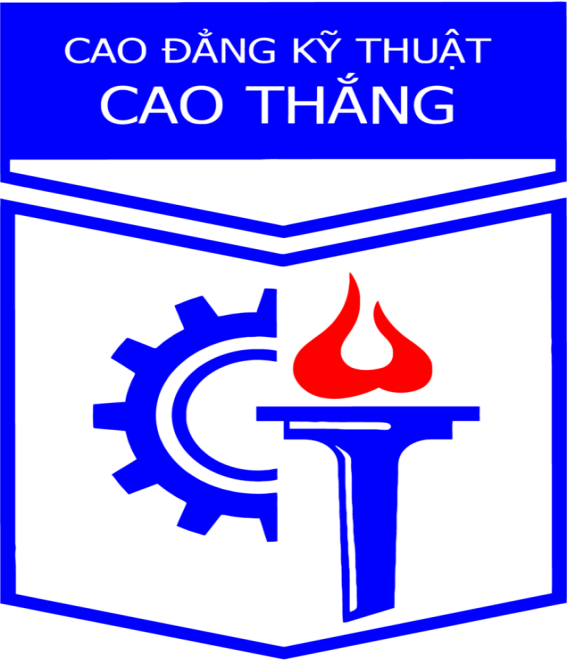
**TRƯỜNG CAO ĐẲNG KỸ THUẬT CAO THẮNG**

**KHOA: ĐIỆN ĐIỆNTỬ**



**BÀI TẬP NGHIÊN CỨU**

**ĐỀ TÀI: SỬ DỤNG KẾT HỢP CẢM BIẾN ULTRA SONIC, MÀN HÌNH OLED SSD1306   
VÀ PICAMERA2 TRÊN RASPBERRY PI 4**

**HỌ VÀ TÊN** :PHẠM THANH HẢI

NGÔ BẢO LONG  
 PHAN TÙNG HIẾU

LÊ THÁI TRUNG QUÂN  
 VŨ PHẠM ĐÌNH KHANG

**LỚP** : CĐ ĐTTT22B

**GIÁO VIÊN BỘ MÔN** : ĐỖ NGỌC THANH CHÂU

**MỤC LỤC**

[1. PHẦN MỞ ĐẦU: 2](#_Toc3025)

[a) RASPBERRY LÀ GÌ ?: 2](#_Toc5827)

[b) PHẦN CỨNG: 3](#_Toc258)

[c) GPIO: 8](#_Toc14258)

[2. ĐỀ TÀI NGHIÊN CỨU: 9](#_Toc12105)

[a) Nguyên lý hoạt động 9](#_Toc16834)

[b) Sơ đồ phần cứng: 10](#_Toc31239)

[c) Code: 11](#_Toc12249)

[3. KẾT LUẬN: 16](#_Toc10607)

[a) Kết quả: 16](#_Toc1719)

[b) Ứng dụng: 17](#_Toc32566)

1. **PHẦN MỞ ĐẦU:**
2. **RASPBERRY LÀ GÌ** ?:

Raspberry Pi là các ****máy tính bo mạch đơn**** (hay còn gọi là máy tính nhúng) kích thước chỉ bẳng 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.

Các công cụ có sẵn cho Python như là ngôn ngữ lập trình chính, hỗ trợ cho BBC BASIC (thông qua RISC OS image hoặc Brandy Basic clone cho Linux), C, C++, Java, [Perl](https://vi.wikipedia.org/wiki/Perl" \o "Perl) và Ruby

Hỗ trợ các giao thức kết nối:

Raspberry Pi 4 có nhiều hệ thống liên lạc hiện đại. Nó có Wifi bên trong và Bluetooth để giao tiếp dữ liệu không dây. Có thể được sử dụng bên trong hệ thống ở bất cứ đâu mà không có bất kỳ sự xung đột nào.

