BỘ GIÁO DỤC & ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ CÔNG NGHIỆP – Y SINH

ĐỒ ÁN MÔN HỌC

NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ TRUYỀN THÔNG

MÔN HỌC: HỆ THỐNG NHÚNG TRONG CÔNG NGHIỆP

ĐỀ TÀI:

HỆ THỐNG ĐIỀU KHIỂN CÁC THIẾT BỊ THÔNG QUA MOBILE APP

GVHD: TS. Nguyễn Thanh Nghĩa

Nhóm sinh viên thực hiện: Nhóm 01

HỌ VÀ TÊN	MSSV
Lê Công Tuấn Anh	20161290
Lâm Tấn Dũng	20161299
Nguyễn Đức Quân	20161360
Phạm Đức Thắng	20161373

Tp. Hồ Chí Minh – 11/2022

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN

1.1 Đặt vấn đề

Xã hội ngày càng hiện đại, con người phải tất bật hơn với công việc và cuộc sống, tạo cho họ nhiều áp lực hơn. Đôi khi, vì quá bận rộn, chúng ta có thể quên tắt điện và các thiết bị trong nhà khi đi làm hay đi học, việc này gây ra lãng phí lớn. Hay là trong khi chúng ta bận làm việc nhưng có một hệ thống từ xa ví dụ như hệ thống bơm nước cho vườn cây, hệ thống đèn chiếu sáng ở nhà, V.v… cần chúng ta khởi động vào cùng thời điểm. Những vẫn đề này tuy nhỏ nhưng việc xử lý lại tốn thời gian do chúng ta phải di chuyển xa và sẽ gặp khó khăn khi chúng ta đang bận một việc quan trọng khác. Để xử lý cho vấm đề này, theo cách thông thường chúng ta có thể thuê người giúp việc để quản lý nhà hay công nhân để quản lý các hệ thống sản xuất từ xa, nhưng cách truyền thống này sẽ tốn một khoảng chi phí lớn hàng tháng, không phù hợp với đại đa số mọi người trong xã hội. Chính vì lý do đó, nhóm có ý tưởng là sẽ tạo nên một hệ thống điều khiển từ xa thông qua app điện thoại, chỉ cần một thao tác nhanh gọn, dù bất cứ đâu, bất cứ nơi nào chúng ta có thể bật tắt những gì chúng ta muốn, khởi động, giám sát những hệ thống sản xuất từ xa một cách dễ dàng. Với cách này sẽ rất là tiện lợi, tiết kiệm thời gian, tiết kiệm chi phí vì chỉ đầu tư một lần, và lại rất hiện đại.

1.2 Mục tiêu

- Thiết kế được một hệ thống các thiết bị cần điều khiển.

- Xây dựng chương trình điều khiển cho Raspberry thực hiện điều khiển hệ thống.

- Xây dựng giao diện điều khiển trên Mobile App.

1.3 Giới hạn

Do hạn chế về mặt thời gian và kinh phí thực hiện nên hệ thống được xây dựng của nhóm còn một số hạn chế như:

- Hệ thông các thiết bị chỉ được xây dựng trên mô phỏng bằng phần mềm Proteus.

- Giao diện app điều khiển được xây dựng trên nền tảng phần mềm đã có sẵn trên CH Play.

- Hệ thống điều khiển qua app chỉ điều khiển được qua các thao tác ON, OFF và chuyển chế độ, chưa xây dựng được hệ thống giám sát, cập nhật dữ liệu từ các cảm biến hệ thống lên app hay truyền đạt âm thanh từ xa.

1.4 Nội dung nghiên cứu

- Nội dung 1: Tìm hiểu về giao thức truyền thông MQTT
- Nội dung 2: Thiết kế mô hình các thiết bị cần điều khiển
- Nội dung 3: Chương trình điều khiển cho Raspberry bằng ngôn ngữ Python

- Nội dung 4: Thiết kế mô hình điều khiển trên Mobile App
- Nội dung 5: Nhận xét và đánh giá kết quả chạy hệ thống

1.5 Bố cục

- Chương 1: Tổng quan
- Chương 2: Cơ sở lý thuyết
- Chương 3: Tính toán và thiết kế
- Chương 4 : Lưu đồ và chương trình thực thi hệ thống
- Chương 5: Kết quả, nhận xét và đánh giá
- Chương 6: Kết luận và hướng phát triển

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Mô tả sơ bộ hệ thống điều khiển thiết bị qua Mobile App

2.1.1 Hệ thống các thiết bị cần điều khiển

- Trong một ngôi nhà những thiết bị có thể được chúng ta điều khiển từ xa như: Hệ thống đèn chiếu sáng, các đèn trang trí, cửa, hệ thống âm thanh báo động, hệ thống tưới cây,…Hay trong một cơ sở sản xuất đó là hệ thống các thiết bị máy móc sản xuất, hệ thống đèn chiếu sáng,…

- Việc ứng dụng Raspberry làm bộ xử lý điều khiển thông qua giao thức truyền thống MQTT, giúp chúng ta có thể điều khiển các thiết bị hệ thống trên chỉ bằng một thao tác nhấn trên giao diện App trên điện thoại ở một khoản cách xa.

- Với hệ thống các thiết bị điều khiển của nhóm sẽ được xây dựng dạng mô phỏng trên phần mềm Proteus với các thiết bị gần giống hoặc thay thế cho các thiết bí thực tế, ví dụ như: Buzzer thay thế cho thiết bị thực tế là còi báo động, động cơ DC 12VDC thay thế cho các thiết bị thực tế là động cơ đóng mở cửa của ngôi nhà hay các động cơ kéo rèm cửa sổ, Đèn dây tóc và các đèn led thay thế cho các thiết bị thực tế là hệ thống đèn chiếu sáng và trang trí,….

2.1.2 Giao diện điều khiển trên Mobile App

- Việc điều khiển được thực hiện qua phần mềm hỗ trợ giao thức truyền thông MQTT đó là phần mềm MQTT Dashboard

Hình 2.1a: *Giao diện ứng dụng MQTT dashboard*

Với phần mềm này ta thực hiện tạo một tài khoản sử dụng, điền địa chỉ URL của máy chủ trung gian giao tiếp MQTT. Sau đó ta thực hiện tạo giao diện điều khiển bằng các nút nhấn ảo, thực hiện liên kết điều khiển giữa các nút bằng cách liên kết với chủ đề quản lý điều khiển được tạo trên máy chủ MQTT và các lệch điều khiển các thiết bị tương ứng.

2.2 Giới thiệu phần cứng

Với đề tài thiết kế: "Hệ thống điều khiển các thiết bị thông qua Mobile App" thì những phần cứng cấu thành nên hệ thống như:

- ***Thiết bị đầu vào:*** Điện thoại thông qua các phím điều khiển tự tạo trên ứng dụng MQTT Dashboard
- ***Thiết bị đầu ra:*** Màn hình LCD 16x4, led đơn 5mm màu vàng, Buzzer, động cơ DC 12VDC, đèn dây tóc 12VAC.
- ***Thiết bị điều khiển trung tâm:*** Raspberry Pi3
- ***Các chuẩn truyền dữ liệu điều khiển:*** Chuẩn giao tiếp truyền thông MQTT, Internet
- ***Thiết bị giao diện điều khiển:*** Điện thoại (với ứng dụng MQTT Dashboard)

2.2.1 Giao thức truyền thông điệp/ giao tiếp MQTT

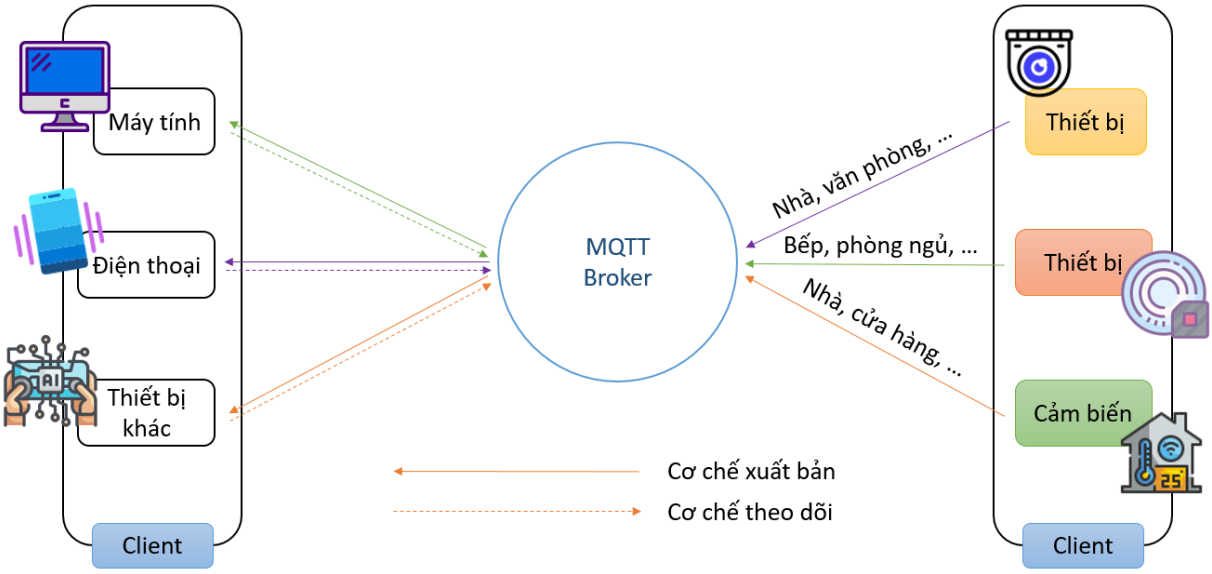
MQTT (Message Queuing Telemetry Transport) là giao thức truyền thông điệp (message) theo mô hình publish/subscribe (cung cấp / thuê bao), được sử dụng cho các thiết bị IoT với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định. Nó dựa trên một Broker (tạm dịch là "Máy chủ môi giới") "nhẹ" (khá ít xử lý) và được thiết kế có tính mở (tức là không đặc trưng cho ứng dụng cụ thể nào), đơn giản và dễ cài đặt.

MQTT được phát minh bởi Andy Stanford - Clark (IBM) và Arlen Nipper (EUROTECH) cuối năm 1999 khi mà nhiệm vụ của họ là tạo ra một giao thức sao cho sự hao phí năng lượng và băng thông là thấp nhất để kết nối đến đường ống dẫn dầu thông qua sự kết nối của vệ tinh.

Năm 2011, IBM và Eurotech đã trao lại MQTT cho một dự án của Eclipse có tên là Paho.

Năm 2013 MQTT đã được đệ trình lên OASIS (Organization for the Advancement of Structured Information Standards) để chuẩn hóa.

Một số ưu điểm nổi bật của MQTT như: băng thông thấp, độ tin cậy cao và có thể sử dụng ngay cả khi hệ thống mạng không ổn định, tốn rất ít byte cho việc kết nối với server và connection có thể giữ trạng thái open xuyên suốt, có thể kết nối nhiều thiết bị (MQTT client) thông qua một MQTT server (broker). Bởi vì giao thức này sử dụng băng thông thấp trong môi trường có độ trễ cao nên nó là một giao thức lý tưởng cho các ứng dụng IoT.



