TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA ĐIỆN – ĐIỆN TỬ**

Logo, company name

Description automatically generated

**CAO TRỌNG KỶ**

**THIẾT KẾ, THI CÔNG VÀ ĐIỀU KHIỂN MÁY BÁN HÀNG TỰ ĐỘNG**

**ĐỒ ÁN TỔNG HỢP**

**KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HOÁ**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA ĐIỆN ‒ ĐIỆN TỬ**

Logo, company name

Description automatically generated

**CAO TRỌNG KỶ ‒ 41801155**

**THIẾT KẾ, THI CÔNG VÀ ĐIỀU KHIỂN MÁY BÁN HÀNG TỰ ĐỘNG**

**ĐỒ ÁN TỔNG HỢP**

**KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HOÁ**

Người hướng dẫn

**TS. Trần Đức Anh Minh**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

**LỜI CẢM ƠN**

Để thực hiện đồ án này, em dựa vào sự quan tâm, giúp đỡ và chỉ dạy nhiệt tình của các thầy cô giảng viên và bạn bè. Em xin chân thành cảm ơn đến TS. Trần Đức Anh Minh khoa Điện ‒ Điện tử trường Đại Học Tôn Đức Thắng đã tận tình hướng dẫn cho em trong suốt quá trình thực hiện. Em cũng xin chân thành cảm ơn các thầy cô trường Đại học Tôn Đức Thắng nói chung và các thầy cô bộ môn ngành Kỹ thuật điều khiển và Tự động hoá nói riêng đã dạy dỗ, truyền lại cho em kiến thức về các môn đại cương cũng như các môn chuyên ngành. Giúp em vững vàng kiến thức và chỉ bảo cho em cách để áp dụng nó vào tương lai cống hiến cho tổ quốc. Em xin chân thành cảm ơn!

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của TS. Trần Đức Anh Minh. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong luận văn còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung luận văn của mình.** Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*Cao Trọng Kỷ*

**THIẾT KẾ, THI CÔNG VÀ ĐIỀU KHIỂN MÁY BÁN HÀNG TỰ ĐỘNG**

**TÓM TẮT**

Hiện nay, đời sống của con người ngày càng được nâng cao. Nhu cầu cho sự tiện dụng là cần thiết. Vì vậy em muốn chế tạo một loại máy bán hàng tự động áp dụng các công nghệ giám sát từ xa hiện nay mà giá thành lại phải chăng. Mô hình máy bán hàng tự động sẽ sử dụng một máy tính nhúng Raspberry Pi làm bộ điều khiển trung tâm. Nó có chức năng nhận tín hiện đầu vào, tính toán sau đó điều khiển các động cơ để đẩy sản phẩm ra cho khách hàng. Mô hình có thể được giám sát từ xa thông qua cơ sở dữ liệu thời gian thực Realtime Firebase Database, áp dụng Node -RED và lưu trữ trên trang tính Google Sheets để truy xuất về sau.

**Design, implementation, and control of a vending machine**

**ABSTRACT**

Nowadays, people's lives are improving. Convenience is necessary. That's why I want to make an automatic vending machine that applies the latest remote monitoring technology and has an affordable price. The automatic vending machine model will use a Raspberry Pi embedded computer as the central control unit. It will receive inputs, perform calculations, and control motors to dispense products to customers. The model can be monitored remotely through the Realtime Firebase Database, applied with Node-RED, and stored on Google Sheets for future access.

**MỤC LỤC**

[DANH MỤC HÌNH ẢNH iv](#_Toc127219732)

[DANH MỤC CÁC TỪ VIẾT TẮT v](#_Toc127219733)

[CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI 1](#_Toc127219734)

[1.1 Giới thiệu đề tài 1](#_Toc127219735)

[1.1.1 Giới thiệu sơ lược 1](#_Toc127219736)

[1.1.2 Mô tả chức năng 3](#_Toc127219737)

[1.2 Mục đích nghiên cứu 5](#_Toc127219738)

[1.3 Đối tượng nghiên cứu 5](#_Toc127219739)

[1.4 Phạm vi nghiên cứu 5](#_Toc127219740)

[1.5 Dự kiến kết quả 5](#_Toc127219741)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 6](#_Toc127219742)

[2.1 Sơ đồ khối của hệ thống 6](#_Toc127219743)

[2.2 Các khối và linh kiện 7](#_Toc127219744)

[2.2.1 Khối xử điều khiển (Raspberry Pi 3 mode B) 7](#_Toc127219745)

[2.2.2 Màn hình cảm ứng điện dung 5 inch 11](#_Toc127219746)

[2.2.3 Nguồn tổ ong 12VDC 30A 13](#_Toc127219747)

[2.2.4 Module cảm biến vật cản hồng ngoại 14](#_Toc127219748)

[2.2.5 Module điều khiển động cơ DC L298N 15](#_Toc127219749)

[2.2.6 Động cơ giảm tốc 12VDC 17](#_Toc127219750)

[2.2.7 Cơ sở dữ liệu Firebase Realtime Database 18](#_Toc127219751)

[2.2.8 Chương trình lập trình Node RED 18](#_Toc127219752)

[2.2.9 Chương trình bảng tính Google Sheets 20](#_Toc127219753)

[CHƯƠNG 3. THIẾT KẾ VÀ THI CÔNG 21](#_Toc127219754)

[3.1 Thi công 21](#_Toc127219755)

[3.1.1 Linh kiện sử dụng 21](#_Toc127219756)

[3.1.2 Thi công mạch điều khiển 21](#_Toc127219757)

[3.1.3 Thi công mô hình 22](#_Toc127219758)

[3.2 Thiết kế 23](#_Toc127219759)

[3.2.1 Thiết kế giao diện người dùng 23](#_Toc127219760)

[3.2.2 Thiết kế giao diện giám sát Node Red 24](#_Toc127219761)

[CHƯƠNG 4. GIẢI THUẬT VÀ ĐIỀU KHIỂN 26](#_Toc127219762)

[4.1 Mô tả hoạt động của hệ thống 26](#_Toc127219763)

[4.2 Lưu đồ giải thuật 26](#_Toc127219764)

[4.2.1 Lưu đồ giải thuật trên chương trình chính 27](#_Toc127219765)

[4.2.2 Lưu đồ giải thuật chương trình thực thi 28](#_Toc127219766)

[CHƯƠNG 5. THỰC NGHIỆM 29](#_Toc127219767)

[5.1 Tiến trình thực nghiệm 29](#_Toc127219768)

[5.2 Kết quả thực nghiệm 29](#_Toc127219769)

[CHƯƠNG 6. KẾT LUẬN 31](#_Toc127219770)

[6.1 Ưu điểm 31](#_Toc127219771)

[6.2 Nhược điểm 31](#_Toc127219772)

[6.3 Hướng phát triển 31](#_Toc127219773)

[Phụ lục 1: Chương trình điều khiển 33](#_Toc127219774)

[Code Raspberry Pi ( Chương trình điều khiển) 33](#_Toc127219775)

[Code HTML ( Chương trình tạo website bán hàng) 44](#_Toc127219776)

DANH MỤC HÌNH ẢNH

[Hình 1.1 Máy bán hàng tự động 1](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219777)

[Hình 1.2 Giá thành máy bán hàng tự động 2](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219778)

[Hình 1.3 Máy bán hàng tự động bị đập phá 3](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219779)

[Hình 2.1. Sơ đồ chân của Raspberry Pi 3 mode B 8](#_Toc127219780)

[Hình 2.2 Cấu trúc phần cứng của Raspberry Pi 3 mode B 8](#_Toc127219781)

[Hình 2.3 Hệ điều hành cho Raspberry Pi 3 mode B 10](#_Toc127219782)

[Hình 2.4 Mặt trước màn hình 5 inch Waveshare 12](#_Toc127219783)