Pi chuyển đổi dữ liệu dễ dàng trong cùng một mạng do có Wifi. Thiết bị cũng có hỗ trợ mạng LAN trong trường hợp không có Wifi và là mạng truyền thông có dây:

LAN - Gigabyte Ethernet

Bluetooth - 5.0

Wifi - 2.4 với tốc độ 5GHz

**Các thông số kỹ thuật của raspberry pi 4:**

|  |  |
| --- | --- |
| **CPU** | Quad-core Cortex-A72 (64-bit) @ 1.5GHz |
| **GPU** | H264 (1080p60 decode, 1080p30 encode) OpenGL ES 3.0 graphics, H.265 (4kp60 decode) |
| **RAM** | 1GB, 2GB, 4GB. |
| **Dải điện áp hoạt động** | 5V với dòng tối thiểu 3A |
| **CỔNG GPIO** | 28 chân I / O |
| **LAN** | Có |
| **PoE** | Cho phép |
| **WifI** | Có |
| **Bluetooth** | 5.0 |
| **Thẻ SD** | Có |
| **HDMI** | 2 Cổng Màn hình 4k (mini-HDMI) |
| **PWR Exp Header** | Không có |
| **Nguồn cấp** | Giắc cắm nguồn DC, Cổng USB-C mini |
| **Kết nối mở rộng** | 40 chân (SPI, I 2 C, LCD, UART, PWM, SDIO) |
| **USB** | 2 × 2.0, 2 × 3.0 |
| **Máy ảnh** | CSI |
| **Display** | DSI |
| **Nhiệt độ hoạt động** | 0-50 độ |

1. **PHẦN CỨNG:**

**Sơ đồ chân:**

****

**Mô tả tính năng các chân kết nối của Raspberry pi 4**

Raspberry Pi 4 có thể sử dụng trong hệ thống nhúng bên ngoài để giao tiếp tín hiệu. Có tổng cộng 40 chân , trong đó 28 chân là chân GPIO và các chân còn lại là chân nguồn.

 Các chân GPIO không chỉ thực hiện các chức năng I / O đơn giản mà còn hỗ trợ giao thức UART, SPI và I 2 C. Các giao thức này dành riêng cho mọi chân và tất cả chức năng của chúng được thảo luận dưới đây:

**Chân cấp nguồn**

Power In: Trong Raspberry pi, có hai cách cấp nguồn, một là từ cổng nguồn USB-C và thứ hai là từ các chân 5V. Chân 5V được kết nối trực tiếp với cổng adapter USB-C.

Đầu vào ở chân 5V phải ổn định và theo đúng thông số kỹ thuật. Trong trường hợp có điện áp cao hơn, thiết bị có thể bị cháy.

Các chân đầu vào 5V sẽ không có cầu chì và bộ điều chỉnh điện áp khi được sử dụng làm đầu vào cấp nguồn, do đó nguồn điện 5V phải theo đúng thông số kỹ thuật để tránh hư hại. Chân đầu vào cấp nguồn cho Raspberry Pi 4 được cung cấp bên dưới:

* Từ chân 2 đến 6 -> + 5V
* Chân 6 —–> GND

Power Out: Có hai loại chân nguồn ra trong Raspberry pi 4 là 3V3 và 5V. 5V được kết nối trực tiếp với cổng USB nhưng 3V3 được kết nối thông qua bộ điều chỉnh điện áp cho ra đầu ra 3V ổn định. Tất cả các chân nguồn ra được cung cấp bên dưới:

* 3V3 - Chân 1, chân 17
* 5V - Chân 2, chân 6

Ground: Raspberry Pi 4 có nhiều chân ground được kết nối bên trong và các chân này này có thể làm điểm nối đất chung cho nguồn điện hoặc thiết bị bên ngoài. Danh sách các chân ground được đưa ra dưới đây:

* Chân 6
* Chân 9
* Chân 14
* Chân 20
* Chân 25
* Chân 30
* Chân 34
* Chân 39

**Chân I/O digital**

Hầu hết mọi thiết bị đều cần các chân đầu vào và đầu ra để giao tiếp. Trong thiết bị này có 28 chân GPIO được sử dụng làm đầu vào và đầu ra digital. Các chân GPIO trong bộ điều khiển có một số giá trị mặc định.

Các chân GPIO từ 0-9 sẽ ở trạng thái logic cao và từ 10 trở lên các chân sẽ ở trạng thái logic thấp. Tất cả các chân đó trong Raspberry Pi 4 đều được cung cấp bên dưới:

* GPIO0 - Chân 27
* GPIO1 - Chân 28
* GPIO2 - Chân 3
* GPIO3 - Chân 5
* GPIO4 - Chân 7
* GPIO5 - Chân 29
* GPIO6 - Chân 31
* GPIO7 - Chân 26
* GPIO8 - Chân 24
* GPIO9 - Chân 21
* GPIO10 - Chân 19
* GPIO11 - Chân 23
* GPIO12 - Chân 32
* GPIO13 - Chân 33
* GPIO14 - Chân 8
* GPIO15 - Chân 10
* GPIO16 - Chân 36
* GPIO17 - Chân 11
* GPIO18 - Chân 12
* GPIO19 - Chân 35
* GPIO20 - Chân 38
* GPIO21 - Chân 40
* GPIO22 - Chân 15
* GPIO23 - Chân 16
* GPIO24 - Chân 18
* GPIO25 - Chân 22
* GPIO26 - Chân 37
* GPIO27 - Chân 13

Tất cả các chân GPIO trong Raspberry Pi 4 không có chức năng đầu vào-đầu ra. Mỗi chân GPIO có thể thực hiện các chức năng khác thông qua lập trình.

**Module giao tiếp dữ liệu nối tiếp Raspberry Pi**

**Các chân UART trong Raspberry Pi**

Có nhiều giao thức nối tiếp và UART là một trong số đó. Nó khá phổ biến vì có hệ thống giao tiếp đơn giản và phụ thuộc vào hầu hết các phần mềm. Có nhiều chân giao tiếp UART trong Raspberry pi 4 được đưa ra bên dưới:

* TXD1 - GPIO14 - Chân 8
* RXD1 - GPIO15 - Chân 10
* TXD2 - GPIO0 - Chân 27
* RXD2 - GPIO1 - Chân 28
* TXD3 - GPIO5 - Chân 29
* RXD3 - GPIO4 - Chân 7
* TXD4 - GPIO8 - Chân 24
* RXD4 - GPIO9 - Chân 21
* TXD5 - GPIO12 - Chân 32
* RXD5 - GPIO13 - Chân 33

**Chân giao tiếp SPI**

Một số thiết bị sử dụng giao thức SPI và nó giúp điều khiển nhiều thiết bị bằng cách sử dụng 1 đường truyền dữ liệu duy nhất. Trong Raspberry pi 4 có nhiều chân SPI được sử dụng cho giao tiếp SPI . Chân SPI của Raspberry Pi 4 được cung cấp bên dưới:

* SPI3 CEO N - GPIO0 - Chân 27
* SPI3 MISO - GPIO1 - Chân 28
* SPI3 MOSI - GPIO2 - Chân 3
* SPI3 SCLK - GPIO3 - Chân 5
* SPI4 CEO N - GPIO4 - Chân 7
* SPI4 MISO - GPIO5 - Chân 29
* SPI4 MOSI - GPIO6 - Chân 31
* SPI4 SCLK - GPIO7 - Chân 26
* SPI0 CE1 N - GPIO8 - Chân 24
* SPI0 CE0 N - GPIO9 - Chân 21
* SPI0 MISO - GPIO10 - Chân 19
* SPI0 MOSI - GPIO11 - Chân 23
* SPI5 CEO N / SPI0 SCLK - GPIO12 - Chân 32
* SPI5 MISO - GPIO13 - Chân 33
* SPI5 MOSI - GPIO14 - Chân 8
* SPI5 SCLK - GPIO15 - Chân 10
* CTS0 - GPIO16 - Chân 36
* RTS0 - GPIO17 - Chân 11
* SPI6 CEO N - GPIO18 - Chân 12
* SPI6 MISO - GPIO19 - Chân 35
* SPI6 MOSI - GPIO20 - Chân 38
* SPI6 SCLK - GPIO21 - Chân 40

**Chân giao tiếp I 2 C**

Raspberry Pi 4 có hỗ trợ giao thức I2C. Là giao thức được sử dụng ở một số cảm biến và động cơ. Tất cả các chân này được đưa ra bên dưới:

* SDA0 / SDA6 - GPIO0 - Chân 27
* SCL0 / SCL6 - GPIO1 - Chân 28
* SDA1 / SDA3 - GPIO2 - Chân 3
* SCL1 / SCL3 - GPIO3 - Chân 5
* SDA3 - GPIO4 - Chân 7
* SCL3 - GPIO5 - Chân 29
* SDA4 - GPIO6 - Chân 31
* SCL4 - GPIO7 - Chân 26
* SDA4 - GPIO8 - Chân 24
* SCL4 - GPIO9 - Chân 21
* SDA5 - GPIO10 - Chân 19
* SCL5 - GPIO11 - Chân 23
* SDA5 - GPIO12 - Chân 32
* SCL5 - GPIO13 - Chân 33
* SDA6 - GPIO22 - Chân 15
* SCL6 - GPIO23 - Chân 16

**Các chân GPIO PWM**

Để tạo ra tín hiệu đầu ra xung mong muốn Raspberry Pi 4 có một số chân PWM. Các chân này cấp dữ liệu trực tiếp cho các thiết bị ngoại vi điện áp thấp. Để tạo ra tín hiệu đầu tiên các chân phải được lập trình trước. Tất cả các chân PWM được đưa ra bên dưới:

* PWM0 - GPIO12 - Chân 32
* PWM1 - GPIO13 - Chân 33
* PWM0 - GPIO18 - Chân 12
* PWM1 - GPIO19 - Chân 35

**Các chân SDIO Raspberry Pi**

Trong Raspberry Pi 4 có một khe cắm thẻ SD nhưng các cgaab GPIO cũng hỗ trợ khả năng tương thích với thẻ SD. Chân SDIO trên thiết bị có thể được sử dụng cho thẻ SD khi có ứng dụng yêu cầu:

* SD0CLK / SD1 CLK - GPIO22 - Chân 15
* SD0 CMD / SD1 CMD - GPIO23 - Chân 16
* SD0 DATA0 / SD1 DAT0 - GPIO24 - Chân 18
* SD0 DAT1 / SD1 DAT1 - GPIO25 - Chân 22
* SD1 DAT2 / SD1 DAT2 - GPIO26 - Chân 37
* SD0 DAT3 / SD1 DAT3 - GPIO27 - Chân 13

**Các thiết bị ngoại vi chính khác:**

USB: Có bốn cổng USB trong Raspberry Pi 4. Hai cổng hỗ trợ 2.0 nhưng hai cổng còn lại là 3.0. USB 3.0 này cho phép người dùng truyền dữ liệu nhanh.

PoE Header: Do việc phổ biến sử dụng Raspberry Pi trong IoT và các dự án thông minh khác, PoE Header cũng đã xuất hiện trong Pi. PoE là một bo mạch cho phép người dùng cấp điện cho thiết bị thông qua Ethernet Wire. Trong trường hợp dùng PoE, một PoE HAT bên ngoài sẽ được yêu cầu.

Máy ảnh: Thiết bị có hỗ trợ máy ảnh. Có một cổng máy ảnh MIPI CSI 2-lane có thể được sử dụng để kết nối pi trực tiếp với máy ảnh và sử dụng mà không cần các giao tiếp thứ ba nào.

Màn hình: Raspberry Pi 4 có thể được kết nối với màn hình LCD bên ngoài. Không sử dụng header mở rộng để giao tiếp với màn hình LCD như các thiết bị khác. Nó có một cổng MIPI DSI 2-lane riêng biệt có thể được sử dụng để giao tiếp với màn hình LCD tương thích bên ngoài.

Âm thanh: Dữ liệu âm thanh có thể được truyền từ pi đến thiết bị display thông qua HDMI nhưng nó có một cổng âm thanh 4 cực riêng biệt có thể được sử dụng để gửi và nhận tín hiệu âm thanh. Tín hiệu từ thiết bị có thể được sử dụng bởi chương trình bên trong hoặc các thiết bị khác kết nối ở header mở rộng.

THẺ SD: Đây là phần cần thiết nhất của Raspberry Pi. Hệ điều hành của Pi sẽ được lưu trong thẻ SD và sau đó được sử dụng thông qua khe cắm Thẻ SD.

1. **GPIO:**

****GPIO**** là từ viết tắt của General Purpose Input / Output, nghĩa là Đầu vào / Đầu ra có mục đích chung. Chúng là các chân có thể được cấu hình để thực hiện các chức năng khác nhau, do đó chúng có mục đích chung chứ không phải cho một mục đích sử dụng cụ thể.

Tính năng:

Và giữa **ĐẶC ĐIỂM KỸ THUẬT **của GPIO**** Nổi bật nhất:

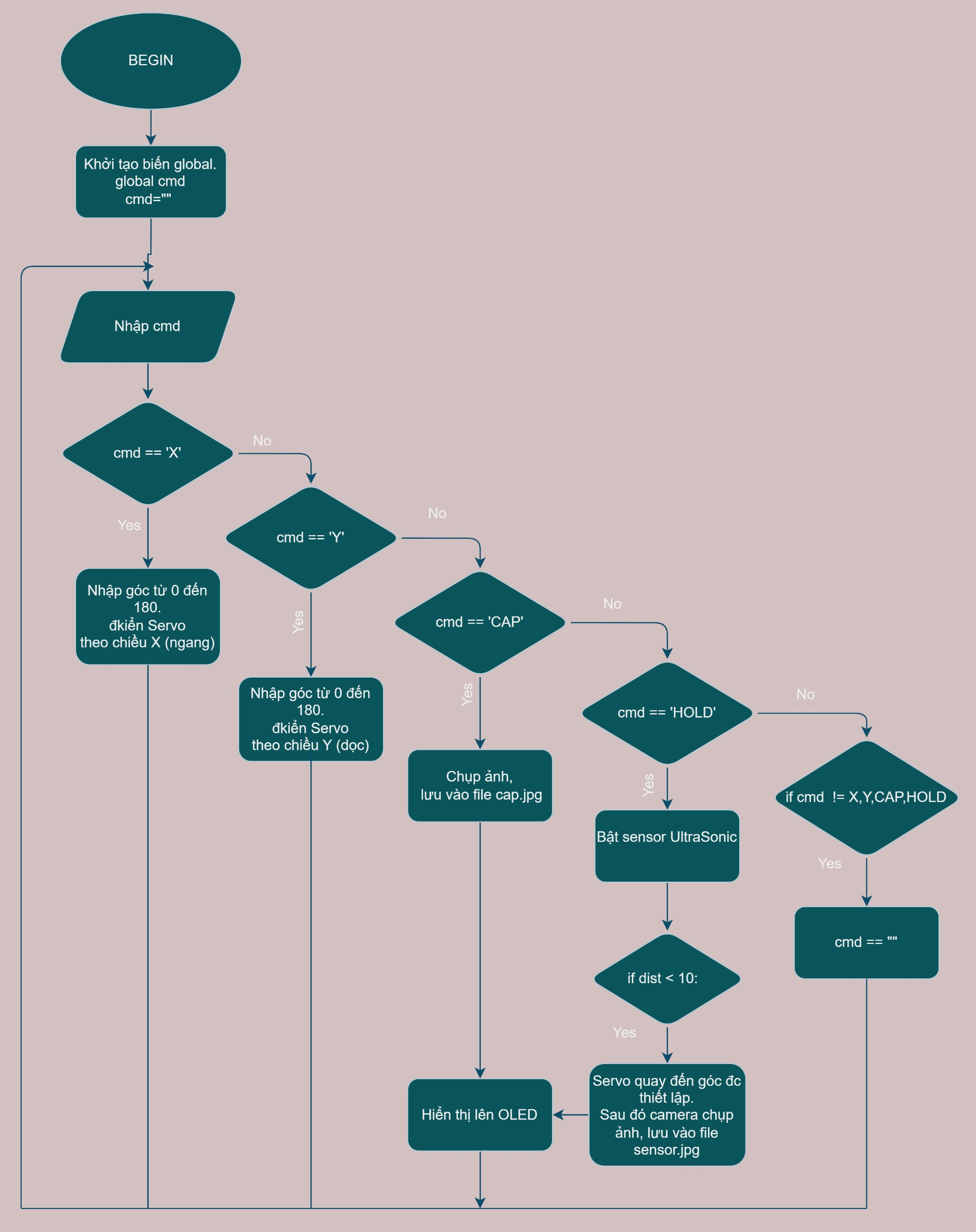
* Các chân GPIO cũng vậy ****có thể được kích hoạt và hủy kích hoạt**** bằng mã. Tức là chúng có thể được đặt thành 1 (mức điện áp cao) hoặc 0 (mức điện áp thấp).
* Tất nhiên GPIO có thể****đọc dữ liệu nhị phân****, như 1 và 0, có nghĩa là, tín hiệu điện áp hoặc không có nó.
* Giá trị đầu ra của ****Đọc và viết****.
* Các giá trị đầu vào có thể được định cấu hình trong một số trường hợp như****Sự kiện**** để chúng tạo ra một số loại hành động trên bảng hoặc hệ thống. Một số hệ thống nhúng sử dụng chúng làm IRQ. Một trường hợp khác là định cấu hình khi một hoặc nhiều chân hoạt động bởi một số cảm biến nhất định, hãy thực hiện một số hành động ...
* Đối với điện áp và cường độ, bạn phải biết rõ về công suất tối đa có thể chấp nhận được đối với bo mạch, trong trường hợp này là Raspberry Pi 4 hoặc 3. Bạn không nên vượt qua chúng để tránh làm hỏng nó.

Nhân tiện, khi một nhóm các chân GPIO được nhóm lại, như trường hợp của Raspberry Pi, nhóm này được gọi là ****Cổng GPIO****.

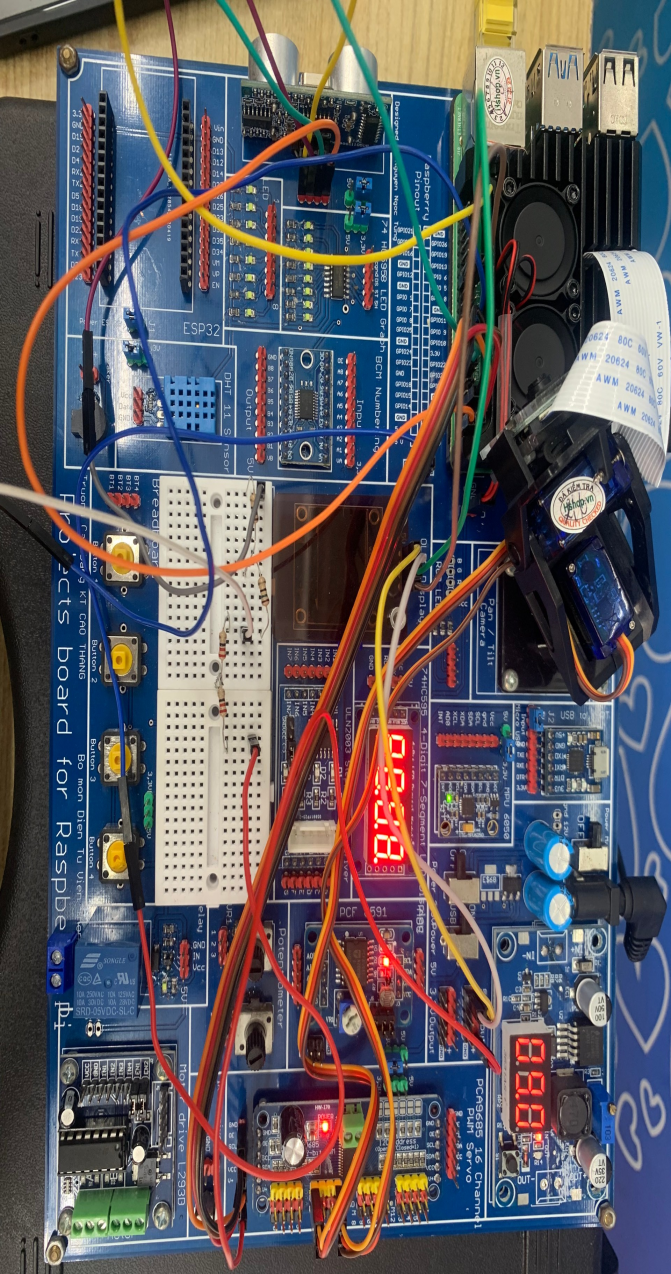
1. **ĐỀ TÀI NGHIÊN CỨU:**

Điều khiển picamera 2 trên raspberry pi 4 bằng cách nhập tọa độ X và Y, đồng thời khi nhập lệnh sẽ khởi động cảm biển HC-SR04 bắt được chuyển động của 1 vật thể bất kỳ trong 1 khoảng cách nhất định thì picamera 2 sẽ lập tức quay về phía của vật thể và chụp hình lại và hiển thị lên màn hình Oled.

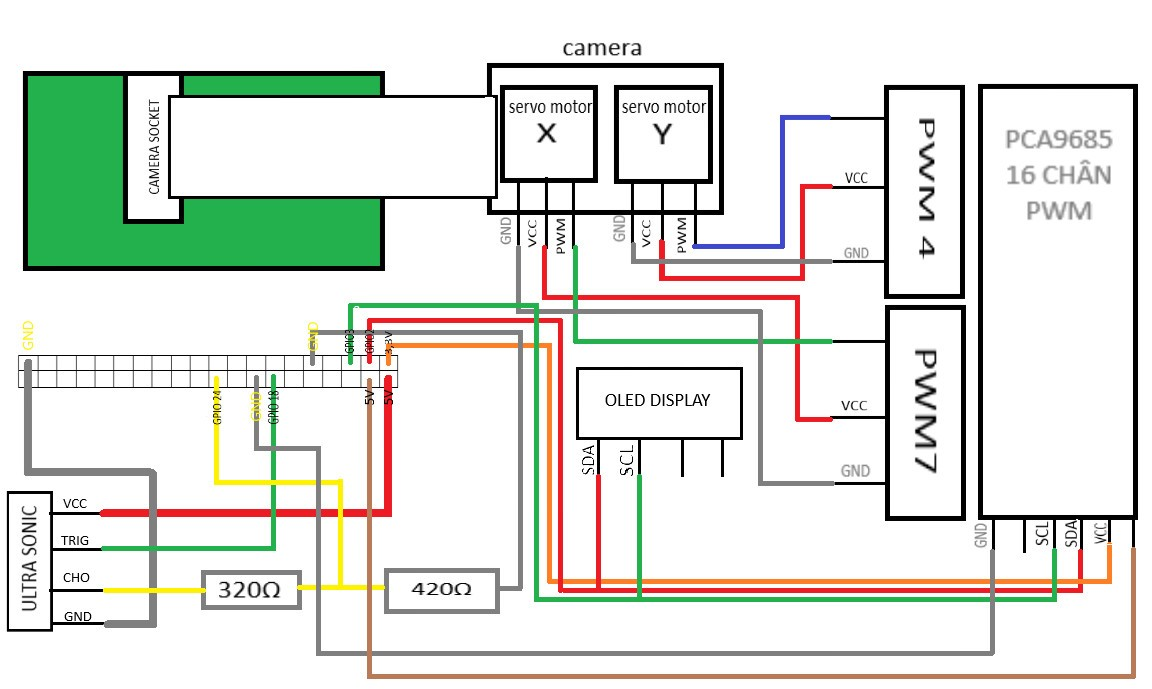
1. **Nguyên lý hoạt động**



1. **Sơ đồ phần cứng:**



**Hình ảnh phần cứng**



**Hình ảnh sơ đồ mô phỏng phần cứng**

1. **Code:**

from time import sleep, time

#--------UltraSonic setup

import RPi.GPIO as GPIO

#GPIO Mode (BOARD / BCM)

GPIO.setmode(GPIO.BCM)

#set GPIO Pins

GPIO\_TRIGGER = 18

GPIO\_ECHO = 24

#set GPIO direction (IN / OUT)

GPIO.setup(GPIO\_TRIGGER, GPIO.OUT)

GPIO.setup(GPIO\_ECHO, GPIO.IN)

def distance():

# set TRIGGER to HIGH

GPIO.output(GPIO\_TRIGGER, True)

# set TRIGGER after 0.01ms to LOW

sleep(0.00001)

GPIO.output(GPIO\_TRIGGER, False)

StartTime = time()

StopTime = time()

# save StartTime

while GPIO.input(GPIO\_ECHO) == 0:

StartTime = time()

# save time of arrival

while GPIO.input(GPIO\_ECHO) == 1:

StopTime = time()

# time difference between start and arrival

TimeElapsed = StopTime - StartTime

# multiply with the sonic speed (34300 cm/s)

# and divide by 2, because there and back

distance = (TimeElapsed \* 34300) / 2

return distance

#-------Camera setup

from picamera2 import Picamera2, Preview

from libcamera import Transform

picam2 = Picamera2()

#Xoay hình ảnh từ camera 180 độ theo chiều dọc

camera\_config = picam2.create\_preview\_configuration(transform=Transform(vflip=True))

picam2.configure(camera\_config)

picam2.start\_preview(Preview.QTGL)

picam2.start()

#-------Servo setup

from adafruit\_servokit import ServoKit

kit = ServoKit(channels=8)

def doc ():

truc\_Y=int(input("Nhap goc:"))

kit.servo[4].angle = truc\_Y

if truc\_Y < 0:

truc\_Y = 0

if truc\_Y > 180:

truc\_Y =180

def ngang ():

truc\_X = int(input("Nhap goc:"))

kit.servo[7].angle = truc\_X

if truc\_X > 180:

truc\_X =180

if truc\_X < 0:

truc\_X = 0

buf=truc\_X

#---------------oled setup

import board

import digitalio

from PIL import Image, ImageDraw, ImageFont

import adafruit\_ssd1306

RESET\_PIN = digitalio.DigitalInOut(board.D4)

i2c = board.I2C()

oled = adafruit\_ssd1306.SSD1306\_I2C(128, 64, i2c)

oled.fill(0)

oled.show()

image = Image.new("1", (oled.width, oled.height))

draw = ImageDraw.Draw(image)

font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 18)

global cmd #Khởi tạo biến global

cmd = ""

while True:

if(cmd==""): #Lựa chọn thực hiện tác vụ nào

cmd = input("Nhap: ")

if(cmd == 'Y'): #Điều khiển servo dọc

doc()

cmd=""

if(cmd == 'X'): #Điều khiển servo ngang

ngang()

cmd=""

if(cmd == 'CAP'): #Chụp ảnh, lưu vào file cap.jpg

picam2.capture\_file("cap.jpg")

cmd=""

#Sau khi chup hien thi len oled

image = (

Image.open("cap.jpg")

.resize((oled.width, oled.height), Image.BICUBIC)

.convert("1")

)

oled.image(image)

oled.show()

sleep(1)

if(cmd == 'HOLD'): #Bật sensor UltraSonic

dist = distance()

#Khi có vật cản trong khoảng dưới 10cm

#Điều khiển servo cho camera hướng về phía UltraSonic và chụp hình vật cản

if dist < 10:

kit.servo[4].angle = 90

kit.servo[7].angle = 20

sleep(0.7)

picam2.capture\_file("sensor.jpg")

#Sau khi chup hien thi len oled

image = (

Image.open("sensor.jpg")

.resize((oled.width, oled.height), Image.BICUBIC)

.convert("1")

)

oled.image(image)

oled.show()

cmd=""

if(cmd == 'SHOWCAP'): #Xuất ảnh chụp từ file cap.jpg

image = (

Image.open("cap.jpg")

.resize((oled.width, oled.height), Image.BICUBIC)

.convert("1")

)

oled.image(image)

oled.show()

sleep(1)

if(cmd == 'SHOWSEN'): #Xuất ảnh chụp từ file sensor.jpg

image = (

Image.open("sensor.jpg")

.resize((oled.width, oled.height), Image.BICUBIC)

.convert("1")

)

oled.image(image)

oled.show()

sleep(1)

if(cmd!='HOLD' and cmd!='X' and cmd!='Y' and cmd!='CAP' and cmd!=""):

cmd=""

1. **KẾT LUẬN:**
2. **Kết quả:**

Sau khi dành ra rất nhiều thời gian, đặc biệt là những ngày lễ quý giá để nghiên cứu và tìm hiểu về nguyên lý hoạt động của cảm biến HC-SR0, màn hình oled và Camera trên Raspberry pi 4, thì chúng mình đã có thể điều khiển được Servo Motor để di chuyển Camera bằng cách nhập tọa độ 2 trục X và Y trên cmd của Raspberry và chụp hình gửi vào file hệ thống của nhóm hoặc là điều khiển cho Servo Motor qua về 1 góc được chỉ định và chụp hình lại rồi hiển thị lên màn hình Oled đồng thời lưu vào file sensor.jpg khi mà CB Ultra Sonic bắt được chuyển động bằng câu lệnh “HOLD”. Ngoài ra có thể chụp lại mọi thứ trong tầm nhìn của camera rồi hiển thị những hình đó lên màn hình Oled bằng lệnh “CAP”.



Hình ảnh thu được từ camera hiển thị lên Oled

1. **Ứng dụng:**

Ứng dụng có lẽ quen thuộc và phù hợp nhất đối với nhóm mình là hệ thống chống trộm, hệ thống bảo mật. Nhưng nếu để có thể đi xa hơn nữa thì sẽ là hệ thống bảo mật nhận diện khuôn mặt, hệ thống cảm biến và ghi hình vật thể và rất nhiều ứng dụng khác mà mình không thể kể hết được.

*CẢM ƠN CÁC BẠN ĐÃ DÀNH THỜI GIAN ĐỂ ĐỌC  
BÀI NGHIÊN CỨU NÀY*