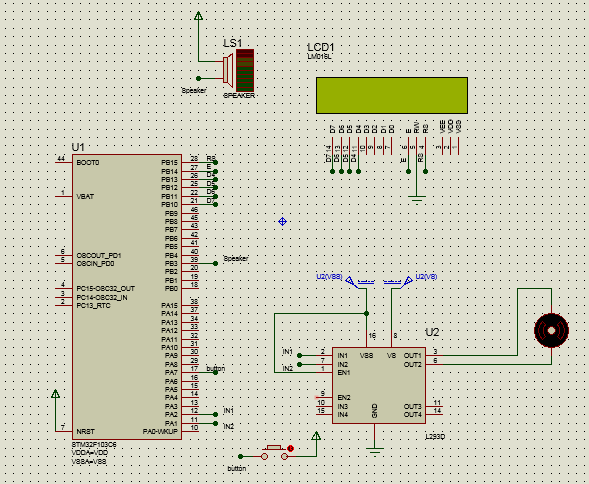
Khi nhấn vào mục Lịch sử thanh toán, Ứng dụng chuyển hướng người dùng tới giao diện như hình dưới. Các thông tin sẽ được hiển thị bao gồm: mã thẻ, loại dịch vụ, số tiền, trạng thái.



Hình 28: Màn hình lịch sử giao dịch

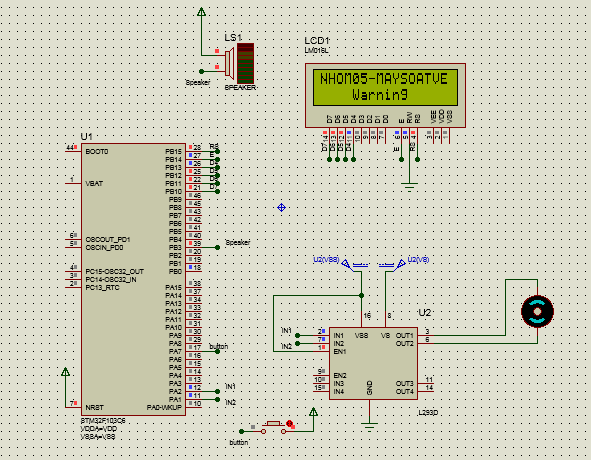
# Chương VII. Triển khai, mô phỏng hệ thống

Qua các bước đã thực hiện ở các phần trên, chúng em sử dụng công cụ mô phỏng mạch Proteus để tiến hành mô phỏng lại hoạt động của hệ thống. Ta có sơ đồ mô phỏng như hình dưới:

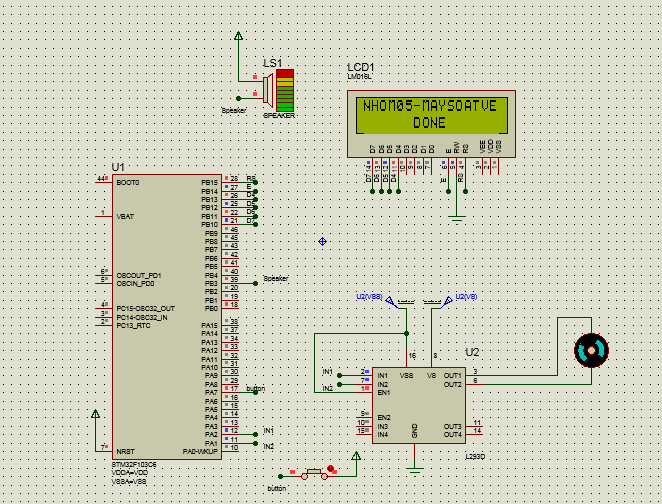


Hình 29: Sơ đồ mạch mô phỏng

Các kết quả mô phỏng thu được:



Hình 30: Mạch mô phỏng kia chua có tín hiệu vào



Hình 31: Khi có tín hiệu vào

# Chương VIII. Phân tích, đánh giá kết quả

Nghiên cứu và thiết kế một hệ thống vẫn luôn là một xu hướng phát triển mạnh mẽ trong nhiều năm trở lại đây. Việc tối ưu hóa hệ thống cũng cần được chú ý phát triển. Sau một thời gian nghiên cứu, thiết kế và thử nghiệm phần cứng cho hệ thống, nhóm chúng em đã đạt được cơ bản các mục tiêu ban đầu đặt ra:

• Nắm được các phương thức truyền nhận dữ liệu thông qua nhiều giao thức.

• Cách xây dựng một mô hình Client-Server cỡ nhỏ, từ đó phát triển lên thành một mô hình lớn.

• Có khả năng thiết kế phần cứng cho một hệ thống thu thập dữ liệu dựa trên yêu cầu có sẵn, từ đó có thể phát triển, nâng cấp dựa trên nền tảng hệ thống cũ khi có yêu cầu mới.

Trong thời gian thực hiện đề tài nhóm chúng em thấy rằng đề tài này là một vấn đề rất rộng, đòi hỏi cần có thêm thời gian để thực hiện nghiên cứu chuyên sâu hơn nữa. Dịch Covid-19 cũng là một trở ngại trong quá trình tìm hiểu và thực hiện đề tài nên chưa thể triển khai hệ thống vào thực tế. Đây là cơ hội cho những bạn sinh viên nghiên cứu về sau, cũng như là thách thức cho chúng em trong việc tối ưu hệ thống.