**TÀI LIỆU EDGE COMPUTING**

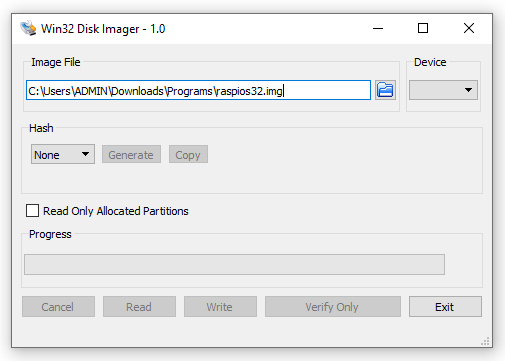
# GIỚI THIỆU VỀ RASPBERRY PI

## Thông số

|  |  |
| --- | --- |
| Modem | Raspberry Pi 3B+ |
| Bộ nhớ tối đa | 32GB |
| Ram | 1GB |
| Nguồi cấp tối ưu | 5v – 1.2A |

## Hướng dẫn cài đặt hệ điều hành raspbian

* [Tải OS](https://www.raspberrypi.org/software/)
* Tạo file boot vào thẻ SD. Yêu cầu:
* Thẻ nhớ dung lượng tối thiểu 8GB Class 10 trở lên
* Đầu đọc thẻ nhớ hoặc các thiết bị có chức năng tương tự
* Phần mềm tạo file boot. Gợi ý sử dụng [**Win32 Disk Imager (WDI)**](https://sourceforge.net/projects/win32diskimager/)
* Các bước tạo file boot:
* Bước 1: Chèn thẻ MicroSD vào đầu đọc thẻ ở máy tính và kiểm tra tên ổ được gán cho thẻ nhớ (ví dụ ổ H:), tránh nhầm ổ dẫn đến mất dữ liệu vì phần mềm sẽ format thẻ.
* Bước 2: Mở phần mềm Win32 Disk Imager, phần mềm này chỉ cần download về rồi chạy mà không cần cài đặt.
* Bước 3: Lựa chọn file hệ điều hành đã tải về. (Lưu ý, hệ điều hành cần phải ở định dạng .img. Thông thường, hệ điều hành của RPI được nén dưới dạng .zip hoặc .tar.gz, … Khi tải về cần giải nén nó ra để có file hệ điều hành dạng .img). Sau đó lựa chọn ổ thẻ nhớ cần ghi.



Sau khi trỏ tới đúng đường link file .img, nhấn nút Write để tiến hành ghi OS vào thẻ. Cài đặt hệ điều hành:

* Trường hợp 1: Có màn hình rời: Khi có màn hình rời, chỉ tháo thẻ nhớ, lắp vào RPI, kết nối RPI với màn hình qua dây HDMI, bật lên và thực hiện các thao tác cài đặt ban đầu (Ngôn ngữ, thời gian, kết nối Wifi,…)
* Trường hợp 2: Nếu không có màn hình, có thể remote Pi (kết nối Wifi)
* Thiết lập mạng Wifi: tạo file cấu hình wpa\_supplicant.conf trong thư mục boot
* Mở file và thêm các dòng lệnh sau:

