

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA TP.HỒ CHÍ MINH
KHOA ĐIỆN – ĐIỆN TỬ



TIỂU LUẬN MÔN HỌC
THIẾT KẾ HỆ THỐNG NHÚNG NÂNG CAO

Remote Control

GVHD: TS.Trương Quang Vinh

Nhóm 3

HVCH: Cao Xuân Thiện - 1870067

Nguyễn Thanh Tùng – 1870249

Nguyễn Tuấn Quang - 1670329

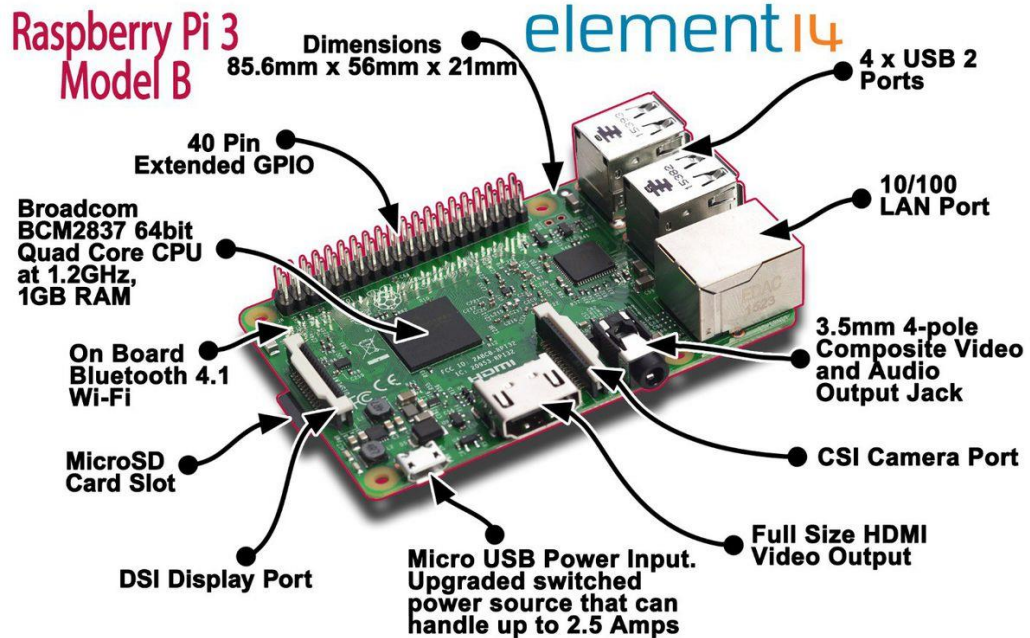
Tp.HCM , tháng 01 năm 2019

Phần 1 : Product requirement

Các bước trong việc viết spec.

1. **Name:**

Remote Control with Raspberry Pi 3.



Hình minh họa board mạch của Raspberry Pi 3

2. **Purpose:**

Điều khiển các Relay (bật tắt các thiết bị ngoại vi) với nhiều cách thức khác nhau thông qua giao diện mạng sử dụng Raspberry.

3. **Input:**

Các lệnh điều khiển của người dùng thông qua giao diện web hoặc lệnh qua giọng nói của người dùng thông qua micro điện thoại.

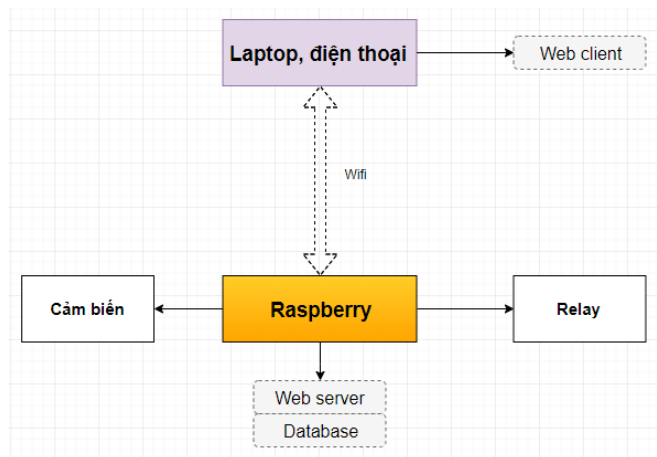
Output: Các lệnh điều khiển ngoại vi.

4. **User case:**

Người dùng theo dõi thông số cảm biến qua giao diện hiển thị trên web. Khi các thông số vượt quá ngưỡng cho trước sẽ bật/tắt relay và có cảnh báo cho người dùng.

Khi thời gian chạm đến ngưỡng hẹn giờ trước sẽ bật/tắt theo ý muốn.

5. **Function:**



Giản đồ phần cách thức hoạt động của hệ thống

- Theo dõi thông số: các thông số sẽ được hiển thị trên user interface.
- Điều khiển Relay:
Hẹn giờ: bật tắt theo lịch hẹn được người dùng đặt trước.
Trực tiếp: thông qua giao diện web, thông qua giọng nói của người dùng qua micro điện thoại.

6. Performance:

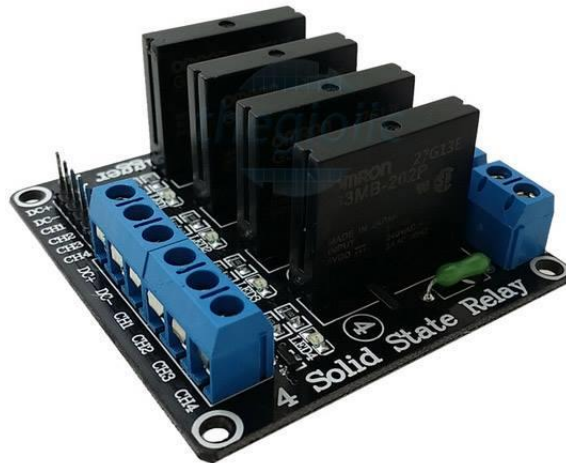
- Điều khiển tối đa 8 Relay.
- Delay lúc điều khiển không quá 2s.

7. Cost:

- Raspberry pi 3 + thẻ nhớ : 1.320.000 đ.



- Relay 4: 95.000 đ.



- Cảm biến nhiệt độ, độ ẩm: 38.000 đ.



Tổng: 1.517.000 đ.

8. Power:

Raspberry cấp nguồn: 5V, 2A.

Relay 4: điện áp kích 3.3V.

Cảm biến cấp nguồn: 3~5v.

9. Physical size/weight.

Specifications

Generation	Model A	Compute module*
	1	3
Power source	5 V via MicroUSB or GPIO header	
Size	85.60 mm × 56.5 mm (3.370 in × 2.224 in), excluding protruding connectors	67.6 mm × 31 mm (2.66 in × 1.22 in)
Weight	31 g (1.1 oz)	7 g (0.25 oz)

20 hàng khác

[Raspberry Pi - Wikipedia](https://en.wikipedia.org/wiki/Raspberry_Pi)

https://en.wikipedia.org/wiki/Raspberry_Pi

Kích thước của Raspberry Pi 3

10. Installation:

Người dùng cần có 1 trình duyệt web để load giao diện sử dụng, tool Tasker để thu thập giọng nói người dùng và gửi chuỗi kí tự lên web-server xử lí.