[Hình 2.5 Mặt sau màn hình 5 inch Waveshare 12](#_Toc127219784)

[Hình 2.6 Nguồn tổ ong 12V 30A 14](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219785)

[Hình 2.7 Module cảm biến hồng ngoại phát hiện vật cản 15](#_Toc127219786)

[Hình 2.8 Cấu tạo module cảm biến hồng ngoại phát hiện vật cản 15](#_Toc127219787)

[Hình 2.9 Sơ đồ mạch điện của Module L298N 16](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219788)

[Hình 2.10 Module L298N 16](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219789)

[Hình 2.11 Động cơ giảm tốc 12VDC 17](#_Toc127219790)

[Hình 2.12 Cơ sở dữ liệu Firebase Realtime Database 18](#_Toc127219791)

[Hình 2.13 Node-Red Websever 19](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219792)

[Hình 2.14 Chương trình bảng tính Google Sheets 20](#_Toc127219793)

[Hình 3.1 Mạch điều khiển 21](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219794)

[Hình 3.2 Mô hình đã thi công 22](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219795)

[Hình 3.3 Mô hình đã thi công 23](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219796)

[Hình 3.4 Giao diện nhập hàng 24](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219797)

[Hình 3.5 Giao diện người dùng 24](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219798)

[Hình 3.6 Giao diện giám sát 25](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219799)

[Hình 3.7 Giao diện Node RED 25](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219800)

[Hình 5.1 Kết quả thực nghiệm 29](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219801)

[Hình 5.2 Cơ sở dữ liệu 30](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219802)

[Hình 5.3 Kết quả Google Sheets 30](file:///I:\TS.TranDucAnhMinh_DATH_CaoTrongKy_41801155.docx#_Toc127219803)

DANH MỤC CÁC TỪ VIẾT TẮT

ESP: Espressif Systems

A: Ampe.

IC: Integrated Circuit.

LCD: Liquid Crystal Display.

LDR: Light-Dependent Resistor.

PIC: Programmable Intelligent Computer.

PLC: Programmable Logic Controller.

R: Resistor.

V: Volt.

VDC: Volts-Direct Current.

# TỔNG QUAN VỀ ĐỀ TÀI

## Giới thiệu đề tài

### Giới thiệu sơ lược

* Các nước phát triển trên thế giới có tỷ lệ người cao tuổi và dân số đông nên máy bán hàng đa dạng các loại sản phẩm như máy bán thực phẩm tươi sống như tôm hùm, máy bán tôm, máy bán thực phẩm đông lạnh. Họ thích máy bán hàng tự động hơn để mua sắm vì máy bán hàng tự động thuận tiện hơn.

**A picture containing vending machine, indoor, several

Description automatically generated**

Hình 1.1 Máy bán hàng tự động

* Hiện nay, máy bán hàng tự động tại Việt Nam đang được ưa chuộng bởi những ưu điểm như bán hàng theo giá niêm yết, mua hàng nhanh chóng, tiết kiệm thời gian… Tuy nhiên, nó cũng nảy sinh nhiều vấn đề. gây ra nhiều vấn đề cho khách hàng và người mua vì những hạn chế mà nó phát sinh:
* Khí hậu không đảm bảo, Việt Nam có khí hậu nhiệt đới gió mùa, thời tiết thay đổi thất thường nên dễ làm hư hỏng thực phẩm trong máy.
* Graphical user interface, text

  Description automatically generated Giá thành cao, khó mở rộng quy mô.

Hình 1.2 Giá thành máy bán hàng tự động

* Hầu hết các máy bán hàng tự động ở Việt Nam không có chức năng trả lại tiền lẻ cho khách hàng mua hàng nên mọi người phải đổi số tiền lẻ tương ứng khi mua hàng.
* Khó giám sát, thống kê doanh thu.
* Trộm cắp vẫn thường xảy ra do ý thức của một số người chưa tốt. Hệ thống an ninh chưa đảm bảo, hầu hết máy đặt ở nơi công cộng nên không được kiểm tra thường xuyên.

A picture containing text, person, outdoor, standing

Description automatically generated

* Hình 1.3 Máy bán hàng tự động bị đập phá
* Nhận thấy thực tế đó, em mong muốn thiết kế ra một máy bán hàng tự động với giá thành phải chăng, áp dụng các công nghệ hiện nay và có khả năng giám sát từ xa.

### Mô tả chức năng

* Chọn sản phẩm trên màn hình.
* Thanh toán qua ví điện tử.
* Hệ thống xác nhận thanh toán sau đó đẩy sản phẩm rơi xuống bộ phận nhận hàng
* Tự động điều chỉnh sản phẩm sau khi châm hàng.
* Thống kê, hiển thị và giám sát số lượng, doanh thu từ xa.
* Hoạt động:

Hệ thống được cấp nguồn 220 VAC.

Máy tính nhúng Raspberry Pi hiển thị thông tin sản phẩm, số lượng lên màn hình LCD, điều khiển hoạt động của các mạch cầu L298N, các cảm biến hồng ngoại .

Máy tính nhúng Raspberry Pi đóng vai trò là một sever chạy chương trình bán hàng. Các thông tin sẽ được hiển thị lên Websever dạng Locall Host.

Màn hình LCD có cảm ứng, người dùng sẽ thao tác chọn sản phẩm trực tiếp trên màn hình.

Hệ thống hiển thị, chọn sản phẩm hoạt động trên giao diện website.

Sau khi khách hàng chọn sản phẩm, máy tính nhúng Raspberry Pi xử lý, tính toán và đưa tín hiệu đến các mạch L298N để điều khiển các động cơ đẩy sản phẩm xuống khay lấy hàng.

Module cảm biến vật cản hồng ngoại có chức năng giám sát, đếm sản phẩm và phản hồi tín hiệu thu được nhằm điều khiển động cơ đẩy sản phẩm chính xác nhất.

Sau khi bán được đơn hàng. Hệ thống sẽ tính toán, gửi dữ liệu gồm: Mã đơn hàng, tên sản phẩm, đơn giá, số lượng và thành tiền về cơ sở dữ liệu FireDatabase.

Cơ sở dữ liệu FireDatabase làm cầu nối nhận dữ liệu từ máy tính nhúng Raspberry Pi, xử lí, lưu trữ và gửi dữ liệu lên Node-red.

Node red sẽ nhận dữ liệu từ FireDatabase sau đó xuất ra một bảng với tiêu đề các cột là các trường dữ liệu mà mày tíng nhúng Raspberry Pi đã gửi lên FireDatabase. Tại đây chúng ta có thể giám sát doanh số bán ra của từng đơn, từng mặt hàng trọng thời gian thực.

Node red cũng gửi dữ liệu về Google Sheets để lưu trữ dữ liệu lâu dài có thể sử dụng trong tương tai.

Hệ thống sẽ vào trạng thái nghỉ khi đã hoàn thành đơn hàng. Hệ thống dừng hoạt động khi ngắt kết nối nguồn điện.

* Ứng dụng:

Ứng dụng vào trong kinh doanh nước giải khát, đồ ăn vặt và các sản phầm nhỏ và vừa.

## Mục đích nghiên cứu

Áp dụng những kiến thức đã học về vi điều khiển, linh kiện điện tử, động cơ điện một chiều, các loại cảm biến và lý thuyết điều khiển tự động. Tiến hành nghiên cứu các linh kiện điện tử như: máy tính nhúng Raspberry Pi, L298N, module cảm biến vật cản hồng ngoại,…. Từ đó áp dụng vào việc tạo ra hệ thống điều khiển thông minh.

Áp dụng kiến thức đã học về lập trình ( cụ thể là ngôn ngữ Python trên máy tính nhúng Raspberry Pi) để viết chương trình điều khiển hệ thống tại máy bán hàng. Sử dụng Websever và cơ sở dữ liệu FireDatabase để truyền thông giữa máy tính nhúng Raspberry Pi, Node red và Google Sheets.