**ctrl\_interface=DIR=/var/run/wpa\_supplicant GROUP=netdev**

**update\_config=1**

**country=VN**

**network={**

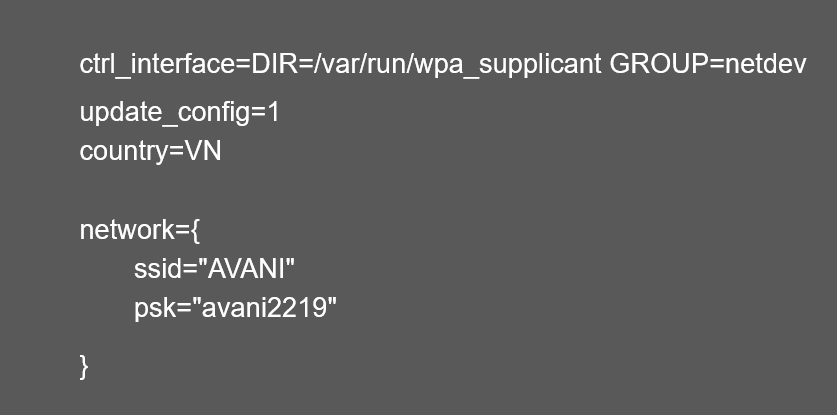
**ssid=" "**

**psk=" "**

**}**

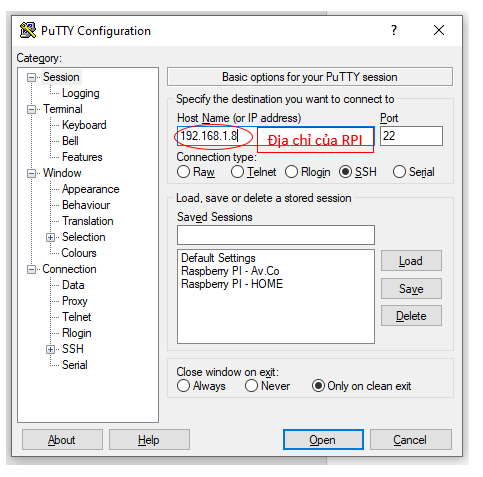
Trong đó ssid là tên wifi, psk là mật khẩu tương ứng.

Ví dụ:



Ngoài ra có thể sử dụng cáp Ethernet để kết nối mạng.

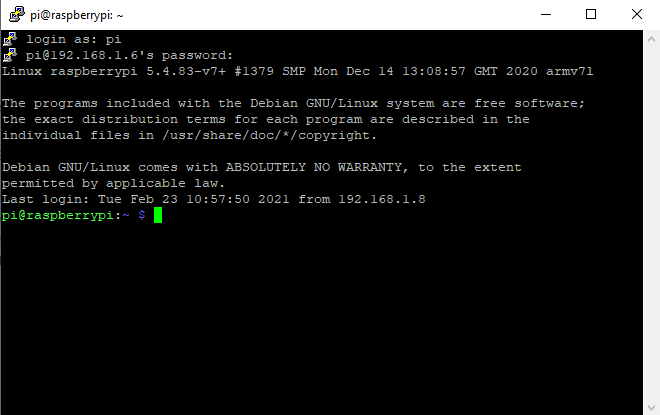
* Cho phép truy cập SSH: tạo một tệp trống trong thư mục boot với tên là ssh (không có phần mở rộng)
* Rút thẻ nhớ vào cắm vào RPI
* Tìm địa chỉ IP của RPI
* Truy cập RPI qua SSH, gợi ý sử dụng phần mềm [Putty](https://www.putty.org/)
* Chạy Putty, gõ địa chỉ IP của RPI vào thanh địa chỉ. Ví dụ:



Nhấn Open, nhập id: pi, password: rapberry



* Đăng nhập thành công: Giao diện hiển thị

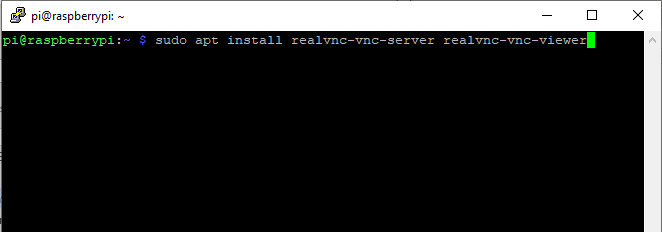


Để thao tác trực quan với giao diện đồ họa của RPI, có thể sử dụng thêm phần mềm VNC.