Hình 2.2a: *Kiến trúc giao thức MQTT*

2.2.2 Bộ điều khiển trung tâm Raspberry Pi

Raspberry Pi là từ để chỉ các máy tính bo mạch đơn (hay còn gọi là máy tính nhúng) kích thước chỉ bằng một thẻ tín dụng, được phát triển tại Anh bởi Raspberry Pi Foundation với mục đích ban đầu là thúc đẩy việc giảng dạy về khoa học máy tính cơ bản trong các trường học và các nước đang phát triển.

Hình 2.2b: *Raspberry Pi*

Trên thực tế có rất nhiều những ứng dụng sử dụng đến Raspberry Pi. Một số ứng dụng phổ biến và gần gũi nhất có thể kể đến như:

- Chúng ta có thể biến những chiếc Raspberry Pi này thành những chiếc đầu xem phim HD tương tự như Android Box có hỗ trợ KODI đầy đủ.
- Đặc biệt phổ biến khi chúng được dùng làm máy chơi game cầm tay.
- Raspberry Pi cũng được dùng như một VPN cá nhân.

- Trong thời đại Internet of Things như hiện nay, thật sự tiện ích khi những chiếc Raspberry Pi này có thể dùng làm thiết bị điều khiển Smart home, nó có thể giúp điều khiển mọi thiết bị điện tử trong nhà. Ngoài ra chúng cũng có thể điều khiển robot, máy in không dây và nhiều thiết bị khác.
- Những chiếc ở cứng thông thường với Raspberry Pi ngay lập tức chúng có thể chuyển thành ổ cứng mạng NAS.
- Hiển thị thời tiết, hiển thị thông tin mạng nội bộ cũng là một ứng dụng khá hữu ích của Raspberry Pi.
- Raspberry Pi cũng có thể trở thành những chiếc máy nghe nhạc hay máy đọc sách.

Trên đây chỉ là một vài ứng dụng dễ thấy của Raspberry Pi, chúng có thể được sử dụng cho vô vàn những mục đích khác nhau.

CHƯƠNG 3: TÍNH TOÁN VÀ THIẾT KẾ

3.1 Giới thiệu tổng quan về hệ thống

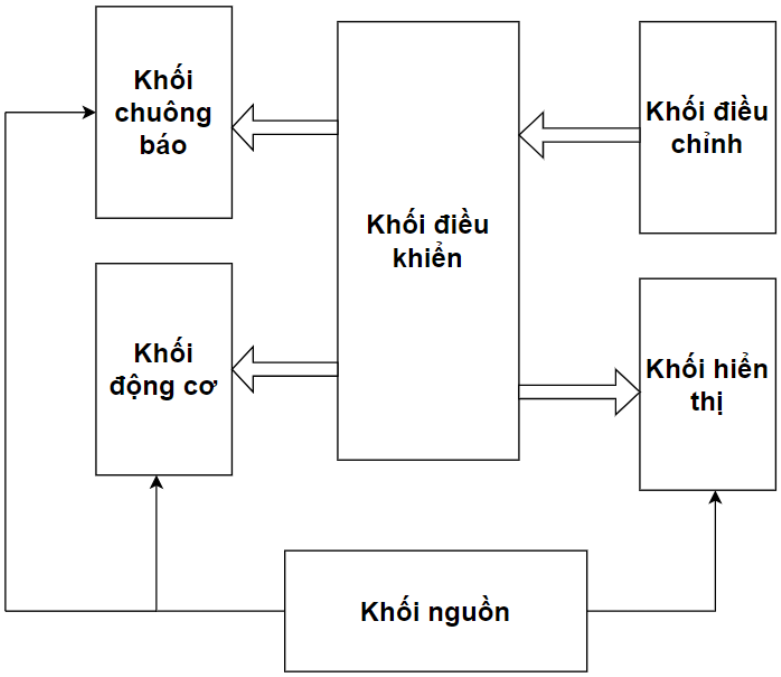
Qua phân tích ở trên, nhóm em đã đi đến quyết định thiết kế một hệ thống dùng để điều khiển động cơ DC, dàn Led và bóng đèn AC sử dụng Rasberry Pi3 điều khiển thông qua ứng dụng di động "Mqtt Dashboard- IOT and Node Red Controller".

❋ Yêu cầu:

- Sử dụng Rasberry Pi3 lập trình bằng ngôn ngữ Python
- Có thể điều khiển chính xác chiều động cơ
- Sáng tắt dàn Led và bóng đèn AC
- Hệ thống báo động sử dụng buzzer

3.2 Tính toán và thiết kế hệ thống

3.2.1 Thiết kế sơ đồ khối hệ thống



Hình 3.2a: *Sơ đồ khối hệ thống*

- *Khối nguồn:* Cung cấp nguồn cho các thiết bị của hệ thống.
- *Khối hiển thị:* Hiển thị thông tin trạng thái các thiết bị sử dụng trên màn hình LCD và 2 module đèn led và bóng đèn AC.
- *Khối chuông báo:* Báo chuông khi có tín hiệu.
- *Khối điều chỉnh:* Điều chỉnh trạng thái các thiết bị của hệ thống.

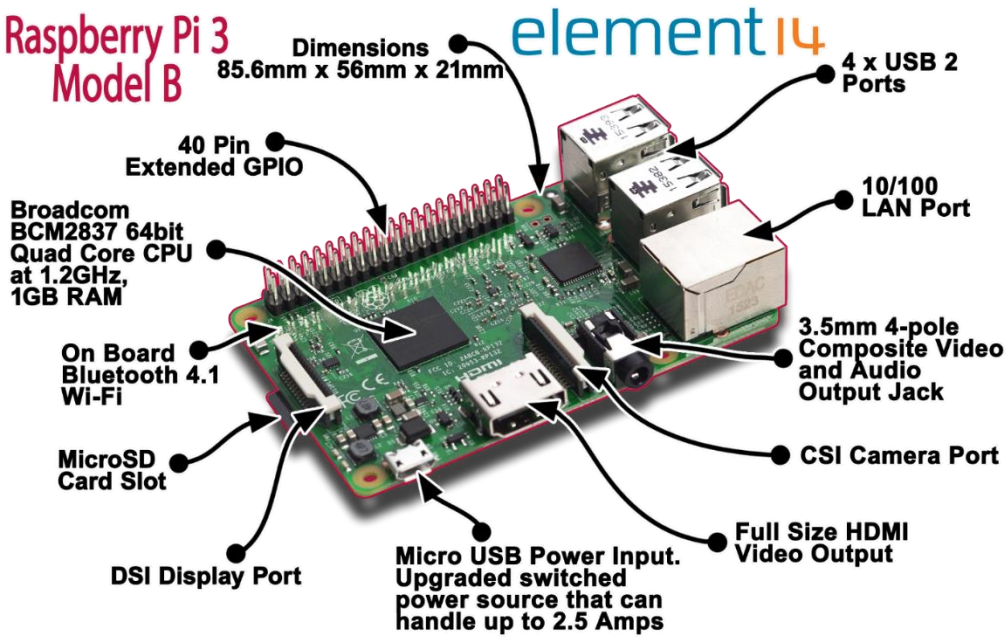
- *Khối điều khiển:* Nhận và gởi tín hiệu cho các module của hệ thống
- Khối động cơ: Điều khiển động cơ khi nhận dữ liệu từ Raspberry Pi3.

3.2.2 Tính toán và thiết kế mạch

a. Lựa chọn linh kiện cho các khối

❋ *Khối điều khiển:*

Sử dụng Rasberry Pi3: Raspberry Pi là một trong những bo mạch vật lý hàng đầu trên thị trường. Từ những người có sở thích xây dựng các dự án DIY cho đến những sinh viên lần đầu học lập trình, mọi người sử dụng Raspberry Pi hàng ngày để tương tác với thế giới xung quanh họ. Python được tích hợp sẵn trên Raspberry Pi, chúng ta có thể dùng Raspberry Pi3 để xây dựng và thiết kế các hệ thống theo yêu cầu. Vì vậy, nhóm chúng em đã thống nhất sử dụng Raspberry Pi3 để thiết kế phần cứng cho đề tài nhóm.



Hình 3.2b: *Cấu tạo của Raspberry Pi 3*

❋ *Khối nguồn:*

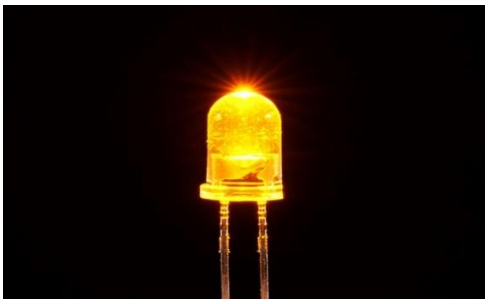
Nguồn 12VDC dùng cấp nguồn cho bóng đèn AC và IC L293D.

Nguồn 5VDC dùng cấp nguồn cho Led đơn và relay.

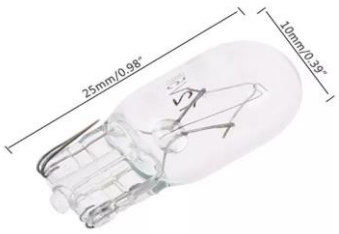
❋ *Khối hiển thị:*

- Sử dụng màn hình LCD 16x4, 5VDC, dữ liệu được đem ra và hiển thị lên màn hình LCD.

- Led đơn(vàng) 5mm: hoạt động ở mức điện áp 2V - 2.2V và dòng 10mA.



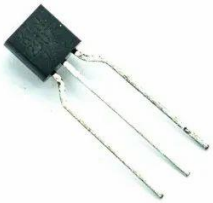
- Bóng đèn AC(vàng): hoạt động ở mức điện áp 12V



❃*Khối chuông báo:*

- Sử dụng buzzer 3.5V - 5.5VDC, dòng điện tiêu thụ: <25mA dùng để phát âm thanh cảnh báo.

- Transitor PNP 2N2907, hoạt động ở mức điện áp 5V,600mA. Dùng cấp dòng cho Buzzer nhằm điều khiển cho Buzzer ON hay OFF.



❋ *Khối điều chỉnh:* Sử dụng ứng dụng di động MQTT Dashboard dùng để cấp tín hiệu cho Raspberry, điều khiển các Module hoạt động theo yêu cầu.

❋ *Khối động cơ:*

- IC L293D

IC L293D là một IC trình điều khiển hay bộ điều khiển động cơ 16 chân thông dụng. Nó có hai mạch cầu H tích hợp có thể điều khiển đồng thời hai động cơ DC theo cả chiều kim đồng hồ và ngược chiều kim đồng hồ. Nó hoạt động như một bộ khuếch đại dòng cao vì nó lấy tín hiệu dòng điện thấp ở đầu vào và cung cấp tín hiệu dòng điện cao hơn ở đầu ra để điều khiển các tải khác nhau, ví dụ động cơ bước và động cơ DC. Các tính năng của nó bao gồm phạm vi điện áp nguồn đầu vào lớn, tín hiệu đầu vào chống nhiễu cao dòng điện đầu ra lớn, ... Các ứng dụng thực tế phổ biến của nó bao gồm trình điều khiển động cơ bước, trình điều khiển relay, trình điều khiển động cơ DC, ...

+ Điện áp động cơ Vcc2 (Vs): 4,5V đến 36V

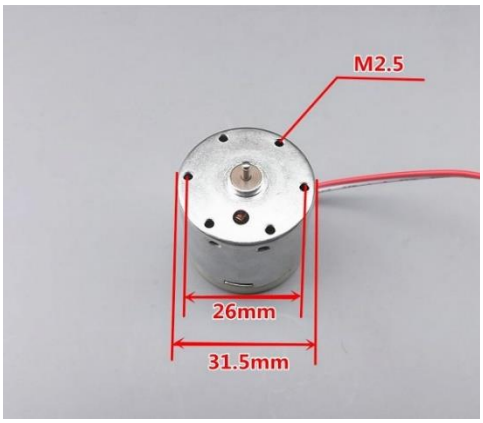
+ Dòng động cơ cao nhất tối đa: 1.2A

+ Dòng động cơ liên tục tối đa: 600mA

+ Điện áp cung cấp cho Vcc1 (vss): 4,5V đến 7V

+ Thời gian chuyển tiếp: 300ns (ở 5V và 24V)

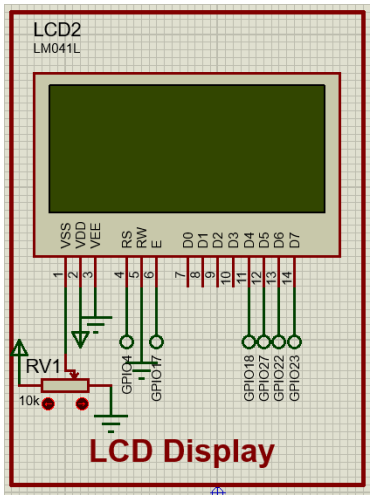
- Động cơ 12VDC



b. Tính toán thiết kế mô phỏng

❋ *Module LCD 16x4:*

- Các chân GPIO18, GPIO27, GPIO22, GPIO23 của Raspberry Pi3 lần lượt giao tiếp với 4 bus data của LCD từ D4 đến D7.
- Hai chân GPIO4 và GPIO17 làm các chân tín hiệu điều khiển LCD tương ứng với RS, E
- Do chỉ sử dụng chức năng ghi nên ta nối mass chân RW. Các chân nguồn VEE, VDD và VSS ta thực hiện nối mas, lên người và chiết áp 10kΩ.



❋ *Module 12 led đơn 5mm:*

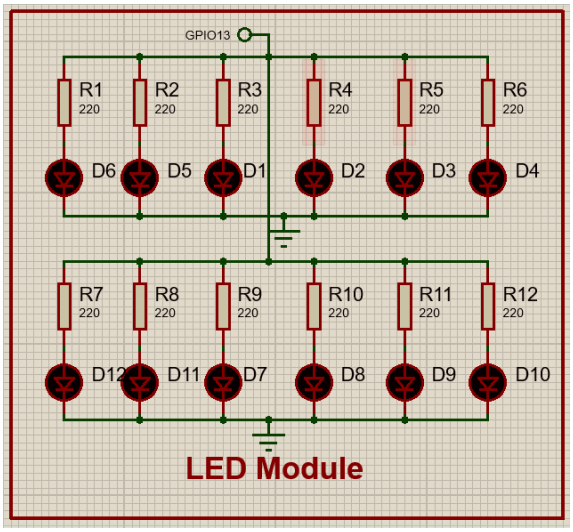
- Led đơn (vàng) hoạt động ở mức điện áp: 1,8V – 2V, dòng: 10mA – 20mA.
- Nên điện trở hạn dòng cho led được tính:

$$R_{min} = \frac{V_A - V_{led(max)}}{I_{max}} = \frac{(5-2)}{20.10^{-3}} = 150(\Omega)$$

$$R_{max} = \frac{V_A - V_{led(min)}}{I_{min}} = \frac{(5-1.8)}{10.10^{-3}} = 320(\Omega)$$

$$R_{hạn\ dòng} = \frac{R_{max} + R_{min}}{2} = \frac{320+150}{2} = 235(\Omega)$$

- Từ kết quả tính điện trở hạn dòng là 235Ω, nhưng trong thực tế không có trở mang giá trị 235Ω, nên ta thực hiện chọn trở giá trị 220Ω
- Các led được mắc song song với nhau, Anode các led được nối đến chân GPIO13 của Raspberry Pi3 và chân Cathode được nối mass

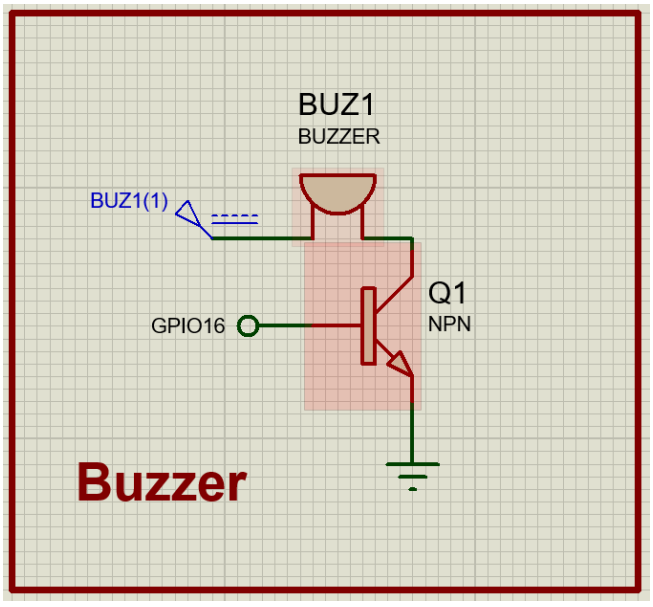


Module Buzzer

- Nối chân dương của Buzzer lên Vcc, chân còn lại nối với cực C của Transistor
- Transistor NPN có cực B nối với chân GPIO16 của Raspberry Pi3, cực E nối mass.
- Khi xuất mức 0 ra cực B=> dòng phân cực IB=0:
- Do mức điện áp giữa cực B và cực C bằng nhau VE=VC

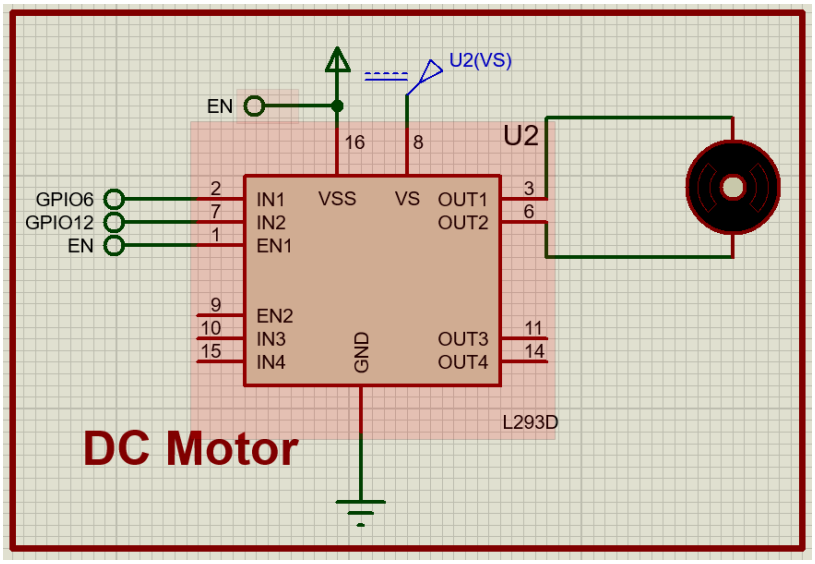
=> không có sự chênh lệch về điện áp nên Transistor ngưng dẫn

Khi xuất mức 0 ra cực B thì dòng phân cực IB khác 0=>Transistor dẫn bão hòa và VE=0 nên Buzzer hoạt động



❋ *Module điều khiển động cơ:*

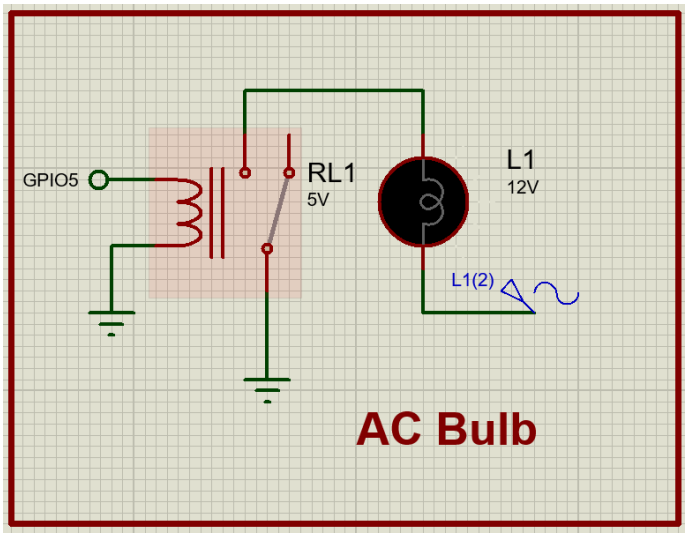
- Chân Ground (GND) phải được nối đất. IC này có 2 chân nguồn, một là Vss (Vcc1) cung cấp điện áp cho IC hoạt động, chân này phải được kết nối với + 5V. Chân còn lại là Vs (Vcc2) cung cấp điện áp cho động cơ chạy, dựa trên thông số kỹ thuật của động cơ bạn có thể kết nối chân này với điện áp trong khoảng từ 4,5V đến 36V, ở đây chúng ta kết nối với + 12V.
- Các chân Enable (Enable 1,2) được sử dụng để cho phép (Enable) các chân ngõ vào cho động cơ 1 và động cơ 2 tương ứng. Vì ta chỉ sử dụng 1 động cơ, nên chỉ cần giữ chân EN1 ở mức cao theo mặc định bằng cách kết nối với nguồn + 5V. Các chân ngõ vào Input 1,2 được sử dụng để điều khiển động cơ và được nối đến các chân GPIO6 và GPIO12 của Raspberry Pi3. Chân Out3 và Out6 dùng kết nối trực tiếp với động cơ.



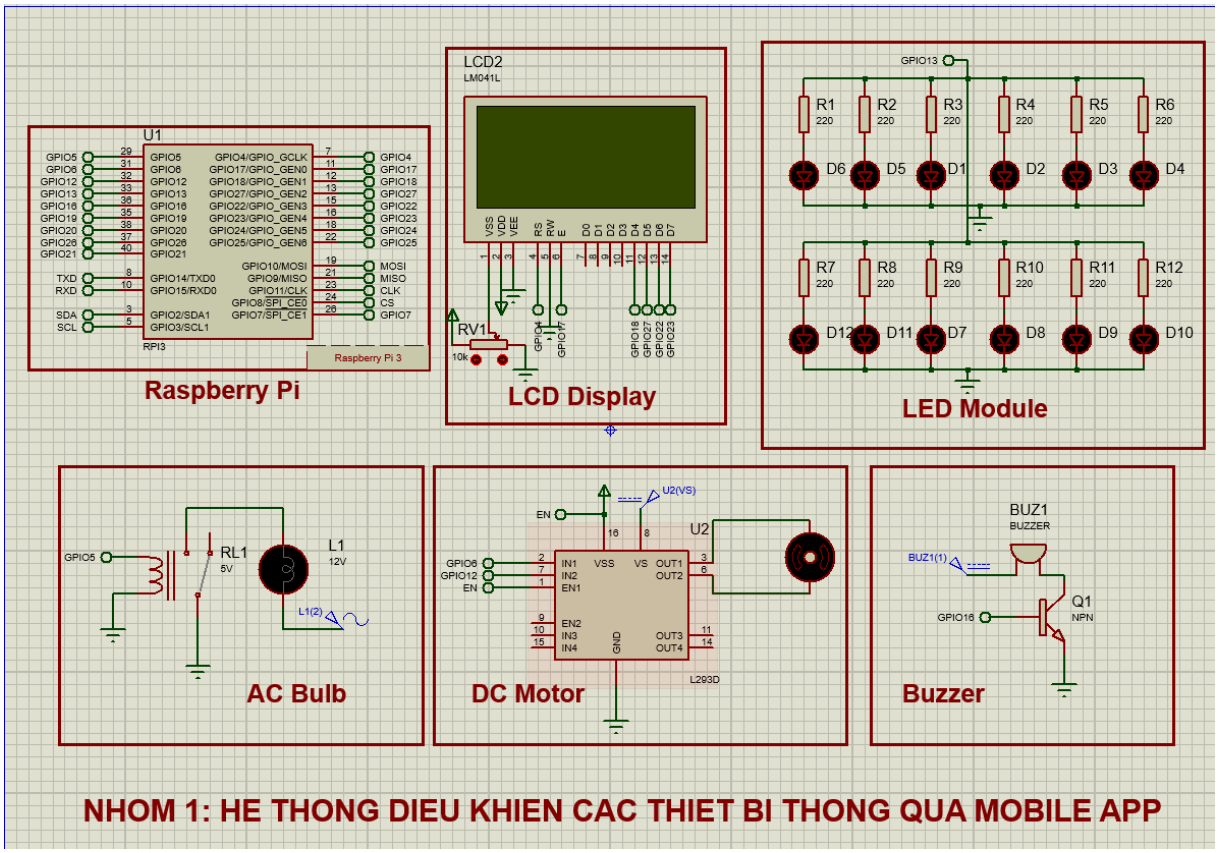
-

❋ *Module bóng đèn AC:*

- Ở đây chúng ta sử dụng Relay 5VDC, 1 chân của Relay nối đến GPIO5, 1 chân nối mass. Khi chân GPIO5 tích cực mức 1, tức là có dữ liệu điều khiển xuất ra thì dòng điện sẽ kích hoạt nam châm điện, tạo ra từ trường, tín hiệu. Từ trường này sẽ thu hút 1 tiếp điểm và đóng khóa K, khi đó 1 cực bóng đèn nối với điện áp 12VAC, 1 cực nối với mass=> đèn sáng
- Khi chân GPIO5 tích cực mức 0, dòng điện bị ngắt, nam châm ngừng hoạt động, không tạo ra thị trường. Lúc này, tiếp điểm sẽ bị lực kéo của lò xo ban đầu kéo về vị trí cũ, tương ứng mạch bóng đèn AC bị hở=> bóng đèn tắt.



❋ *Sơ đồ nguyên lý toàn mạch:*



-

CHƯƠNG 4: LƯU ĐỒ VÀ CHƯƠNG TRÌNH THỰC THI HỆ THỐNG

4.1 Phần mềm mô phỏng

Trong phần này để mô phỏng thì chúng ta dẽ sử dụng phầm mềm Proteus 8.13.

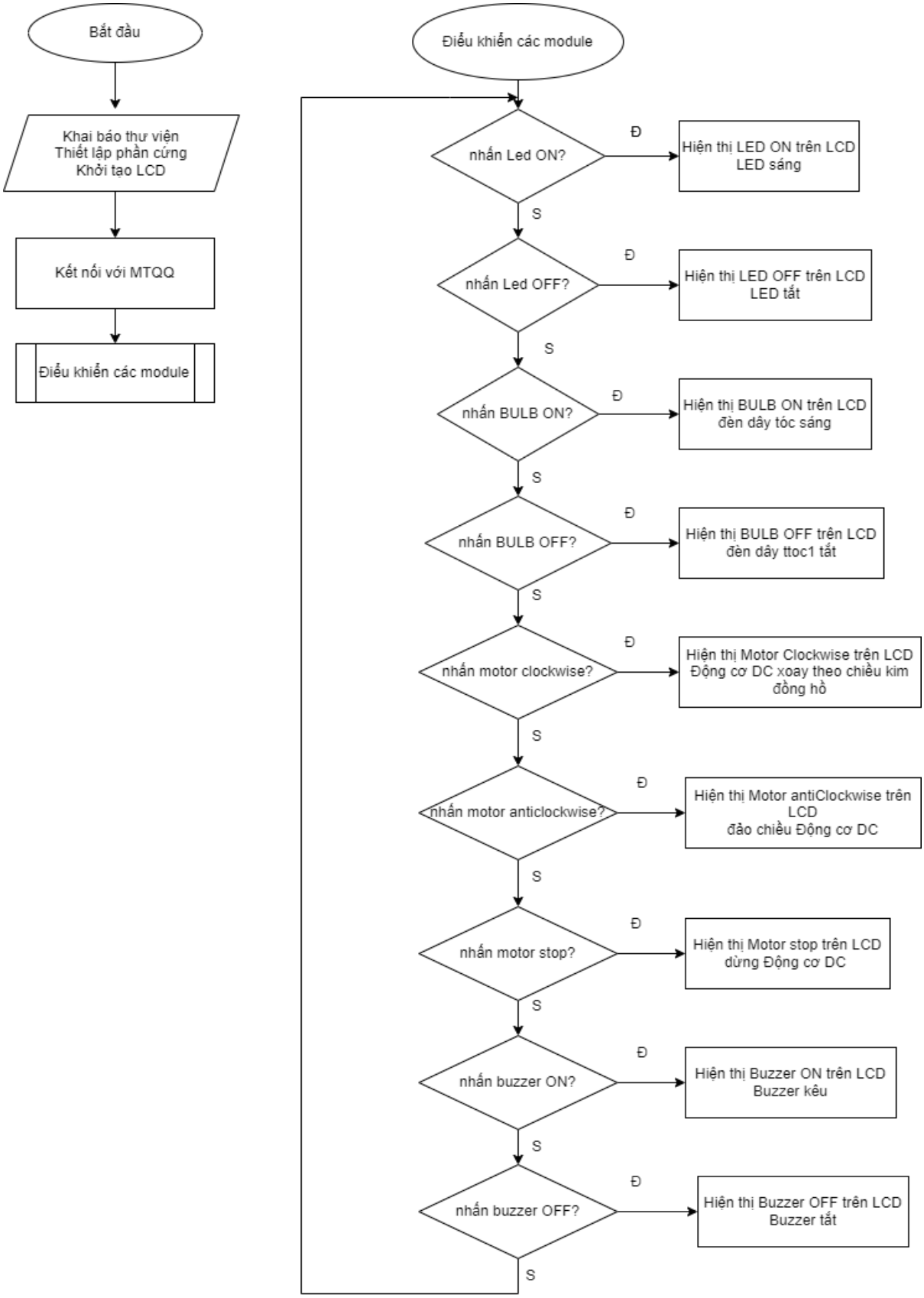
Phần mềm Proteus cho phép mô phỏng hoạt động của mạch điện tử bao gồm phần thiết kế mạch và viết chương trình điều khiển cho các họ vi điều khiển như MCS-51, PIC, AVR, … Proteus là phần mềm mô phỏng mạch điện tử của Labcenter Electronics, mô phỏng cho hầu hết các linh kiện điện tử thông dụng, đặc biệt hỗ trợ cho cả các MCU như PIC, 8051, AVR, Motorola.

Phần mềm bao gồm 2 chương trình: ISIS (Intelligent Schematic Input System) cho phép mô phỏng mạch và ARES (Advanced Routing and Editing Software) dùng để vẽ mạch in.

Khả năng ứng dụng chính của Proteus là mô phỏng, phân tích các kết quả từ các mạch nguyên lý. Proteus giúp người sử dụng có thể thấy trước được mạch thiết kế chạy đúng hay sai trước khi thi công mạch. Nó còn có các công cụ phục vụ cho việc phân tích mạch có độ chính xác khá cao như vôn kế đo điện áp, ampe kế đo dòng điện, máy đao động ký.

4.2 Lập trình hệ thống

4.2.1 Lưu đồ giải thuật



4.2.2 Lập trình cho Raspberry Pi3

```
#!/usr/bin/python
import time
import RPi.GPIO as GPIO
import time
import os,sys
from urllib.parse import urlparse
import paho.mqtt.client as paho
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

'''
define pin for lcd
'''
# Timing constants
E_PULSE = 0.0005
E_DELAY = 0.0005
delay = 1

# Define GPIO to LCD mapping
LCD_RS = 7
LCD_E  = 11
LCD_D4 = 12
LCD_D5 = 13
LCD_D6 = 15
LCD_D7 = 16
bulb_pin = 29
dc_motor1 = 31
dc_motor2 = 32
led_pin = 33
buzzer_pin = 36
GPIO.setup(LCD_E, GPIO.OUT)  # E
GPIO.setup(LCD_RS, GPIO.OUT) # RS
GPIO.setup(LCD_D4, GPIO.OUT) # DB4
GPIO.setup(LCD_D5, GPIO.OUT) # DB5
GPIO.setup(LCD_D6, GPIO.OUT) # DB6
GPIO.setup(LCD_D7, GPIO.OUT) # DB7
GPIO.setup(bulb_pin, GPIO.OUT)
GPIO.setup(dc_motor1, GPIO.OUT)
GPIO.setup(dc_motor2, GPIO.OUT)
GPIO.setup(led_pin, GPIO.OUT)
GPIO.setup(buzzer_pin, GPIO.OUT)
GPIO.setup(dc_motor2, GPIO.OUT)
GPIO.setup(led_pin, GPIO.OUT)
GPIO.setup(buzzer_pin, GPIO.OUT)
```

```
# Define some device constants
LCD_WIDTH = 16    # Maximum characters per line
LCD_CHR = True
LCD_CMD = False
LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line
LCD_LINE_3 = 0x90# LCD RAM address for the 3nd line
LCD_LINE_4 = 0xD0# LCD RAM address for the 3nd line

def on_connect(self, mosq, obj, rc):
        self.subscribe("led", 0)

def on_message(mosq, obj, msg):
    #print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload))
    if(msg.payload == b"led_ON"):
        lcd_string("LED ON ",LCD_LINE_1)
        GPIO.output(led_pin,GPIO.HIGH)  #LED ON
    elif(msg.payload== b"led_OFF"):
        lcd_string("LED OFF ",LCD_LINE_1)
        GPIO.output(led_pin,GPIO.LOW)   # LED OFF
    elif(msg.payload== b"bulb_ON"):
        lcd_string("Bulb ON ",LCD_LINE_2)
        GPIO.output(bulb_pin,GPIO.HIGH)   # LED OFF
    elif(msg.payload== b"bulb_OFF"):
        lcd_string("Bulb OFF ",LCD_LINE_2)
        GPIO.output(bulb_pin,GPIO.LOW)   # LED OFF
    elif(msg.payload== b"motor_clock"):
        lcd_string("Motor Clockwise ",LCD_LINE_3)
        GPIO.output(dc_motor1,GPIO.HIGH)   # LED OFF
        GPIO.output(dc_motor2,GPIO.LOW)   # LED OFF
    elif(msg.payload== b"motor_anti"):
        lcd_string("Motor anticlockwise ",LCD_LINE_3)
        GPIO.output(dc_motor1,GPIO.LOW)   # LED OFF
        GPIO.output(dc_motor2,GPIO.HIGH)   # LED OFF
    elif(msg.payload== b"motor_stop"):
        lcd_string("Motor stop ",LCD_LINE_3)
        GPIO.output(dc_motor1,GPIO.LOW)   # LED OFF
        GPIO.output(dc_motor2,GPIO.LOW)   # LED OFF
    elif(msg.payload== b"buzzer_ON"):
        lcd_string("Buzzer ON ",LCD_LINE_4)
        GPIO.output(buzzer_pin,GPIO.HIGH)   # LED OFF
    elif(msg.payload== b"buzzer_OFF"):
        lcd_string("Buzzer OFF ",LCD_LINE_4)
        GPIO.output(buzzer_pin,GPIO.LOW)   # LED OFF
```

```
def on_publish(mosq, obj, mid):
    print("mid: " + str(mid))


def on_subscribe(mosq, obj, mid, granted_qos):
    print("Subscribed:   "   +   str(mid)   +   "   "   +
str(granted_qos))

mqttc = paho.Client()                                  # object
declaration
# Assign event callbacks
mqttc.on_message                =                on_message
# called as callback
mqttc.on_connect = on_connect
mqttc.on_publish = on_publish
mqttc.on_subscribe = on_subscribe

url_str         =         os.environ.get('CLOUDMQTT_URL',
'tcp://broker.emqx.io:1883')
url = urlparse(url_str)
mqttc.connect(url.hostname, url.port)

'''
Function Name :lcd_init()
Function  Description  :  this  function  is  used  to
initialized lcd by sending the different commands
'''
def lcd_init():
  # Initialise display
  lcd_byte(0x33,LCD_CMD) # 110011 Initialise
  lcd_byte(0x32,LCD_CMD) # 110010 Initialise
  lcd_byte(0x06,LCD_CMD)    #    000110    Cursor    move
direction
  lcd_byte(0x0C,LCD_CMD)   #   001100   Display   On,Cursor
Off, Blink Off
  lcd_byte(0x28,LCD_CMD) # 101000 Data length, number
of lines, font size
  lcd_byte(0x01,LCD_CMD) # 000001 Clear display
  time.sleep(E_DELAY)
'''
Function Name :lcd_byte(bits ,mode)
Fuction  Name  :the  main  purpose  of  this  function  to
convert the byte data into bit and send to lcd port
'''
def lcd_byte(bits, mode):
  # Send byte to data pins
  # bits = data
  # mode = True  for character
  #        False for command
```

```
        GPIO.output(LCD_RS, mode) # RS

        # High bits
        GPIO.output(LCD_D4, False)
        GPIO.output(LCD_D5, False)
        GPIO.output(LCD_D6, False)
        GPIO.output(LCD_D7, False)
        if bits&0x10==0x10:
          GPIO.output(LCD_D4, True)
        if bits&0x20==0x20:
          GPIO.output(LCD_D5, True)
        if bits&0x40==0x40:
          GPIO.output(LCD_D6, True)
        if bits&0x80==0x80:
          GPIO.output(LCD_D7, True)

        # Toggle 'Enable' pin
        lcd_toggle_enable()

        # Low bits
        GPIO.output(LCD_D4, False)
        GPIO.output(LCD_D5, False)
        GPIO.output(LCD_D6, False)
        GPIO.output(LCD_D7, False)
        if bits&0x01==0x01:
          GPIO.output(LCD_D4, True)
        if bits&0x02==0x02:
          GPIO.output(LCD_D5, True)
        if bits&0x04==0x04:
          GPIO.output(LCD_D6, True)
        if bits&0x08==0x08:
          GPIO.output(LCD_D7, True)
# Toggle 'Enable' pin
  lcd_toggle_enable()
'''
Function Name : lcd_toggle_enable()
Function Description:basically this is used to toggle
Enable pin
'''
def lcd_toggle_enable():
  # Toggle enable
  time.sleep(E_DELAY)
  GPIO.output(LCD_E, True)
  time.sleep(E_PULSE)
  GPIO.output(LCD_E, False)
  time.sleep(E_DELAY)
```

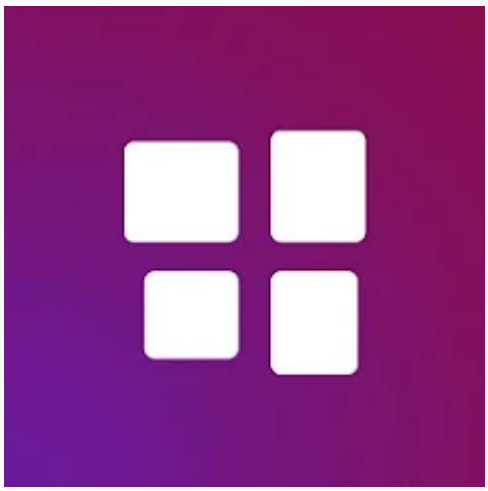
```
'''
Function Name :lcd_string(message,line)
Function  Description :print the data on lcd
'''
def lcd_string(message,line):
  # Send string to display

  message = message.ljust(LCD_WIDTH," ")

  lcd_byte(line, LCD_CMD)

  for i in range(LCD_WIDTH):
    lcd_byte(ord(message[i]),LCD_CHR)

lcd_init()
lcd_string("welcome ",LCD_LINE_1)
time.sleep(0.5)
lcd_byte(0x01,LCD_CMD) # 000001 Clear display
lcd_string("LED OFF ",LCD_LINE_1)
lcd_string("Bulb OFF ",LCD_LINE_2)
lcd_string("Motor stop ",LCD_LINE_3)
lcd_string("Buzzer OFF ",LCD_LINE_4)
# Define delay between readings
delay = 5

while 1:
  # Print out results
  rc = mqttc.loop()
  time.sleep(0.5)
```

4.2.3 Phần mềm MQTT DÁHBOARD

Hiện nay, có nhiều app hỗ trợ các hệ thống có giao thức MTQQ thì trong phần này chúng ta sẽ sử dụng app MTQQ DASHBOARD. MTQQ DASHBOARD là một công cụ đơn giản và đẹp mắt để điều khiển các thiết bị có giao thức MQTT và quản lý hệ thống tự động hóa. Nó tương thích với Node-RED, Tasmota Sonoff, tất cả bo mạch Arduino hỗ trợ internet và hơn thế nữa…

4.3 Quy trình thao tác điều khiển

Đầu tiền chúng ta cần tải phần mềm MTQQ DASHBOARD về điện thoại.



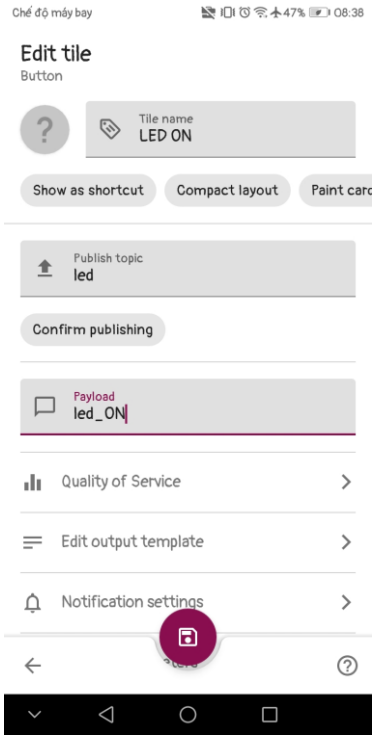
Hình 4.2 Phần mềm MTQQ DASHBOARD

Sau khi tải phần mềm về điện thoại, chúng ta thêm tên và địa chỉ của broker để kết nối với hệ thống, hoàn thành xong chúng ta nhấn vào biểu tượng Save để lưu lại.

Sau khi đã thêm địa chỉ Broker, chúng ta bắt đầu tạo các nút nhấn để diều khiển các module trong hệ thống. Chúng ta sẽ đặt tên và chọn giao diện cho nút nhấn.

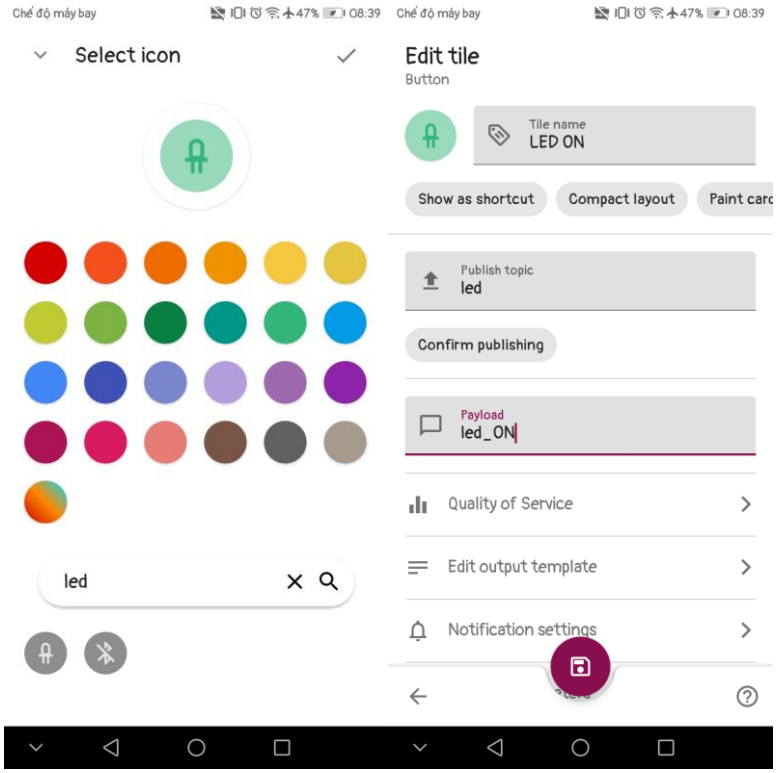
Lưu ý: Chúng ta phải đặt payload giống với chuỗi ký tự payload mà chúng ta đã lập trình

Ví dụ: Tạo nút nhấn LED ON



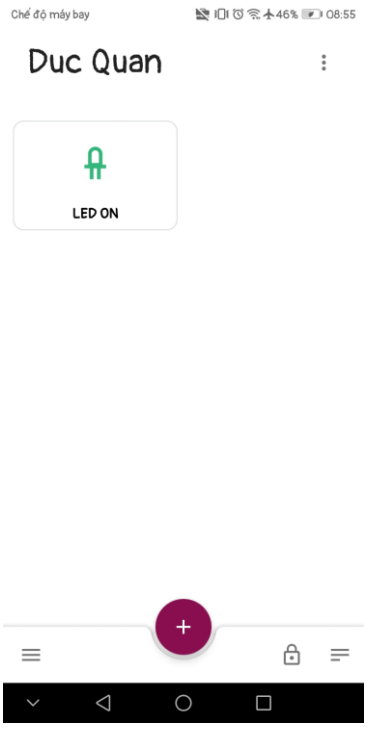
Hình 4.3 Hiệu chỉnh Broker

Hình 4.4 cho thấy chúng ta đã đã đạt tên và thiết lập payload cho nhấn, hoàn thành xong thì chúng ta nhấn vào biểu tượng hình tròn có dấu chấm hỏi để bắt đầu tạo giao diện cho nút nhấn.



Hình 4.5 Tạo giao diện cho nút LED ON

Ở Hình 4.5 khi chúng vào ô tìm kiếm và gõ led phầm mềm sẽ hiện thị những biểu tượng và màu của led để chúng ta chọn, sau khi chọn xong chúng ta sẽ có được giao diện của nút nhấn LED ON



Hình 4.6 Giao diện của nút nhấn LED ON

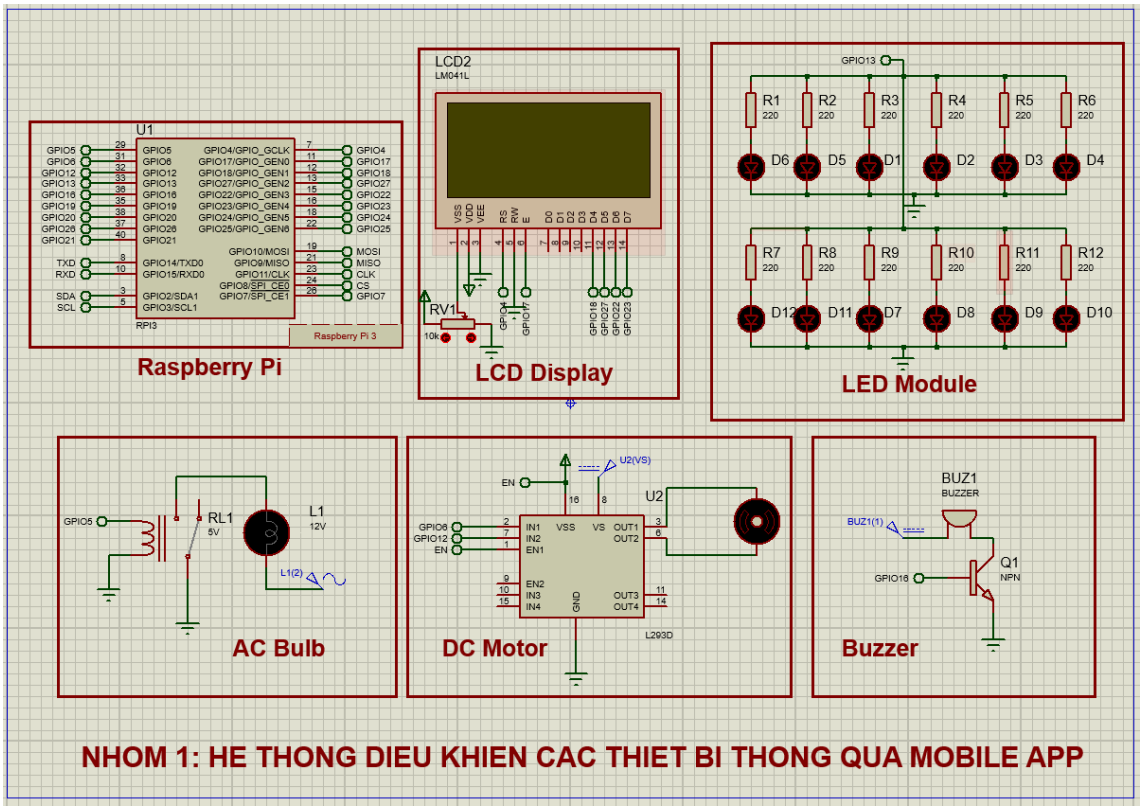
Lặp lại tương tự các bước trên với các nút còn lại chúng ta sẽ có được bảng điều khiển:

Sau khi đã thiết lập xong bảng điều khiển chúng ta kiểm tra kết nối giữa bảng điều khiển với hệ thống và lần lượt nhấn các nút để diều khiển module mà chúng ta mong muốn.

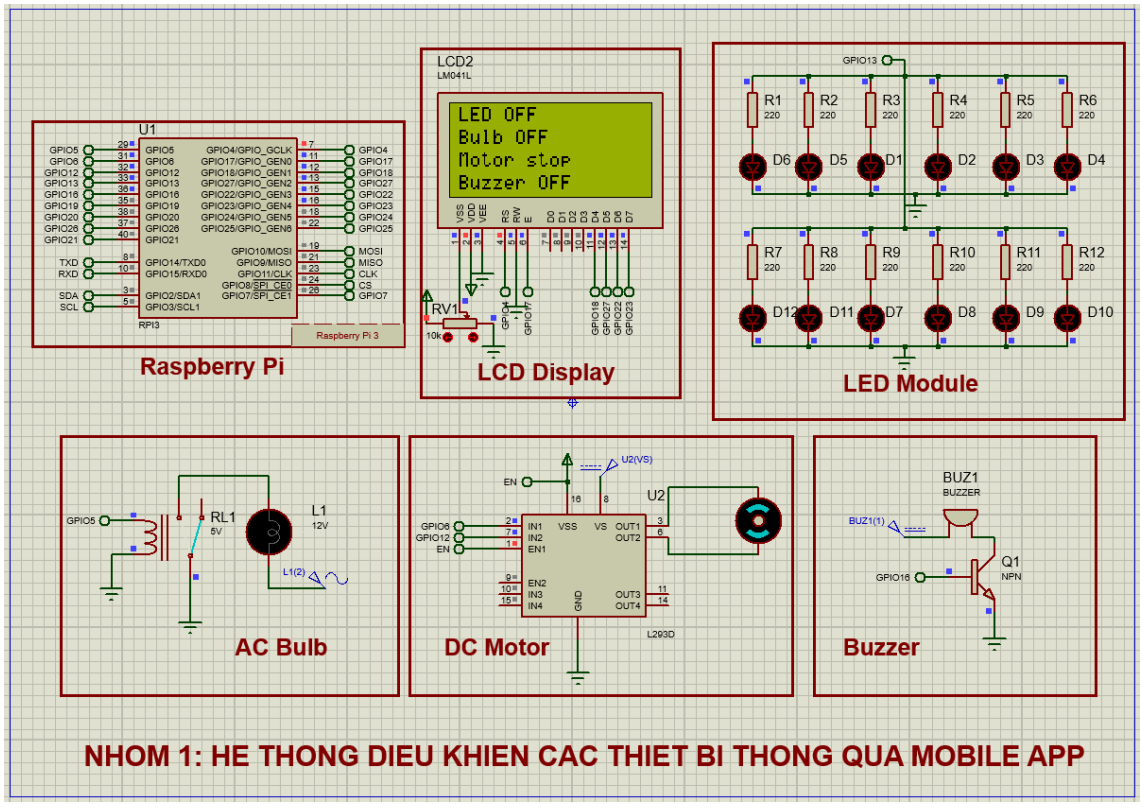
Ví dụ: Nếu chúng ta muốn bật đèn led thì chúng ta nhấn nút LED ON trẹn bảng diều khiển và tín hiệu led_ON (payload) sẽ dược gửi đến hệ thống và so sánh với chuỗi payload mà chúng ta đã lập trình, nếu tín hiệu đúng thì đèn led sẽ sáng. Chúng ta thực hiện tương tự với các nút còn lại.

CHƯƠNG 5: KẾT QUẢ, NHẬN XÉT, ĐÁNH GIÁ

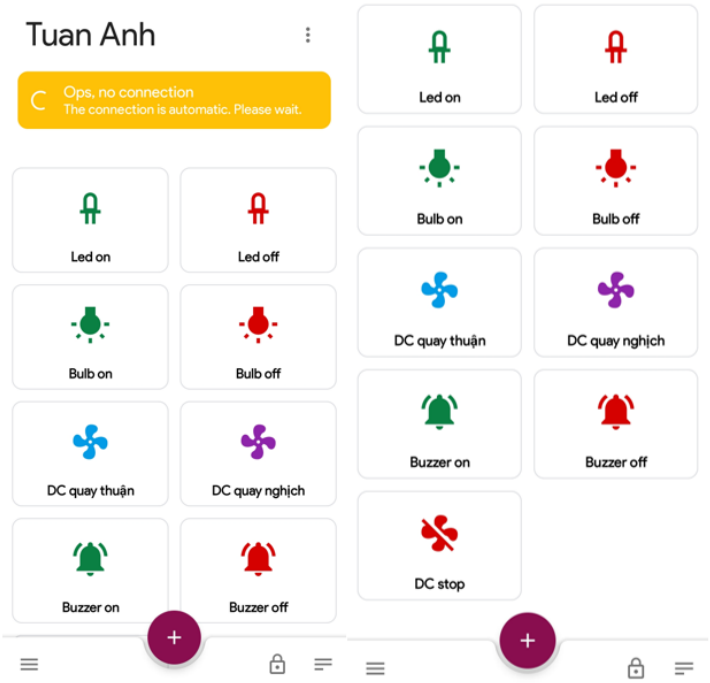
5.1 Kết quả mô phỏng



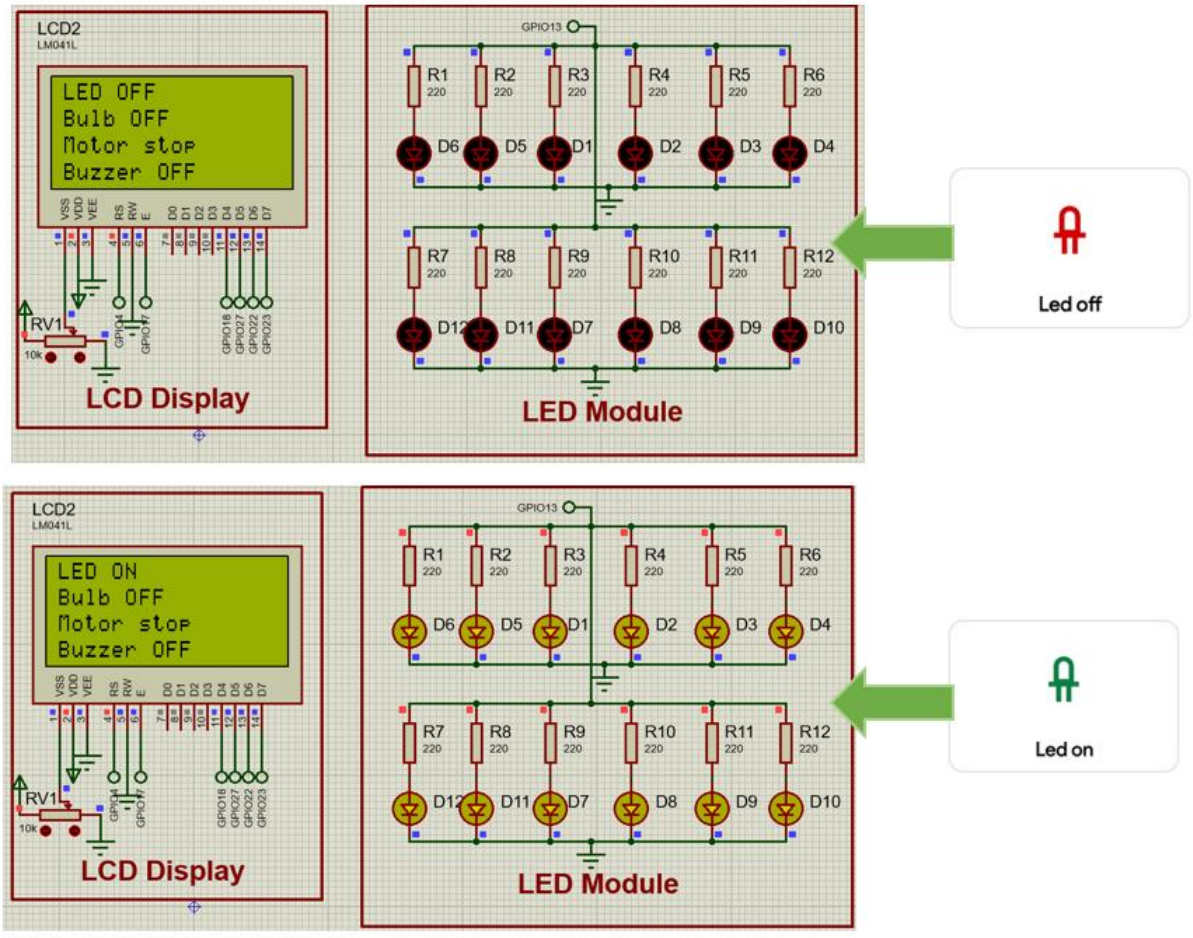
Hình 5.1a: *Giao diện hệ thống khi chưa chạy mô phỏng*



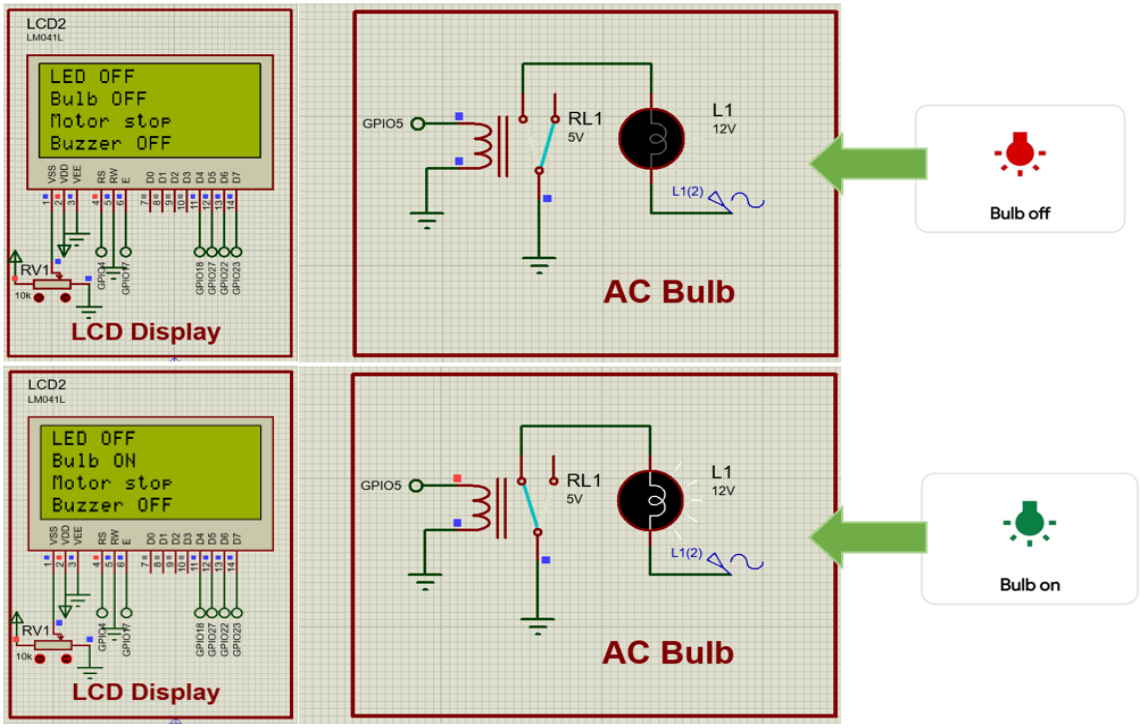
Hình 5.1b: *Giao diện hệ thống khi mới chạy mô phỏng*



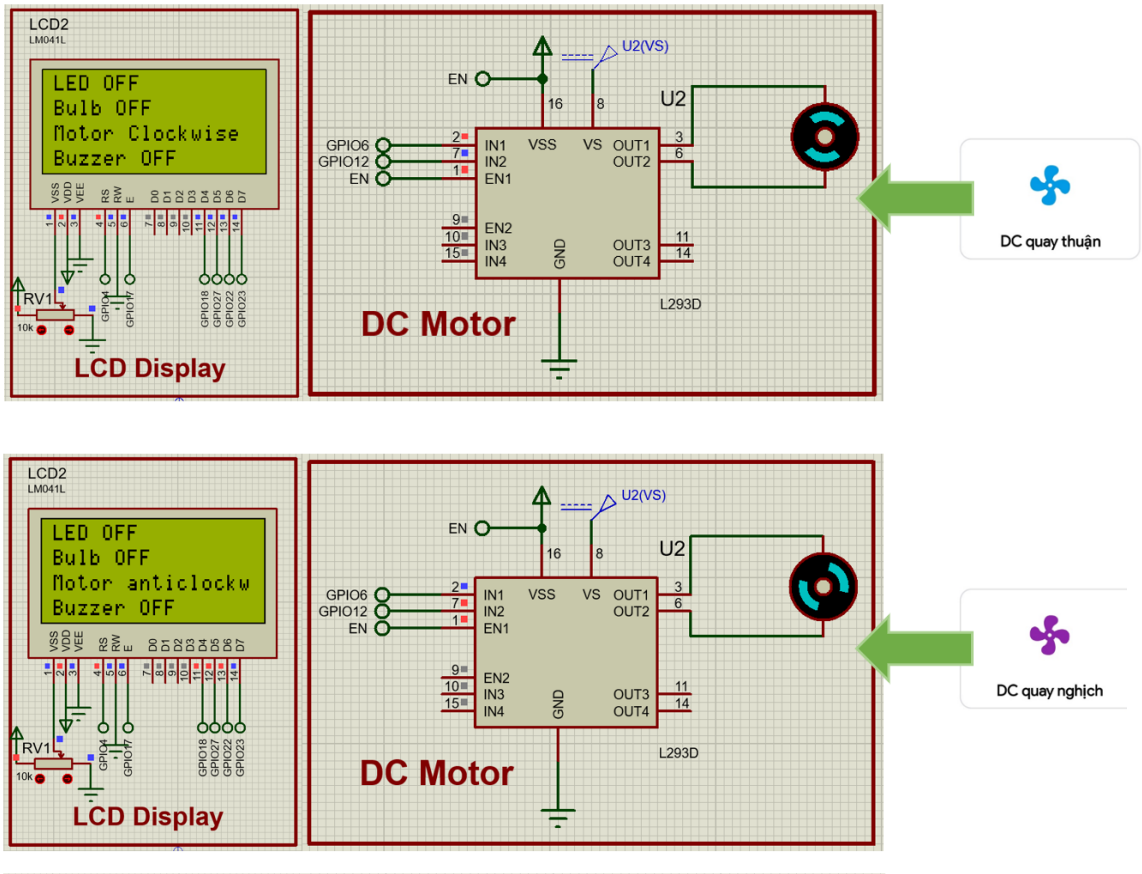
Hình 5.1c *Giao diện điều khiển trên app MQTT Dashboard khi chưa kết nối và khi đã kết nối với hệ thống thành công*



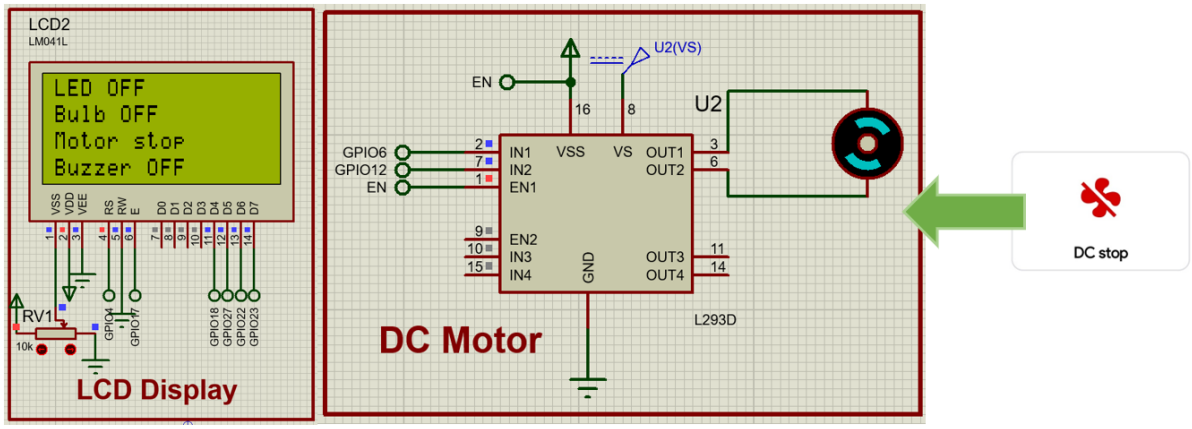
Hình 5.1d: *Điều khiển module led sáng tắt qua 2 nút "Led on" và "Led off" của App*



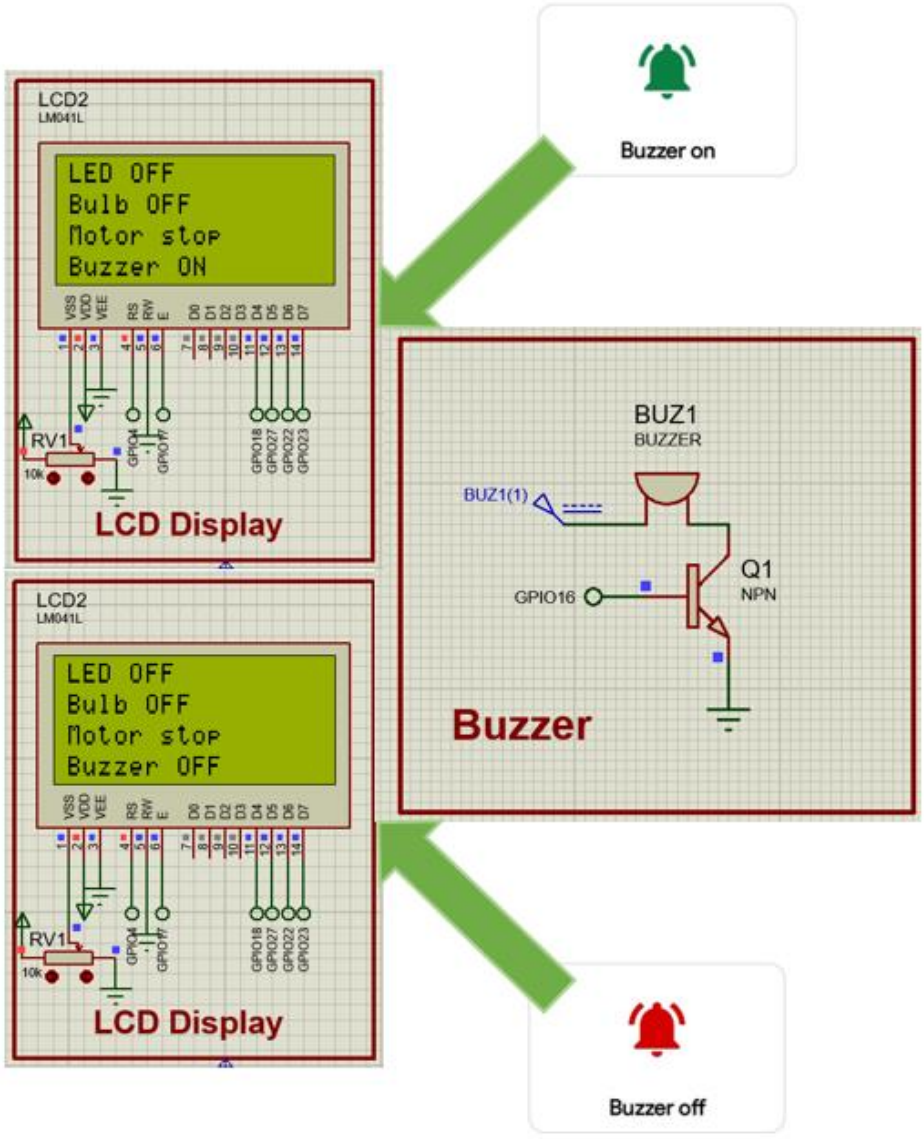
Hình 5.1e: *Điều khiển led bulb sáng tắt qua 2 nút "Bulb on" và "Bulb off"của App*



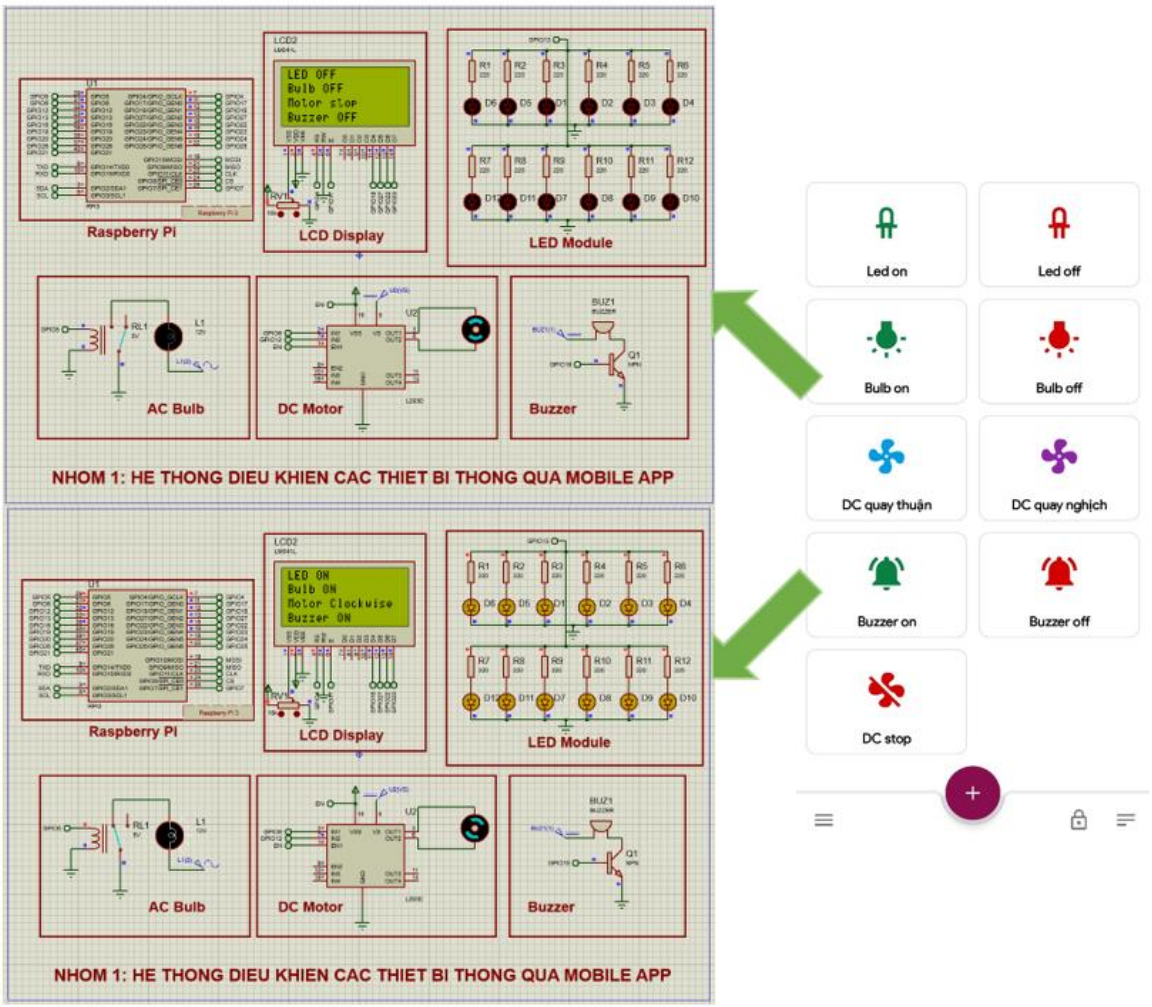
Hình 5.1f: *Điều khiển động cơ quay thuận và quay nghịch qua App*



Hình 5.1f: *Điều khiển động cơ quay thuận và quay nghịch qua App*



Hình 5.1g: *Điều khiển Buzzer "On", "Off" thông qua App*



Hình 5.1h: *Điều khiển cả hệ thống qua App MQTT Dashboard*

5.2 Nhận xét kết quả mô phỏng

- Giao diện mô phỏng hệ thống các thiết bị đều giống như lý thuyết đã trình bày, hệ thống điều khiển thuận tiện.
- Tốc độ mà raspberry Pi3 nhận tín hiệu từ app khi thao tác phím nhấn trên màn hình android nhanh và nhạy.
- Sau quá trình raspberry Pi3 truyền và nhận dữ liệu thông qua app MQTT dashboard (android), ứng dụng đã cho kết quả hoạt động và hiển thị các hệ thống thiết bị một cách chính xác và nhanh chóng.
- Tránh được các thao tác thủ công bật tắt, kiểm soát được các thiết bị một cách chủ động, biết được tình trạng hoạt động của các thiết bị khi ta lắp các mạch thực tế.
- Giao diện dễ sử dụng và bắt mắt, hiển thị đầy đủ thông tin cho người dùng.

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Sau khoản thời gian nhóm thực hiện nghiên cứu và xây dựng đề tài báo cáo, cùng với sự nỗ lực của các thành viên trong nhóm và sự giúp đỡ tận tình của Thầy Nguyễn Thanh Nghĩa, cùng với sự giúp đỡ của các bạn, đến nay nhóm chúng em đã hoàn thành đề tài báo cáo.

Trong đề tài của chúng em đã tìm hiểu và thực hiện được các yêu cầu sau:

- Điều khiển bật tắt các thiết bị thông qua app mobile (MQTT Dashboard,..)
- Tìm hiểu về các thiết bị hệ thống để thực thi mô phỏng (led đơn, led bulb, động cơ DC, buzzer,…)
- Hiểu hơn về phần mềm lập trình python.

Do hạn chế về mặt thời gian và khả năng lập trình còn hạn hẹp nên đề tài chưa thể có những tính năng cao hơn ví dụ: Tự lập trình App và thiết kế giao diện App riêng, vấn đề về bảo mật thông tin riêng tư cho App điều khiển, và hệ thống thiết kế chưa được xây dựng để đo bằng các cảm biến và hiển thị thông số trên App,…

❃ Ưu điểm:

- Giao diện android đẹp dễ dùng, dễ dàng cài đặt trên hầu hết các thiết bị điện thoại máy tính bảng chạy android.
- Tốc độ xử lý nhanh.
- Hệ thống đơn giải dễ thực hiện .

❃ Nhược điểm:

- Vấn đề về truyền dữ liệu còn trì hoãn khi ngắt một khoảng thời gian không điều khiển.
- Hệ thống chưa được thử nghiệm và thi công sản phẩm thực tế nên vẫn chưa đánh giá được độ ổn định và kết quả của hệ thống.

6.2 Hướng phát triển

- Phát triển App để kiểm soát số lần đóng mở thiết bị.
- Tích hợp điều khiển các thiết bị chống trộm với App.
- Điều khiển đóng mở các thiết bị bằng mật khẩu hoặc thẻ RFID.
- Kết hợp nhiều mô hình lại với nhau để mở rộng phạm vi và thiết bị điều khiển.
- Có thể xây dựng hệ thống mạng để quản lí thiết bị và hoạt động của hệ thống.

LIỆU THAM KHẢO

[1] Nguyễn Đình Phú, "Giáo trình vi xử lý II", NXB ĐH Quốc Gia Tp.HCM, 2007

[2] Nguyễn Văn An và Hồ Thanh Hùng, "Thiết Kế và Thi Công Nhà Thông Minh Dùng Vi Điều Khiển", Đồ Án Tốt Nghiệp ĐH, Trường ĐH Sư Phạm Kỹ Thuật Tp.HCM, 2015.

[3] Manuel Jiménez, Rogelio Palomera, Isidoro Couvertier "Introduction to Embedded Systems_ Using Microcontrollers and the MSP430-Springer-Verlag New York"

[4] Mark Lutz, "Learning Python, 5th Edition Fifth Edition"

[5] Vũ Cao Minh Đức, "Giới thiệu về giao thức truyền thông điệp/giao tiếp MQTT", Bộ TT&TT, Cục chuyển đổi số Quốc gia, 06/12/2021 https://aita.gov.vn/gioi-thieu-giao-thuc-truyen-thong-diepgiao-tiep-mqtt

[6] Bkaii, "Nguyên lí hoạt động và vai trò của Raspberry Pi", https://bkaii.com.vn/tin-tuc/401-nguyen-li-hoat-dong-va-vai-tro-cua-raspberry-pi