## Đối tượng nghiên cứu

* Máy tính nhúng Raspberry Pi, L298N, màn hình LCD, module cảm biến vật cản hồng ngoại, motor giảm tốc.
* Phần mềm lập trình Python, cơ sở dữ liệu Realtime FireDatabase, công cụ lập trình Node red, và chương trình bảng tính Google Sheets.
* Bộ nguồn 12VDC, đèn led, điện trở và các linh kiện khác
* Mô hình máy bán hàng tự động.

## Phạm vi nghiên cứu

* Ở phạm vi là đồ án tổng hợp, nghiên cứu mô hình tỉ lệ nhỏ. Ưu tiên giải quyết vấn đề được đặt ra.

## Dự kiến kết quả

* Máy bán hàng hoạt động ổn đinh.
* Hệ thống giám sát làm việc chính xác.

# CƠ SỞ LÝ THUYẾT

## Sơ đồ khối của hệ thống

Graphical user interface, diagram

Description automatically generated with medium confidence

* Mô tả hoạt động:

Hệ thống hoạt động khi được cấp nguồn 12 VDC vào mạch.

Máy tính nhúng Raspberry điều khiển hoạt động của mạch cầu L298N, cảm biến vật cản hồng ngoại.

Máy tính nhúng Raspberry tạo ra một Sever Local-Host ngay trên chính nó. Có nhiệm vụ duy trì Website trình chiếu các mặt hàng, nhận hành động từ khách hàng và gửi thông tin về máy tính nhúng để xử lí, tính toán và đưa ra tính hiệu điều khiển cho mạch cầu L298N.

Mạch cầu L298N nhận tín hiệu digital từ máy tính nhúng Raspberry về các chân IN1, IN2, IN3, IN4 để điều khiền chiều của các động cơ qua các cổng OUT1, OUT2, OUT3, OUT4. Nhận xung PWM từ máy tính nhúng Raspberry về các chân ENA và ENB để điều khiển tốc độ động cơ.

Cảm biến vật cản hồng ngoại có chức năng đo đạc tín hiệu. Xác định vật cản phía trước, gửi dữ liệu về máy tính nhúng Raspberry.

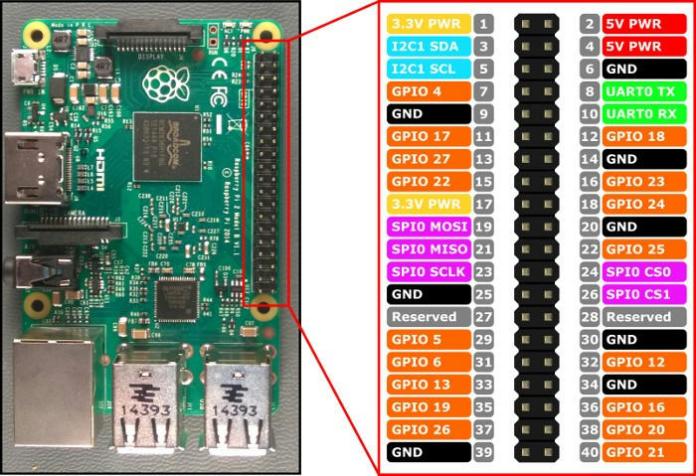
## Các khối và linh kiện

### Khối xử điều khiển (Raspberry Pi 3 mode B)

* Chức năng: Nhận tín hiệu từ các cảm biến, tính toán và xử lý tín hiệu nhận được rồi điều khiển các module L298N. Gửi dữ liệu lên websever
* Linh kiện chính sử dụng: Raspberry Pi 3 mode B
* Lý do chọn Raspberry Pi 3 mode B: Raspberry Pi 3 mode B có thể kết nối wifi, bluetooh. Công suất làm việc trung bình, nhiều chân GPIO. Dễ làm quen cũng như dễ tìm được các tài liệu hướng dẫn.
* Thông số kỹ thuật của linh kiện chính: Raspberry Pi 3 mode B.

Raspberry Pi 3 là một board máy tính đơn nhỏ, giá rẻ, kích thước chỉ bằng một thẻ tín dụng, tiết kiệm điện năng (vì nguồn điện cung cấp cho RPi chỉ có 5V) được giới thiệu bởi Raspberry Pi Foundation, đi kèm với CPU, GPU, cổng USB và các chân I/O và có khả năng thực hiện một số chức năng đơn giản như một máy tính thông thường.

Sơ đồ chân của Raspberry Pi 3:



* Hình 2.1. Sơ đồ chân của Raspberry Pi 3 mode B

Cấu trúc phần cứng:



* Hình 2.2 Cấu trúc phần cứng của Raspberry Pi 3 mode B

Raspberry Pi 3 Model B đi kèm với bộ xử lý lõi tứ 64 bit, trên một board mạch với các tính năng WiFi và Bluetooth và USB.

Nó có tốc độ xử lý từ 700 MHz đến 1,4 GHz trong đó bộ nhớ RAM dao động từ 256 đến 1GB.

CPU của thiết bị này được coi là bộ não của thiết bị chịu trách nhiệm thực thi các câu lệnh dựa trên hoạt động toán học và logic.

GPU (bộ xử lý đồ họa) là một chip tiên tiến khác được tích hợp trong board mạch có chức năng tính toán hình ảnh. Board mạch được trang bị cáp lõi video Broadcam chủ yếu được sử dụng để chơi các trò chơi video thông qua thiết bị.

Pi 3 đi kèm với các chân GPIO (General Purpose Input Output) rất cần thiết để duy trì kết nối với các thiết bị điện tử khác. Các chân đầu ra đầu vào này nhận lệnh và hoạt động dựa trên chương trình của thiết bị.

Cổng Ethernet được tích hợp trên thiết bị này để thiết lập một đường giao tiếp với các thiết bị khác. Bạn có thể kết nối cổng Ethernet với bộ định tuyến để duy trì kết nối cho internet.

Board có bốn cổng USB được sử dụng có thể sử dụng để kết nối với bàn phím, chuột hoặc có thể kết nối USB 3G để truy cập internet và thẻ SD được thêm vào để lưu trữ hệ điều hành.

Đầu nối nguồn điện là một phần cơ bản của board mạch được sử dụng để cung cấp nguồn 5 V cho bo mạch. Bạn có thể sử dụng bất kỳ nguồn nào để thiết lập nguồn cho board mạch, tuy nhiên, bạn ưu tiên kết nối cáp nguồn qua cổng USB của máy tính xách tay để cung cấp 5 V.

Pi 3 hỗ trợ hai tùy chọn kết nối bao gồm HDMI và RCA Video. Cổng HDMI được sử dụng để kết nối LCD hoặc TV, có thể hỗ trợ cáp phiên bản 1.3 và 1.4. Cổng RCA Video được sử dụng để kết nối các màn hình TV đời cũ sử dụng jack cắm 3,5mm mà không hỗ trợ cổng HDMI.

Cổng USB được tích hợp trên board mạch được sử dụng để khởi động thiết bị. Vì RPi chạy hệ điều hành Linux, nên chỉ cần cắm bàn phím và chuột vào là có thể sử dụng mà không cần cài thêm driver.

Hệ điều hành cho Raspberry Pi 3:

Graphical user interface, application

Description automatically generated

* Hình 2.3 Hệ điều hành cho Raspberry Pi 3 mode B

Hệ điều hành Raspbian Linux chính thức chạy trên Pi 3. Hệ điều hành bên thứ ba khác có thể hoạt động trên thiết bị này là RISC OS, OSMC, Windows 10 loT Core, Ubuntu MATE,…

Thông số kỹ thuật chính:

* 1,4 GHz 64 bit, Bộ xử lý lõi tứ Broadcom BCM2387 ARM Cortex-A53, nhanh hơn 10 lần so với Raspberry Pi 1.
* RAM 1GB (LPDDR2 SDRAM) cho phép bạn chạy các ứng dụng nâng cao.
* 802.11 b/g/n Wireless LAN.
* On-board Bluetooth 4.1.
* 4 cổng USB 2.0.
* Ethernet 300Mbit/s.
* 40 chân GPIO.
* HDMI hỗ trợ phiên bản 1.3/1.4 và Composite RCA (PAL and NTSC).
* 10/100 BaseT Ethernet socket.
* Camera interface (CSI), để kết nối với camera.
* Display interface (DSI): được sử dụng để kết nối Raspberry Pi với màn hình cảm ứng.
* Khe cắm thẻ microSD: đễ lưu trữ dữ liệu.
* Micro USB power source.
* VideoCore IV multimedia/3D graphics core @ 400MHz/300MHz.

### Màn hình cảm ứng điện dung 5 inch

- Chức năng: Trình chiếu bảng chọn sản phẩm, nhận lựa chọn sản phẩm của khách hàng, gửi dữ liệu về máy tính nhúng Raspberry Pi.

- Linh kiện chính: 5 inch Led Display Waveshare.

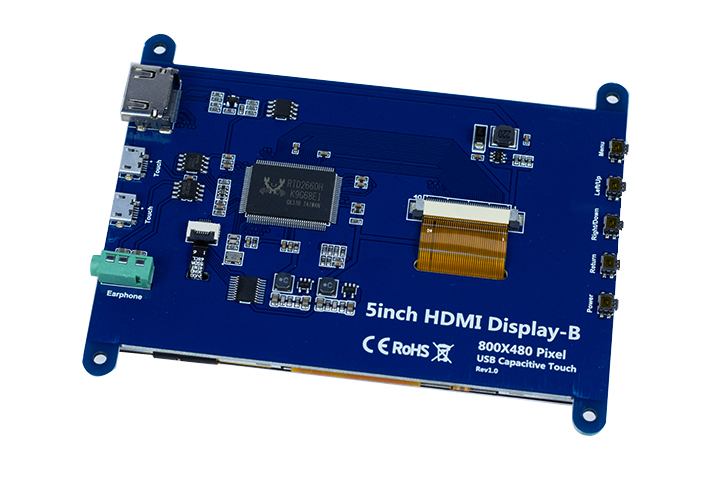
- Lý do chọn: Dễ sử dụng.

- Thông số kỹ thuật của linh kiện chính:

* + - Kích thước: 5 inch
    - Độ phân giải phần cứng: 800x480 pixel.
    - Thực hiện chức năng cảm ứng qua cổng micro USB
    - Phù hợp cho các phiên bản của Raspberry: Pi B, B+, 2, Raspberry Pi 3
    - Cổng kết nối : HDMI.
    - Hỗ trợ Ban ana Pi / Banana Pro (có hệ điều hành Lubuntu, Raspbian)
    - Hỗ trợ BB Black (có hệ điều hành Angstrom đi kèm)
    - Có driver riêng để cài đặt tích hợp với các thiết bị khác
    - Có công tắc bật/tắt màn hình
    - Khối lượng: 200 gram



* Hình 2.4 Mặt trước màn hình 5 inch Waveshare



* Hình 2.5 Mặt sau màn hình 5 inch Waveshare

### Nguồn tổ ong 12VDC 30A

- Chức năng: Cung cấp nguồn cho toàn bộ hệ thống hoạt động.

- Linh kiện chính: Nguồn tổ ong 12V 30A.

- Lý do chọn: Giá thành vừa phải, dễ sử dụng, chịu tải cao.

- Thông số kỹ thuật của linh kiện chính:

* Điện áp đầu vào 110VAC hoặc 220VAC thay đổi qua nút gạt chìm bên trong.
* Điện áp đầu ra 9VDC~14VDC chỉnh được thông qua núm chỉnh ADJ có sẵn trên nguồn.
* Công suất: 360W.
* Dòng điện đầu ra: Max 30A.
* Tần số đầu vào: 50-60Hz.
* Nhiệt độ hoạt động: 0 ℃ ~ 40 ℃.
* Kích thước sản phẩm: 215 \* 13 \* 50 mm.
* Cổng vào: N (dây nguội) – L (dây nóng ) – Cổng tiếp chất chống dòng rò.
* Cổng ra: 3 (+) và 3 (-).
* Hình 2.6 Nguồn tổ ong 12V 30A

### Module cảm biến vật cản hồng ngoại

- Chức năng: Phát hiện sản phẩm, đếm sản phẩm.

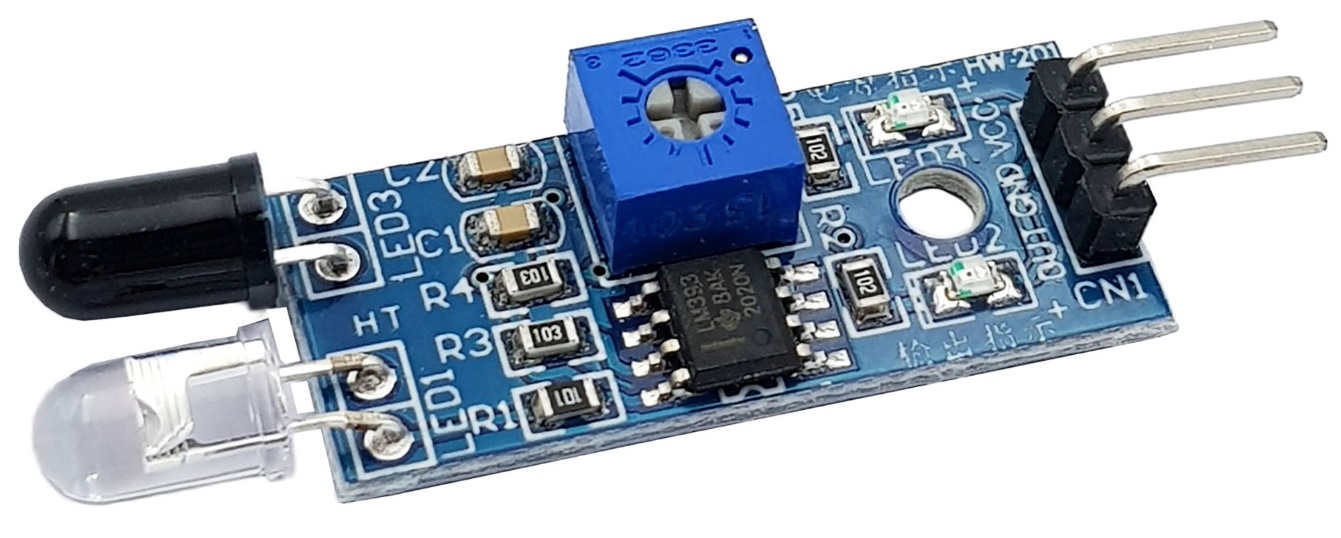
- Linh kiện chính: LM393 .

- Lí do chọn: Tiêu hao điện năng thấp, giá thành rẻ, dễ tiếp cận.

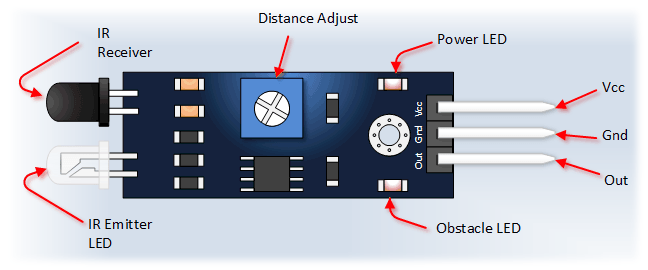
- Thông số kỹ thuật của linh kiện chính:

* + - Bộ so sánh sử dụng LM393, làm việc ổn định
    - Điện áp làm việc: 3.3V – 5V DC.
    - Khi bật nguồn, đèn báo nguồn màu đỏ sáng.
    - Lỗ vít 3 mm, dễ dàng cố định, lắp đặt.

Kích thước: 3.2cm \* 1.4cm



* Hình 2.7 Module cảm biến hồng ngoại phát hiện vật cản



* Hình 2.8 Cấu tạo module cảm biến hồng ngoại phát hiện vật cản

### Module điều khiển động cơ DC L298N

- Chức năng: Điều khiển tốc độ, chiều của động cơ DC.

- Linh kiện chính: IC L298.

- Lí do chọn: Tiêu hao điện năng thấp, giá thành rẻ, bền, dễ tiếp cận.

- Thông số kỹ thuật của linh kiện chính:

Driver: L298N tích hợp hai mạch cầu H

Điện áp điều khiển: +5 V ~ +35 V

Dòng tối đa cho mỗi cầu H: 2A

Điện áp tín hiệu điều khiển: +5 V ~ +7 V

Dòng tín hiệu điều khiển: 0 ~ 36mA

Công suất hao phí: 20W (khi nhiệt độ T = 75 ℃)

Diagram, schematic

Description automatically generatedNhiệt độ vận hành: -25 ℃ ~ +130 ℃Graphical user interface

Description automatically generated

* Hình 2.9 Sơ đồ mạch điện của Module L298N
* Hình 2.10 Module L298N

### Động cơ giảm tốc 12VDC

- Chức năng: Đóng ngắt hệ thống bơm.

- Linh kiện chính: Motor 12VDC.

- Lý do chọn: Cấu tạo đơn giản, giá thành rẻ, độ chính xác tốt.

- Thông số kỹ thuật của linh kiện chính:

* Điện áp định mức: DC12V.
* Dòng không tải: 0,1A.
* Công suất định mức: 5W.
* Tốc độ không tải: 60 vòng / phút.
* Điện áp thích hợp: 5-12V (25-60 vòng / phút).
* Kích thước:
  + Chiều dài thân động cơ: 67mm.
  + Đường kính thân: 39.7mm.
  + Chiều dài trục đầu ra: 21mm.
  + Đường kính trục: 5mm (trục có lỗ 2.5mm).
  + Trọng lượng: 114g.



* Hình 2.11 Động cơ giảm tốc 12VDC

### Cơ sở dữ liệu Firebase Realtime Database

- Chức năng: Giao tiếp với khối xử lý để điều khiển, nhận và gửi dữ liệu.

- Websever: Firebase.

- Giao thức: NoSQL.

- Lý do chọn: Dễ tiếp cận, miễn phí, phù hợp với các dạng đồ án nhỏ và vừa.

- Khái niệm: Firebase là dịch vụ cơ sở dữ liệu hoạt động trên nền tảng đám mây – cloud. Kèm theo đó là hệ thống máy chủ cực kỳ mạnh mẽ của Google. Chức năng chính là giúp người dùng lập trình ứng dụng bằng cách đơn giản hóa các thao tác với cơ sở dữ liệu.

Cụ thể là những giao diện lập trình ứng dụng API đơn giản. Mục đích nhằm tăng số lượng người dùng và thu lại nhiều lợi nhuận hơn.

Đặc biệt, còn là dịch vụ đa năng và bảo mật cực tốt. Firebase hỗ trợ cả hai nền tảng Android và IOS. Không có gì khó hiểu khi nhiều lập trình viên chọn Firebase làm nền tảng đầu tiên để xây dựng ứng dụng cho hàng triệu người dùng trên toàn thế giới.

Logo, company name

Description automatically generated

* Hình 2.12 Cơ sở dữ liệu Firebase Realtime Database

### Chương trình lập trình Node RED

- Chức năng: Giao tiếp với khối xử lý để điều khiển, hiện thị các chức năng.

- Websever: Node Red.

- Giao thức: MQTT, HTTP, Json,…..

- Lý do chọn: Dễ tiếp cận, miễn phí, phù hợp với các dạng đồ án nhỏ và vừa.

- Khái niệm:

Graphical user interface, application

Description automatically generatedNode RED là một công cụ lập trình dùng để kết nối các thiết bị phần cứng, API và các dịch vụ trực tuyến với nhau. Về cơ bản, đây là một công cụ trực quan được thiết kế cho IoT (Internet of Things), nhưng cũng có thể được sử dụng cho các ứng dụng khác nhằm liên kết nhanh các luồng (flow) dịch vụ khác nhau.

* Hình 2.13 Node-Red Websever

MQTT (Message Queueing Telemetry Transport) là một giao thức mạng kích thước nhỏ (lightweight), hoạt động theo cơ chế publish - subscribe (tạm dịch: xuất bản - đăng ký) theo tiêu chuẩn ISO (ISO/IEC 20922) và OASIS mở để truyền tin nhắn giữa các thiết bị. Giao thức này hoạt động trên nền tảng TCP/IP; tuy nhiên, bất kỳ giao thức mạng nào cung cấp các kết nối theo tuần tự, không mất dữ liệu (lossless), kết nối hai chiều đều có thể hỗ trợ MQTT. MQTT được thiết kế cho các kết nối cho việc truyền tải dữ liệu cho các thiết bị ở xa, các thiết bị hay vi điều khiển nhỏ có tài nguyên hạn chế,hoặc trong các ứng dụng có băng thông mạng bị hạn chế.

### Chương trình bảng tính Google Sheets

- Chức năng: Lưu trữ dữ liệu.

- Websever: Google.

- Lý do chọn: Dễ tiếp cận, miễn phí, phù hợp với các dạng đồ án nhỏ và vừa.

- Khái niệm:

* Google Sheets là một chương trình bảng tính được bao gồm như một phần của bộ phần mềm văn phòng dựa trên web miễn phí được cung cấp bởi Google trong dịch vụ Google Drive của mình.
* Dịch vụ này cũng bao gồm Google Docs và Google Slides, một trình xử lý văn bản và chương trình trình bày tương ứng. Google Sheets có sẵn dưới dạng một ứng dụng web, ứng dụng di động cho Android, iOS, Windows, BlackBerry và dưới dạng ứng dụng trên máy tính để bàn trên ChromeOS của Google. Ứng dụng này tương thích với các định dạng file Microsoft Excel.
* Ứng dụng cho phép người dùng tạo và chỉnh sửa các file trực tuyến trong khi cộng tác với những người dùng khác trong thời gian thực. Chỉnh sửa được theo dõi bởi người dùng với lịch sử sửa đổi trình bày các thay đổi. Vị trí của biên tập viên được tô sáng bằng màu sắc và con trỏ dành riêng cho biên tập viên và hệ thống quyền điều chỉnh những gì người dùng có thể làm.



* Hình 2.14 Chương trình bảng tính Google Sheets

# THIẾT KẾ VÀ THI CÔNG

## Thi công

### Linh kiện sử dụng

* Máy tính nhúng Raspberry Pi 3 mode B.
* Màn hình Waveshare 5 inch cảm ứng điện dung.
* Module cảm biến hồng ngoại phát hiện vật cản.
* Module điều khiển động cơ L298N.
* Nguồn tổ ong 12VDC 30A
* Động cơ giảm tốc 12VDC
* Điện trở và dây điện

### Thi công mạch điều khiển

* Hình 3.1 Mạch điều khiển

### Thi công mô hình

* Hình 3.2 Mô hình đã thi công

## A picture containing indoor, wall, open, appliance Description automatically generatedThiết kế

* Hình 3.3 Mô hình đã thi công

### Thiết kế giao diện người dùng

Calendar

Description automatically generatedGraphical user interface, text, application

Description automatically generated

* Hình 3.4 Giao diện nhập hàng
* Hình 3.5 Giao diện người dùng

### Thiết kế giao diện giám sát Node Red

Chart, waterfall chart

Description automatically generatedDiagram

Description automatically generated

* Hình 3.6 Giao diện giám sát
* Hình 3.7 Giao diện Node RED

# GIẢI THUẬT VÀ ĐIỀU KHIỂN

## Mô tả hoạt động của hệ thống

* Tự động điều chỉnh sản phẩm sau khi châm hàng.
* Chọn sản phẩm trên màn hình.
* Kiểm tra sản phẩm, số lượng đã chọn.
* Thanh toán qua ví điện tử.
* Sau khi chọn sản phẩm máy tính nhúng Raspberry Pi sẽ xửu lí và kích xung cho module L298N điều khiển động cơ đẩy sản phẩm ra khỏi khay chứa theo thứ tự đã cài đặt.
* Khi đã bán được sản phẩm thì máy tính nhúng Raspberry Pi sẽ gửi dữ liệu lên cơ sỏ dữ liệu Firebase Realtime Database.
* Thống kê, hiển thị và giám sát số lượng, doanh thu từ xa qua Node RED.
* Lưu trữ dữ liệu thu được qua Google Sheets

## Lưu đồ giải thuật

### Diagram Description automatically generatedLưu đồ giải thuật trên chương trình chính

### Lưu đồ giải thuật chương trình thực thi

Diagram

Description automatically generated

# THỰC NGHIỆM

## Tiến trình thực nghiệm

Bước 1: Khởi động mô hình, khởi động Cơ sở dữ liệu, Node RED.

Bước 2: Kiểm tra các module, kiểm tra cơ sở dữ liệu, kiểm tra giao diện Node-Red.

Bước 3: Chọn sản phẩm trên màn hình.

Bước 4: Đợi hệ thống xử lí và trả sản phẩm.

Bước 5: So sánh số lượng trả về và số lượng đã mua.

Bước 6: Kiểm tra dữ liệu được gửi lên cơ sở dữ liệu, Google Sheets và Node RED

Bước 7: Kết thúc kiểm tra.

## Kết quả thực nghiệm

A picture containing text, indoor, appliance

Description automatically generated

* Hình 5.1 Kết quả thực nghiệm

Graphical user interface, text, application, email

Description automatically generated Table

Description automatically generated

* Hình 5.2 Cơ sở dữ liệu
* Hình 5.3 Kết quả Google Sheets

# KẾT LUẬN

## Ưu điểm

* Giao diện trực quan, dễ sử dụng.
* Cơ chế hoạt động đơn giản, dễ tiếp cận và ổn định.
* Giá thành thấp, phù hợp cho thị trường khách hàng vừa và nhỏ.
* Có thể giám sát từ xa qua mạng internet.

## Nhược điểm

* Chưa có nhiều chức năng hỗ trợ người bán.
* Chưa có chức năng bảo mật và chống trộm.
* Mô hình chưa được tối ưu hóa.
* Cần phải phát triển thêm nếu muốn đưa ra thực tế

## Hướng phát triển

* Tiếp tục nghiên cứu cho ra mô hình có chức năng nhiều hơn, độ ổn định cao hơn.
* Áp dụng một số hệ thống tiên tiến, những module có công suất cao hơn.

Tài liệu tham khảo

Tiếng việt

1. Lê Ngọc Bích, Phạm Quang Huy (2016), Ứng dụng vi xử lý và vi điều khiển, Nhà xuất bản bách khoa Hà Nội.
2. TS. Nguyễn Tất Bảo Thiện, KS. Phạm Quang Huy, Lập Trình Hệ Thống Nhúng Với Raspberry , Nhà xuất bản Thanh Niên.
3. Node RED là gì? Những kiến thức cơ bản về Node-RED

<https://dientuviet.com/node-red-la-gi/>

1. Tìm hiểu về Firebase Realtime Database  
   <https://viblo.asia/p/tim-hieu-ve-firebase-realtime-database-Az45bxzVZxY>

# Phụ lục 1: Chương trình điều khiển

## Code Raspberry Pi ( Chương trình điều khiển)

import RPi.GPIO as GPIO

import time

import pyrebase

from datetime import datetime

from flask import Flask, render\_template, request, redirect, url\_for

config = {

      "apiKey": "AIzaSyCz-FHXfyY1nr5tkQfiw8G6rqJYipF24JA",

      "authDomain": "vending-machine-6f41b.firebaseapp.com",

      "databaseURL": "https://vending-machine-6f41b-default-rtdb.firebaseio.com",

      "projectId": "vending-machine-6f41b",

      "storageBucket": "vending-machine-6f41b.appspot.com",

      "messagingSenderId": "866803929412",

      "appId": "1:866803929412:web:630ba649987ebc142d4711",

      "measurementId": "G-9T3KV64TYF"

     }

firebase = pyrebase.initialize\_app(config)

storage = firebase.storage()

database = firebase.database()

app = Flask(\_\_name\_\_)

enA = 18

in1 = 23

in2 = 24

in3 = 16

in4 = 20

enB = 12

enC = 13

in5 = 19

in6 = 26

cb1 = 22

cb2 = 27

cb3 = 5

temp1=1

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

GPIO.setup(in1,GPIO.OUT)

GPIO.setup(in2,GPIO.OUT)

GPIO.setup(in3,GPIO.OUT)

GPIO.setup(in4,GPIO.OUT)

GPIO.setup(enA,GPIO.OUT)

GPIO.setup(enB,GPIO.OUT)

GPIO.setup(in5,GPIO.OUT)

GPIO.setup(in6,GPIO.OUT)

GPIO.setup(enC,GPIO.OUT)

GPIO.output(in1,GPIO.LOW)

GPIO.output(in2,GPIO.LOW)

GPIO.output(in3,GPIO.LOW)

GPIO.output(in4,GPIO.LOW)

GPIO.output(in5,GPIO.LOW)

GPIO.output(in6,GPIO.LOW)

GPIO.setup(cb1, GPIO.IN)

GPIO.setup(cb2, GPIO.IN)

GPIO.setup(cb3, GPIO.IN)

p=GPIO.PWM(enA,10000)

p.start(50)

q=GPIO.PWM(enB,10000)

q.start(50)

k=GPIO.PWM(enC,10000)

k.start(50)

id\_1 = {

           'name' : 'Coca',

           'price' : 5000,

           'quantity' : 0,

           'stock' : 0,

           'total' : 0

        }#end id\_1

id\_2 = {

           'name' : 'Pepsi',

           'price' : 4000,

           'quantity' : 0,

           'stock' : 0,

           'total' : 0

        }#end id\_2

id\_3 = {

           'name' : 'Mitom',

           'price' : 3000,

           'quantity' : 0,

           'stock' : 0,

           'total' : 0

        }#end id\_3

slot\_1 = None

slot\_2 = None

slot\_3 = None

DS1 = 0

DS2 = 0

DS3 = 0

id1Input = 0

id2Input = 0

id3Input = 0

productPlus = 0

subTotal = 0

def prepareBegin():

    temp = 0

    while GPIO.input(cb1) != 0:

        GPIO.output(in1,GPIO.LOW)

        GPIO.output(in2,GPIO.HIGH)

        temp += 1

        time.sleep(1)

        if temp > 5:

            GPIO.output(in1,GPIO.LOW)

            GPIO.output(in2,GPIO.LOW)

            break

        else:

            GPIO.output(in1,GPIO.LOW)

            GPIO.output(in2,GPIO.LOW)

    time.sleep(2)

    temp = 0

    while GPIO.input(cb2) != 0:

        GPIO.output(in3,GPIO.LOW)

        GPIO.output(in4,GPIO.HIGH)

        temp += 1

        time.sleep(1)

        if temp > 5:

            GPIO.output(in3,GPIO.LOW)

            GPIO.output(in4,GPIO.LOW)

            break

        else:

            GPIO.output(in3,GPIO.LOW)

            GPIO.output(in4,GPIO.LOW)

    time.sleep(2)

    temp = 0

    while GPIO.input(cb3) != 0:

        GPIO.output(in5,GPIO.LOW)

        GPIO.output(in6,GPIO.HIGH)

        temp += 1

        time.sleep(1)

        if temp > 5:

            GPIO.output(in5,GPIO.LOW)

            GPIO.output(in6,GPIO.LOW)

            break

        else:

            GPIO.output(in5,GPIO.LOW)

            GPIO.output(in6,GPIO.LOW)

@app.route('/pickup/<status>/<productPlus>')

def pick(productPlus, status):

    global slot\_1

    global slot\_2

    global slot\_3

    global id\_1

    global id\_2

    global id\_3

    if productPlus == 'item1' and status == 'plus':

        cart(1)

        id\_1['total'] = id\_1['price']\*id\_1['quantity']

        subTotal = id\_1['total'] + id\_2['total'] + id\_3['total']

        return render\_template('index.html', id\_1=id\_1, id\_2=id\_2, id\_3=id\_3, slot\_1=slot\_1, slot\_2=slot\_2, slot\_3=slot\_3, subTotal=subTotal)

    elif productPlus == 'item1' and status == 'minus':

        minusCart(1)

        id\_1['total'] = id\_1['price']\*id\_1['quantity']

        subTotal = id\_1['total'] + id\_2['total'] + id\_3['total']

        return render\_template('index.html',id\_1=id\_1, id\_2=id\_2, id\_3=id\_3, slot\_1=slot\_1, slot\_2=slot\_2, slot\_3=slot\_3, subTotal=subTotal)

    if productPlus == 'item2' and status == 'plus':

        cart(2)

        id\_2['total'] = id\_2['price']\*id\_2['quantity']

        subTotal = id\_1['total'] + id\_2['total'] + id\_3['total']

        return render\_template('index.html',id\_1=id\_1, id\_2=id\_2, id\_3=id\_3, slot\_1=slot\_1, slot\_2=slot\_2, slot\_3=slot\_3, subTotal=subTotal)

    elif productPlus == 'item2' and status == 'minus':

        minusCart(2)

        id\_2['total'] = id\_2['price']\*id\_2['quantity']

        subTotal = id\_1['total'] + id\_2['total'] + id\_3['total']

        return render\_template('index.html',id\_1=id\_1, id\_2=id\_2, id\_3=id\_3, slot\_1=slot\_1, slot\_2=slot\_2, slot\_3=slot\_3, subTotal=subTotal)

    if productPlus == 'item3' and status == 'plus':

        cart(3)

        id\_3['total'] = id\_3['price']\*id\_3['quantity']

        subTotal = id\_1['total'] + id\_2['total'] + id\_3['total']

        return render\_template('index.html',id\_1=id\_1, id\_2=id\_2, id\_3=id\_3, slot\_1=slot\_1, slot\_2=slot\_2, slot\_3=slot\_3, subTotal=subTotal)

    elif productPlus == 'item3' and status == 'minus':

        minusCart(3)

        id\_3['total'] = id\_3['price']\*id\_3['quantity']

        subTotal = id\_1['total'] + id\_2['total'] + id\_3['total']

        return render\_template('index.html',id\_1=id\_1, id\_2=id\_2, id\_3=id\_3, slot\_1=slot\_1, slot\_2=slot\_2, slot\_3=slot\_3, subTotal=subTotal)

def cart(id\_input):

        global slot\_1

        global slot\_2

        global slot\_3

        global id\_1

        global id\_2

        global id\_3

#         id\_pick = dict()

        if id\_input == 1 and id\_1['quantity'] < id\_1['stock']:

            id\_1['quantity'] = id\_1['quantity'] + 1

            if slot\_1 == None or slot\_1['name'] == id\_1['name']:

                slot\_1 = id\_1

            elif slot\_2 == None or slot\_2['name'] == id\_1['name']:

                slot\_2 = id\_1

            elif slot\_3 == None or slot\_3['name'] == id\_1['name']:

                slot\_3 = id\_1

        elif id\_input == 2 and id\_2['quantity'] < id\_2['stock']:

            id\_2['quantity'] = id\_2['quantity'] + 1

            if slot\_1 == None or slot\_1['name'] == id\_2['name']:

                slot\_1 = id\_2

            elif slot\_2 == None or slot\_2['name'] == id\_2['name']:

                slot\_2 = id\_2

            elif slot\_3 == None or slot\_3['name'] == id\_2['name']:

                slot\_3 = id\_2

        elif id\_input == 3 and id\_3['quantity'] < id\_3['stock']:

            id\_3['quantity'] = id\_3['quantity'] + 1

            if slot\_1 == None or slot\_1['name'] == id\_3['name']:

                slot\_1 = id\_3

            elif slot\_2 == None or slot\_2['name'] == id\_3['name']:

                slot\_2 = id\_3

            elif slot\_3 == None or slot\_3['name'] == id\_3['name']:

                slot\_3 = id\_3

        return slot\_1, slot\_2, slot\_3

def minusCart(id\_input):

        global slot\_1

        global slot\_2

        global slot\_3

        global id\_1

        global id\_2

        global id\_3

        id\_pick = dict()

        if id\_input == 1 and id\_1['quantity'] > 0:

            id\_1['quantity'] = id\_1['quantity']-1

            if slot\_1['name'] == id\_1['name'] and id\_1['quantity'] > 0:

                slot\_1['quantity'] = id\_1['quantity']

            elif slot\_1['name'] == id\_1['name'] and id\_1['quantity'] == 0:

                slot\_1 = None

            elif slot\_2['name'] == id\_1['name'] and id\_1['quantity'] > 0:

                slot\_2['quantity'] = id\_1['quantity']

            elif slot\_2['name'] == id\_1['name'] and id\_1['quantity'] == 0:

                slot\_2 = None

            elif slot\_3['name'] == id\_1['name'] and id\_1['quantity'] > 0:

                slot\_3['quantity'] = id\_1['quantity']

            elif slot\_3['name'] == id\_1['name'] and id\_1['quantity'] == 0:

                slot\_3 = None

        elif id\_input == 2 and id\_2['quantity'] > 0:

            id\_2['quantity'] = id\_2['quantity']-1

            if slot\_1['name'] == id\_2['name'] and id\_2['quantity'] > 0:

                slot\_1['quantity'] = id\_2['quantity']

            elif slot\_1['name'] == id\_2['name'] and id\_2['quantity'] == 0:

                slot\_1 = None

            elif slot\_2['name'] == id\_2['name'] and id\_2['quantity'] > 0:

                slot\_2['quantity'] = id\_2['quantity']

            elif slot\_2['name'] == id\_2['name'] and id\_2['quantity'] == 0:

                slot\_2 = None

            elif slot\_3['name'] == id\_2['name'] and id\_2['quantity'] > 0:

                slot\_3['quantity'] = id\_2['quantity']

            elif slot\_3['name'] == id\_2['name'] and id\_2['quantity'] == 0:

                slot\_3 = None

        elif id\_input == 3 and id\_3['quantity'] > 0:

            id\_3['quantity'] = id\_3['quantity']-1

            if slot\_1['name'] == id\_3['name'] and id\_3['quantity'] > 0:

                slot\_1['quantity'] = id\_3['quantity']

            elif slot\_1['name'] == id\_3['name'] and id\_3['quantity'] == 0:

                slot\_1 = None

            elif slot\_2['name'] == id\_3['name'] and id\_3['quantity'] > 0:

                slot\_2['quantity'] = id\_3['quantity']

            elif slot\_2['name'] == id\_3['name'] and id\_3['quantity'] == 0:

                slot\_2 = None

            elif slot\_3['name'] == id\_3['name'] and id\_3['quantity'] > 0:

                slot\_3['quantity'] = id\_3['quantity']

            elif slot\_3['name'] == id\_3['name'] and id\_3['quantity'] == 0:

                slot\_3 = None

        return slot\_1, slot\_2, slot\_3

@app.route('/')

def index():

        global slot\_1

        global slot\_2

        global slot\_3

        global id\_1

        global id\_2

        global id\_3

        global subTotal

        global subQuantity

        slot\_1 = None

        slot\_2 = None

        slot\_3 = None

        quanS1 = 0

        quanS2 = 0

        quanS3 = 0

        id\_1['quantity'] = 0

        id\_2['quantity'] = 0

        id\_3['quantity'] = 0

        id\_1['total'] = 0

        id\_2['total'] = 0

        id\_3['total'] = 0

        subTotal = 0

        prepareBegin()

        ton()

        return render\_template('index.html', id\_1=id\_1, id\_2=id\_2, id\_3=id\_3, slot\_1=slot\_1, slot\_2=slot\_2, slot\_3=slot\_3)

@app.route('/pay')

def pay():

    return render\_template('pay.html')

@app.route('/payment')

def action():

    global id\_1, id\_2, id\_3

    global cb1, cb2, cb3

    global in1, in2, in3, in4

    count1 = int(id\_1['quantity'])

    for i in range(1, count1 + 1 ):

        while(True):

            time.sleep(0.25)

            GPIO.output(in1,GPIO.LOW)

            GPIO.output(in2,GPIO.HIGH)

            if GPIO.input(cb1) != 0:

                GPIO.output(in1,GPIO.LOW)

                GPIO.output(in2,GPIO.HIGH)

                time.sleep(0.25)

                break

        while(True):

            if GPIO.input(cb1) == 0:

                GPIO.output(in1,GPIO.LOW)

                GPIO.output(in2,GPIO.LOW)

                time.sleep(0.25)

                break

            if GPIO.input(cb1) != 0 and i == count1:

                GPIO.output(in1,GPIO.LOW)

                GPIO.output(in2,GPIO.LOW)

                time.sleep(0.25)

                break

    if id\_1['quantity'] > 0:

        SendFB(1)

        if id\_1['stock'] > 0:

            id\_1['stock'] = id\_1['stock'] - id\_1['quantity']

    time.sleep(1)

    count2 = int(id\_2['quantity'])

    for i in range(1, count2 + 1 ):

        while(True):

            time.sleep(0.25)

            GPIO.output(in3,GPIO.LOW)

            GPIO.output(in4,GPIO.HIGH)

            if GPIO.input(cb2) != 0:

                GPIO.output(in3,GPIO.LOW)

                GPIO.output(in4,GPIO.HIGH)

                time.sleep(0.25)

                break

        while(True):

            if GPIO.input(cb2) == 0:

                GPIO.output(in3,GPIO.LOW)

                GPIO.output(in4,GPIO.LOW)

                time.sleep(0.25)

                break

            if GPIO.input(cb2) != 0 and i == count2:

                GPIO.output(in3,GPIO.LOW)

                GPIO.output(in4,GPIO.LOW)

                time.sleep(0.25)

                break

    if id\_2['quantity'] > 0:

        SendFB(2)

        if id\_2['stock'] > 0:

            id\_2['stock'] = id\_2['stock'] - id\_2['quantity']

    time.sleep(1)

    count3 = int(id\_3['quantity'])

    for i in range(1, count3 + 1 ):

        while(True):

            time.sleep(0.25)

            GPIO.output(in5,GPIO.LOW)

            GPIO.output(in6,GPIO.HIGH)

            if GPIO.input(cb3) != 0:

                GPIO.output(in5,GPIO.LOW)

                GPIO.output(in6,GPIO.HIGH)

                time.sleep(0.25)

                break

        while(True):

            if GPIO.input(cb3) == 0:

                GPIO.output(in5,GPIO.LOW)

                GPIO.output(in6,GPIO.LOW)

                time.sleep(0.25)

                break

            if GPIO.input(cb3) != 0 and i == count3:

                GPIO.output(in5,GPIO.LOW)

                GPIO.output(in6,GPIO.LOW)

                time.sleep(0.25)

                break

    if id\_3['quantity'] > 0:

        SendFB(3)

        if id\_3['stock'] > 0:

            id\_3['stock'] = id\_3['stock'] - id\_3['quantity']

    doanhso()

    return redirect(url\_for('index'))

def doanhso():

    global id\_1, id\_2, id\_3

    global DS1, DS2, DS3

    tg = datetime.now()

    dateStr = tg.strftime("%d-%m-%Y")

    DS1 = DS1 + id\_1['quantity']

    DS2 = DS2 + id\_2['quantity']

    DS3 = DS3 + id\_3['quantity']

    sendDs = {

        'Time' : dateStr,

        'DoanhsoItem1' : DS1,

        'DoanhsoItem2' : DS2,

        'DoanhsoItem3' : DS3

        }

    database.child('DoanhSo').set(sendDs)

def ton():

    global id\_1, id\_2, id\_3

    stockDb = {

        'StockItem1' : id\_1['stock'],

        'StockItem2' : id\_2['stock'],

        'StockItem3' : id\_3['stock']

        }

    database.child('Ton').set(stockDb)

def SendFB(id\_com):

    global id\_1, id\_2, id\_3

    if id\_com == 1:

        sendInfo = id\_1

        tg = datetime.now()

        dateStr = tg.strftime("%d%m%Y%H%M%S")

        sendInfo["OrderCode"] = "CC"+ dateStr

        database.child("DH").set(sendInfo)

    elif id\_com == 2:

        sendInfo = id\_2

        tg = datetime.now()

        dateStr = tg.strftime("%d%m%Y%H%M%S")

        sendInfo["OrderCode"] = "PS"+ dateStr

        database.child("DH").set(sendInfo)

    elif id\_com == 3:

        sendInfo = id\_3

        tg = datetime.now()

        dateStr = tg.strftime("%d%m%Y%H%M%S")

        sendInfo["OrderCode"] = "MT"+ dateStr

        database.child("DH").set(sendInfo)

@app.route('/stock')

def stock():

    global id\_1, id\_2, id\_3

    global id1Input, id2Input, id3Input

    id\_1['stock'] = id\_1['stock'] + id1Input

    id\_2['stock'] = id\_2['stock'] + id2Input

    id\_3['stock'] = id\_3['stock'] + id3Input

    id1Input = 0

    id2Input = 0

    id3Input = 0

    return render\_template('stock.html', id\_1=id\_1, id\_2=id\_2, id\_3=id\_3)

@app.route('/stock/<product>/<status>')

def stockin(product, status):

    global id\_1, id\_2, id\_3

    global id1Input, id2Input, id3Input

    if product == 'item1':

        if status == 'plus':

            id1Input = id1Input + 1

        elif status == 'minus' and id1Input > 0:

            id1Input = id1Input - 1

    if product == 'item2':

        if status == 'plus':

            id2Input = id2Input + 1

        elif status == 'minus' and id2Input > 0:

            id2Input =id2Input - 1

    if product == 'item3':

        if status == 'plus':

            id3Input =id3Input + 1

        elif status == 'minus' and id3Input > 0:

            id3Input = id3Input + 1

    return render\_template('stock.html', input1=id1Input, input2=id2Input, input3=id3Input, id\_1=id\_1, id\_2=id\_2, id\_3=id\_3)

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(host = '0.0.0.0',debug=True)

    app.config["TEMPLATES\_AUTO\_RELOAD"] = True