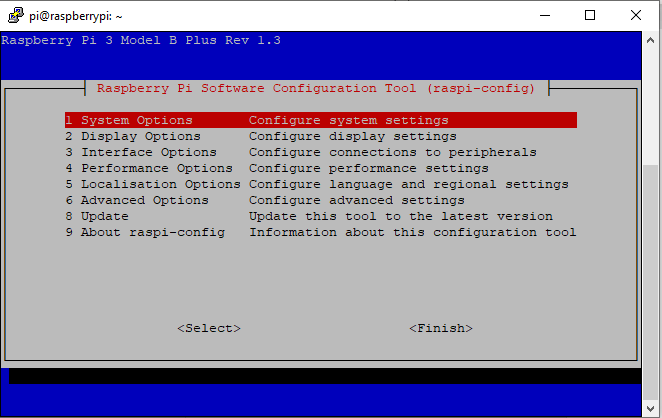
* Gõ câu lệnh cài đặt VNC:

**sudo apt update**

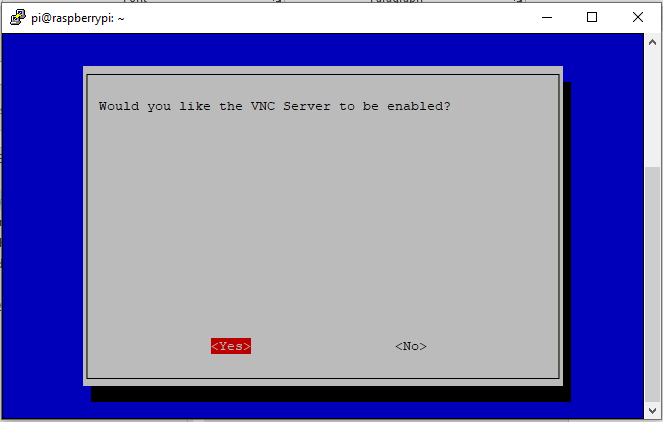
**sudo apt install realvnc-vnc-server realvnc-vnc-viewer**



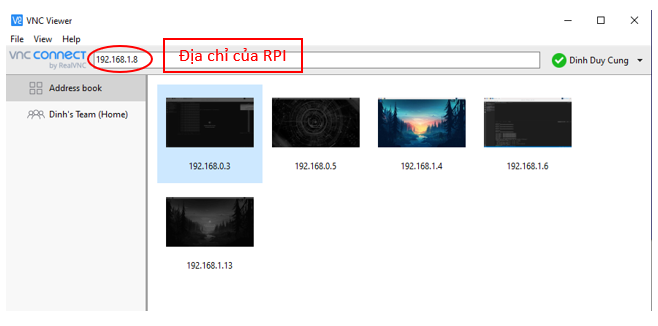
* Gõ câu lệnh sau: **sudo raspi-config** để vào giao diện thiết lập đồ họa cho RPI:



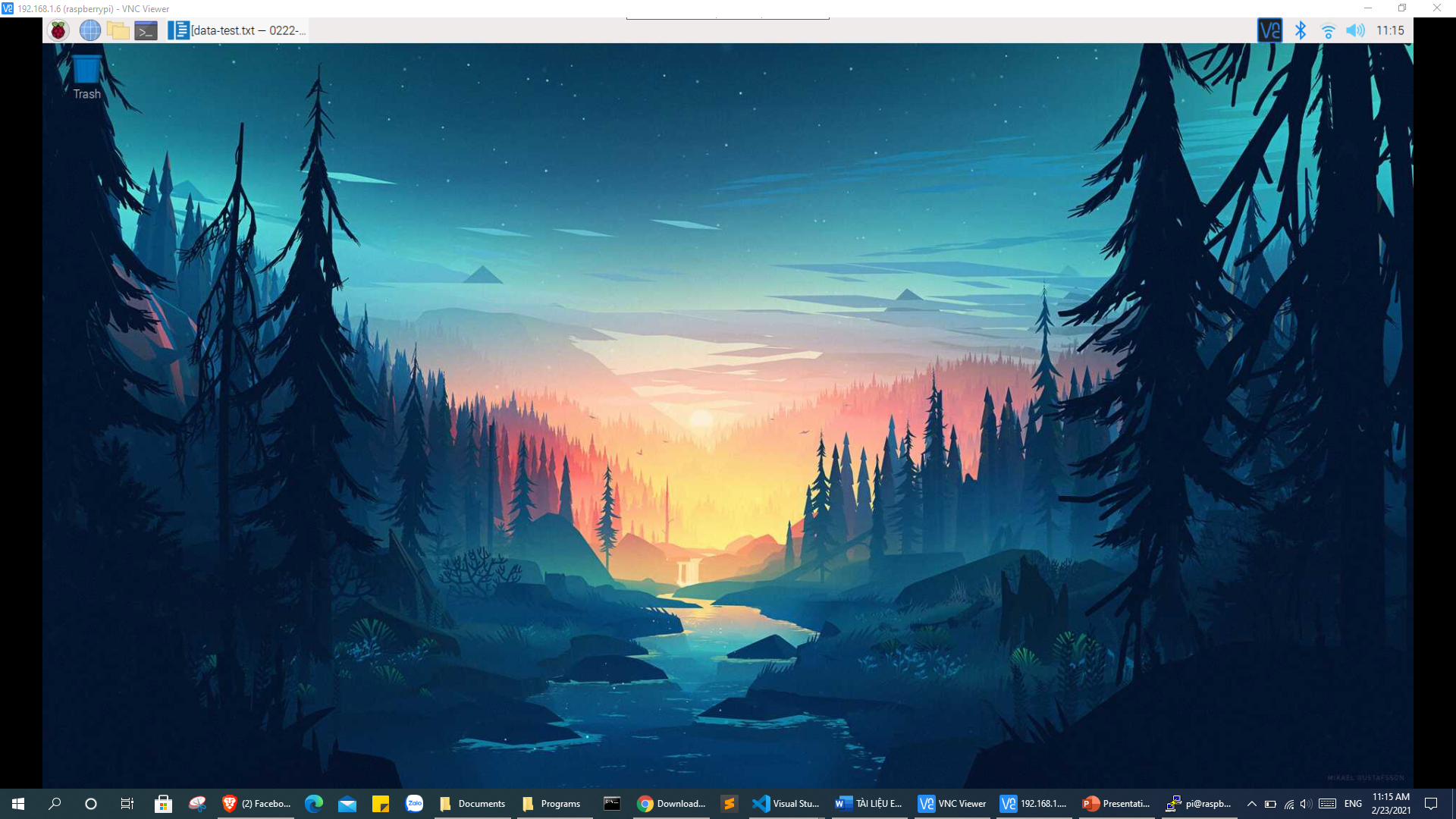
* Chuyển tới mục 3 Interface Options >> P3 VNC >> Chọn Yes



* Cài đặt [VNC Viewer](https://www.realvnc.com/en/connect/download/viewer/) trên máy tính windows.
* Mở VNC và gõ địa chỉ của RPI:



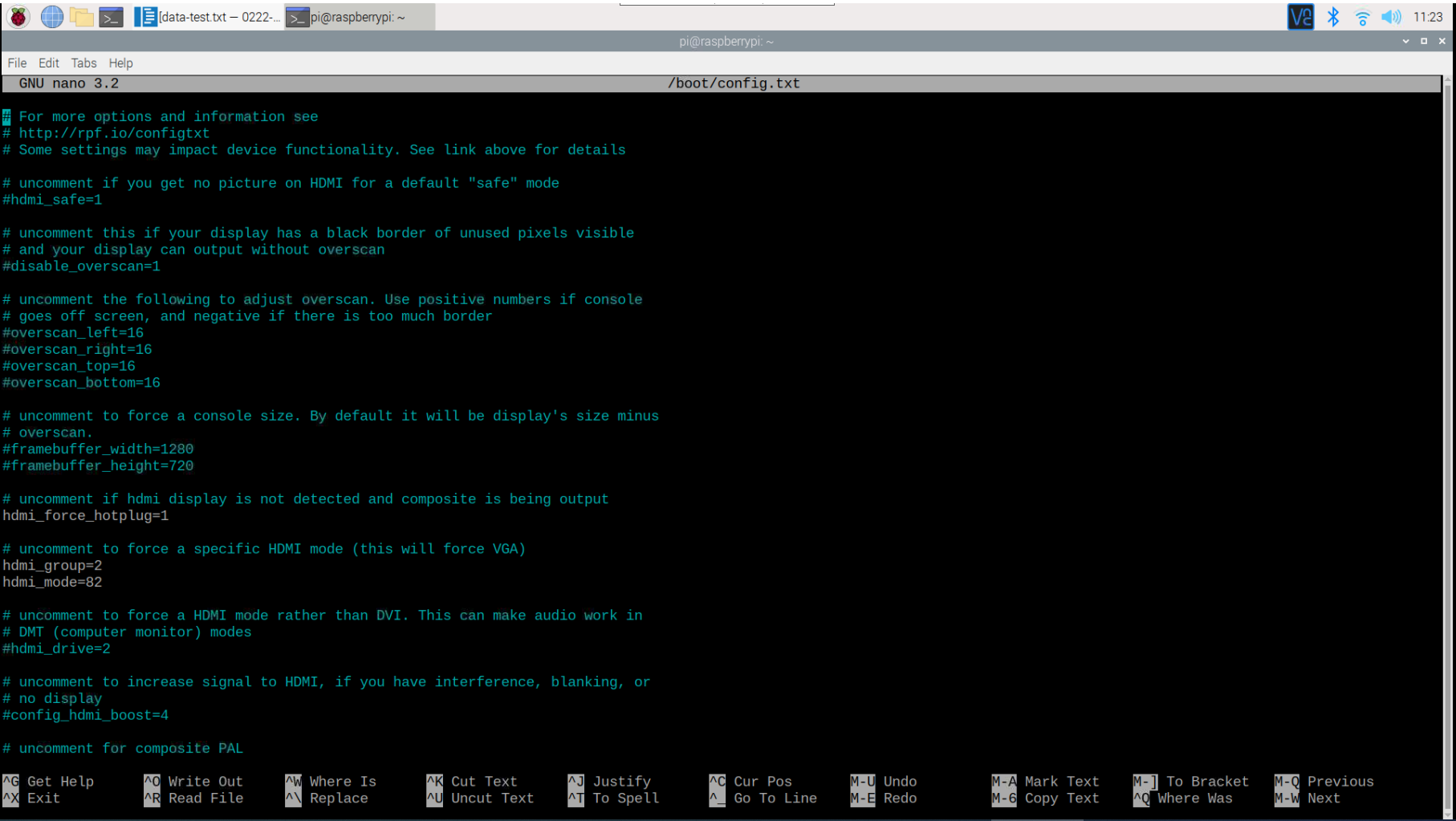
* Truy cập thành công vào RPI:



## Hướng dẫn sử dụng

### Cài đặt RPI khởi động cùng lúc với màn hình

* Mở Terminal (Ctr + Alt + T)
* Gõ câu lệnh sau: **sudo nano /boot/config.txt**

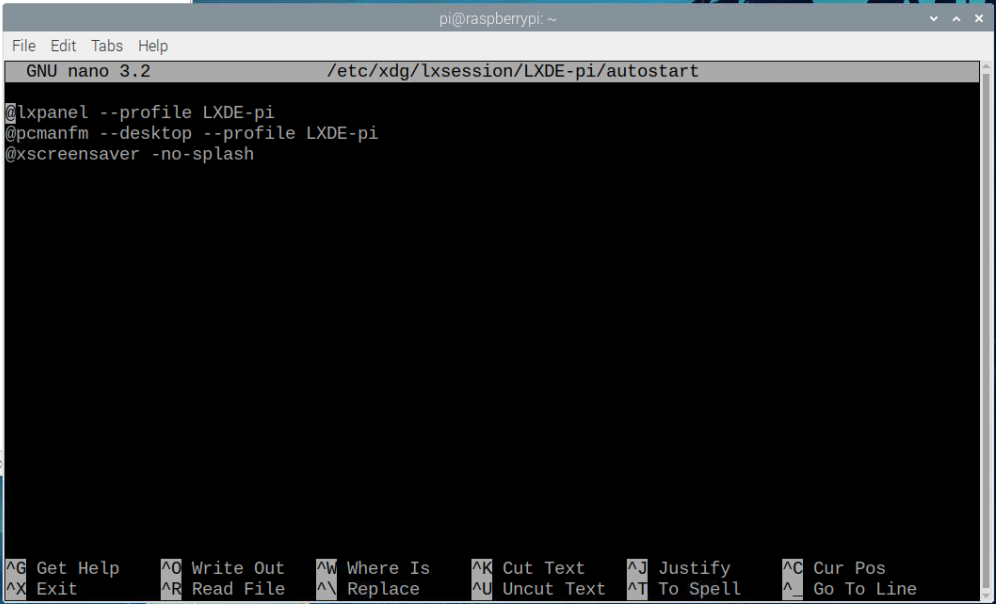


* Chỉnh sửa nội file tại các nội dung sau:
* **disable\_overscan=1**
* **hdmi\_force\_hotplug=1**
* **hdmi\_group=2**
* **hdmi\_mode=82**
* Khởi động lại RPI

### Cài đặt ứng dụng khởi động cùng hệ thống (Ví dụ cho Chromium tự)

* Mở Terminal, gõ câu lệnh:

**sudo nano /etc/xdg/lxsession/LXDE-pi/autostart**



* Thêm vào các câu lệnh sau:

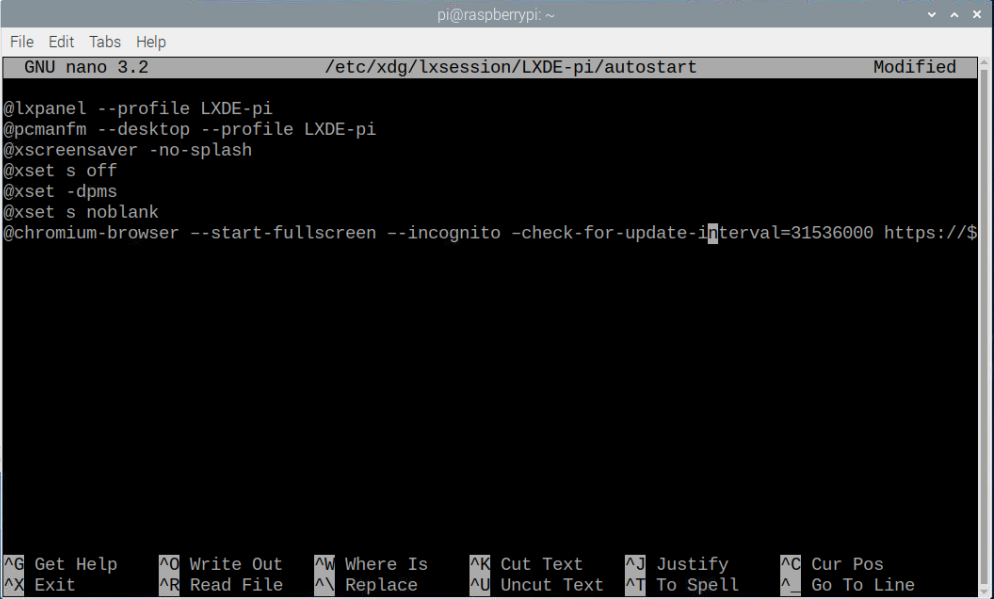
**@xset s off**

**@xset -dpms**

**@xset s noblank**

**@chromium-browser –-start-fullscreen –-incognito –check-for-update-interval=31536000** [**https://www.google.com.vn/?hl=vi**](https://www.google.com.vn/?hl=vi)

(Link tự động mở Google Chrome, với các trang web khác thì thay bằng đường link của trang web đó)