Phần 2 : Engineering specification

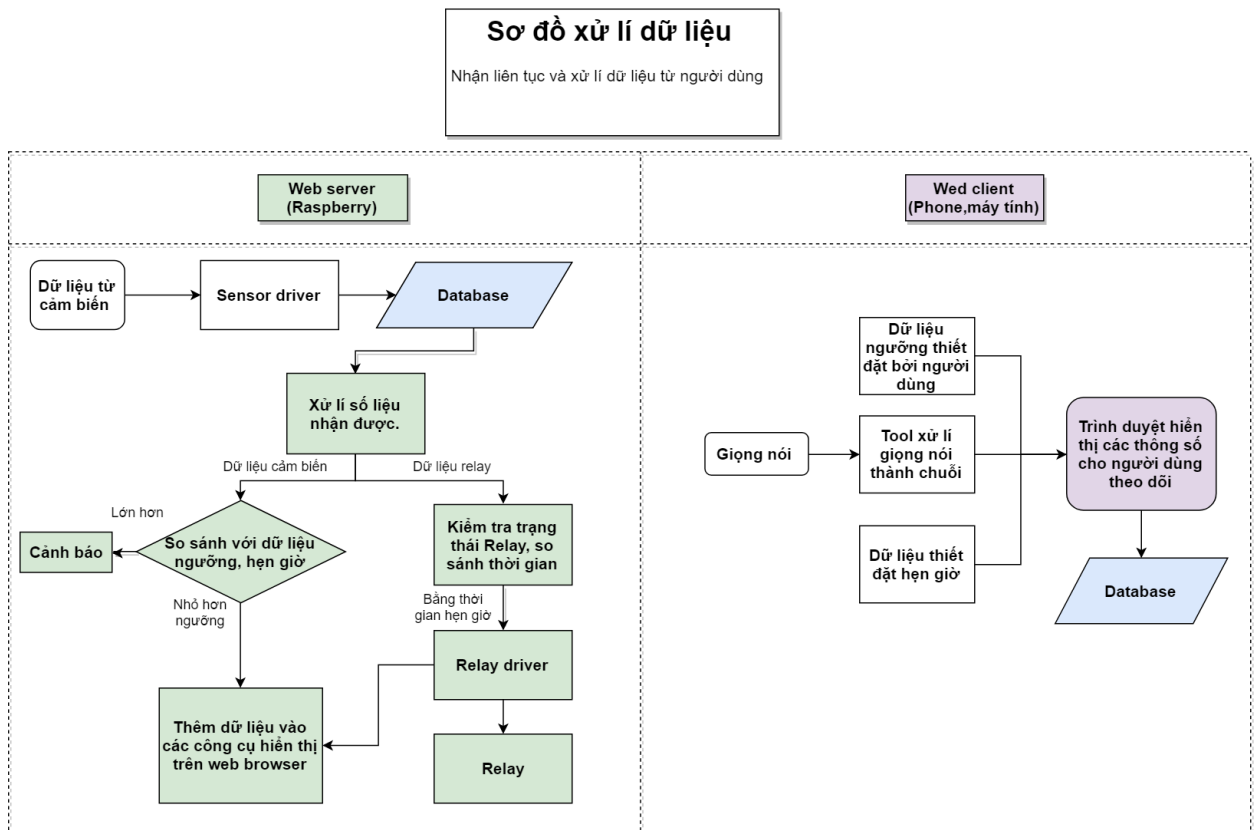
1. Nguyên lý hoạt động

Người dùng theo dõi thông số cảm biến qua giao diện hiển thị trên web. Khi các thông số vượt quá ngưỡng cho trước sẽ bật/tắt relay và có cảnh báo cho người dùng. Khi thời gian chạm đến ngưỡng hẹn giờ trước sẽ bật/tắt theo ý muốn

2. Môi trường hoạt động

- Có kết nối Wifi để nhận tín hiệu điều khiển từ xa
- Thiết bị được điều khiển nằm trong phạm vi cho phép

3. Sơ đồ khối hệ thống



4. Mô tả các khối chính

- Cảm biến : Có nhiệm vụ thu thập thông tin và gửi về server để phân tích và ra quyết định điều khiển .
- Database : Lưu giữ thông tin từ cảm biến .
- Xử lý số liệu : Phân tích số liệu từ cảm biến và tình trạng của relay để đưa ra quyết định điều khiển .
- Relay : Đóng và ngắt thiết bị điều khiển
- Điện thoại hoặc máy tính : Cung cấp giao diện hiển thị thông tin và nhận lệnh điều khiển để gửi đến thiết bị .

5. Phân chia phần cứng phần mềm

- Phần cứng :

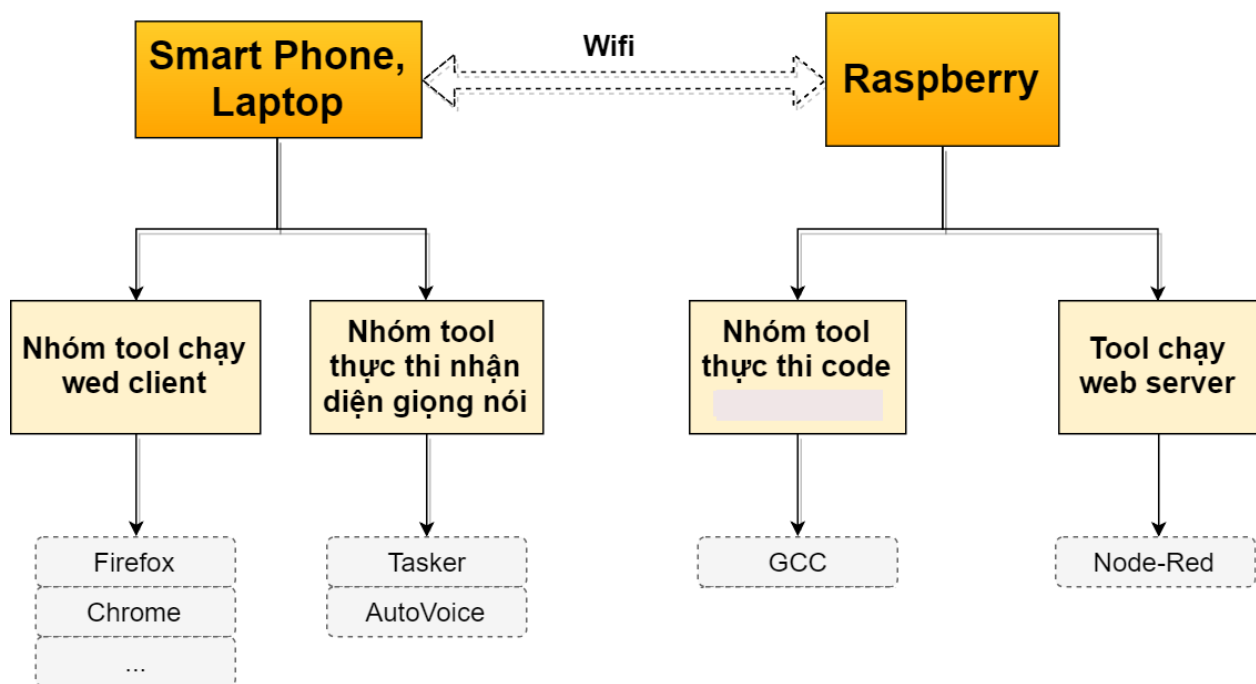
+ Cảm biến

+ Relay

+ Raspberry pi 3 + thẻ nhớ

- Phần mềm :

Sơ đồ tổng quát phần mềm



Phần 3 : Hardware design document.



1. Lựa chọn phần cứng.

a. Lựa chọn MPU giữa Beagle bone black và Raspberry.

Bảng so sánh chi tiết giữa Raspberry pi 3 và Black Beagle bone:

Raspberry Pi 3 Model B vs. BeagleBone Black Rev. C

[Add another board](#)

	Raspberry Pi 3 Model B ✖ <i>Raspberry Pi Foundation</i>	BeagleBone Black Rev. C ✖ △ <i>BeagleBoard</i>
		
SoC	Broadcom BCM2837	Texas Instruments AM3358/9
CPU	ARM Cortex-A53 (64-bit) 1.2GHz quad core	ARM Cortex-A8 (32-bit) 1000MHz single core
GPU	VideoCore IV	PowerVR SGX530
RAM size	1GB	512MB
Built-in RAM	✓	✓
Internal storage	✖	4GB
SD card	MicroSD card slot Max. size: 32GB	MicroSD card slot Max. size: 32GB

USB host	4	1
USB OTG	✓	✓
USB version	2.0	2.0
Ethernet	1 port Type: 10/100	1 port Type: 10/100
Wake-on-Lan	✖	✖
HDMI	✓	✓
VGA	✖	✖
Composite video (CVBS)	✓	✖
Display interfaces	MIPI DSI	✖
Camera interfaces	1	✖
Audio output	✓	✖
Audio connector	3.5mm jack	✖
HDMI audio	✓	✓
SPDIF	✖	✖
I2S	✓	✖

I2S	✓	✗
Line input	✗	✗
Mic input	✗	✗
Onboard mic	✗	✗
SATA	✗	✗
IR sensor	✗	✗
WiFi	802.11b/g/n	✗
Bluetooth	Bluetooth 4.1	✗
RTC	✗	✗
GPIO pins	40	92
PWM pins	1	✗
ADC pins	✗	✗
I2C	1	✗
SPI	1	✗
UART	1	✗
RS232	✗	✗
Arduino pinout	✗	✗
Voltage	4.8V - 5.2V	5V
Power consumption	600mA - 1.6A	600mA - 1.4A
Windows support	Windows 10	✗
Android support	✗	Android 4.4
Size (mm)	85 x 56 x 20	?
Weight (g)	45	?
Operating temperature	0 °C – 45 °C	?
Price	\$35	\$45

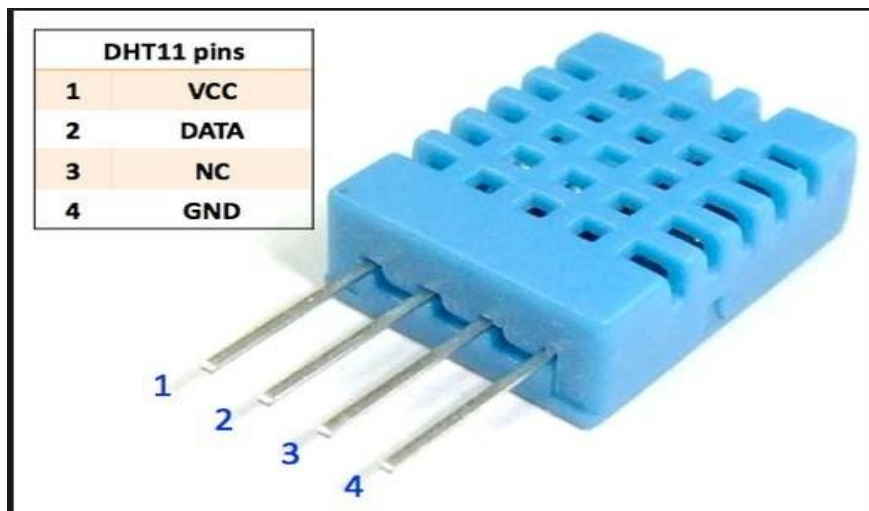
- **Nhận xét:**

- Với việc hỗ trợ 92 Pin GPIO và hỗ trợ Android 4.4, Beagle bone rất phù hợp để chọn trong các project cần xử lý nhiều tác vụ thông qua GPIO hoặc android. Tuy nhiên, điểm quan trọng nhất của bài tập lớn là remote các thiết bị trong cùng lớp mạng IP nên với việc không hỗ trợ module wifi là một điểm trừ lớn.
- Hơn nữa, với giá thành mắc hơn 10\$ nhưng beagle bone lại cho hiệu năng thấp hơn (1GHz so với 1.2 GHz) raspberry.

⇒ Nhóm quyết định chọn Raspberry pi 3 cho project “Remote controller”.

b. Lựa chọn giữa 2 sensor nhiệt độ, độ ẩm DHT11, DHT22.

- Cảm biến nhiệt độ và độ ẩm DHT11:



- Nguồn: 3->5 VDC.
- Dòng sử dụng: 2.5mA max.
- Độ tốt ở độ ẩm 20-80%RH với sai số 5%.
- Độ tốt ở nhiệt độ 0-50°C sai số 2°C.
- Tần số lấy mẫu tối đa 1Hz.
- 4 chân, khoảng cách chân 0.1".
- Giá thành: 38.000 vnd.

- Cảm biến nhiệt độ và độ ẩm DHT22.



DHT22 Pinout
Pin 1: VCC (3V to 5.5V)
Pin 2: Data
Pin 3: Not Connected
Pin 4: Ground

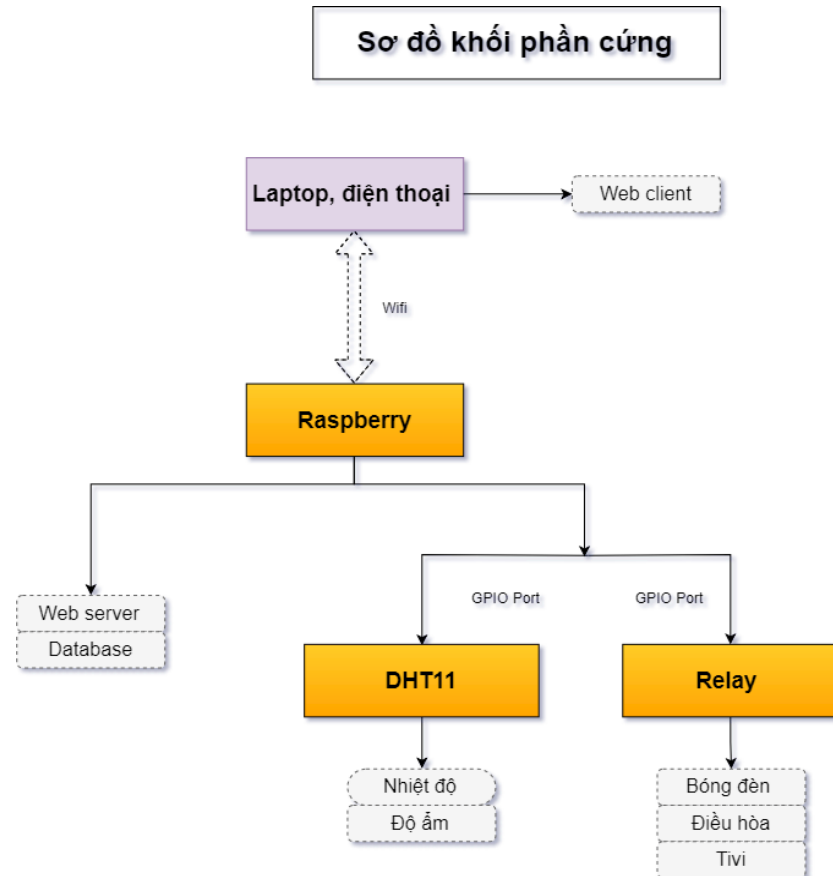
- Nguồn: 3-5 VDC.
- Dòng sử dụng: 2.5mA max.
- Đo tốt ở độ ẩm 100%RH với sai số 2-5%.
- Đo tốt ở nhiệt độ -40 -> 80°C sai số 0.5°C.
- Tần số lấy mẫu tối đa 0.5Hz.
- 4 chân, khoảng cách chân 0.1".
- Giá thành: 102.000 vnd.

- **Nhận xét:**

- Nhìn chung DHT22 có độ chính xác và thân đo rộng hơn DHT11. Nhưng cả hai đều dùng một chân tín hiệu kỹ thuật số duy nhất.
- Với mô hình của bài project không mang tính thương mại thì DHT11 vẫn đáp ứng đủ yêu cầu của project với giá thành thấp hơn nhiều so với việc sử dụng DHT22.

⇒ Nhóm quyết định lựa chọn DHT11 làm sensor đo nhiệt độ và độ ẩm.

2. Sơ đồ khối thiết kế.

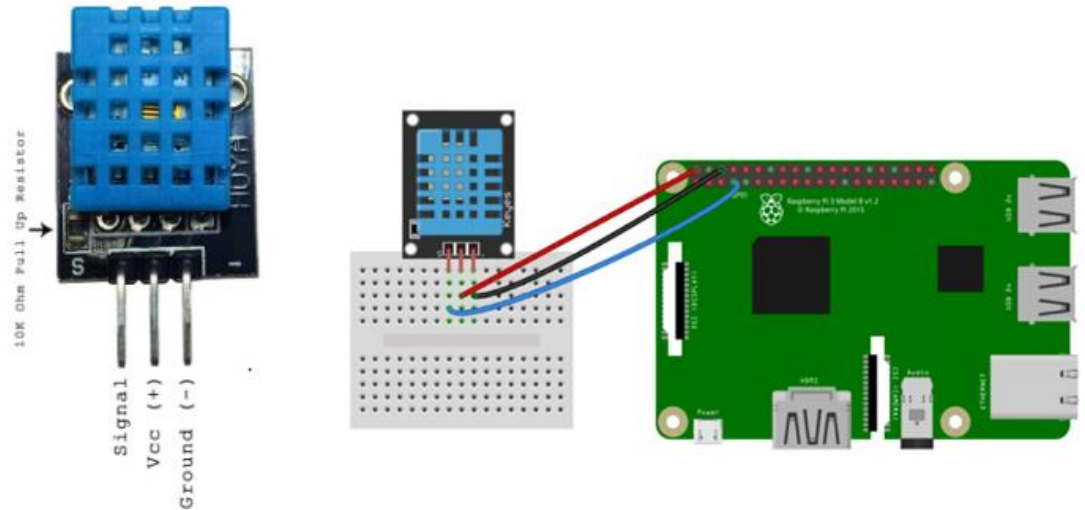


3. Sơ đồ mạch chi tiết cho từng khối.

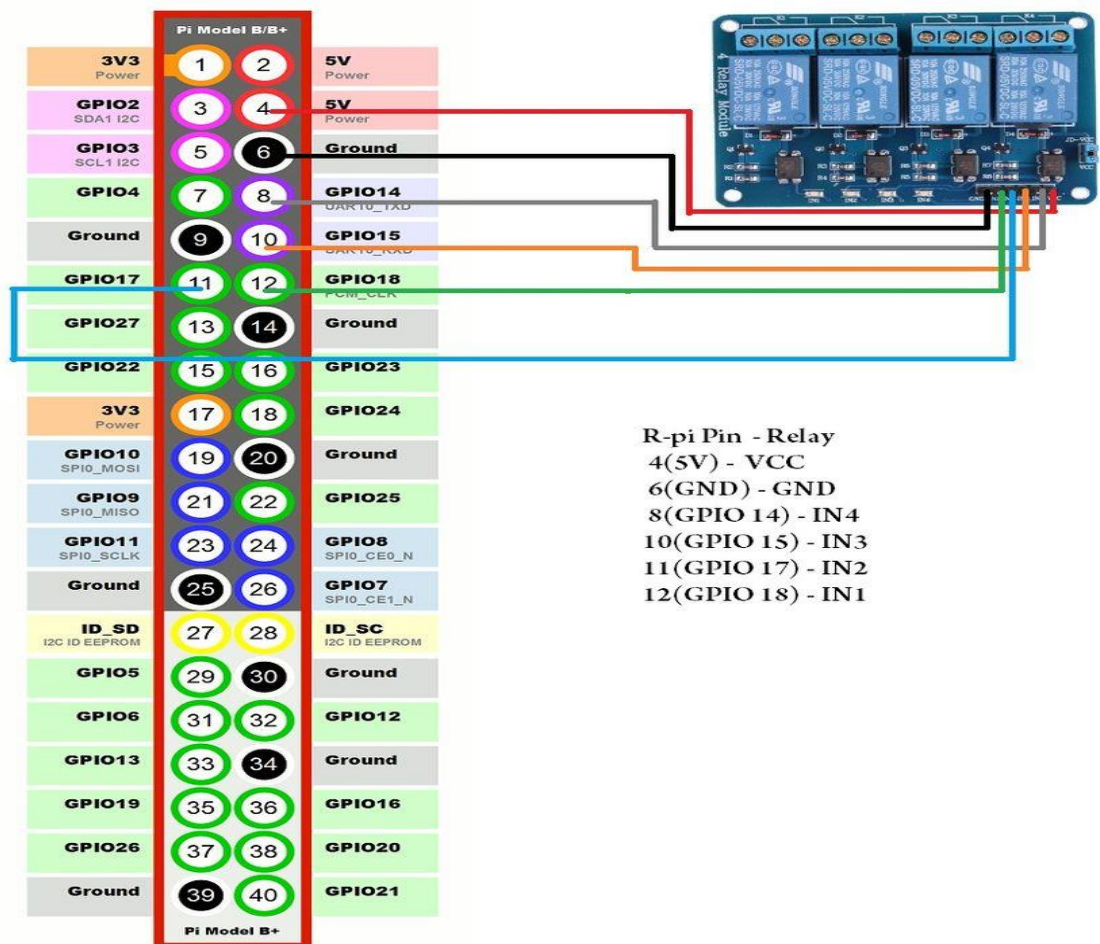
a. Raspberry và DHT11.

Kết nối chân DHT11:

- Vcc nối đến chân 5V trên Raspberry.
- Ground nối chân GND trên Raspberry.
- Signal nối chân GPIO 4 trên Raspberry.



b. Raspberry và Relay 4 chân.



Kết nối chân Relay 4 chân:

- VCC và GND nối vào chân 5V và GND của Raspberry.
- IN1 nối vào GPIO 18.
- IN2 nối vào GPIO 17.

- IN3 nối vào GPIO 15.
- IN4 nối vào GPIO 14.

4. Thông số cho từng khối.

a. Raspberry.

- Bộ nhớ RAM 1G
- 4 cổng USB
- Cổng HDMI, hỗ trợ Full HDMI
- Cổng Ethernet (hay là cổng mạng LAN)
- Jack cắm audio 3.5mm
- Giao tiếp Camera qua CSI
- Hỗ trợ hiển thị DSI
- Khe gắn Micro SD card được hàn chết trên board theo kiểu Push-Pull (nghĩa là bạn muốn gắn vào thì đẩy thẻ vào, lấy ra thì kéo ra), theo như hãng giải thích sẽ tốt hơn kiểu Push-Push trước kia.
- Vi xử lý hình ảnh VideoCore IV 3D.
- CPU 64 bit quad-core bộ vi xử lý ARM Cortex A53, tốc độ 1.2GHz gấp 10 lần so với thế hệ đầu tiên.
- Tích hợp wireless chuẩn 802.11n.
- Tích hợp Bluetooth 4.1 (sở hữu tính năng tiết kiệm năng lượng BLE).

b. DHT11.

- Điện áp hoạt động : 3V - 5V (DC)
- Dải độ ẩm hoạt động : 20% - 90% RH, sai số $\pm 5\%RH$
- Dải nhiệt độ hoạt động : $0^{\circ}C \sim 50^{\circ}C$, sai số $\pm 2^{\circ}C$
- Tần số lấy mẫu tối đa: 1 Hz

c. Relay 4 SSR (5VDC).

- SSR: OMRON G3MB-202P
- Số Relay: 4
- Điện áp kích: 5VDC.
- Dòng tiêu thụ: 20mA/1 Relay SSR
- Điện áp đóng ngắt tối đa: 75 to 240VAC (50/60Hz).
- Dòng điện đóng ngắt: 0.1 - 2A.
- Có cách ly: Photo Triac.
- Kích thước: 57 * 55 * 25 (L * W * H)
- Trọng lượng: 40 g.
- Relay tích cực mức thấp:
- + 0 - 0.5VDC: Mức thấp (SSR đóng).
- + 0.5 - 2.5VDC: (trạng thái bất định).
- + 2.5 - 5VDC: Mức cao (SSR ngắt).

Phần 4 : Software design document.

1. Lựa chọn công cụ phần mềm.

a. Ngôn ngữ lập trình.

JAVASCRIPT



Hình 1.1: JavaScript

JavaScript là một ngôn ngữ lập trình dựa trên nguyên mẫu với cú pháp phát triển từ C. Giống như C, JavaScript có khái niệm từ khóa, do đó, JavaScript gần như không thể được mở rộng.

Cũng giống như C, JavaScript không có bộ xử lý xuất/nhập (input/output) riêng. Trong khi C sử dụng thư viện xuất/nhập chuẩn, JavaScript dựa vào phần mềm ngôn ngữ được gắn vào để thực hiện xuất/nhập.

Trên trình duyệt, rất nhiều trang web sử dụng JavaScript để thiết kế trang web động và một số hiệu ứng hình ảnh thông qua DOM. JavaScript được dùng để thực hiện một số tác vụ không thể thực hiện được với chỉ HTML như kiểm tra thông tin nhập vào, tự động thay đổi hình ảnh,... Ở Việt Nam, JavaScript còn được ứng dụng để làm bộ gõ tiếng Việt giống như bộ gõ hiện đang sử dụng trên trang Wikipedia tiếng Việt. Tuy nhiên, mỗi trình duyệt áp dụng JavaScript khác nhau và không tuân theo chuẩn W3C DOM, do đó trong rất nhiều trường hợp lập trình viên phải viết nhiều phiên bản của cùng một đoạn mã nguồn để có thể hoạt động trên nhiều trình duyệt. Một số công nghệ nổi bật dùng JavaScript để tương tác với DOM bao gồm DHTML, Ajax và SPA. Bên ngoài trình duyệt, JavaScript có thể được sử dụng trong tập tin PDF của Adobe Acrobat và Adobe Reader. Điều khiển Dashboard trên hệ điều hành Mac OS X phiên bản 10.4 cũng có sử dụng JavaScript. Công nghệ kịch bản linh động (active scripting) của Microsoft có hỗ trợ ngôn ngữ JScript làm một ngôn ngữ kịch bản dùng cho hệ điều hành. JScript.NET là một ngôn ngữ tương thích với CLI gần giống JScript nhưng có thêm nhiều tính năng lập trình hướng đối tượng.

Từ khi Node.js ra đời vào năm 2009, Javascript được biết đến nhiều hơn là một ngôn ngữ đa nền khi có thể chạy trên cả môi trường máy chủ cũng như môi trường nhúng.

Mỗi ứng dụng này đều cung cấp mô hình đối tượng riêng cho phép tương tác với môi trường chủ, với phần lõi là ngôn ngữ lập trình JavaScript gần như giống nhau.

PYTHON



Hình 1.2: Python

Python đang là một trong các ngôn ngữ lập trình bậc cao phổ biến được sử dụng rộng rãi cho mọi chương trình máy tính. Đây là một ngôn ngữ đơn giản cả về cú pháp lẫn cách sử dụng và là ngôn ngữ dễ tiếp cận nhất cho người mới học lập trình. Python cũng được sử dụng nhiều trong các bài toán về trí tuệ nhân tạo và học máy.

Python là một ngôn ngữ lập trình dạng thông dịch, do đó có ưu điểm tiết kiệm thời gian phát triển ứng dụng vì không cần phải thực hiện biên dịch và liên kết. Trình thông dịch có thể được sử dụng để chạy file script, hoặc cũng có thể được sử dụng theo cách tương tác. Ở chế độ tương tác, trình thông dịch Python tương tự shell của các hệ điều hành họ Unix, tại đó, ta có thể nhập vào từng biểu thức rồi gõ Enter, và kết quả thực thi sẽ được hiển thị ngay lập tức. Đặc điểm này rất hữu ích cho người mới học, giúp họ nghiên cứu tính năng của ngôn ngữ; hoặc để các lập trình viên chạy thử mã lệnh trong suốt quá trình phát triển phần mềm. Ngoài ra, cũng có thể tận dụng đặc điểm này để thực hiện các phép tính như với máy tính bỏ túi.

BASH SCRIPT



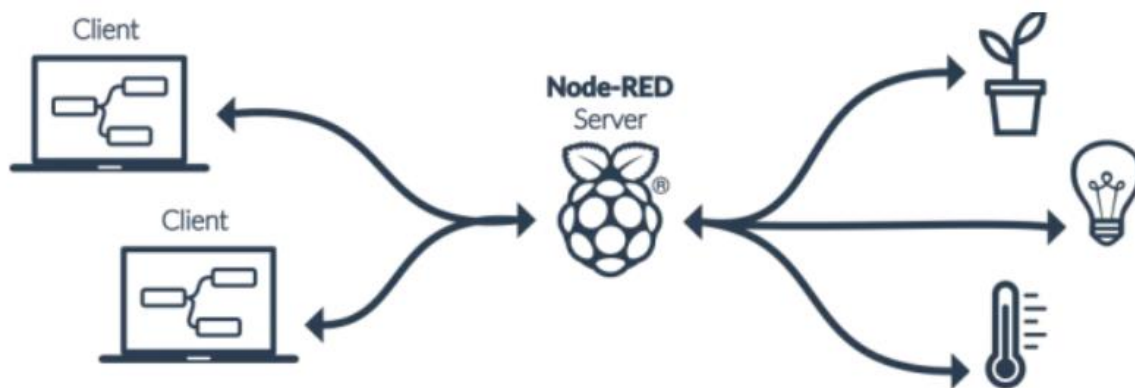
Hình 1.3: Bash Script

Shell là chương trình giao tiếp với người dùng. Có nghĩa là shell chấp nhận các lệnh từ bạn (keyboard) và thực thi nó. Nhưng nếu bạn muốn sử dụng nhiều lệnh chỉ bằng một lệnh, thì bạn có thể lưu chuỗi lệnh vào text file và bảo shell thực thi text file này thay vì nhập vào các lệnh. Điều này gọi là shell script.

Shell script là một chuỗi các lệnh được viết trong plain text file. Shell script thì giống như batch file trong MS-DOS nhưng mạnh hơn.

b. Các công cụ lập trình.

Node-RED



Hình 1.4: Node-RED

Node-RED là một công cụ mã nguồn mở mạnh mẽ để xây dựng Internet of Things (IoT). Ứng dụng với mục tiêu đơn hóa các thành phần lập trình. Nó sử dụng một lập trình trực quan cho phép bạn kết nối các khối mã, được gọi là các nút với nhau để thực hiện một nhiệm vụ. Các nút khi nối với nhau được gọi là dòng chảy. Mỗi ứng dụng Node-red bao gồm các node có thể liên kết được với nhau với các dạng là input, output và operation. Và đặc biệt là Node-Red được tích hợp sẵn trong file image của các phiên bản hệ điều hành Rasbian gốc của Raspberry pi.

Với Node-red ta có thể hình dung cách tương tác và giao tiếp với các thiết bị một cách tổng quan như hình 1.4 bên trên. Ở đây máy tính của chúng ta sẽ đóng vai trò là client kết nối tới server được mở trên raspberry chạy Node-RED.

Tasker và AutoVoice

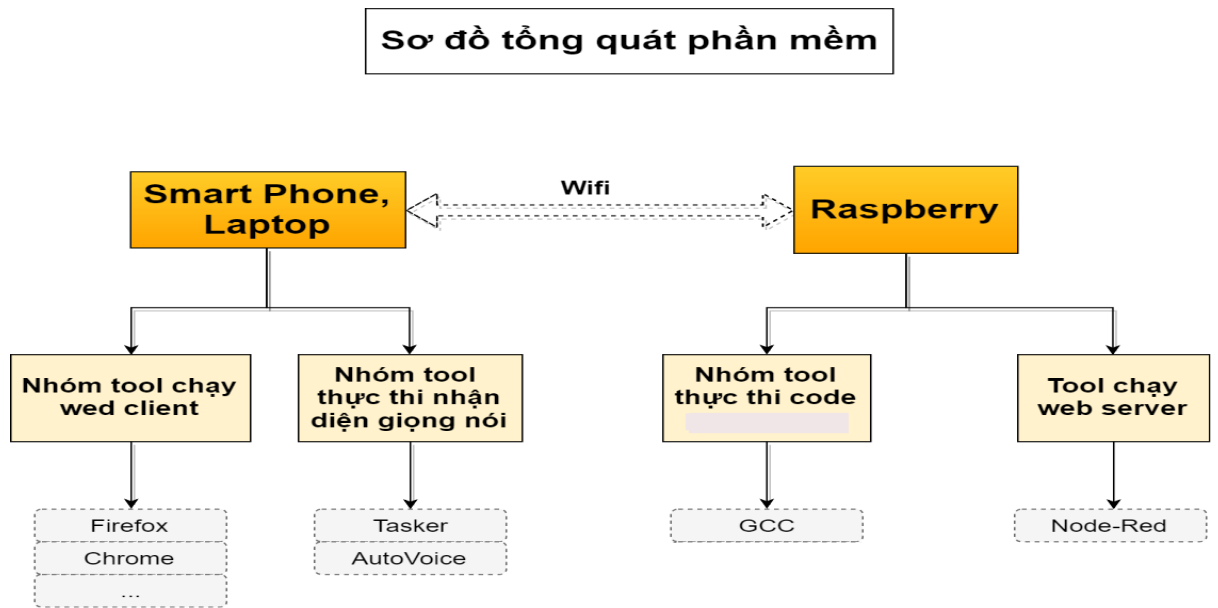


Hình 1.5: Tasker và AutoVoice

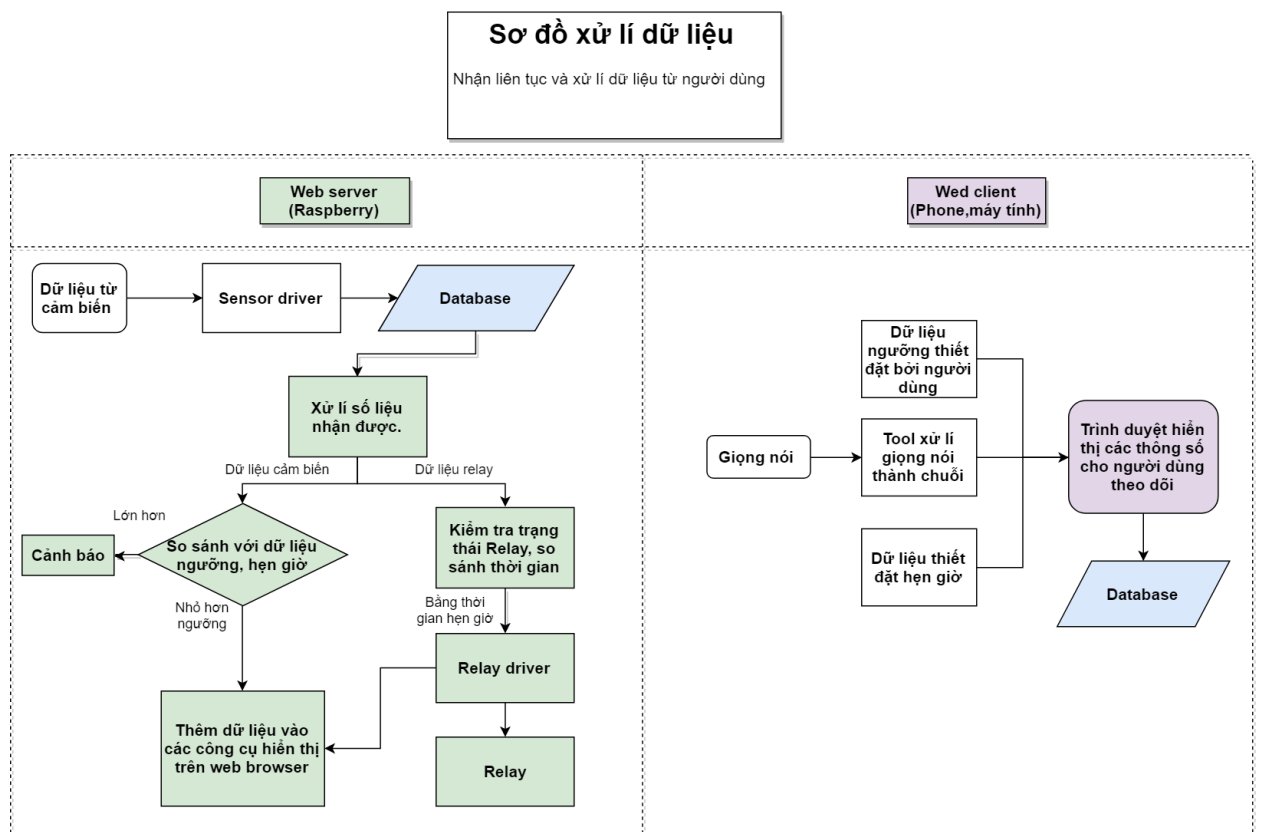
Tasker là một trong số rất ít ứng dụng cho phép người dùng tùy biến lại hệ thống hoạt động của thiết bị Android theo lối cá nhân hóa. Bạn có thể tạo hoặc chỉnh sửa nhiều hoạt động khác nhau như bật/tắt dữ liệu di động, tạo trình tiết kiệm pin, tùy biến trung tâm thông báo, thời gian, vị trí, widget,....

AutoVoice do lập trình viên Joao Dias phát triển từ Play Store. Auto giúp chuyển đổi tiếng nói của người dùng thành dạng văn bản. AutoVoice sử dụng nền tảng Google Now và là một Add-on của tasker.

2. Lưu đồ giải thuật chính.



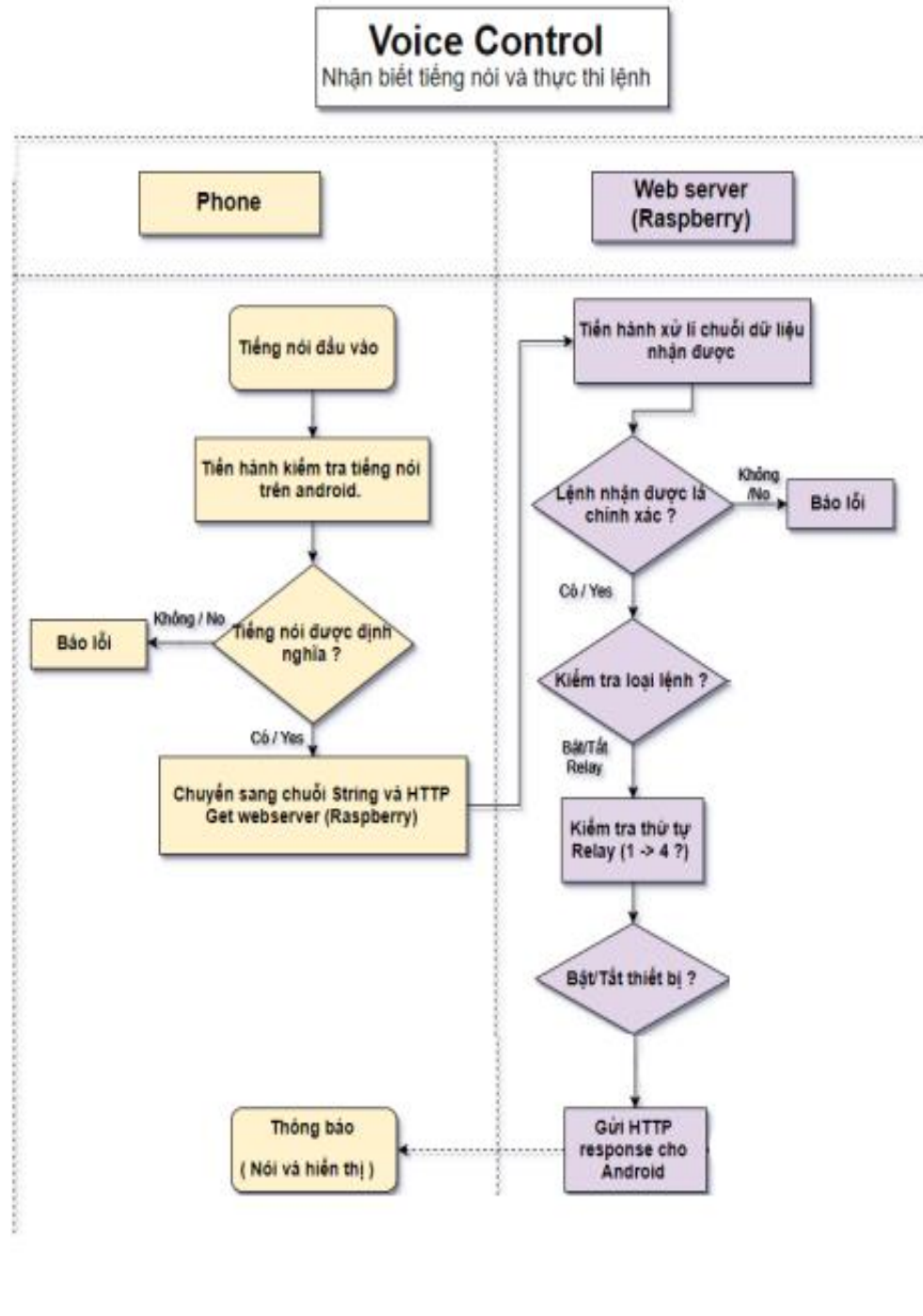
Hình 2.1: Sơ đồ tổng quát phần mềm.



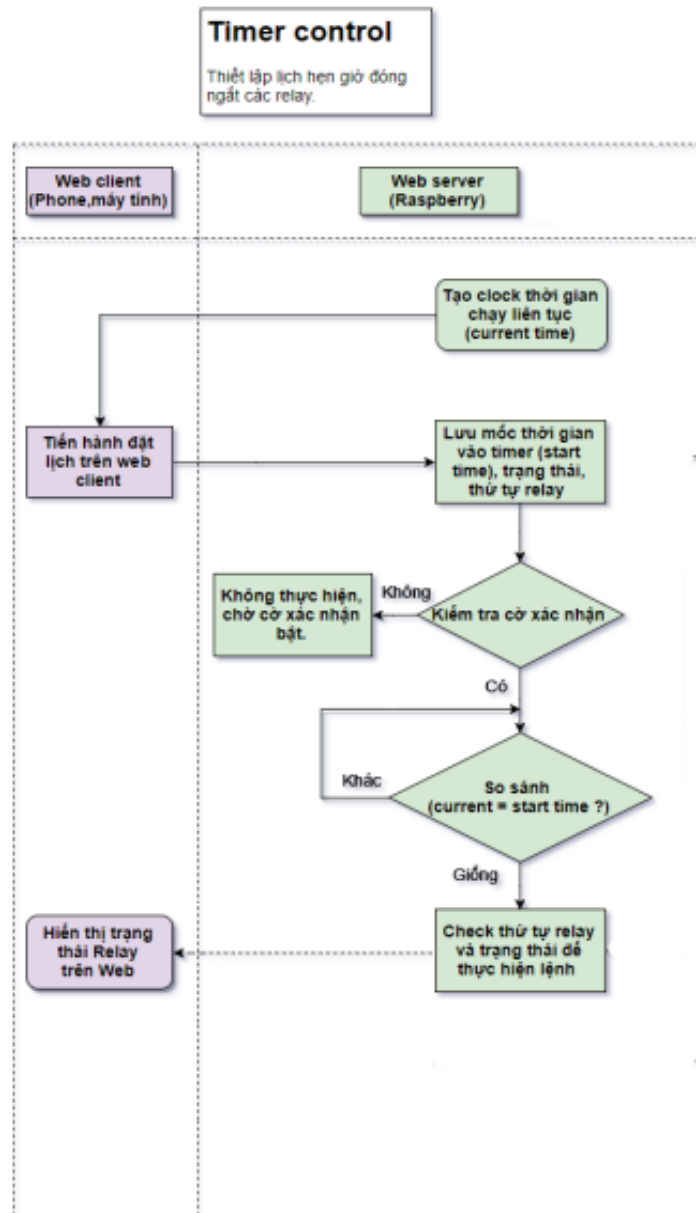
Hình 2.2: Sơ đồ xử lý dữ liệu.

3. Lưu đồ giải thuật các chương trình con.

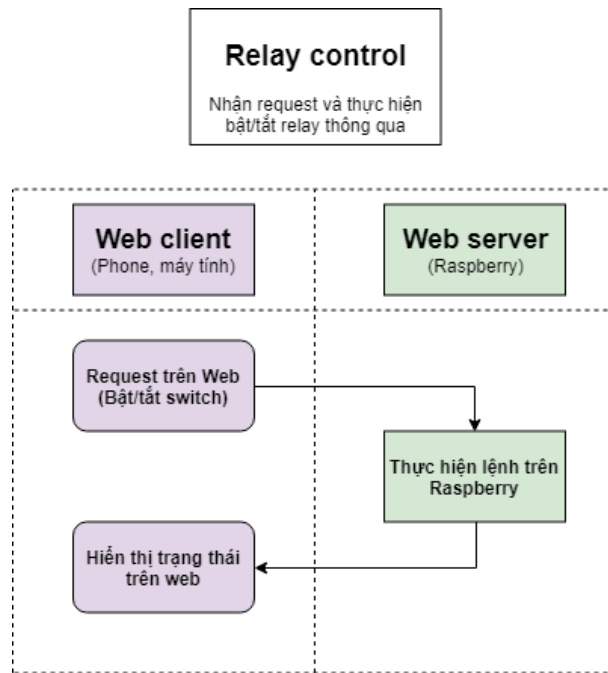
Quá trình xử lý dữ liệu giọng nói.



Quá trình xử lý hện giờ:



Quá trình xử lý Relay:



4. Mô tả và lập trình các chương trình con.

Phần xử lý chuỗi giọng nói trên webserver Node Red (JavaScript):

```

//var temp = msg.payload
var temp = msg.req.query.text;
msg.payload.command = 0;
msg.payload.success = false;
msg.payload.voice_thing=0;
msg.payload.voice_state=0;
// removing rubbish
temp = temp.toLowerCase();

// check if the command is to turn something on/off
if (temp.indexOf("đèn")>-1) {
    temp = temp.replace("đèn ", "");
    msg.payload.command = 1;
}
if (temp.indexOf("kiểm tra")>-1)
    temp = temp.replace("kiểm tra ", "");

// // check if the command is about the solar panels
if (temp.indexOf("thông số")>-1) {
    temp = temp.replace("thông số", "");
    msg.payload.command = 2;
}

// let's try finding the thing and state
// the turn/switch commands are expected as 'turn <state> the
<thing>'
var lastIndex= temp.indexOf(" ");
var voice_state = temp.substring(0, lastIndex).trim();
var voice_thing =
temp.substring(lastIndex+1,temp.length).trim();

msg.payload.voice_state=voice_state;
msg.payload.voice_thing=voice_thing
// evaluate the state

```

```

if (voice_state=="bật") {
    msg.payload.command_value = "1";
}
if (voice_state=="tắt") {
    msg.payload.command_value = "2";
}

// handle the solar request
if (msg.payload.command===2) {
    msg.payload.response="Thông số môi trường:\n Nhiệt độ " +
    global.get("tmp") + " độ C\n Độ ẩm : " + global.get("humi") + "
    %\n";
}
// handle the switch commands
if (msg.payload.command===1) {
    switch (voice_thing) {
        case "1":
            msg.payload.response="Xác nhận, đã " + voice_state +
            " đèn " + voice_thing;
            msg.payload.success = true;
            msg.payload.command_thing="1";
            break;
        case "2":
            msg.payload.response="Xác nhận, đã " + voice_state +
            " đèn " + voice_thing;
            msg.payload.success = true;
            msg.payload.command_thing="2";
            break;
        case "3":
            msg.payload.response="Xác nhận, đã " + voice_state +
            " đèn " + voice_thing;
            msg.payload.success = true;
            msg.payload.command_thing="3";
            break;
        case "4":
            msg.payload.response="Xác nhận, đã " + voice_state +
            " đèn " + voice_thing;
            msg.payload.success = true;
            msg.payload.command_thing="4";
            break;
        default:
            msg.payload.response="Xin lỗi, đèn " + voice_thing +
            "chưa được thiết lập ";
            msg.payload.success = false;
            break;
    }
}
return msg;

```

Code bật/tắt relay trên raspberry(Shell Script):

```

#!/bin/bash

# Common path for all GPIO access
BASE_GPIO_PATH=/sys/class/gpio

# Assign names to states
ON="1"
OFF="0"

# Utility function to export a pin if not already exported
exportPin()
{

```

```

    if [ ! -e $BASE_GPIO_PATH/gpio$1 ]; then
        echo "$1" > $BASE_GPIO_PATH/export
    fi
}

# Utility function to set a pin as an output
setOutput()
{
    echo "out" > $BASE_GPIO_PATH/gpio$1/direction
}

# Utility function to change state of a light
setState()
{
    echo $2 > $BASE_GPIO_PATH/gpio$1/value
}

exportPin $1
setOutput $1
setState $1 $2

```

Code đọc sensor DHT11 trên raspberry (Python):

```

import sys

import Adafruit_DHT
# Parse command line parameters.
sensor_args = { '11': Adafruit_DHT.DHT11,
                 '22': Adafruit_DHT.DHT22,
                 '2302': Adafruit_DHT.AM2302 }
if len(sys.argv) == 3 and sys.argv[1] in sensor_args:
    sensor = sensor_args[sys.argv[1]]
    pin = sys.argv[2]
else:
    print('Usage: sudo ./Adafruit_DHT.py [11|22|2302] <GPIO pin number>')
    print('Example: sudo ./Adafruit_DHT.py 2302 4 - Read from an AM2302 connected to GPIO pin #4')
    sys.exit(1)

# Try to grab a sensor reading.  Use the read_retry method which will retry up
# to 15 times to get a sensor reading (waiting 2 seconds between each retry).
humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

# Un-comment the line below to convert the temperature to Fahrenheit.
# temperature = temperature * 9/5.0 + 32

# Note that sometimes you won't get a reading and
# the results will be null (because Linux can't
# guarantee the timing of calls to read the sensor).
# If this happens try again!
if humidity is not None and temperature is not None:
    print('Temp={0:0.1f}* Humidity={1:0.1f}%'.format(temperature, humidity))
else:
    print('Failed to get reading. Try again!')
    sys.exit(1)

```