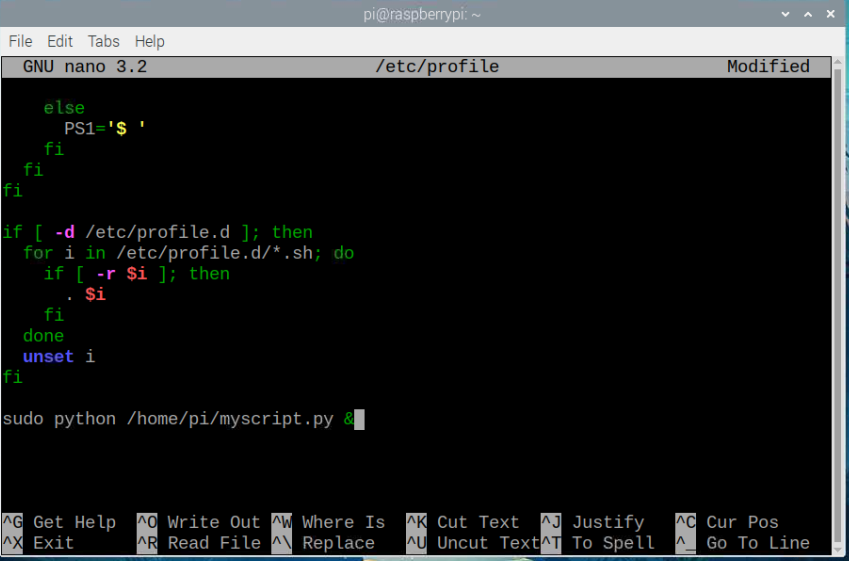


* Nhấn Ctr + X và nhấn Y để lưu lại file.

### Cài đặt cho một file python tự động chạy khi khởi động RPI

* Mở Terminal, gõ câu lệnh sau: sudo nano /etc/profile
* Kéo xuống dưới cùng của tập tin, thêm dòng lệnh sau:

**sudo python /home/pi/myscript.py &**



* Trong đó: **/home/pi/myscript.py** là đường dẫn tới file python cần chạy. Thêm ký tự “**&**” ở cuối để file chạy liên tục (Trong trường hợp file python có vòng lặp vô hạn)

# EDGE computing

## Cài đặt RabbitMQ trên RPI

* Cài đặt Erlang:

**sudo apt-get update**

**sudo apt-get install wget**

**sudo apt-get install libssl-dev**

**sudo apt-get install ncurses-dev**

**wget http://www.erlang.org/download/otp\_src\_23.2.tar.gz**

**tar -xzvf otp\_src\_23.2.tar.gz**

**cd otp\_src\_23.2/**

**./configure**

**make**

**sudo make install**

**cd ..**

**rm otp\_src\_23.2.tar.gz**

**sudo rm -R otp\_src\_23.2/**

* Cài đặt RabbitMQ:

**wget** [**https://github.com/rabbitmq/rabbitmq-server/releases/download/rabbitmq\_v3\_6\_10/rabbitmq-server\_3.6.10-1\_all.deb**](https://github.com/rabbitmq/rabbitmq-server/releases/download/rabbitmq_v3_6_10/rabbitmq-server_3.6.10-1_all.deb)

**dpkg -i rabbitmq-server\_3.6.10-1\_all.deb**

***Lưu ý:***

* *Quá trình cài đặt có thể yêu cầu thêm một số packet Erlang đi kèm, theo dõi phần báo lỗi của màn hình Terminal để cài đặt*
* *Sửa lại file cấu hình trong cài đặt (Nếu báo lỗi không thể truy cập vào máy chủ):*

**sudo nano /etc/rabbitmq/rabbitmq-env.conf**

Thêm các dòng lệnh sau vào file:

**NODENAME=rabbit@localhost**

**NODE\_IP\_ADDRESS=127.0.0.1**

hoặc **NODE\_IP\_ADDRESS=0.0.0.0**

Lưu lại file.

## Demo code

Bài toán: Tạo một EDGE service trên Raspberry Pi. Service có nhiệm vụ:

* Nhận bản tin gửi về từ các device.
* Gửi bản tin về Rabbit cài đặt trên máy chủ
* Trong trường hợp kết nối về máy chủ bị lỗi, lưu trữ lại các dữ liệu dưới dạng file raw. Mỗi phút sẽ check kết nối một lần, nếu kết nối lại thành công thì gửi dữ liệu trong file đi. Mỗi lần gửi dữ liệu trong vòng 1 phút, dữ liệu sau khi gửi thành công sẽ bị xóa.

### Giả lập code cho client

import socket

import time

HOST = "192.168.1.3"

PORT = 1234

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_address = (HOST, PORT)

print("connecting to %s port " + str(server\_address))

s.connect(server\_address)

try:

    while True:

        # msg = input('Client: ')

        while True:

            msg = ":D0004503000000000000000000000000111\r\n"

            s.sendall(bytes(msg, "utf8"))

            time.sleep(1)

            print(msg)

        # if msg == "quit":

        #     break

        # data = s.recv(1024)

        # print('Server: ', data.decode("utf8"))

finally:

    s.close()

### Code service trên RPI

import socket

import os

import pika

import \_thread

import time

from datetime import datetime

import threading

from threading import Timer,Thread,Event

# os.system('fuser -k 1234/tcp')

ServerSideSocket = socket.socket()

host = '192.168.1.3'

port = 1234

ThreadCount = 0

credentials = pika.PlainCredentials('avani', 'avani')

try:

    ServerSideSocket.bind((host, port))

except socket.error as e:

    print(str(e))

print('Socket is listening..')

ServerSideSocket.listen(20)

class perpetualTimer():

   def \_\_init\_\_(self,t,hFunction):

      self.t=t

      self.hFunction = hFunction

      self.thread = Timer(self.t,self.handle\_function)

   def handle\_function(self):

      self.hFunction()

      self.thread = Timer(self.t,self.handle\_function)

      self.thread.start()

   def start(self):

      self.thread.start()

   def cancel(self):

      self.thread.cancel()

# initial rabbit connect

def initial\_rb\_connect():

    try:

        rb\_connnection = pika.BlockingConnection(

            pika.ConnectionParameters(

                "192.168.1.8",

                5672,

                "/",

                credentials,

                heartbeat=0,

                socket\_timeout=1,

            )

        )

        return rb\_connnection

    except:

        return False

def try\_rb\_connect():

    try:

        rb\_connnection = pika.BlockingConnection(

            pika.ConnectionParameters(

                "192.168.1.8",

                # "localhost",

                5672,

                "/",

                credentials,

                heartbeat\_interval=0,

                blocked\_connection\_timeout=1,

            )

        )

        return rb\_connnection

    except:

        return False

def process\_data(tcp\_connnection):

    global channel

    while True:

        data = tcp\_connnection.recv(1024).decode('latin-1')

        err\_data = time.strftime("%H:%M:%S", time.localtime()) + " - " + data

        file = open("data-test.txt", "a")

        if not data:

            break

        print(err\_data)

        rb\_connect = initial\_rb\_connect()

        if rb\_connect == False:

            print("Connect fail !")

            file.write(err\_data)

        else:

            print("Connect OK, send : ", data)

            channel = rb\_connect.channel()

            channel.queue\_declare(queue="data")

            channel.basic\_publish(exchange="", routing\_key="data", body=data)

def scan():

    # time.sleep(10)

    # global channel

    # global connection

    try:

        connection = pika.BlockingConnection(

            pika.ConnectionParameters(

                "192.168.1.3",

                5672,

                "/",

                credentials,

                heartbeat=0,

                # socket\_timeout=1,

            ))

        channel = connection.channel()

        channel.queue\_declare(queue='err-data')

        a\_file = open('data-test.txt', 'r')

        Lines = a\_file.readlines()

        a\_file.close()

        count = 0

        try:

            new\_file = open("data-test.txt", "w")

            for line in Lines:

                if line != "\n":

                    count += 1

                    channel.basic\_publish(exchange='', routing\_key='err-data', body=line)

                    # print("Line{}: {}".format(count, line.strip()))

                    print("Line{}: {}".format(count, line))

                    # del Lines[1]

        except:

            connection.close()

    except:

        pass

    # time.sleep(delay)

def Main():

    try:

        while True:

            Client, address = ServerSideSocket.accept()

            \_thread.start\_new\_thread(process\_data, (Client, ))

            # time.sleep(1)

            # # print(i)

            # \_thread.start\_new\_thread(scan,())

            t = perpetualTimer(60,scan)

            t.start()

    except:

        print("Lỗi hàm main")

        ServerSideSocket.close()

if \_\_name\_\_ == '\_\_main\_\_':

    Main()

Source Code: