Important Links

https://www.hackster.io/

Watch When have time:

ttps://www.youtube.com/watch?v=z55nt4qJvdI --> outubro 2017
https://www.youtube.com/watch?v=MNypwrbHMgU --> maio 2018 - placas-montagem - configuracao
https://www.youtube.com/watch?v=eZ8RlTEIkoE --> junho 2018 - continuacao - montagem - varios video relacionados ao lado
https://www.youtube.com/watch?v=NsLmnxv4ii8 --> maio 2020 - montagem - placa
https://www.youtube.com/watch?v=a7bkImtkgUo --> maio 2020 continuacao
https://www.youtube.com/watch?v=moCOf6vRQyA --> INSTALANDO APLICAÇÃO MAGIC MIRROR
https://www.youtube.com/watch?v=q9OsWBEHDO4 --> abril 2020 - Chromecast - 6
https://www.youtube.com/watch?v=D6LRa0M4LBI --> BUILD YOUR OWN GOOGLE HOME 2.0!
https://www.youtube.com/watch?v=Yfc0dwS402A --> maio 2020 - automacao residencial
https://www.youtube.com/watch?v=gbnpmLS0DvQ --> RaspberryPi

Project Ideas

Umidifier

PhotoResistor when it gets dark it turns on the light in your home

Facial Recognition + Door Opening + Accelerometer (If someone tries to steal, an alarm rings)

Voice Recognition + Call Number + Control Lights + Control Blinds

Robot that follows you

Robot that helps you with luggage

Robot that helps you with garbage

Cleaning Robot that detects shit in the floor and avoids it as well as make sounds to avoid animals coming close

Make a glass that can be transparent and opaque when you want (12:40 building automation event video)

Cooker Robot

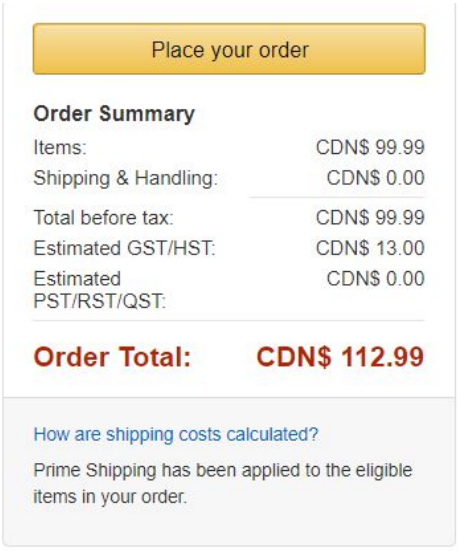
Infrared Camera

Measure the Efficiency of a Compressor

Arduino and Raspberry PI communication
https://www.raspberrypi.org/forums/viewtopic.php?p=1007463#p1007463

Cost

Raspberry Pi 3 Kit



Links to buy Raspberry Pi

https://www.amazon.ca/Freenove-Raspberry-Processing-Tutorials-Components/dp/B06W54L7B5/ref=sr_1_8?crid=G06RSDBGX9AY&keywords=raspberry+pi&qid=1590850084&sprefix=rasp%2Caps%2C200&sr=8-8

https://www.buyapi.ca/product/raspberry-pi-4b-budget-bundle-1gb/

https://www.canakit.com/raspberry-pi-kit-for-dummies.html

https://www.amazon.ca/CanaKit-Raspberry-Starter-Premium-Black/dp/B07BCC8PK7/ref=sr_1_3?dchild=1&keywords=raspberry+pi+3&qid=1590852890&sr=8-3

Raspberry Berry PI Cheat Sheet

-Ciro’s Folder:
file:///D:/CIRO%20(Estudos%20Compactados)/CENTENNIAL/CENTENNIAL%20(ENERGY%20ENGINEERING)/Mat%C3%A9rias/Python/Raspberry/Raspberry%20Terminal%20Cheat%20Sheet.pdf

Raspberry Pi vs Arduino

Raspberry Pi vs Arduino

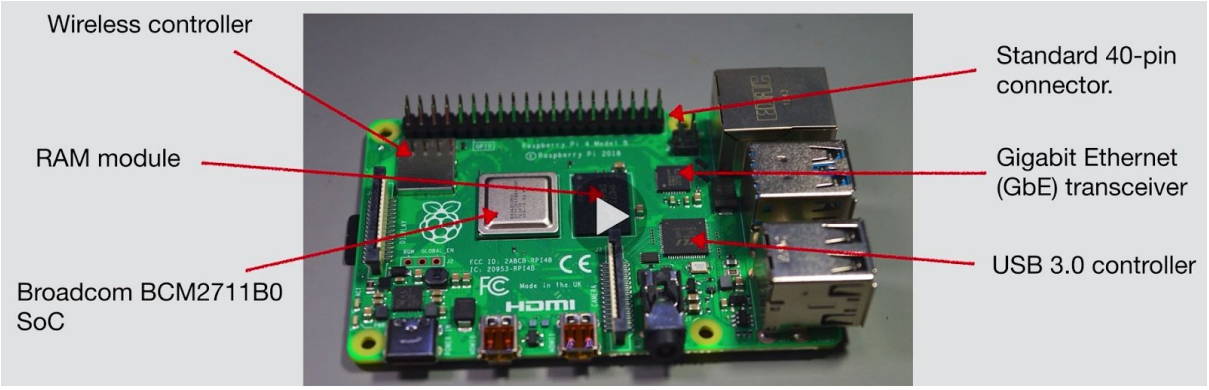
Raspberry Pi	Arduino
Microcomputer	Microcontroller
Needs an operating system Linux	Does not need an operating system
Complicated	Simple
Video out, Camera, Ethernet ports, Wifi, Bluetooth, USB, I2C, SPI, UART etc. on board	USB only for power and serial in/out, I2C, SPI, UART
Best for general computer	Best for small tasks that constantly repeat
Capable of performing a huge range of tasks	Optimised for sensing and controlling the world around it
Best for more advanced makers	Best for beginners
Programmed in many languages, including C/C++, Python, Ruby	Programmed in C/C++
Relatively high power consumption	Relatively low power consumption

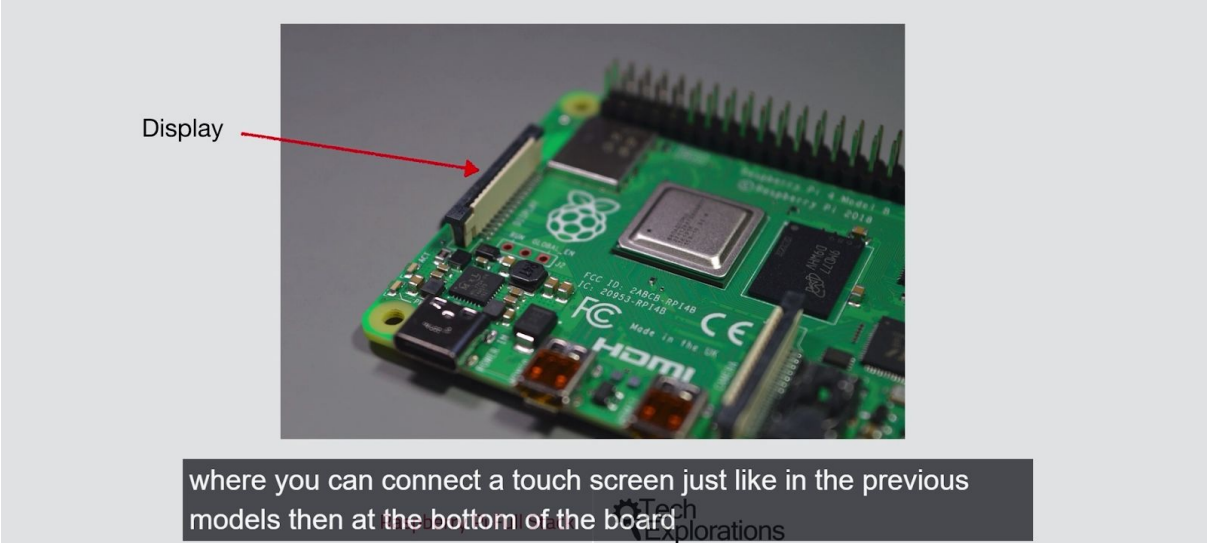
Hardware (Pi3 vs Pi4)

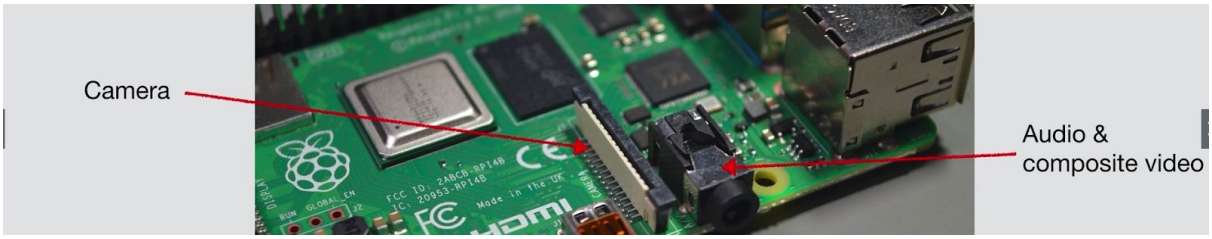
Raspberry Pi 3 vs Raspberry Pi 4

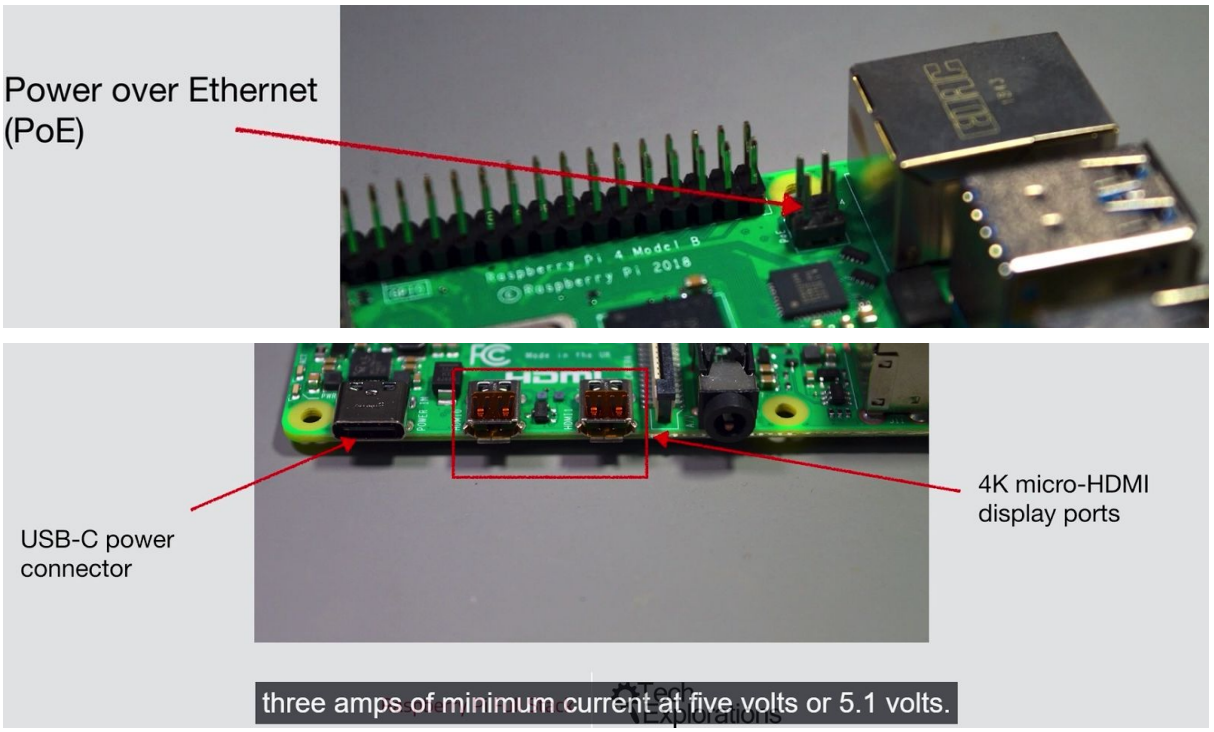
Raspberry Pi 3	Raspberry Pi 4
1.2 GHz CPU	1.5 GHz CPU
1 GB RAM	1GB, 2GB, 4GB RAM
100 Base Ethernet	Gigabit Ethernet
4 USB 2.0 Ports	2 USB 2.0 Ports 2 USB 3.0 Ports
1 Full size HDMI	2 Micro HDMI with 4K support
Micro USB Power Support	USB Type -C Power Support

Raspberry PI Hardware

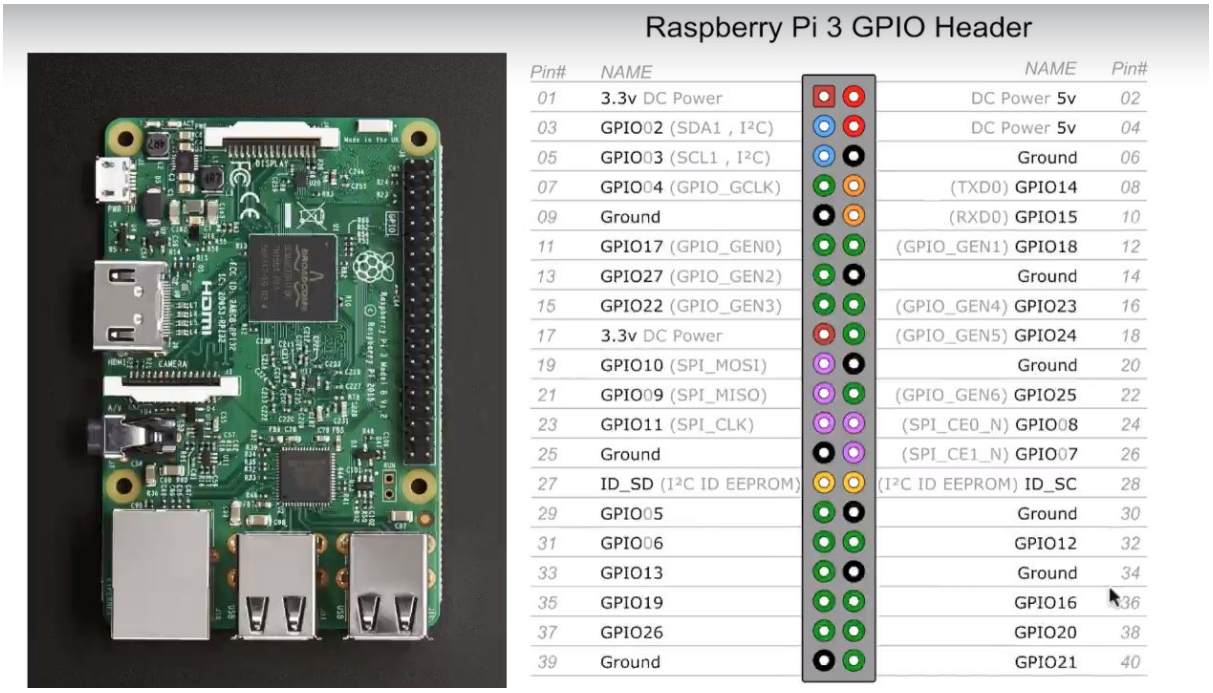








Raspberry PI pins



Acronyms for the Pins

No analog Pins, so in order to process analog readings, an external analog to digital converter must be used.

GPIO=> General Purpose Input Output Pins (operates with 3.3V)

SDA=> Serial Sata

SCL=> Serial Clock

MOSI=> Master of Slave In

MISO=> Master of Slave Out

Reducing Damage to the Board

Tips for reducing the risk of damage to your Raspberry Pi

- Ensure your Pi is powered off when connecting circuitry
- Do not put more than 3.3 V on any GPIO pin
- Do not draw more than 3 mA per output.
- Do not poke at the GPIO connector with a screwdriver or metal object when the Pi is powered on
- Do not power the Pi with more than 5V
- Do not draw more than a total of 50 mA from the 3.3 V supply pins
- Do not draw more than a total of 250 mA from the 5V supply pins

Connecting Raspberry Pi to the PC

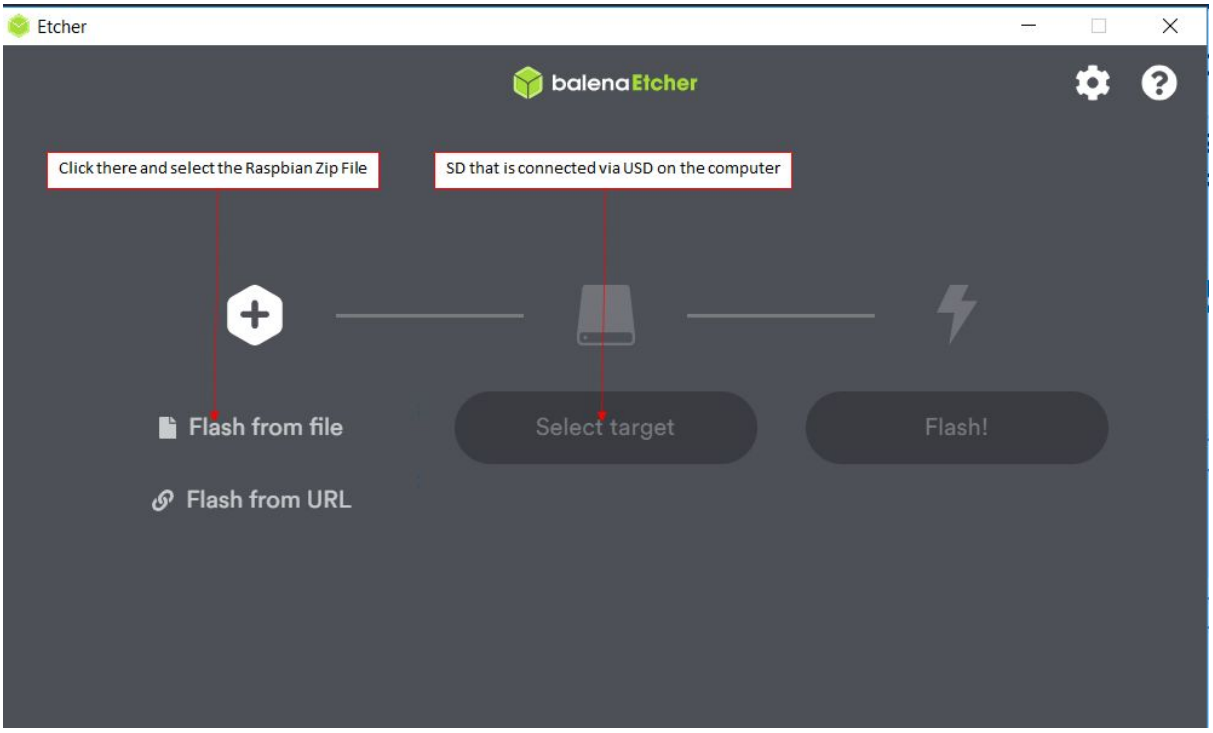
https://github.com/leeassam/pi-ultimate-guide/blob/master/setup_headless_mode/instructions.md

Headless Setup Process

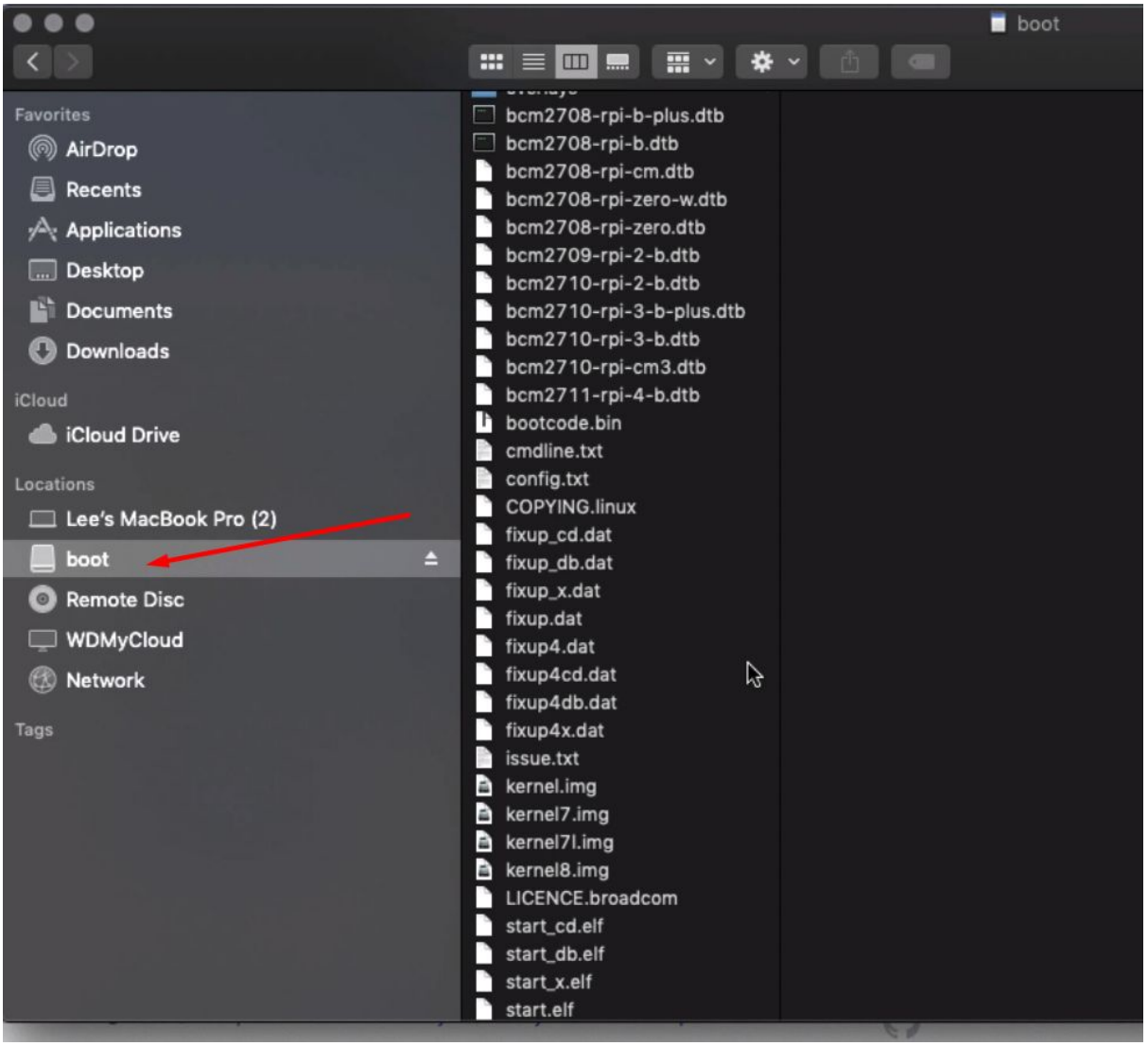
- Flash SD card with latest Raspbian
- Create two files (ssh and wpa_supplicant.conf)
- Get IP address of Pi
- Connect via SSH and enable VNC Viewer, Boot to Desktop and Set Screen Resolution in raspi-config settings
- Use VNC Viewer to connect to your Pi

https://github.com/leeassam/pi-ultimate-guide/blob/master/setup_headless_mode/intstructions.md

1) Download the latest Raspbian version https://www.raspberrypi.org/downloads/raspbian/
2) Download Etcher https://www.balena.io/etcher/
3) Open Etcher



4) After the Flashing process is over, the SD card is going to be removed from the computer, so just take out ther USB reader and insert it again, so the computer will recognized it back.
5) The name of the location when you put the SD card back is probably going to be Boot.

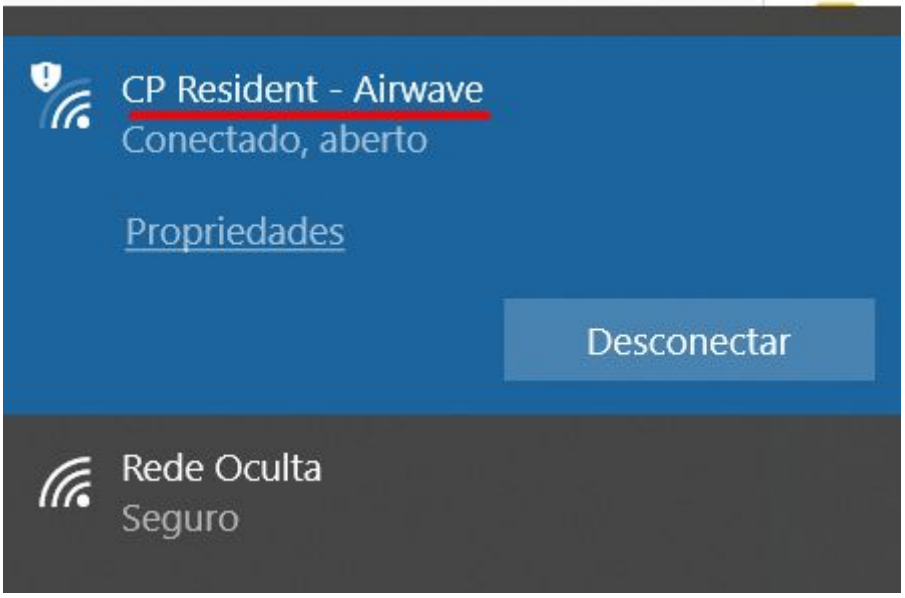


6) Create a file WITHOUT EXTENSION named: ssh
7) Create another file named: wpa_suplicant.conf and paste the below text there:

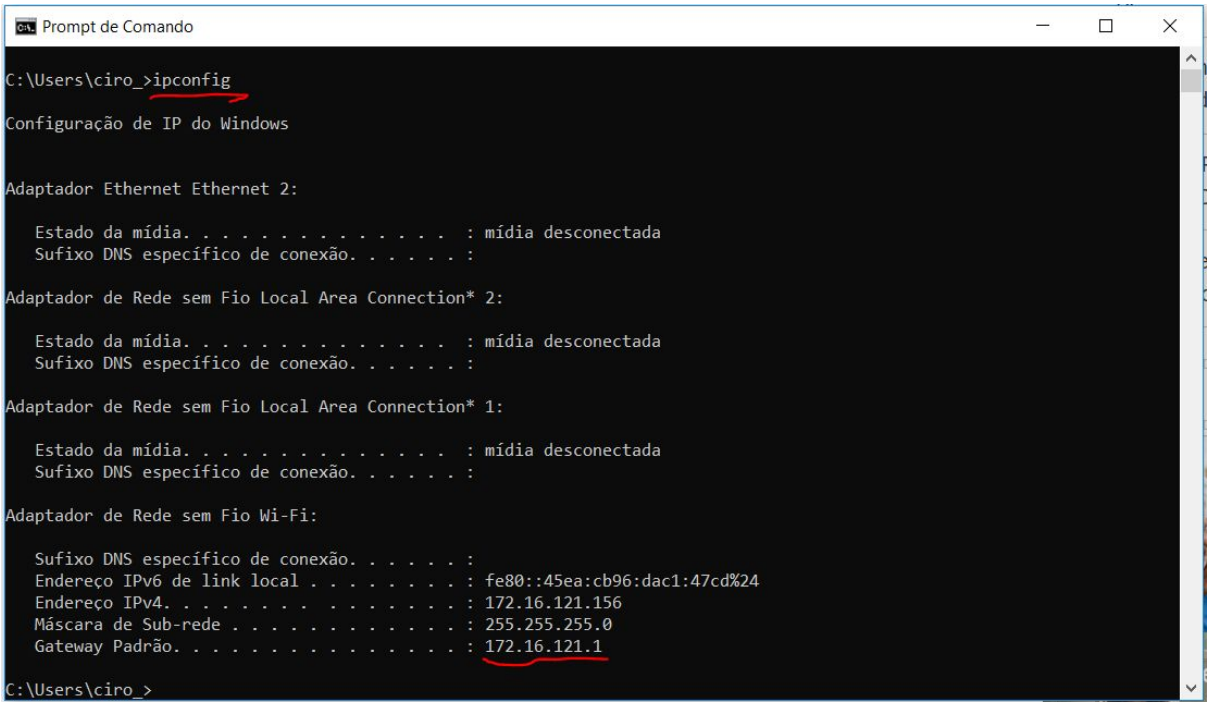
```
ctrl_interface=DIR=/var/run/wpa_supplicant
GROUP=netdev
update_config=1

country=CA

network={

   ssid="CP Resident - Airwave"

   psk=""

}
```

8) In the country part on the code above, change it for the country code, in this case CA for Canada. If you don't know just search: https://www.nationsonline.org/oneworld/canada.htm

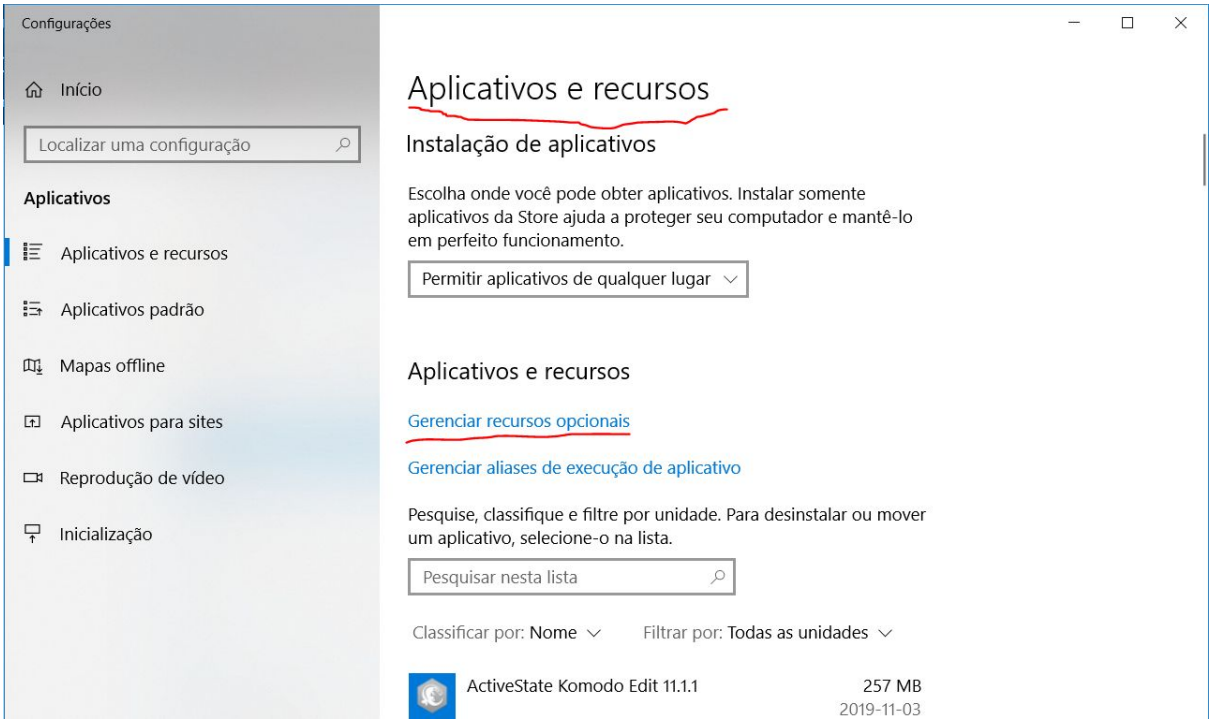
9) In the SSID part on the code above, enter the name of your network and in the PSK, enter the password.

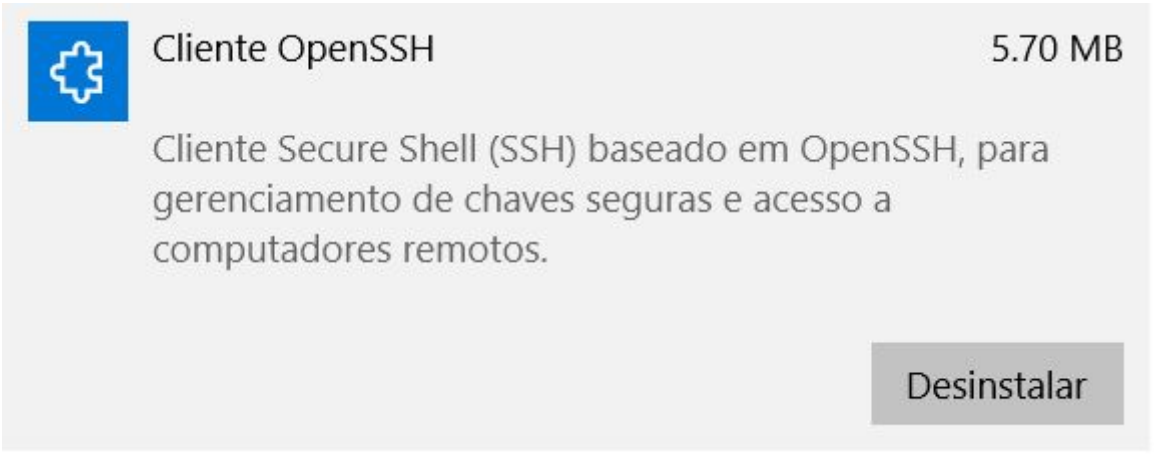


10) Remove the SD card and put in the Raspberry Pi
11) Get the IP address from your computer:
https://www.howtogeek.com/233952/how-to-find-your-routers-ip-address-on-any-computer-smartphone-or-tablet/

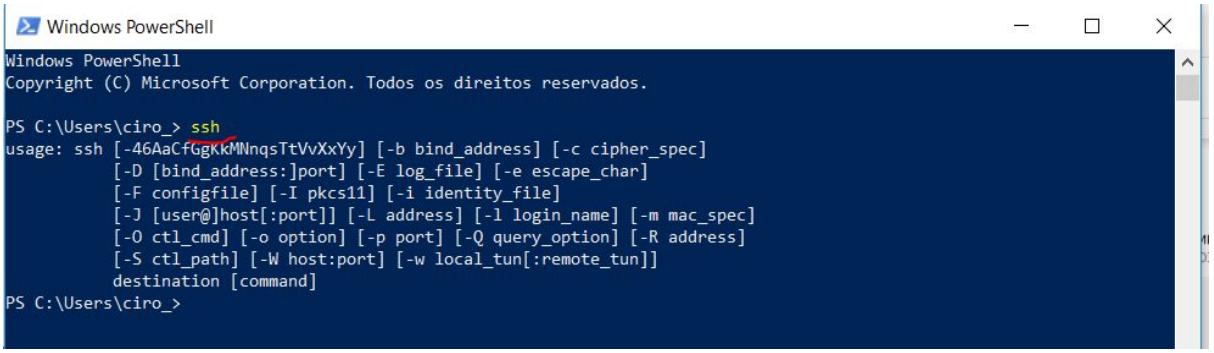


12) Download this software to get the IP address of your Raspberry Pi:
https://www.advanced-ip-scanner.com/
13) Enable SSH:
https://www.howtogeek.com/336775/how-to-enable-and-use-windows-10s-built-in-ssh-commands/
14) Go to Apps=> Manage Optional Feature=> Install OpenSSH Client

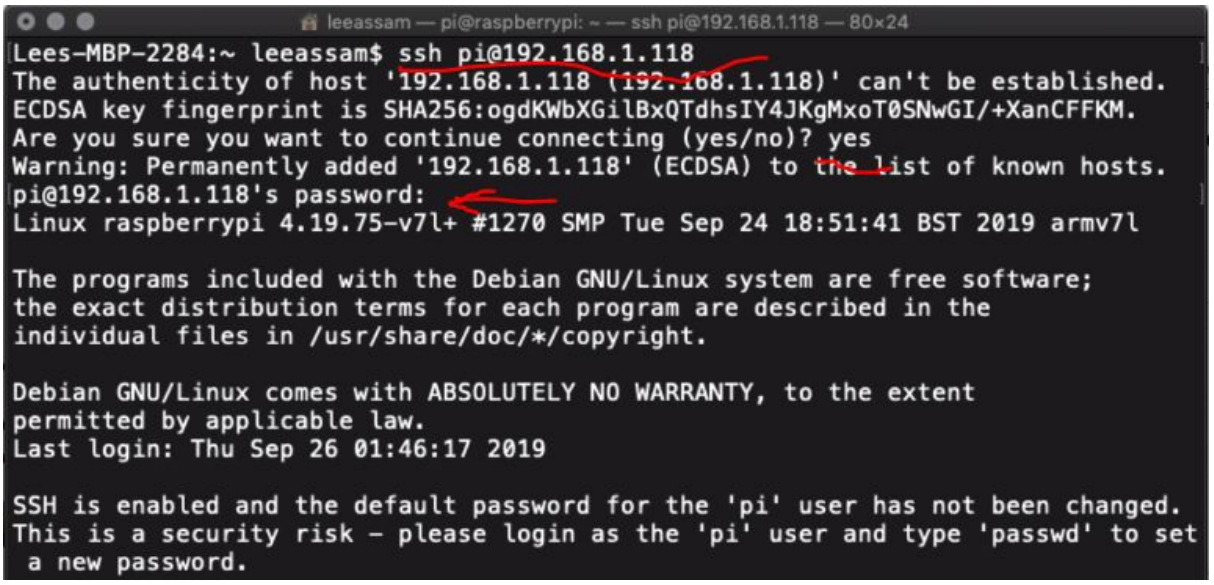




15) Got to the Terminal and type ssh:



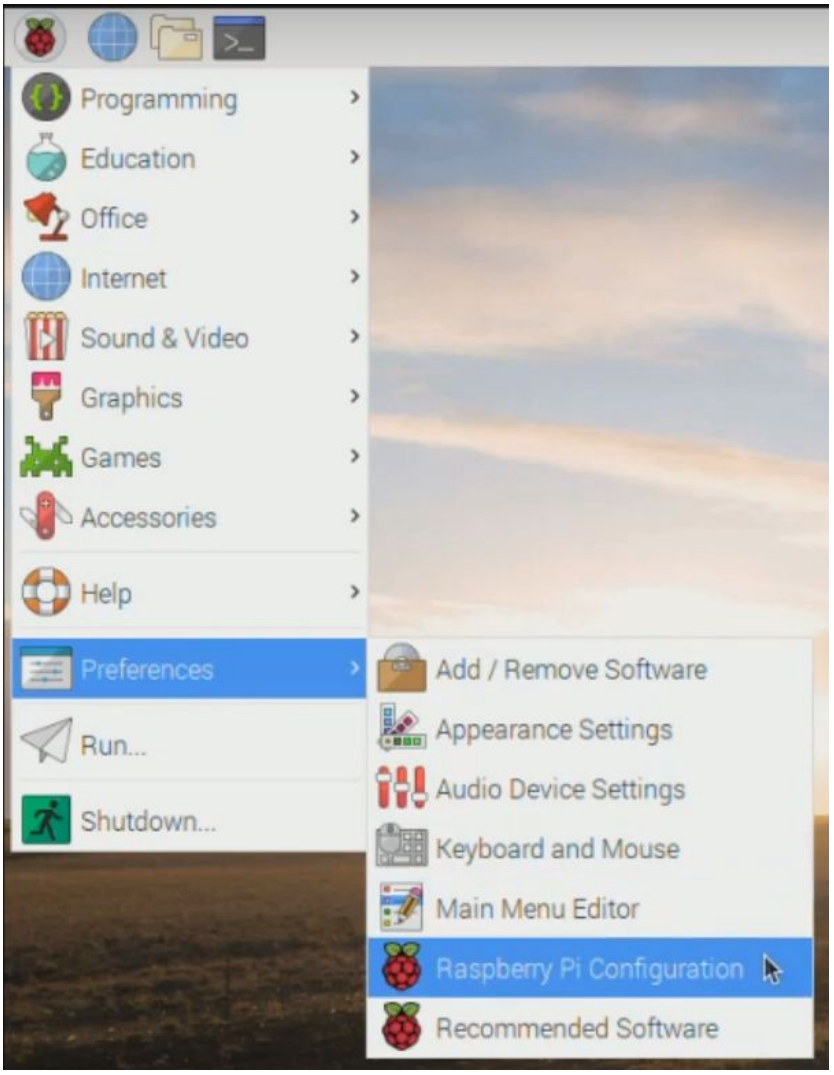
16) In the Terminal type ssh pi@IP address of your Raspberry Pi
17) Type yes
18) Put the password: RASPBERRY



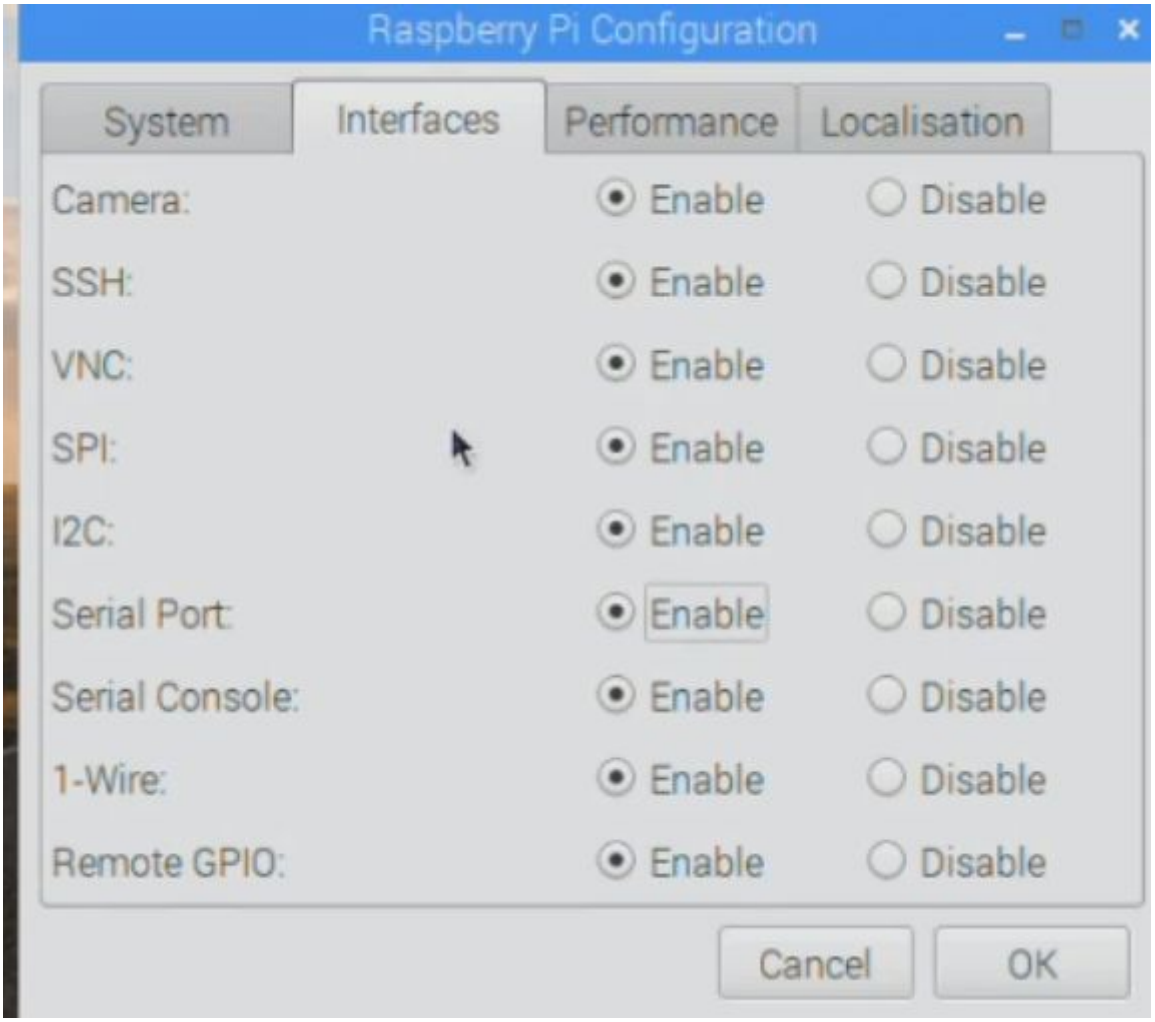
19) After this you are connected with the Raspberry
20) To configure the raspberry py type in the Terminal: sudo raspi-config
21) Use the keyboard arrows to go to Interfacing Options=>VNC=> Yes
22) Press Enter
23) Go to Boot Options=> Desktop/CLI=> Desktop Autologin
24) Advanced Options=> Resolution=> 1920x1080=> OK
25) Finish
26) Yes
27) Download the VNC
    https://www.realvnc.com/en/connect/download/viewer/
28) Enter the Raspberry Pi IP Address

29) Continue
30) Log in information:
    Username: pi
    Password: raspberry
31) Remember the Password
32) After pressing next and setting the language,you can change the default password:

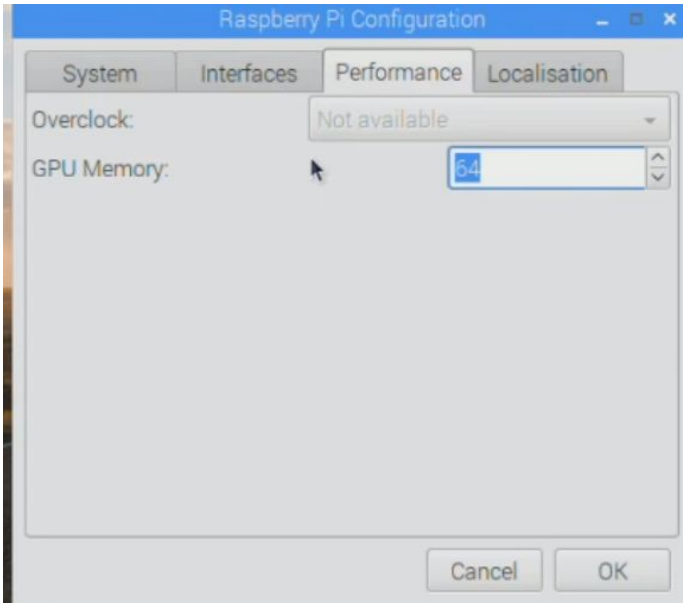
33) Select the Wifi network and put the password
34) Click NEXT
35) REBOOT
36) Click at the top in the Raspberry Pi=> Preferences=> Raspberry PI Configuration



37) Go to Interfaces and Enable everything



38) If you want to OVERCLOCK (AVOID DOING THAT), go to performance:

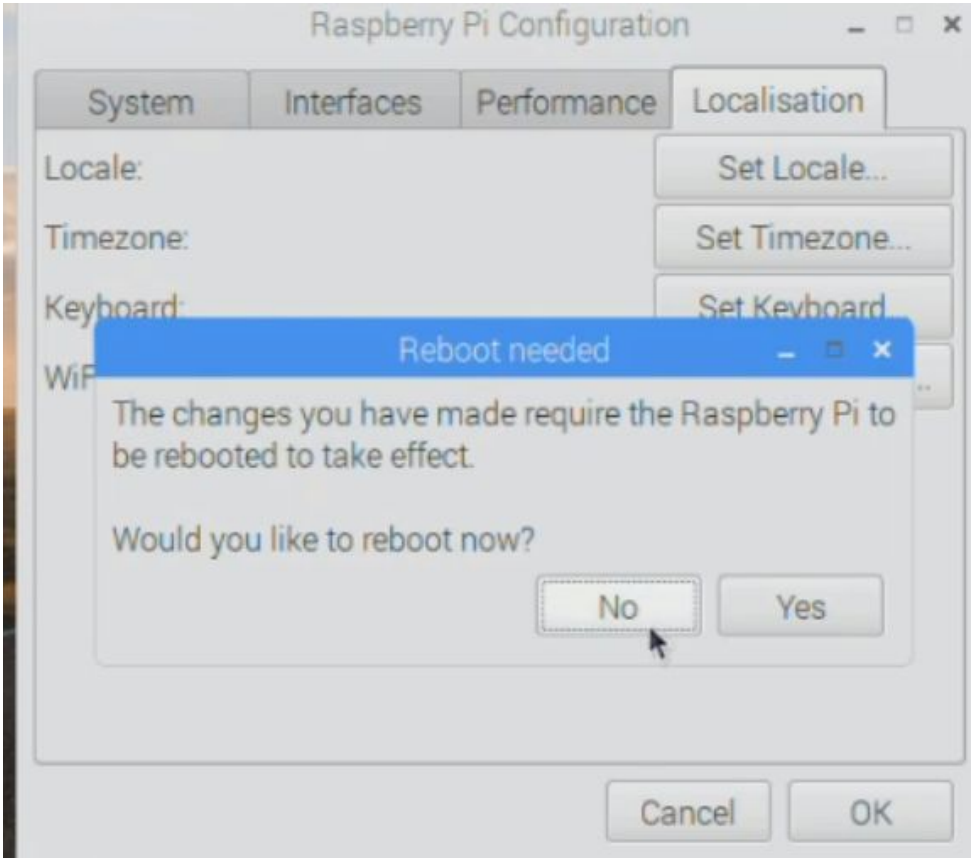


39) Click OK

40) Click Yes (reboot), so the changes will take effect.

You can also go at the top in the Raspberry Pi=> Shutdown=> Reboot

Or type in the terminal: sudo reboot now


Raspberry Pi Configuration
System
Interfaces
Performance
Localisation
Locale:
Set Locale...
Timezone:
Set Timezone...
Keyboard:
Set Keyboard...
Reboot needed
The changes you have made require the Raspberry Pi to be rebooted to take effect.
Would you like to reboot now?
No
Yes
Cancel
OK

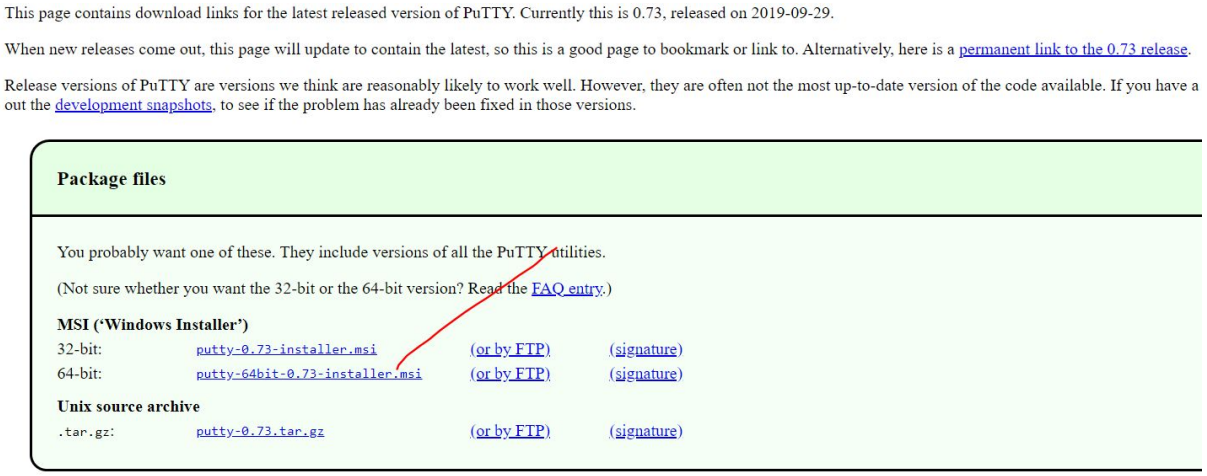
41) s

Connecting Remotely with Raspberry Pi

SSH

For Windows

1) Go to the Putty website: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
2) Select this one to download:



This page contains download links for the latest released version of PuTTY. Currently this is 0.73, released on 2019-09-29.

When new releases come out, this page will update to contain the latest, so this is a good page to bookmark or link to. Alternatively, here is a permanent link to the 0.73 release.

Release versions of PuTTY are versions we think are reasonably likely to work well. However, they are often not the most up-to-date version of the code available. If you have a out the development snapshots, to see if the problem has already been fixed in those versions.

Package files

You probably want one of these. They include versions of all the PuTTY utilities.

(Not sure whether you want the 32-bit or the 64-bit version? Read the FAQ entry.)

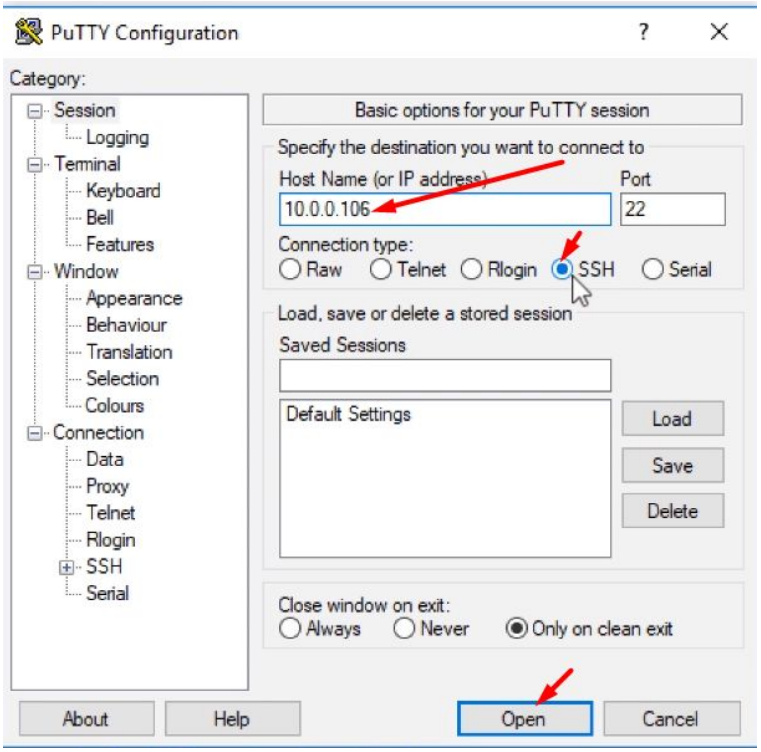
MSI ('Windows Installer')

32-bit:	putty-0.73-installer.msi	(or by FTP)	(signature)
64-bit:	putty-64bit-0.73-installer.msi	(or by FTP)	(signature)

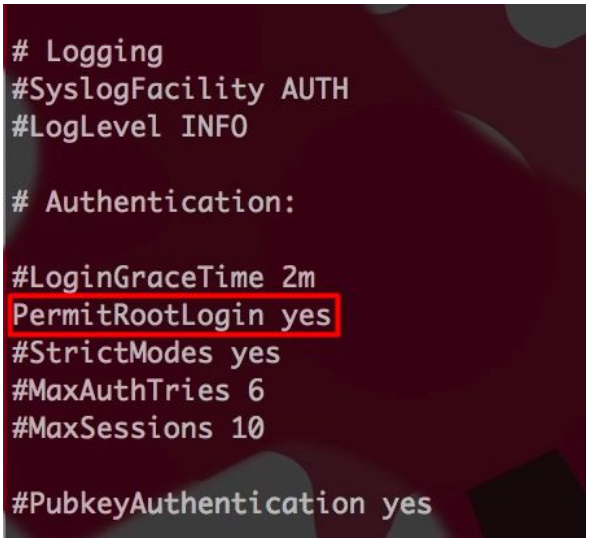
Unix source archive

.tar.gz:	putty-0.73.tar.gz	(or by FTP)	(signature)

3) Install it
4) Open the software
5) Type your Raspberry PI IP address
6) Make sure the SSH is selected
7) Press Open



8) Select Yes
9) Put your username (pi) and password (raspberry)
10) Go to the Terminal: nano /etc/ssh/sshd_config
11) Scroll down and Change the Parameter below:
    #PermitRootLogin without-password => PermitRootLogin yes

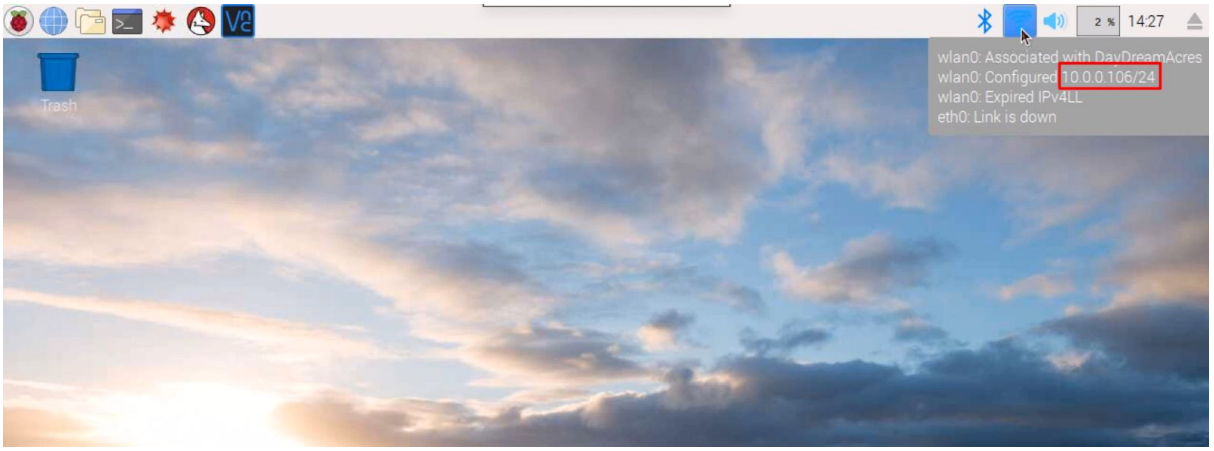


12) Exit the file: CTRL+X
13) Restart the service, so you can log in the Raspberry Pi as root: /etc/init.d/ssh restart
14) Change the password of the root user to raspberry: passwd raspberry
15) To check all the files=> Type: ls
16) To check folder location=> Type: pwd

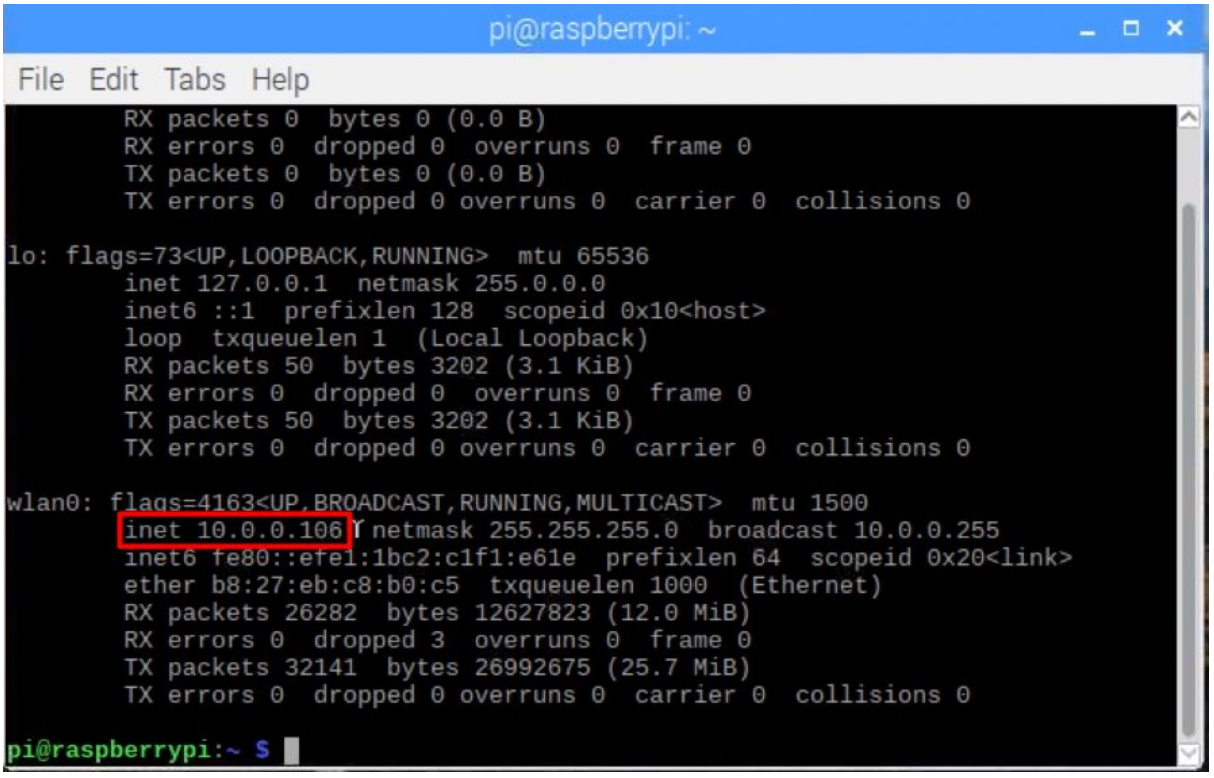
For MAC

1) Check the IP address assigning to the Raspberry Pi:

a) Hover your mouse at the top:



b) Go to the Terminal and type: ifconfig



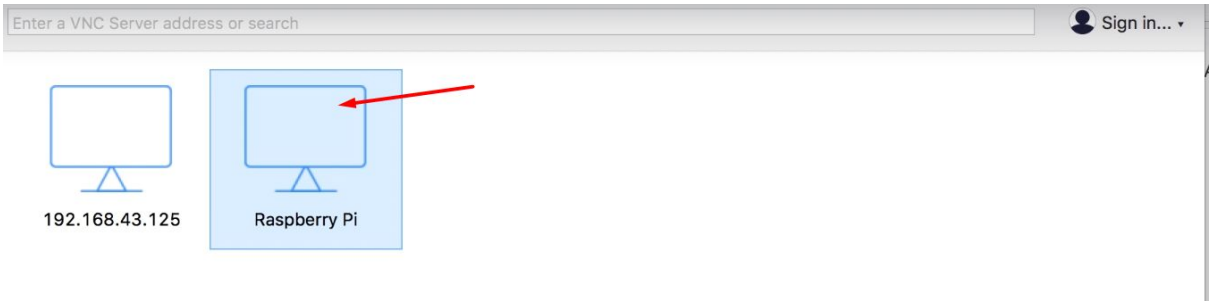
2) Raspberry PI must be connected in the same network with the remote machine
3) Go to the Terminal and type: ssh@IPADRESS
   The IP address is the one got from before WITHOUT THE DIGITS AFTER THE BAR.
   In this case: it would be ssh@10.0.0.106
4) Type yes
5) Put your new password (the reset one or the old one if you didn't reset)
6) To check all the files=> Type: ls
7) To check folder location=> Type: pwd

VNC

1) Download the VNC:

https://www.realvnc.com/en/connect/download/viewer/

2) Install
3) Open the software
4) Go to File=> New
5) The computer MUST be in the same network as your Raspberry
6) VNC Server=> Put the IP address of the Raspberry PI
7) Name=> Up to you, but choose an easy name
8) Click OK
9) Double click in the recent created VNC:



10) Put the login and remember password

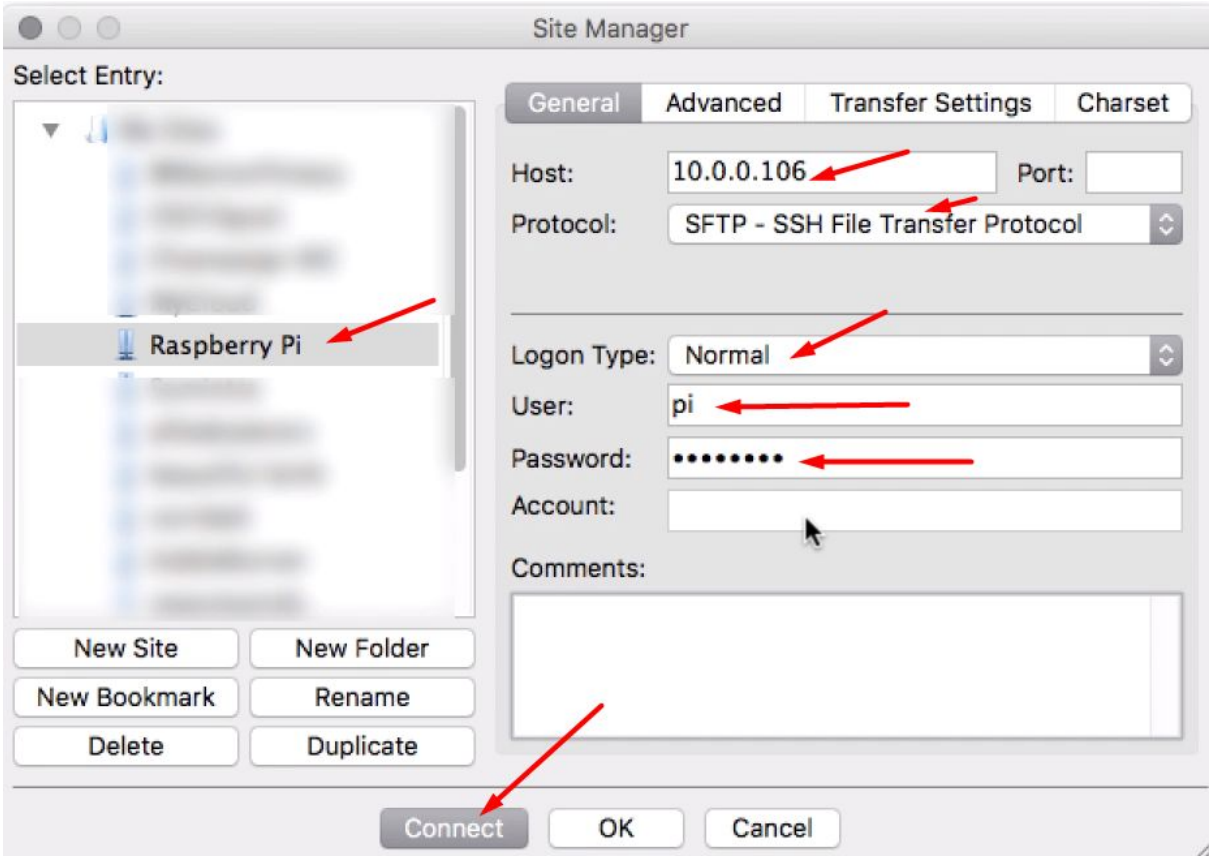
SFTP (file transfer)

Used to transfer files between different computers or the Raspberry Pi to the Computer

1) Go to Filezilla:

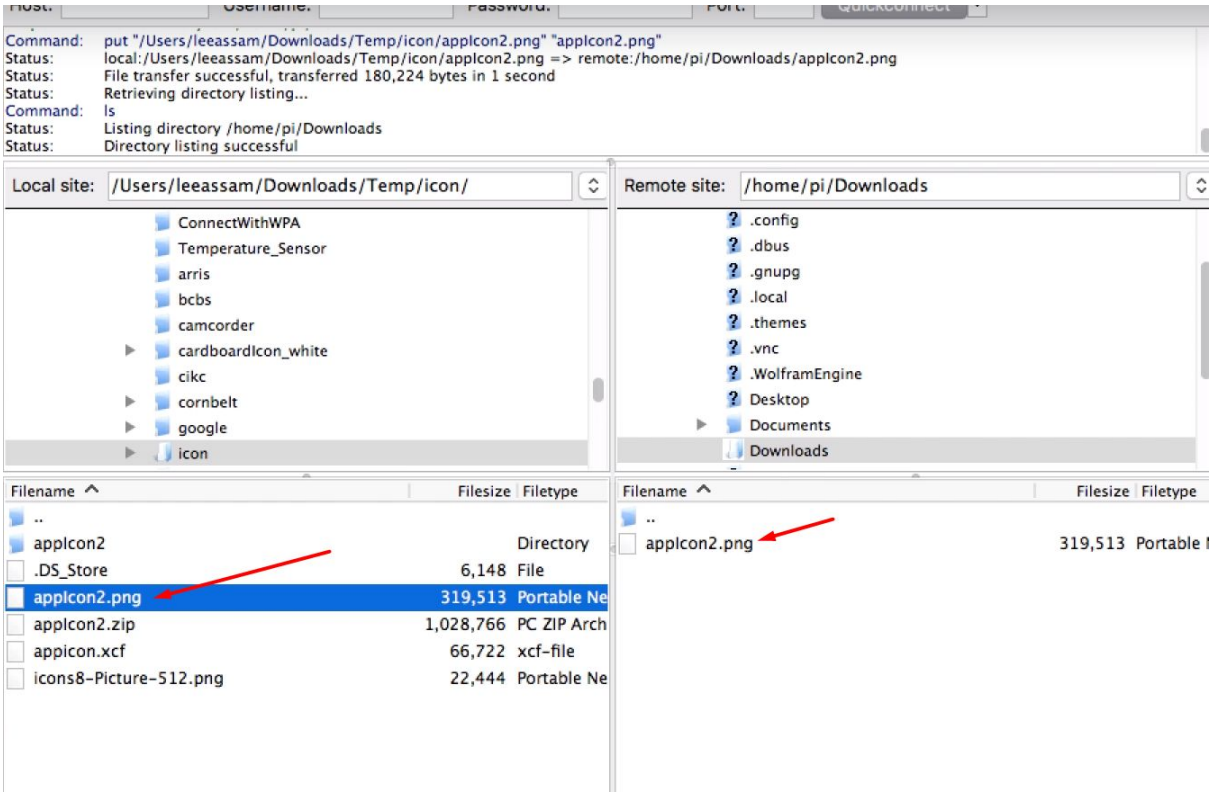
https://filezilla-project.org/

2) File=> Site Manager=> New Site=> Rename to Raspberry PI
3) Host=> Raspberry PI IP Address
4) Protocol=> SFTP
5) Logon Type=> Normal
6) User=> pi
7) Password=> password set for the account
8) Connect



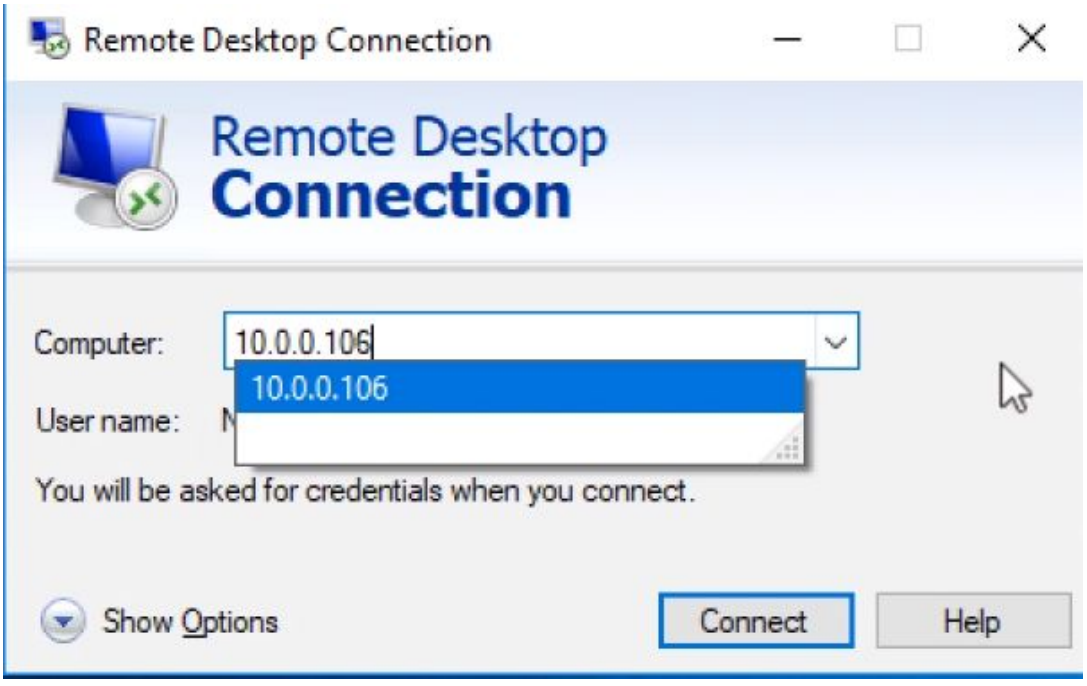
9) Check the box and press okay

10) Now you can double click in one file from the LEFT SIDE and it will go to the right side. So, files are being transferred from the Computer to the Raspberry Pi

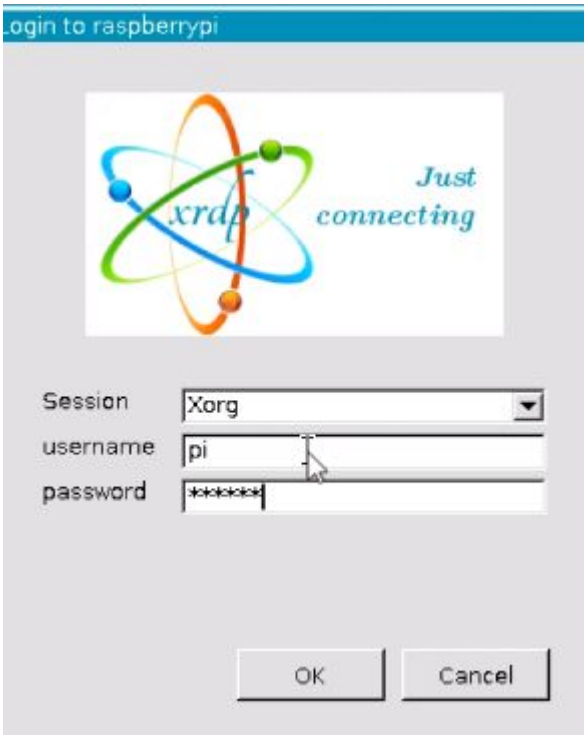


RDP (Remote Desktop Connection)

1) Search in the Windows Remote Desktop Connection
2) The machine must be connected in the same network as the raspberry PI
3) Use the Raspberry PI IP address



4) Connect and Yes
5) Put the login info



SSDs and Raspberry Pi

Choosing a SSD Card

Avoid SSD card problems by:

1) Using genuine SSD cards
2) Using a good quality power supply
3) Using a good quality USB cable
4) BEFORE SHUTTING DOWN YOUR RASPBERRY PI=> TYPE **sudo halt**
5) Overclocking is BAD

https://elinux.org/RPi_SD_cards

SSD Card Physical Size

Model		Size
Raspberry Pi Model A Raspberry Pi Model B		Full-Size SD Card 
Raspberry Pi Model A+ Raspberry Pi Model B+ Raspberry Pi 2 Model B Raspberry Pi Zero Raspberry Pi 3 Model B		Micro SD Card 

SSD Card Sizes

Recommendation: 16 GB

OS SD card size requirements

OS Distribution	Size
 NOOBS  RASPBIAN  UBUNTU MATE  WINDOWS 10 IOT CORE	8GB Recommended
 RASPBIAN Lite	4GB
 OSMC  LIBREELEC	< 1 GB

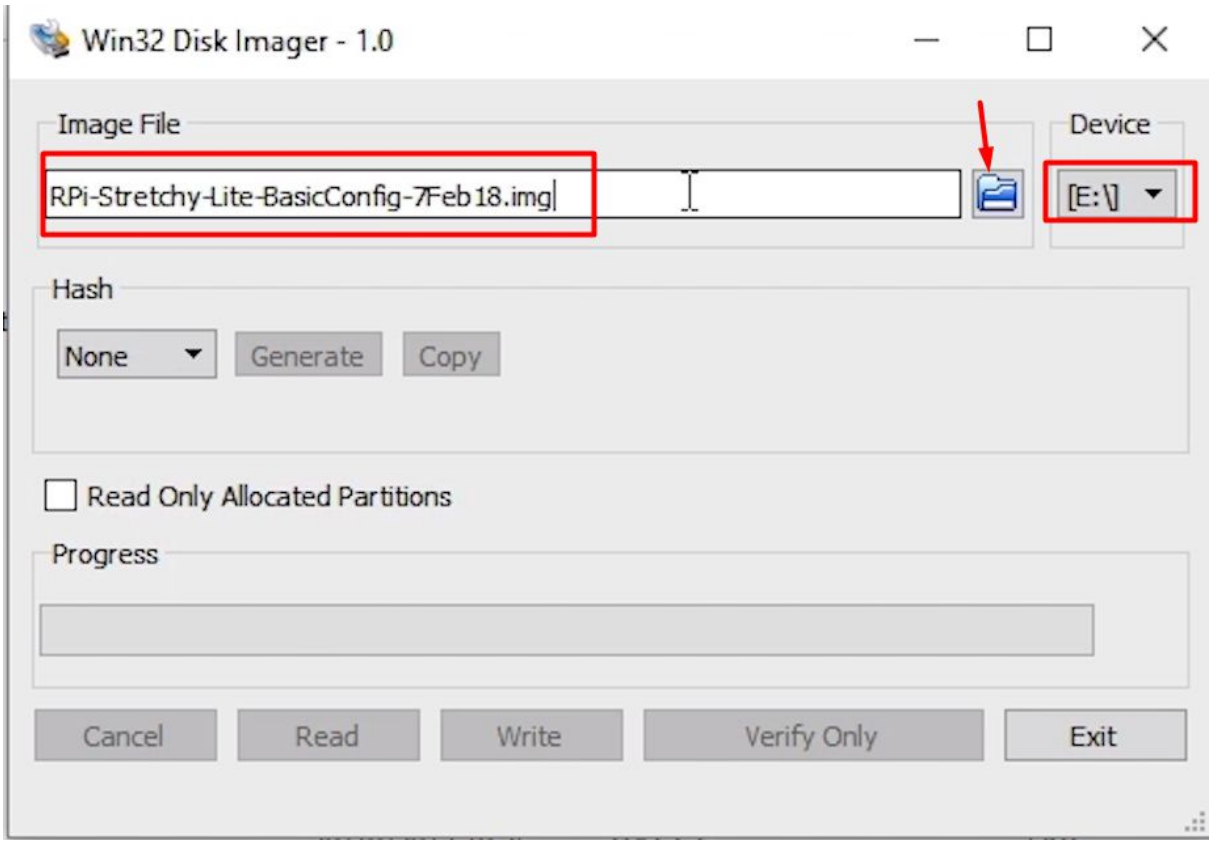
SSD Card Class or Speed

1) Recommendation: Class 10
2) The number of the class equals the Writing Speed. Example: Class 4: 4 Mb/s
3) The higher the writing speed, the lower the reading speed and then slower performance.

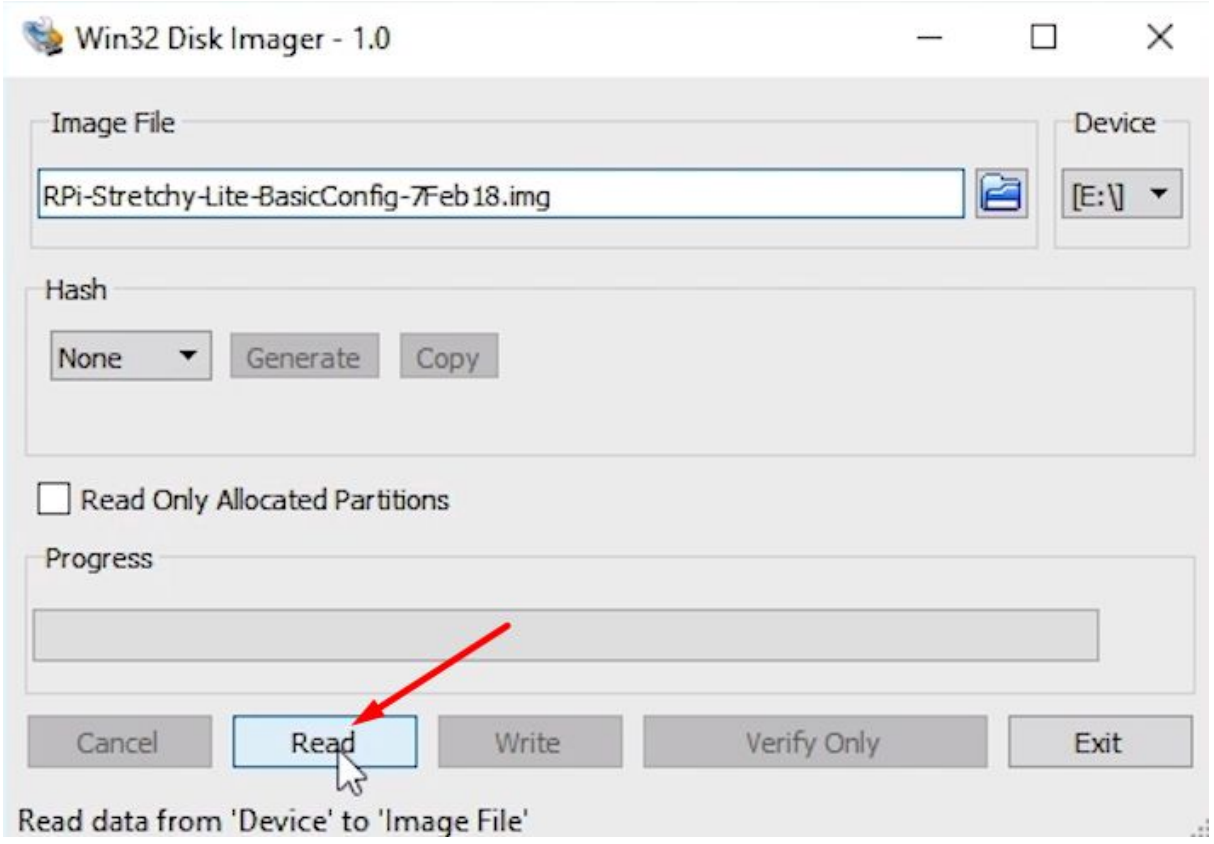
SSD Card Backup

Storing the Data in a Computer

1) Download Win32 Disk Manager
2) Put the SSD card in your laptop
3) Open the Win32 Disk Manager
4) Insert the File name, Select Drive E (referring to the SSD Card) and press the folder icon to save it somewhere in your drive



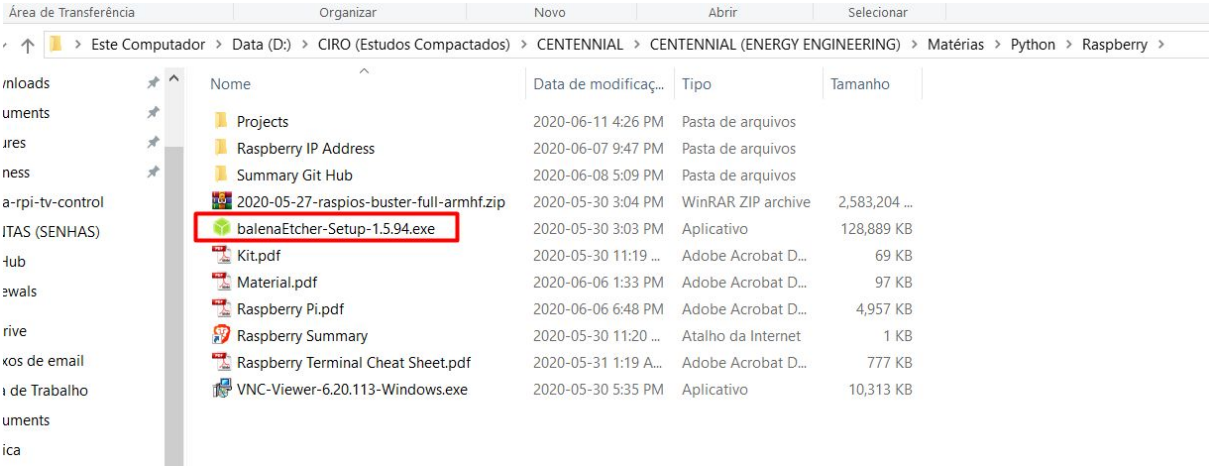
5) Click on the read button:



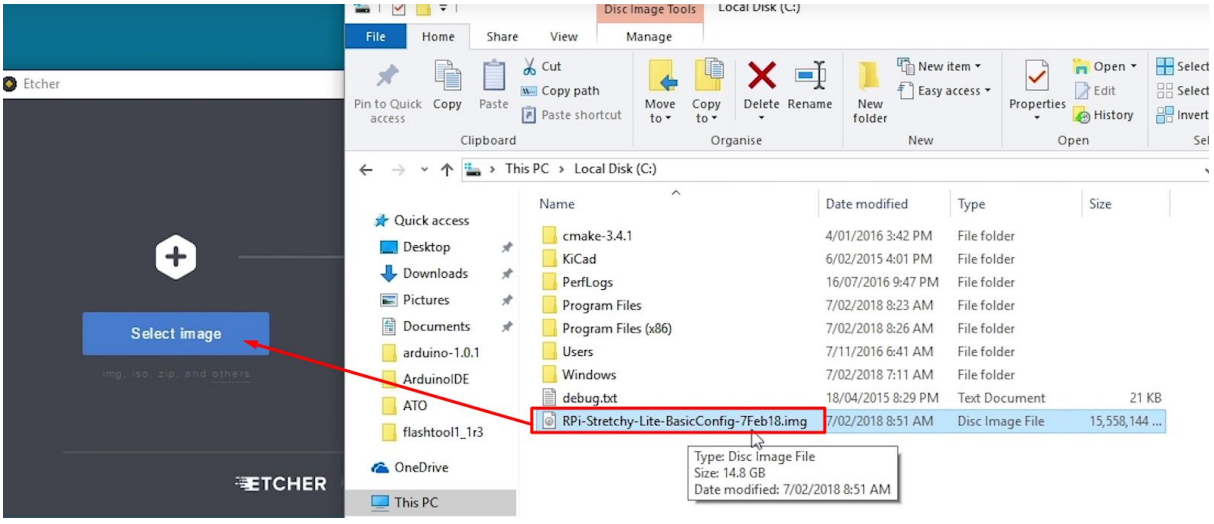
6) OK

Storing the Data back in the Raspberry Pi

1) Open Etcher:

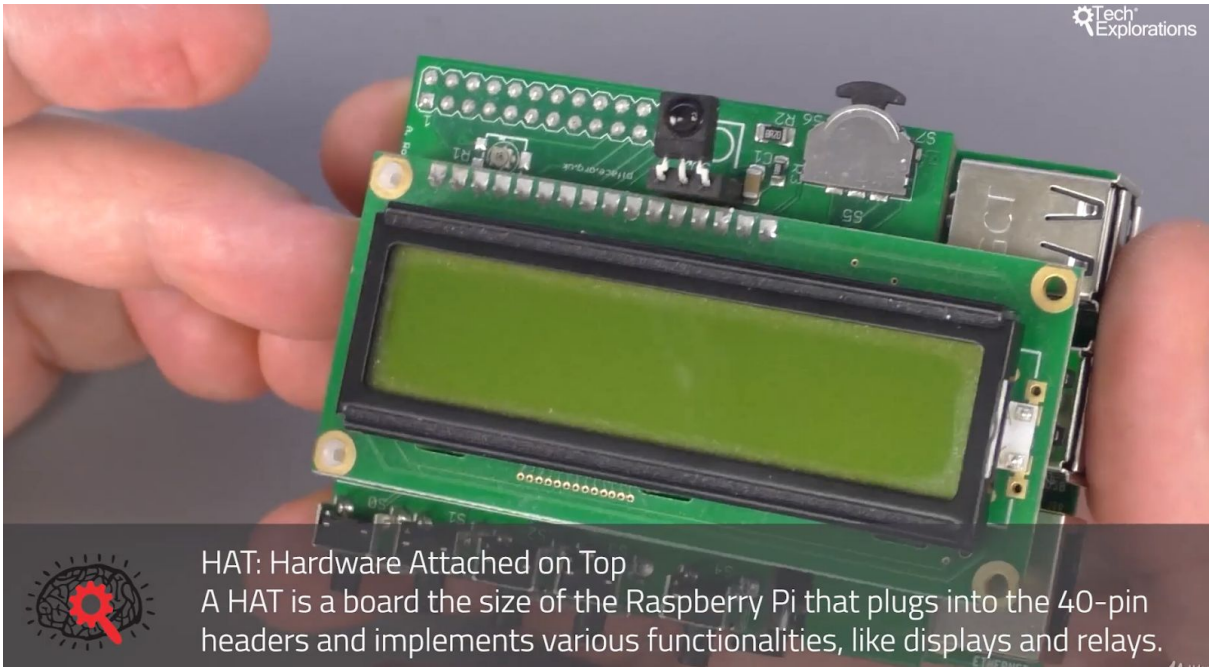


2) Select the image in Etcher as being the backup file:



3) And then select the SSD card and click Flash

HAT (Hardware Attached on Top)



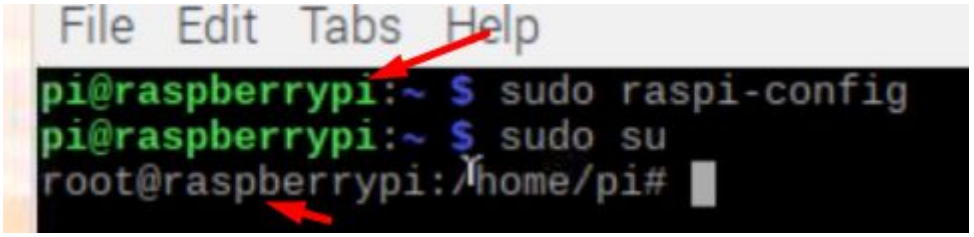
Keyboard Commands

Up and down=> go back and forth through the commands you typed before
Tab=> Autofill
CTRL+C=>Stop a ongoing process

Basic Coding (Files, Folder, Configuration)

Super User

sudo=> runs a command as a super user (user with advanced privileges)
Type before each command: sudo
Or type to change to Super User: sudo su
Super user is names as root@raspberrypi and not pi@raspberrypi



Go back to Normal User

exit

Checking Raspberry PI Configuration (Must be in super user)

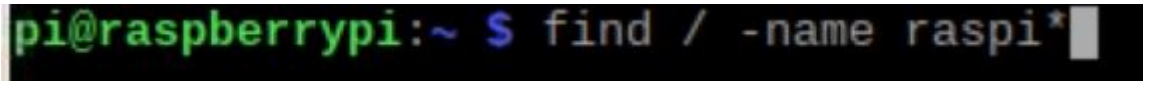
raspi-config

Check the date

date

Locate a File

find /FOLDER -name NAME OF THE FILE
The folder is where you want to start looking for the file
You can use * to look for anything after or before the name
Example:



Find in the root folder the file that starts with the name raspi

Create or Open a File

nano NAME OF THE FILE.EXTENSION
Example: nano test.txt

Create a file in blank

touch new.txt

Delete a file

rm new.txt

Files in the Folder and When Lastly Modified

ls -l

Go to another folder

cd NAME OF THE FOLDER
cd Downloads
Go back to the previous level: cd ..
Go to Home: cd ~

Create a folder or directory

mkdir NAME OF THE FOLDER
mkdir new_folder

Remove a folder

rm -rf new_folder/

Moving Files

mv NAME OF THE FILE NAME OF THE DIRECTORY/
mv test.txt new_folder/

Copying Files

cp NAME OF THE FILE NAME OF THE COPY
Example: cp test.txt test2.txt
This example would create another text file with the name test2

Shutdown Raspberry PI

Shutdown now: shutdown -h now
Shutdown 1:30AM: shutdown -h 01:30

Network Coding

IP Address

hostname -I

Network Configuration

ifconfig

Wireless Configuration

iwconfig

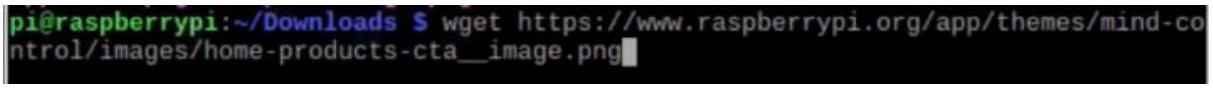
Check if the machine is running in the Network

ping IP_ADRESS (It doesn't need to be from the Raspberry PI)
ping 10.0.0.102
TO STOP THE PROCESS PRESS CTRL+C

OR
ping HOSTNAME (It doesn't need to be from the Raspberry PI)

Getting a File from the Internet

wget URL LINK



System Commands Coding

Check Memory

cat /proc/meminfo

Check SSD Card Utilization

df -h

Or

df /

Check All Packages that were installed

dpkg --get-selections

Check if a Specific Package was installed

dpkg --get-selections | grep NAME OF THE PACKAGE
Example: dpkg --get-selections | grep python

Editors in Raspberry

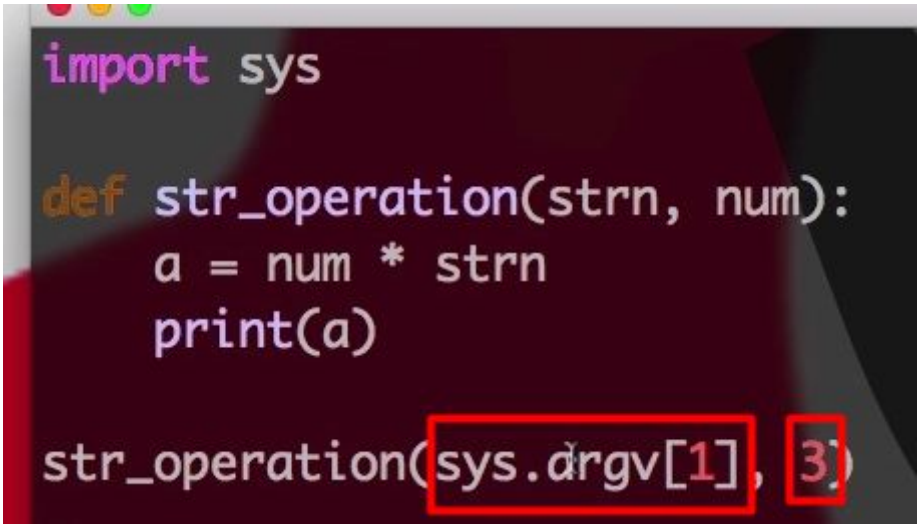
VI or VIM

4) Go to the terminal and type: sudo get-apt install vim
5) vim
6) Command mode: ESC
7) To leave VIM, type: :q and press ENTER
8) To create a file: vim example.py
9) To start editing, press: I
10) Then you can start programming
11) After you finish and want to run the code, press: ESC
12) Leave the Vim: SHIFT + :q
13) Run the code: python 3 example.py

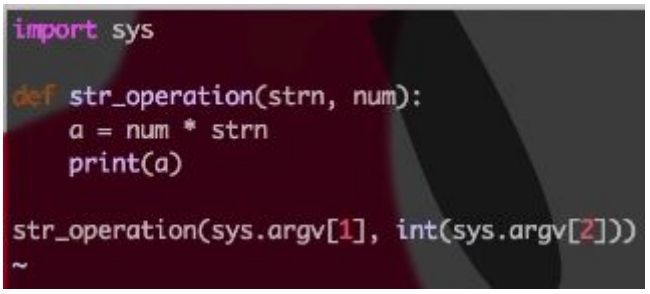
Command to Interact with the User

1) If you make a function in VIM and run the code, this function is going to be always getting its parameters from the file you made not from the user. A way to do that is to:
    a) import sys
    b) Use the sys.argv[#] to allow the user to interact with the variables of a function inside a file without having to opening it. The sys.argv[1] is a variable the user can interact while the 3 is something that is never going to change in the function. Also, the number [1] means that it is going to be the first term the user gives is going to be considered the sys.argv[1] and consequently strn

https://www.udemy.com/course/raspberry-pi-full-stack-raspbian/learn/lecture/9467534?start=60#overview



2) You can also configure this to be a string, int, float:



3) s

Installing a Software in Raspberry

Update packages

sudo apt-get update

AND

sudo apt-get upgrade

OR

sudo apt-get dist-upgrade

Install Software

sudo apt-get install NAME OF THE PACKAGE
Example: sudo apt-get install ssh

Remove Package or Software (Use all the commands below)

sudo apt-get remove NAME OF THE PACKAGE
sudo apt-get purge NAME OF THE PACKAGE
Example: sudo apt-get remove ssh
sudo apt-get purge ssh
sudo apt autoremove
sudo apt-get clean

Fix Broken Dependencies

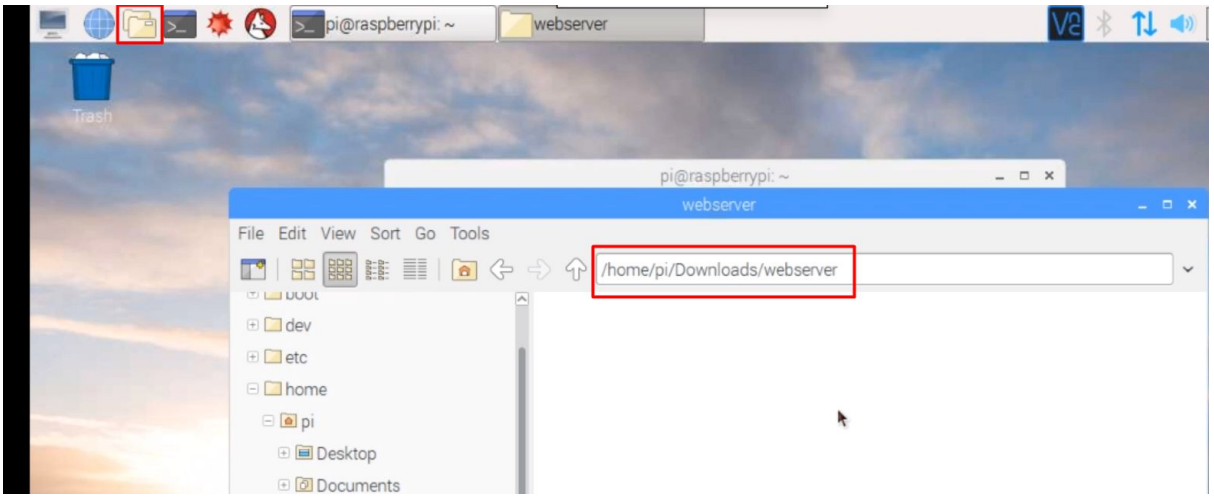
sudo apt-get -f install

Free up Space from downloads

sudo apt-get clean

Creating a Web Server

1) Create a folder:



2) Go to the terminal and find the folder:

   cd ~

   cd NEW FOLDER

3) Create a html file:

   nano index.html

4) Type something or just exit the file (Press Y)
5) Run the server:

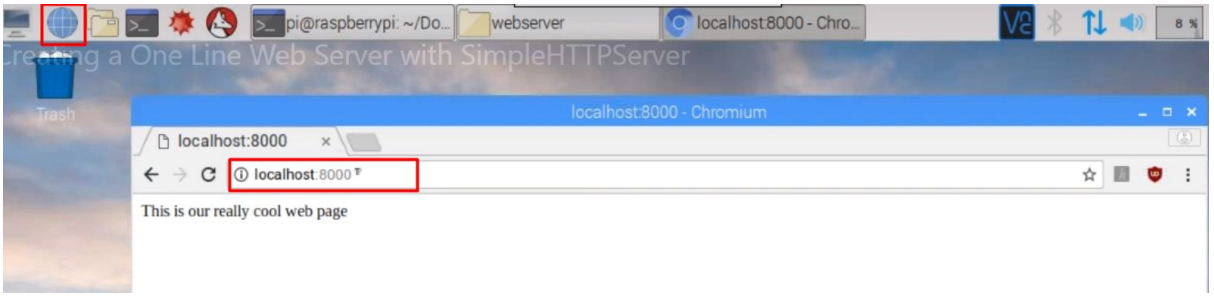
   python -m SimpleHTTPServer

   COPY THE PORT NUMBER

6) Open the browser and type:

   localhost:PORT NUMBER

   Example: localhost:8000



7) Any device that is connected to this network can access the page. So, type the:

   IP Address:Port Number

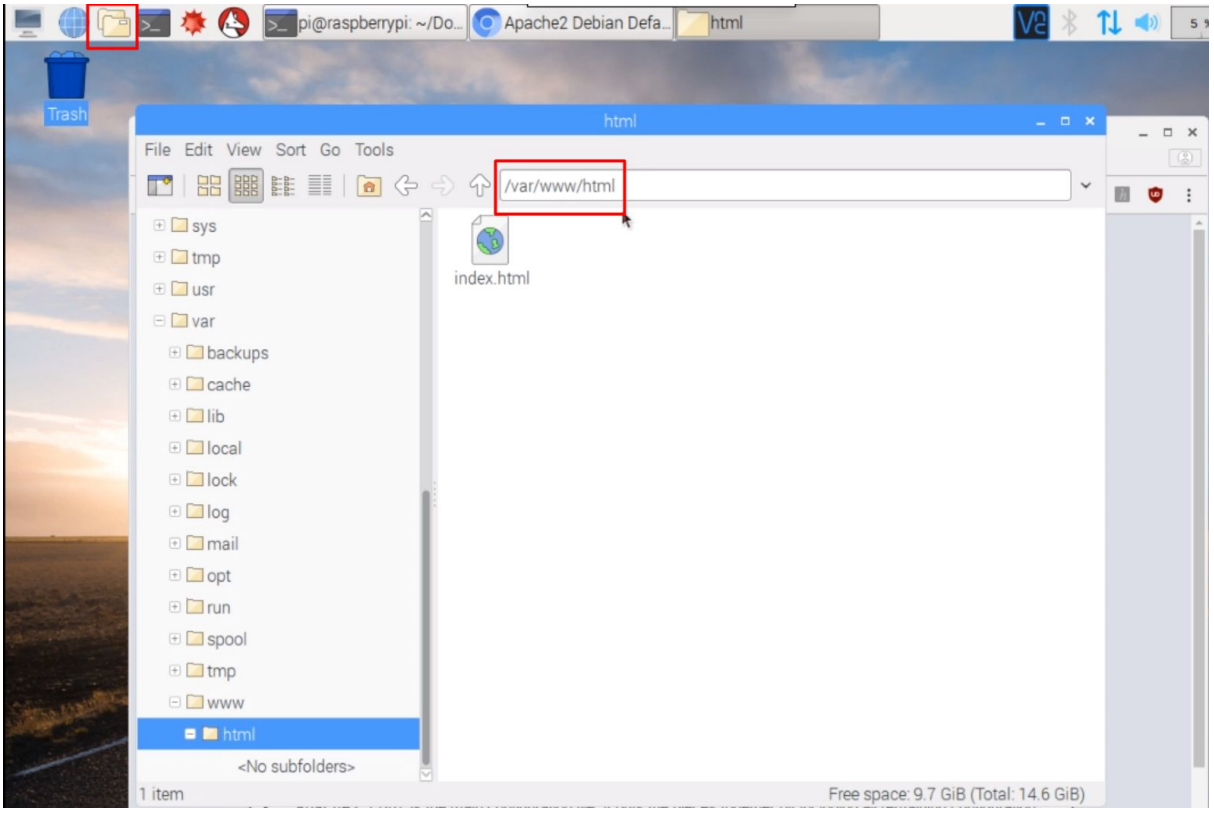
   Example: 10.0.0.117:8000

8) If you want to stop the server from running press CTRL+C

Creating a Apache Web Server

1) Install Apache:
   sudo apt-get install apache2 - y
2) The Apache Web Server Root is placed in:
   var/www/html

   You can also go to the folder in the Terminal:
   cd /var/www/html
   JUST THE ROOT (SUNDO) USER CAN CHANGE THIS FILE



3) To create a file (IT MUST BE IN THE ROOT LOCATION: var/www/html):
   sudo leafpad NAMEOFTHEFILE.html
   Example: sudo leafpad hello.html

4) Open the browser and type:
   localhost/NAMEOFTHEFILE.html
   Example: localhost/hello.html
5) Any device that is connected to this network can access the page. So, type the:

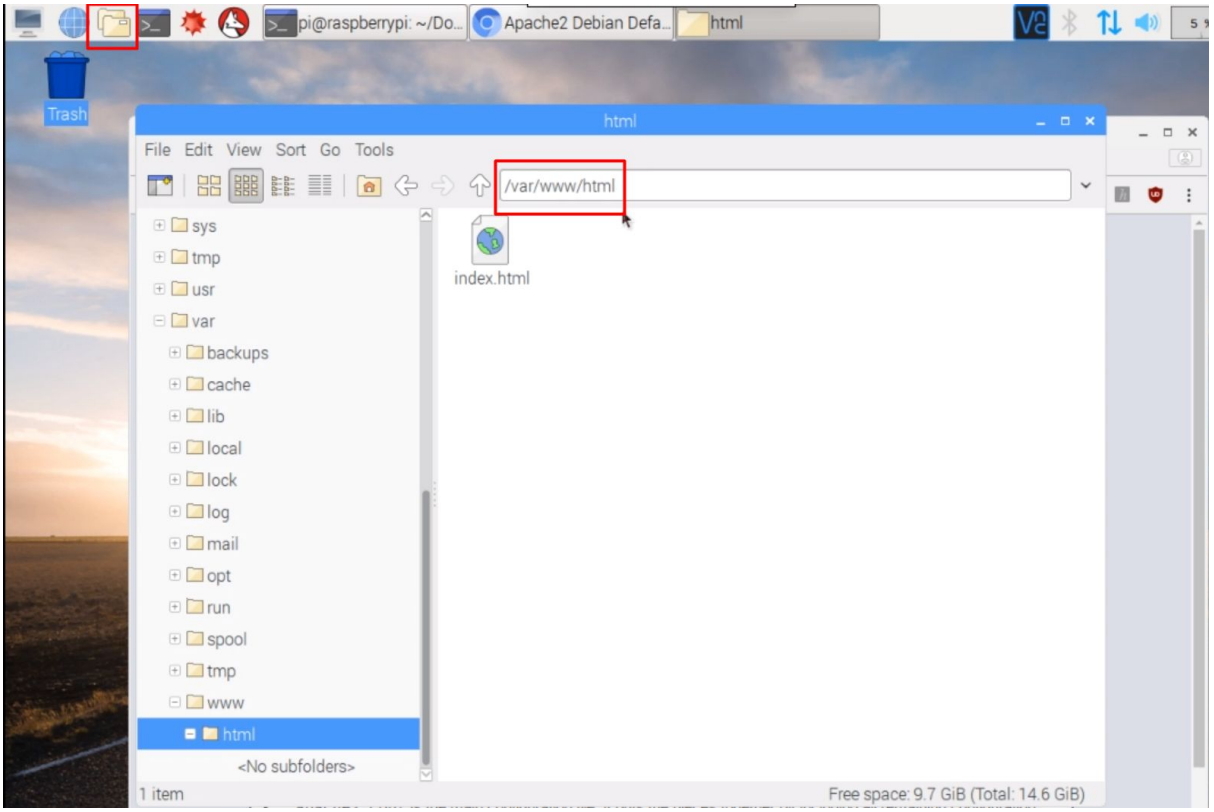
IP Address/NAMEOFTHEFILE.html

Example: 10.0.0.117/hello.html

Creating a PHP Application

6) Install PHP:
   sudo apt-get install php libapache2-mod-php -yapache2 - y
7) Restart Apache
   sudo apachectl restart
8) The Apache Web Server Root is placed in:
   var/www/html

   You can also go to the folder in the Terminal:
   cd /var/www/html
   JUST THE ROOT (SUNDO) USER CAN CHANGE THIS FILE



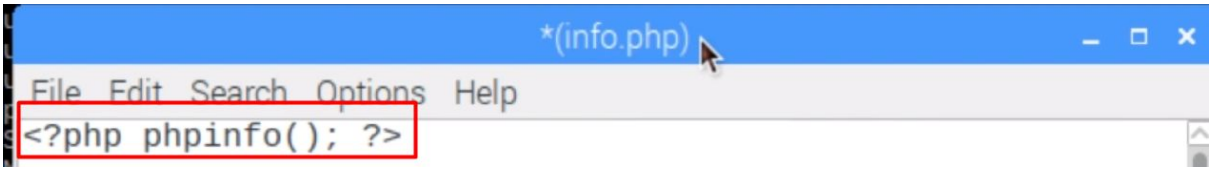
9) To create a file (IT MUST BE IN THE ROOT LOCATION: var/www/html):

   sudo leafpad NAMEOFTHEFILE.php

   Example: sudo leafpad info.php

10) Type (<? php indicates a php block and phpinfo get the information php is going to display):

```
<?php phpinfo(); ?>
```



11) Save and close
12) pen the browser and type:

    localhost/NAMEOFTHEFILE.phpl

    Example: localhost/info.php

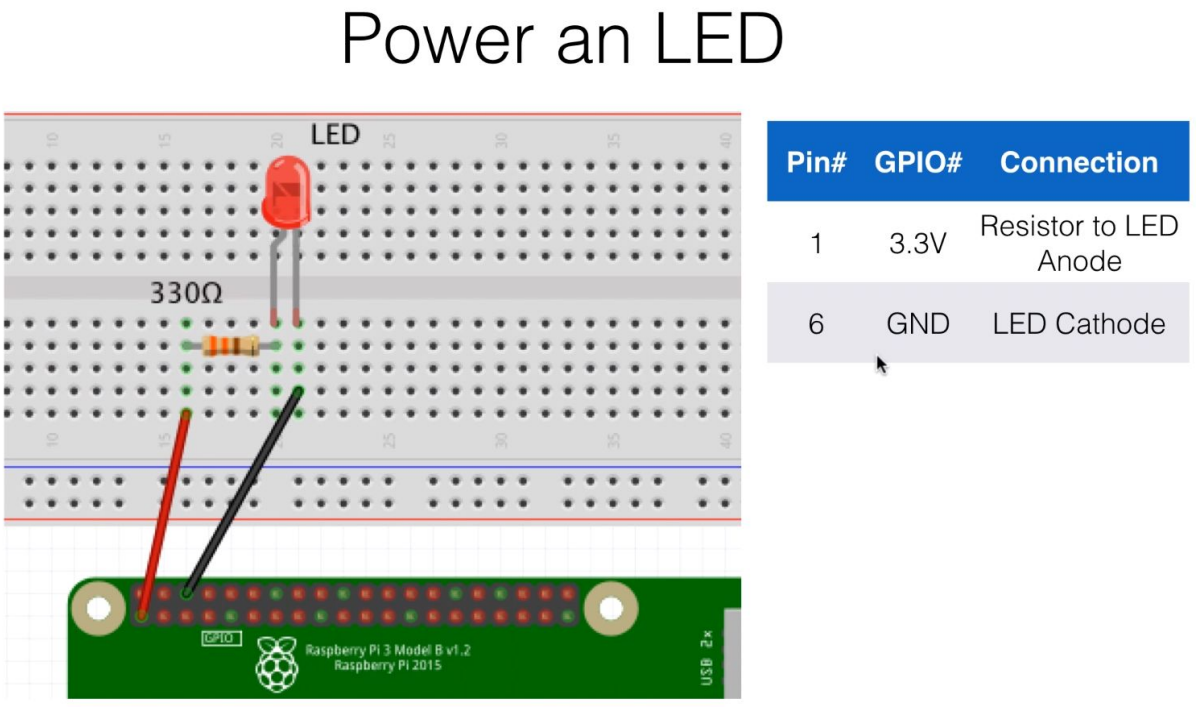
13) Any device that is connected to this network can access the page. So, type the:

    IP Address/NAMEOFTHEFILE.php

    Example: 10.0.0.117/info.php

Completed Projects

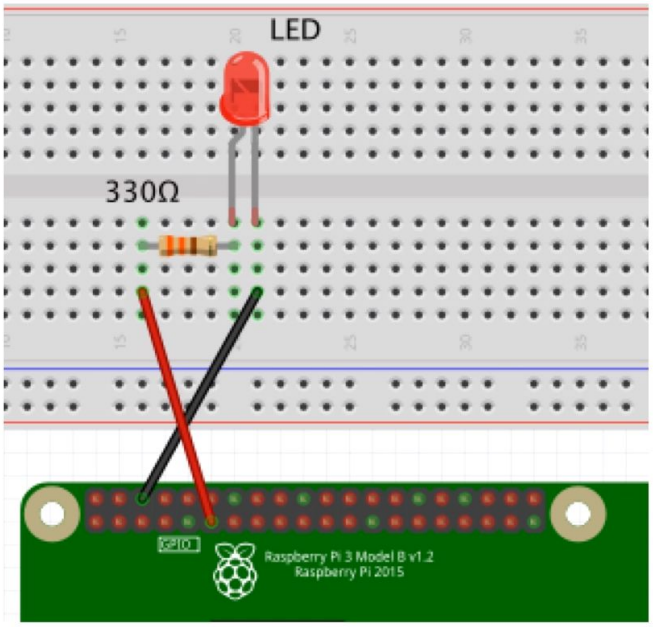
LED connection



Pin#	GPIO#	Connection
1	3.3V	Resistor to LED Anode
6	GND	LED Cathode

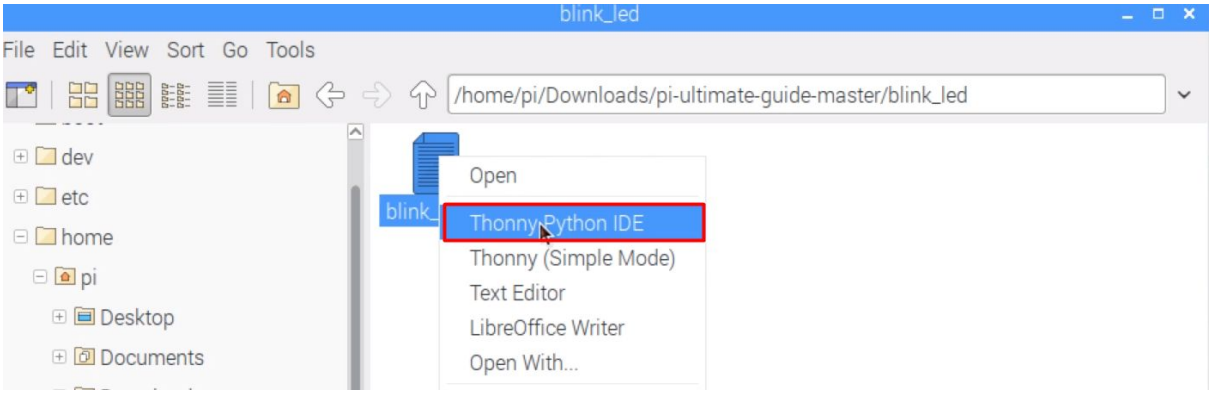
LED Blinking Connection

Blink an LED



Pin#	GPIO#	Connection
11	GPIO17	Resistor to LED Anode
6	GND	LED Cathode

14) Open the Program by selecting Thonny Python IDE:

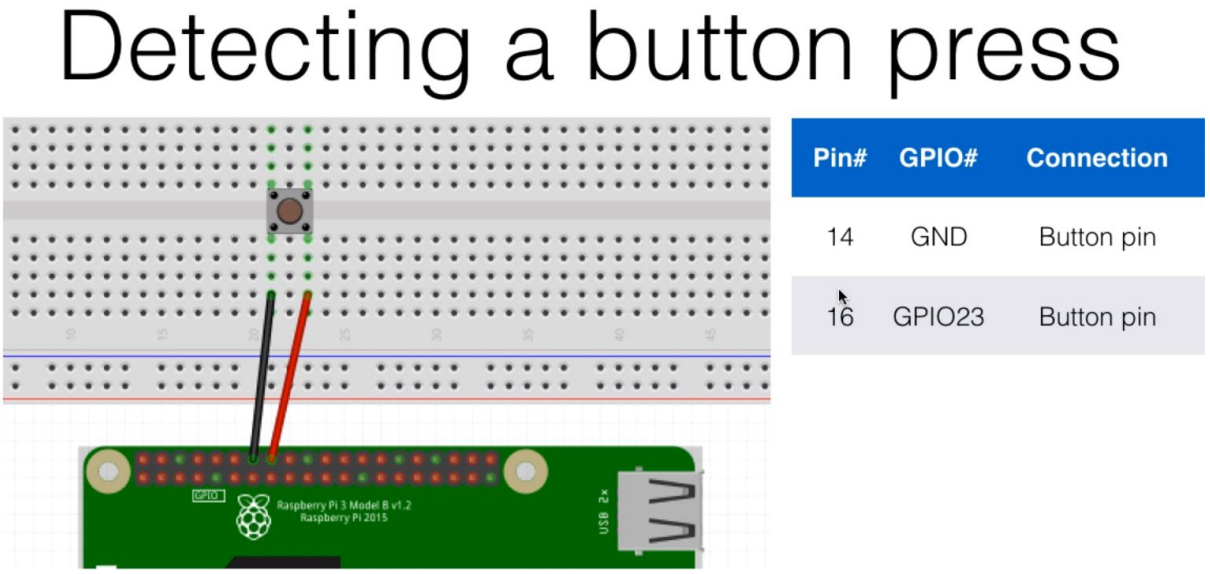


15) Code:

```
import RPi.GPIO as GPIO #Use GPIO library
import time #Use time library

GPIO.setwarnings(False) #Disable any warnings that may appear

ledPin = 11    # pin11 is connected to the led anode (+ve pin)
GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location
GPIO.setup(ledPin, GPIO.OUT)   # Set ledPin's mode as output
GPIO.output(ledPin, GPIO.HIGH) # Set ledPin high(+3.3V) to turn on led

while True: # Continue looping indefinitely
  GPIO.output(ledPin, GPIO.HIGH)  # Turn led on
  time.sleep(1) # Pause for 1 second
  GPIO.output(ledPin, GPIO.LOW) # Turn led off
  time.sleep(1) # Pause for 1 second
```

GPIO.cleanup(); #Clean up when exiting the program

Push Button Connection

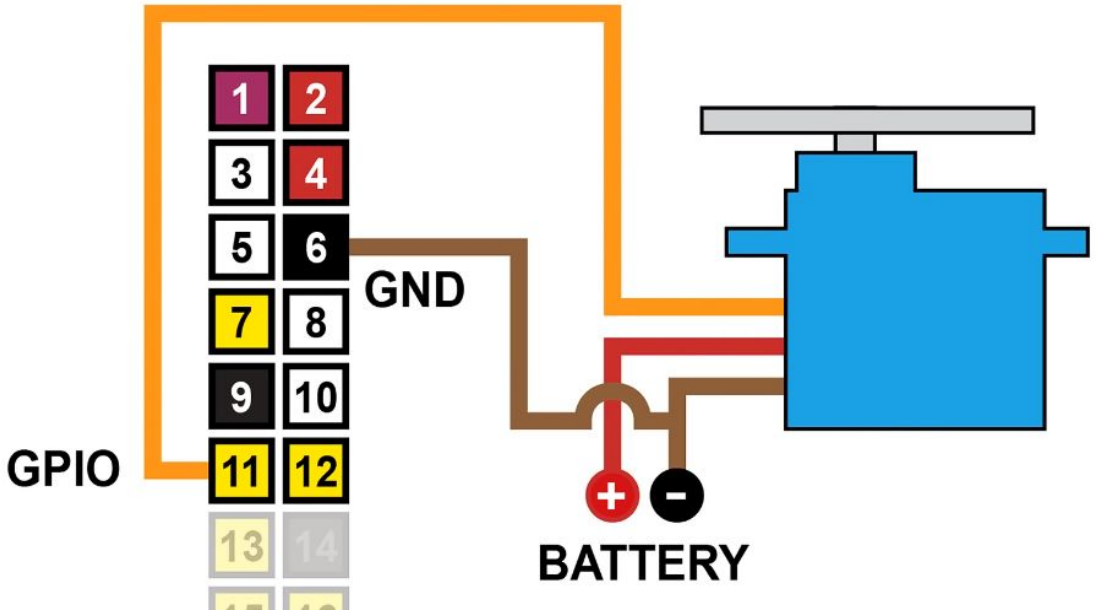


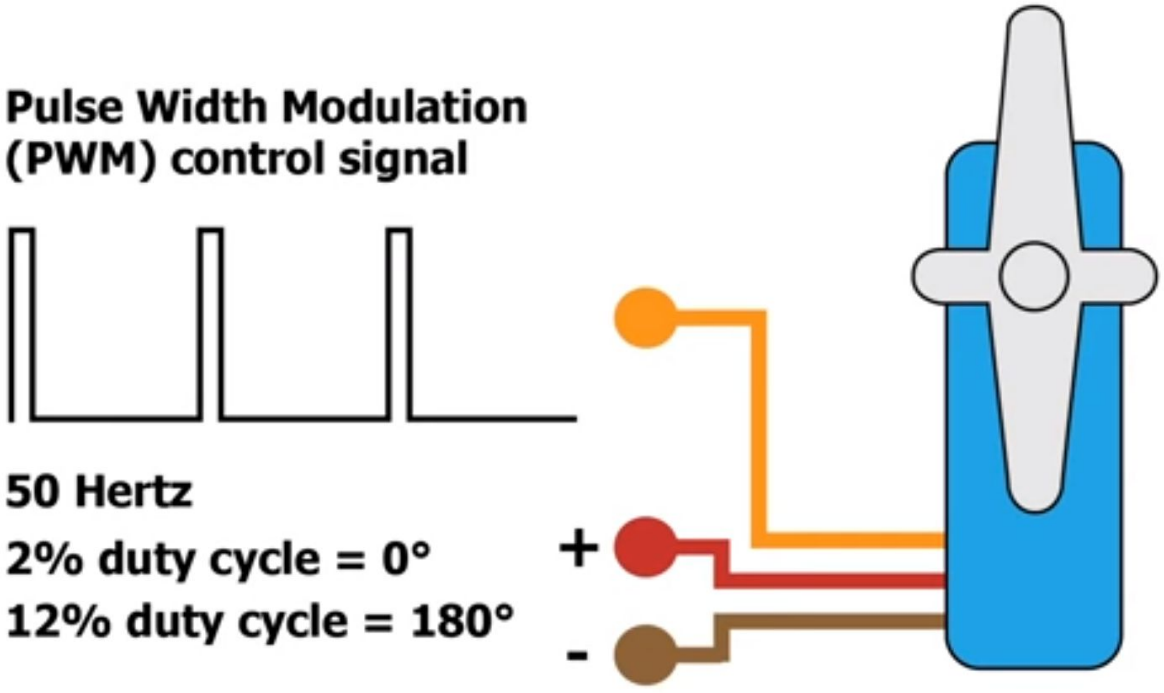
Code:

```
import RPi.GPIO as GPIO #Use GPIO library
import time #Use time library

buttonPin = 16  # pin 16 is connected to the button - other button pin is connected to GND - pin 14
GPIO.setmode(GPIO.BOARD)  # Numbers GPIOs by physical location

GPIO.setup(buttonPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)    #Set buttonPin as input and enable pullup resistor. This pullup resistor avoids short circuit when the button is pressed by using an internal pullup resistor in Raspberry Pi

while True: #Loop indefinitely
  input_state = GPIO.input(buttonPin) #Retrieve the state of the button pin
  if input_state == False:    #If readling is LOW, button has been pressed
    print('Button Pressed') #Display button pressed
    time.sleep(0.3) #Wait for 0.3 seconds for debouncing

GPIO.cleanup(); #Clean up when exiting the program
```

Controlling a Servo Motor

Brown Cable=> Ground
Red Cable=> Positive
Orange/Yellow=> Control
They should be connected to a Battery since they can draw a lot of Power



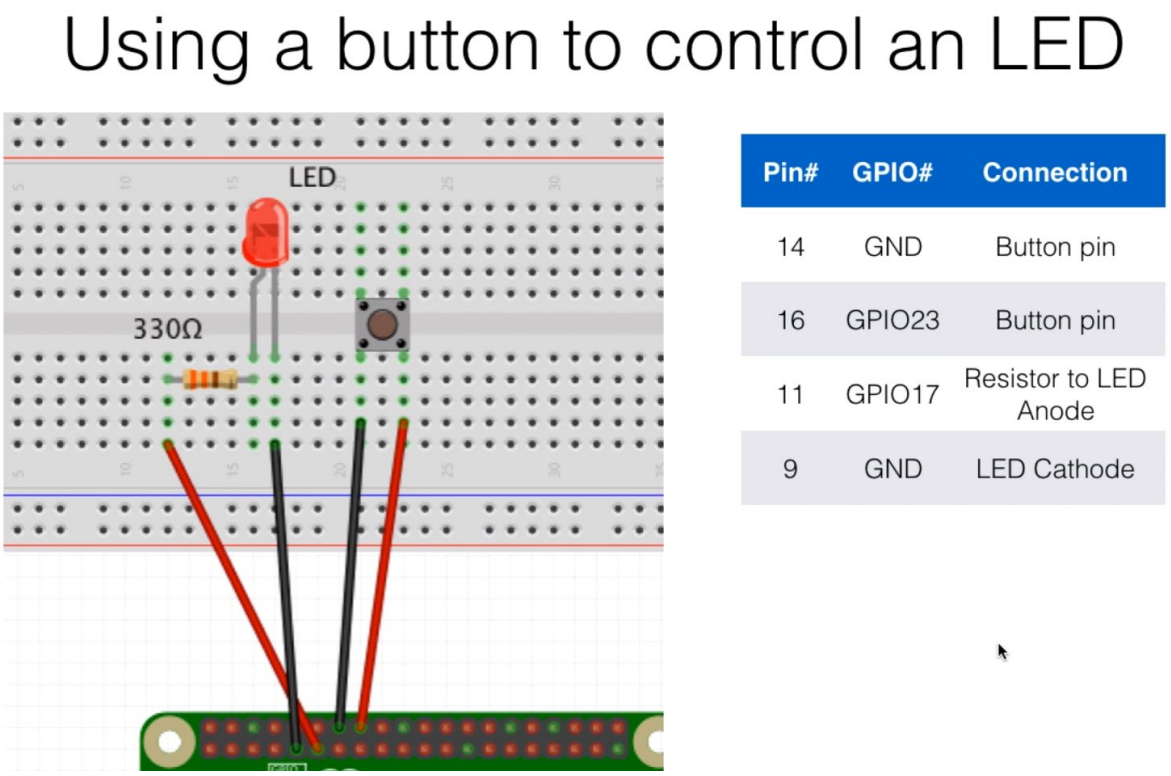


Code:

```
# Import libraries
import RPi.GPIO as GPIO
```

```
import time

# Set GPIO numbering mode
GPIO.setmode(GPIO.BOARD)

# Set pin 11 as an output, and set servo1 as pin 11 as PWM
GPIO.setup(11,GPIO.OUT)
servo1 = GPIO.PWM(11,50) # Note 11 is pin, 50 = 50Hz pulse

#start PWM running, but with value of 0 (pulse off)
servo1.start(0)
print ("Waiting for 2 seconds")
time.sleep(2)

#Let's move the servo!
print ("Rotating 180 degrees in 10 steps")

# Define variable duty
duty = 2

# Loop for duty values from 2 to 12 (0 to 180 degrees)
while duty <= 12:
    servo1.ChangeDutyCycle(duty)
    time.sleep(1)
    duty = duty + 1

# Wait a couple of seconds
time.sleep(2)

# Turn back to 90 degrees
print ("Turning back to 90 degrees for 2 seconds")
servo1.ChangeDutyCycle(7)
time.sleep(2)

#turn back to 0 degrees
print ("Turning back to 0 degrees")
servo1.ChangeDutyCycle(2)
time.sleep(0.5)
servo1.ChangeDutyCycle(0)

#Clean things up at the end
servo1.stop()
GPIO.cleanup()
print ("Goodbye")
```

LED Blinking and Push Button Connection



Pin#	GPIO#	Connection
14	GND	Button pin
16	GPIO23	Button pin
11	GPIO17	Resistor to LED Anode
9	GND	LED Cathode

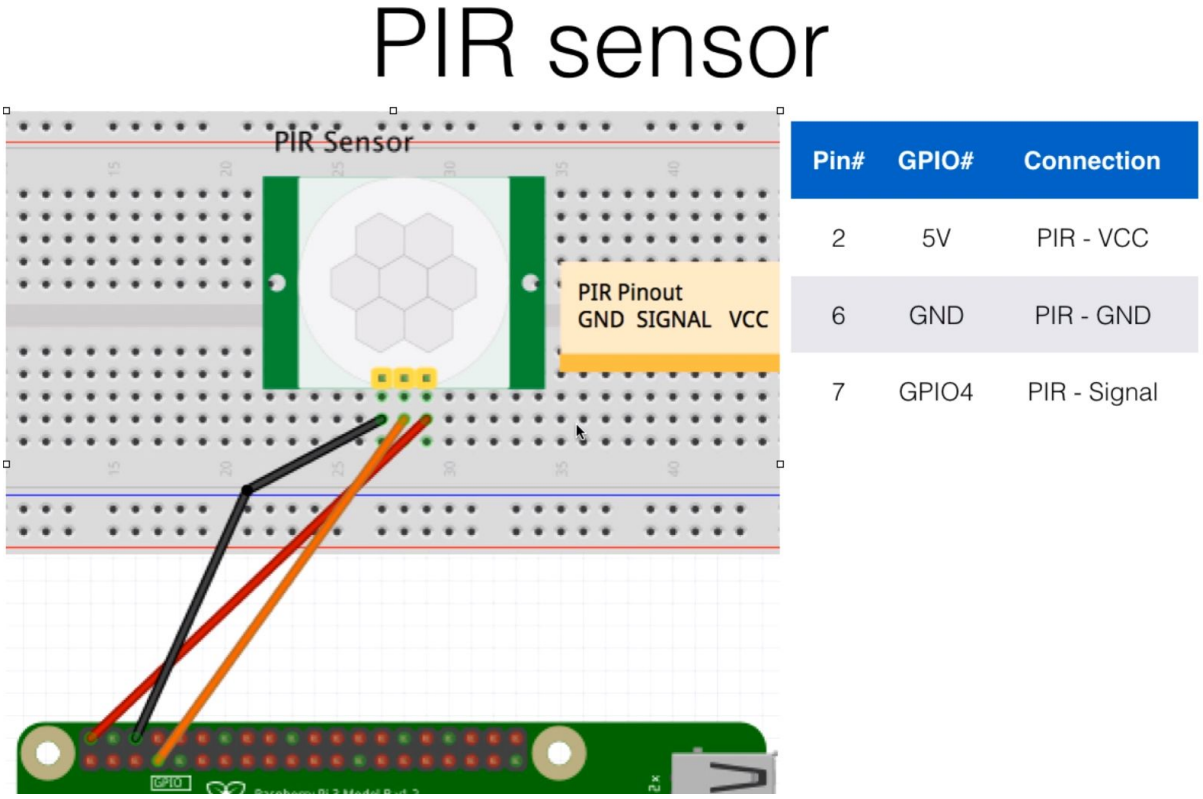
Code:

```
import RPi.GPIO as GPIO #Use GPIO library
import time #Use time library

GPIO.setwarnings(False)

ledPin = 11    # pin11 is connected to the led anode (+ve pin), cathode(-ve pin) connected to pin9 GND
buttonPin = 16  # pin 16 is connected to the button - other button pin is connected to gnd - pin 14
GPIO.setmode(GPIO.BOARD)  # Numbers GPIOs by physical location

GPIO.setup(ledPin, GPIO.OUT)    #set ledPin's mode as output
GPIO.output(ledPin, GPIO.LOW)   #initially turn off the led

GPIO.setup(buttonPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)    #Set buttonPin as input and enable pullup resistor

while True: #Loop indefinitely
  GPIO.wait_for_edge(buttonPin, GPIO.FALLING) #Button was pressed (When the button is not pressed it is HIGH, when it is pressed, it turns into low)
  print('Button Pressed') #Display button pressed
  GPIO.output(ledPin, GPIO.HIGH)  # Turn led on
```

```
  GPIO.wait_for_edge(buttonPin, GPIO.RISING) #Button was released
  GPIO.output(ledPin, GPIO.LOW)  # Turn led off

GPIO.cleanup(); #Clean up when exiting the program
```

PIR (Passive Infrared) Sensor

This sensor detects the change in its voltage when a heat body passes through its range,so it can detect movement



Pin#	GPIO#	Connection
2	5V	PIR - VCC
6	GND	PIR - GND
7	GPIO4	PIR - Signal

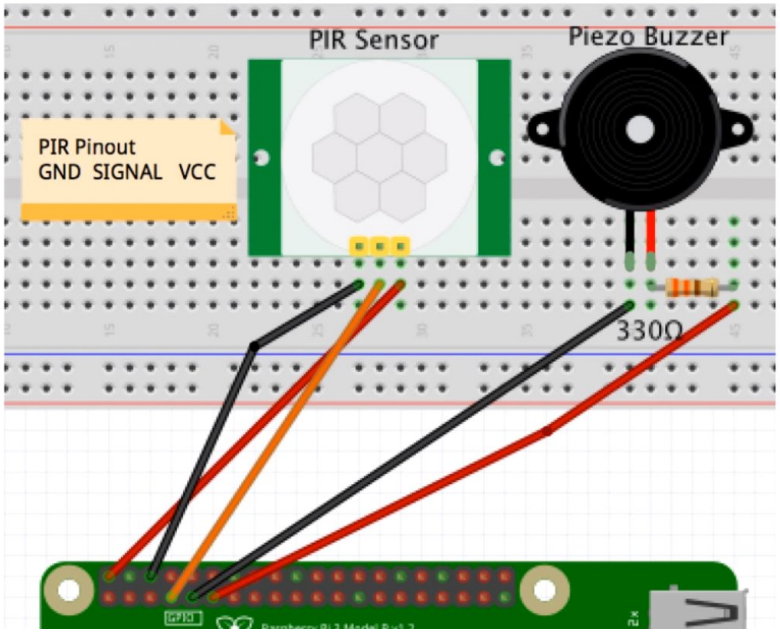
Code:

```
import RPi.GPIO as GPIO #Use GPIO library
import time #Use time library


pirPin = 7  # pin 7 is connected to output from the PIR motion sensor
GPIO.setmode(GPIO.BOARD)  # Numbers GPIOs by physical location
GPIO.setup(pirPin, GPIO.IN) #set the pirPin as an input
count = 0; #number used as a reference to detect motion

while True: #Loop indefinitely
  input_state = GPIO.input(pirPin) #Retrieve the state of the button pin
  if input_state == True:    #If readling is HIGH, there is output from the motion sensor
    print("Motion Detected -- " + str(count))  #Motion was detected
    count+=1  #increment the count
    time.sleep(0.3) #Wait for 0.3 sec
```

GPIO.cleanup(); #Clean up when exiting the program

PIR (Passive Infrared) Sensor + Alarm

PIR sensor alarm



Pin#	GPIO#	Connection
2	5V	PIR - VCC
6	GND	PIR - GND
7	GPIO4	PIR - Signal
11	GPIO17	Resistor to +ve piezo
9	GND	-ve piezo

Code:

```
import RPi.GPIO as GPIO #Use GPIO library
import time #Use time library

pirPin = 7  # pin 7 is connected to output from the PIR motion sensor, 5V to pin 2, GND to pin 6
buzzerPin = 11  # pin 11 is connected to the +ve buzzer pin; -ve connected to pin 9 GND
GPIO.setmode(GPIO.BOARD)  # Numbers GPIOs by physical location

GPIO.setup(pirPin, GPIO.IN); #set the pirPin as input
GPIO.setup(buzzerPin, GPIO.OUT); #set the buzzer pin as output
GPIO.output(buzzerPin, GPIO.LOW) #initially turn off the buzzer

#define alarm events
def soundAlarm(pirPin):
  #sound alarm (for 2 seconds )
  print("Sound Alarm")
  GPIO.output(buzzerPin, GPIO.HIGH)  # Turn buzzer on
  time.sleep(2) # Pause for 2 seconds
  turnOffAlarm()  # Turn buzzer off

def turnOffAlarm():
```

```
  #turn off alarm
  GPIO.output(buzzerPin, GPIO.LOW)  # Turn buzzer off

#adding a callback function when the pir sensor output rises when motion is detected
GPIO.add_event_detect(pirPin, GPIO.RISING, callback=soundAlarm);

#try catch block to perform GPIO cleanup
try:
  while True:
    pass
except KeyboardInterrupt:
  print("You ended the program")
finally:
  #clean up the GPIO pins
  GPIO.cleanup()
```

Temperature/Humidity Sensor

1) Download all the files in your Raspberry Pi: git clone https://github.com/futureshocked/RaspberryPiFullStack_Raspbian.git
2) Install the DHT11 library:
   git clone https://github.com/adafruit/Adafruit_Python_DHT.git
3) Install the sensor file: sudo python3 setup.py install
4)

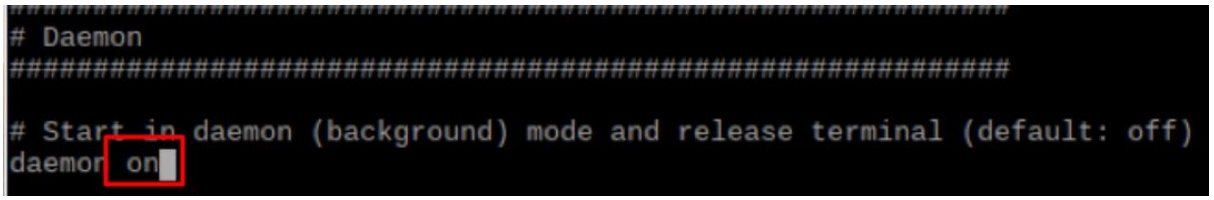
Camera with Motion Package (Just put in the Web Browser)

1) It is easier to get a camera with USB connection
2) IF YOU HAVE THE ANY OTHER CAMERA PACKAGE INSTALLED, YOU HAVE TO UNINSTALL IT (Go to this part - link)
3) Go to the Raspberry PI terminal and install the webcam package by typing:

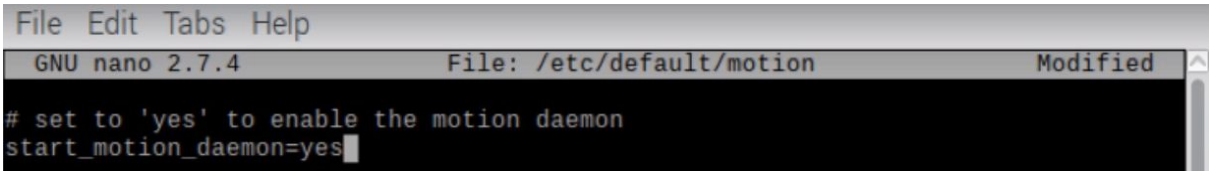
sudo apt-get install motion

4) Type: y
5) Check if the camera is shown as connected:
   lsusb

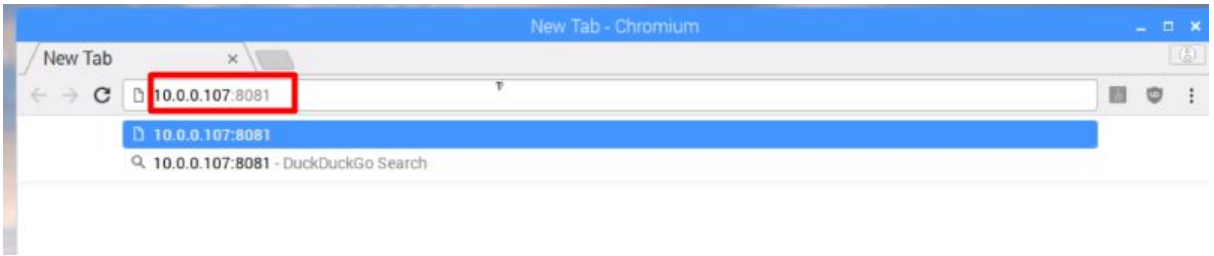
6) sudo nano / etc/motion//motion.config
7) CTRL+W
8) Search for: daemon
9) Change from OFF to ON



10) CTRL+W
11) Search for: localhost
12) Change from ON to OFF
13) CTRL+W
14) Search for: webcontrol_localhost
15) Change from ON to OFF
16) CTRL+X
17) Y
18) Enter
19) sudo nano / etc/default/motion
20) Change from NO to YES



21) CTRL+X
22) Y
23) Enter
24) sudo service motion start
25) If you want to stop the program: sudo service motion stop
26) Open Chromium in the Raspberry Pi
27) Get the Network IP Address
28) Type in the browser:
    IP Address:8081
    Example: 10.0.0.107:8081
    Since the camera is connected to the Raspberry PI, you can also use: localhost:8081



29) s

Camera with FSwebcam Package (Allows you to Take Pics and Record Video)

1) It is easier to get a camera with USB connection

2) IF YOU HAVE THE OTHER CAMERA PACKAGE INSTALLED (MOTION), YOU HAVE TO UNINSTALL IT (Go to this part - link, typing MOTION at the end)
3) Create a folder:
   mkdir ~/Downloads/photos/
4) cd ~/Downloads/photos/
5) Go to the Raspberry PI terminal and install the webcam package by typing:
   sudo apt-get install fswebcam
6) Choosing the image resolution:
   fswebcam -r 1280x720
7) Take a photo with the webcam:
   fswebcam image.jpg
8) You can change the resolution and take a picture
   fswebcam -r 1280x720 image2.jpg
9) Removing the banner of the image:
   fswebcam -r 1280x720 --no banner image3.jpg

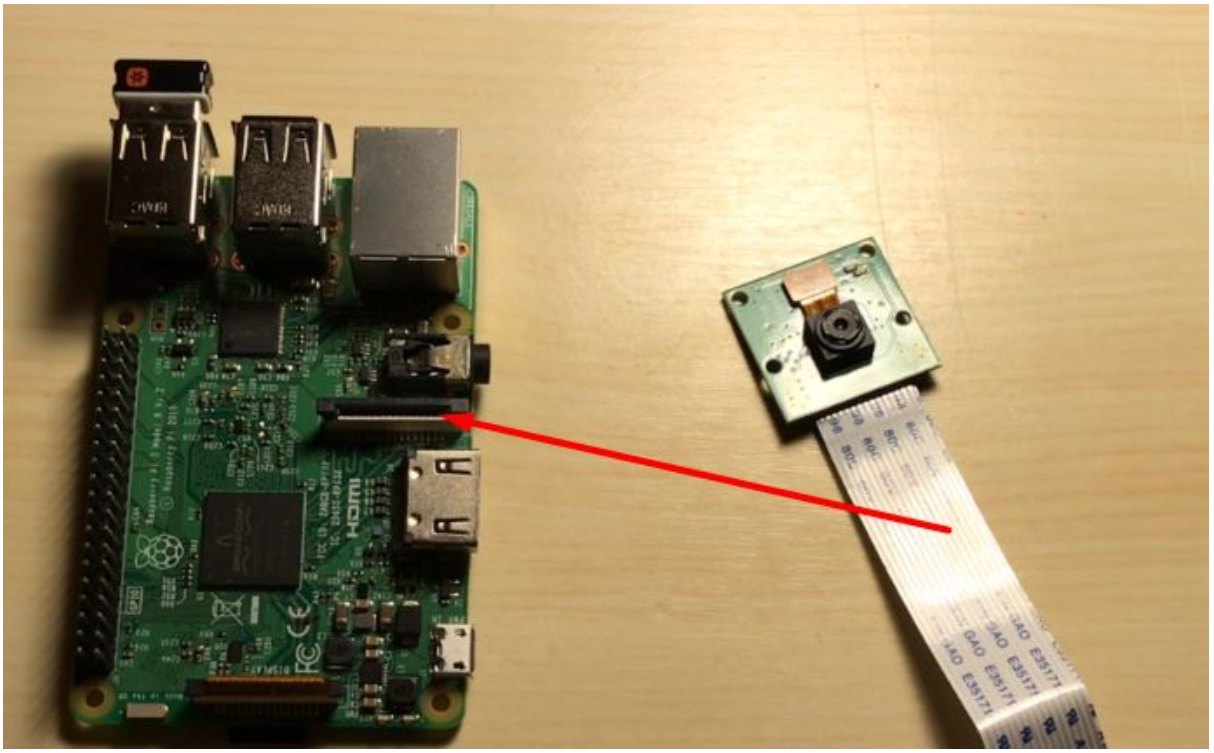
Script to Take Pictures

https://www.udemy.com/course/pi-ultimate-guide/learn/lecture/9385984#questions

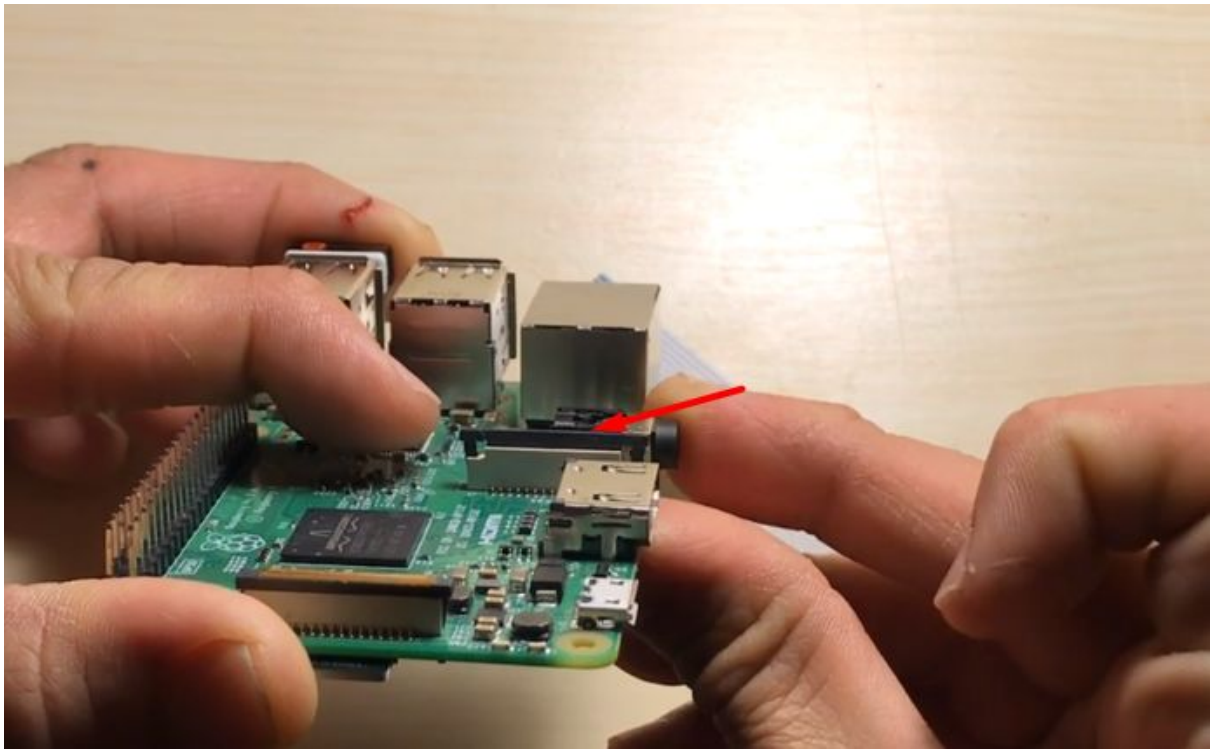
Camera Module

The Camera Module is specific for Raspberry PI

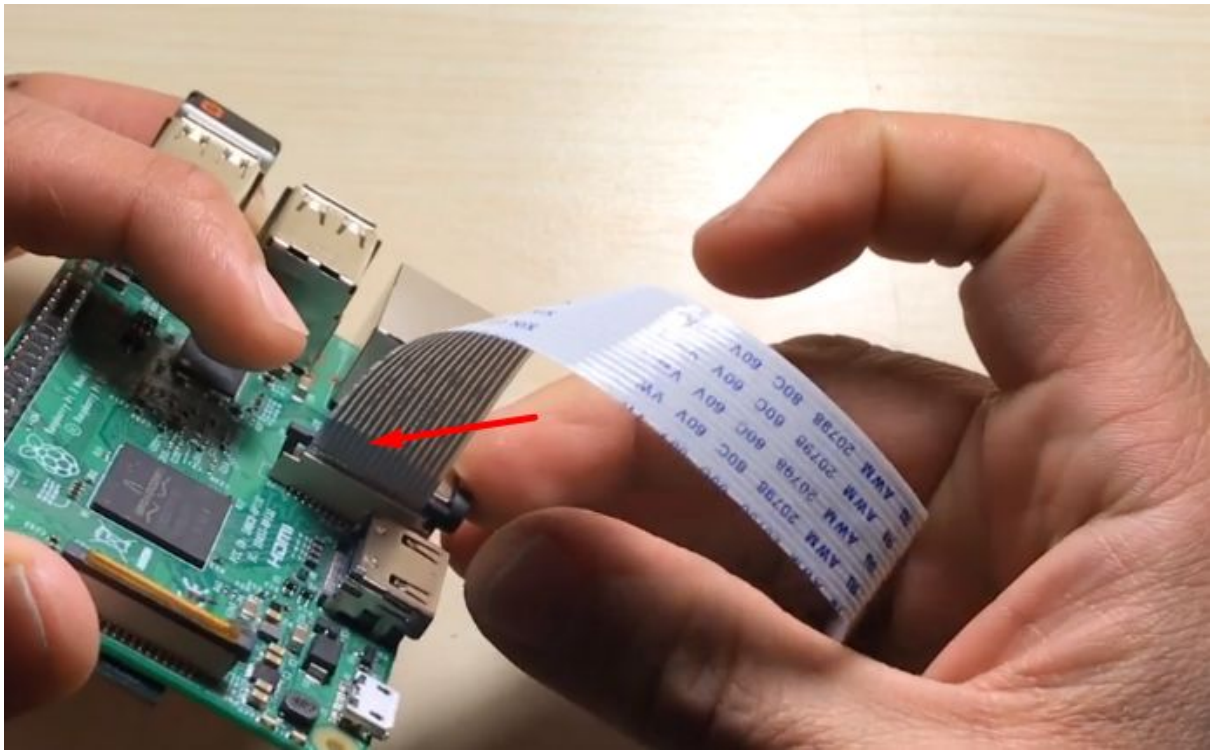
1) The camera module goes to that section



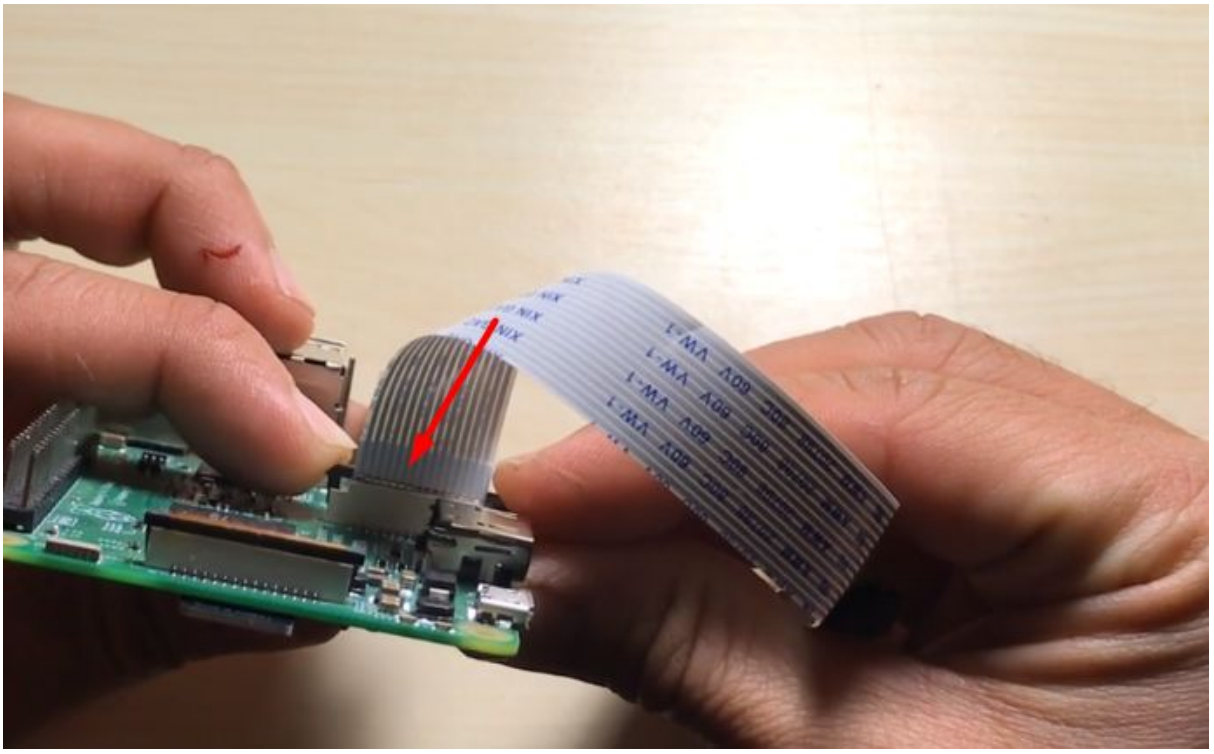
2) Lift the black compartment



3) Put the tape there:



4) Push the black part down:



Adjusting the Brightness

```
camera.brightness = 70
```

Put Text in the Image

```
camera.annotate_text = "Brightness: %s" % i
```

Resolution

```
camera.resolution = (2592, 1944)
```

Frame Rate

```
camera.framerate = 15
```

Camera Effects

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview()
for effect in camera.IMAGE_EFFECTS:
  camera.image_effect = effect
```

```
  camera.annotate_text = "Effect: %s" % effect
  sleep(5)
camera.stop_preview()
```

Showing a Live Video

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview()
sleep(10) # Live for 10 seconds
camera.stop_preview()
```

Capture an Image

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview()
sleep(5)
camera.capture('/home/pi/Downloads/photos/image.jpg')
camera.stop_preview()
```

Capture Multiple Images

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview()
for i in range(5):
  sleep(5)
  camera.capture('/home/pi/Downloads/photos/image%s.jpg' % i)
camera.stop_preview()
```

Record a Video

1) Code:

```
from picamera import PiCamera
```

```
from time import sleep

camera = PiCamera()

camera.start_preview()
camera.start_recording('/home/pi/Downloads/photos/video.h264')
sleep(10)
camera.stop_recording()
camera.stop_preview()

#view image
#omxplayer video.h264
```

2) Play the Video using OMXPLAYER. Type in the terminal:

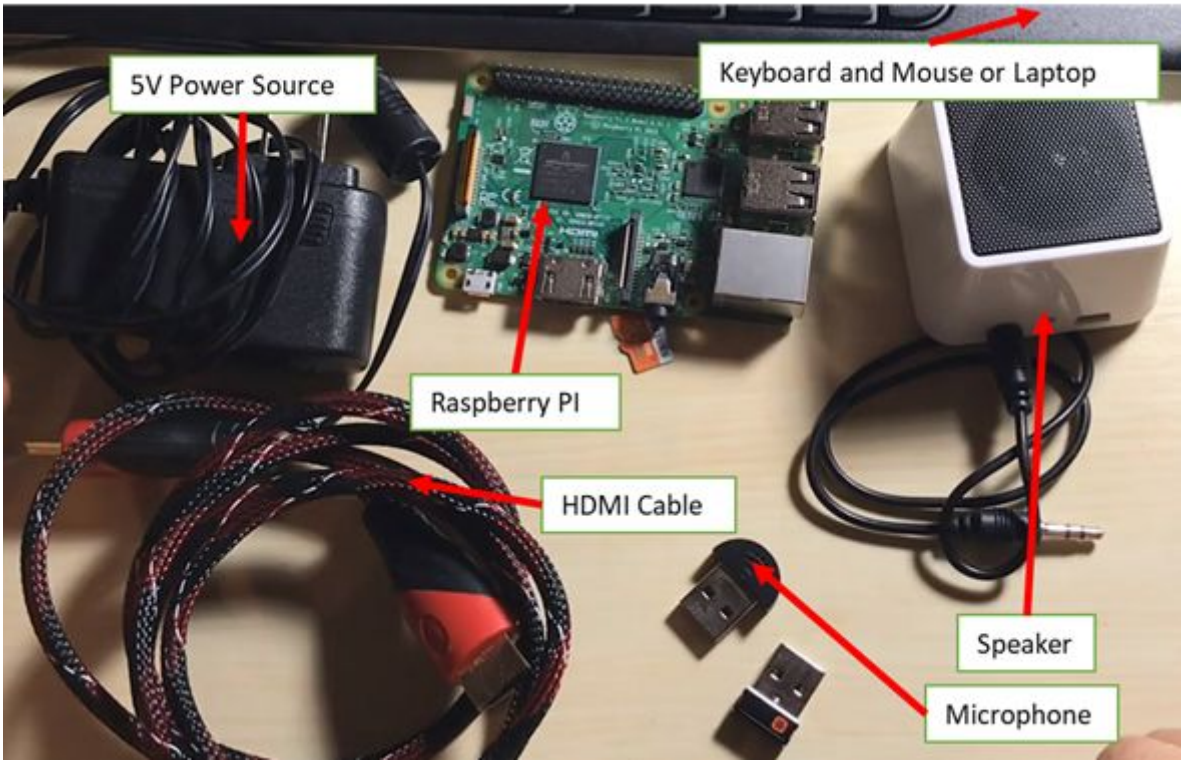
omxplayer ~PATH/NAMEOFTHEFILE

Example: omxplayer ~ /Downloads/photos/video.h264

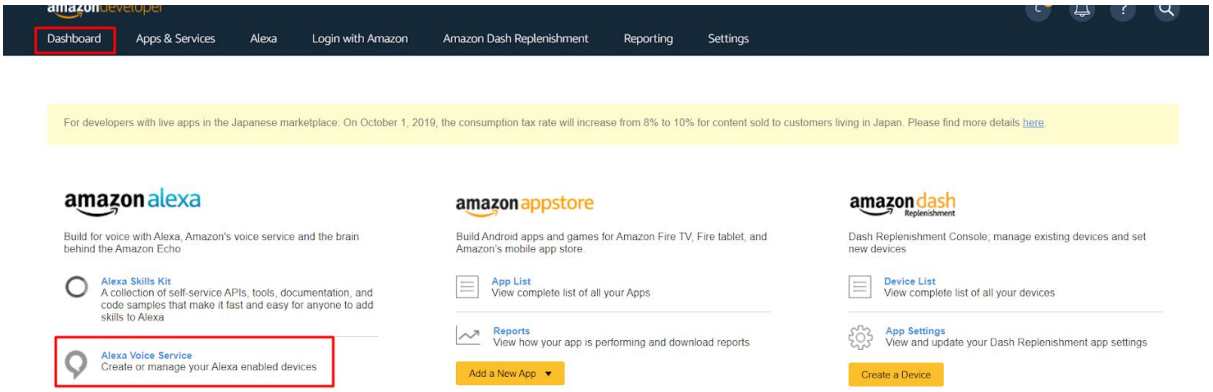
3) s

Amazon Alexa with Raspberry PI

1) Material:



2) Go to the website: developer.amazon.com
3) Log in
4) Go dashboard=> Alexa Voice Service



5) Products
6) Add new products
7) Fill up the fields below:

Product name *

PI_Echo

Product ID *

PI_Echo

Please select your product type. *

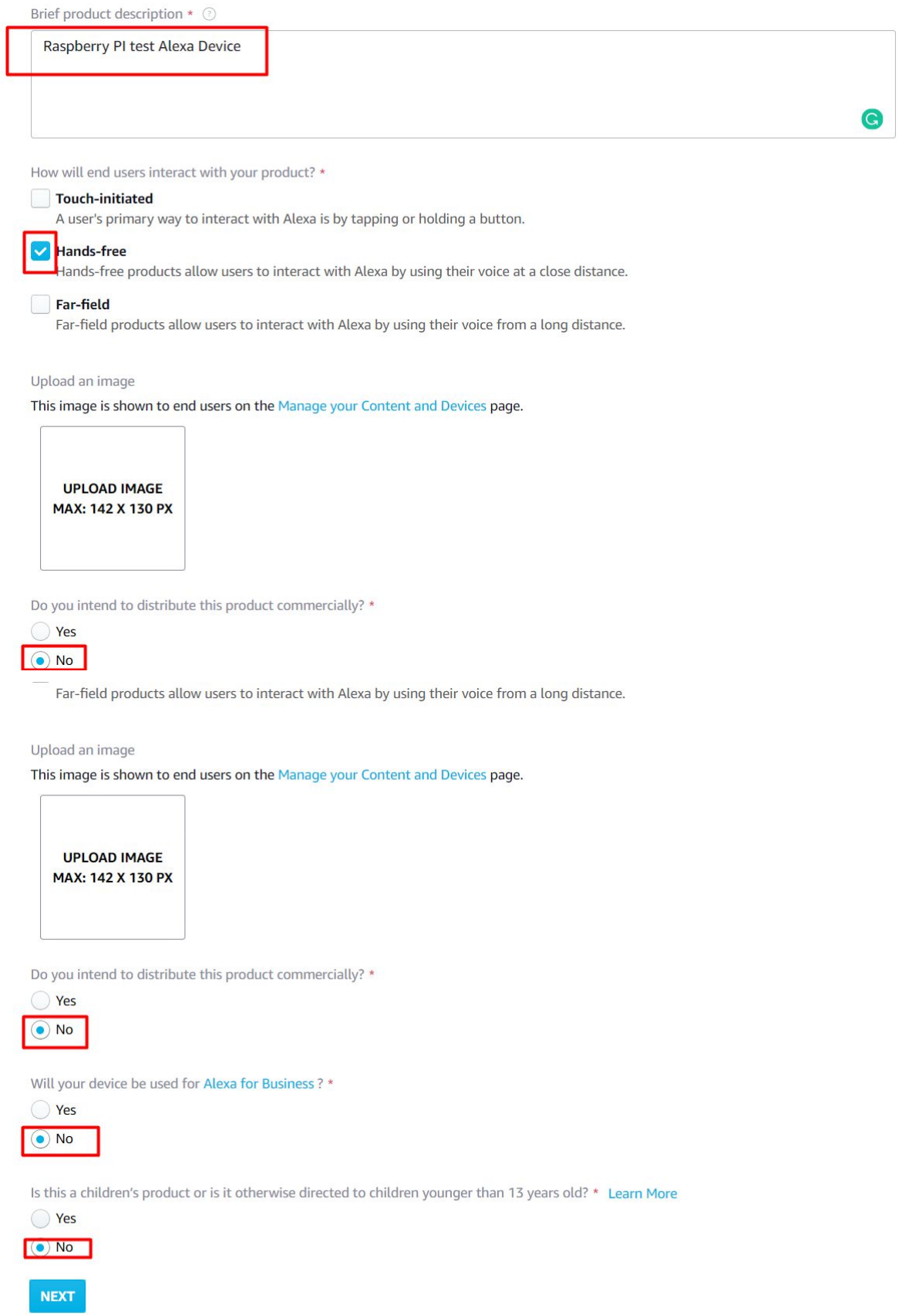
- **Application with Alexa built-in**
  A standalone app. This includes apps on the web, Android, Kindle, iOS, FireTV, AppleTV, etc.
- **Device with Alexa built-in**
  Physical product with the potential to have buttons, knobs, a touch screen, etc. Examples are speakers, headsets, televisions, set top boxes, appliances, etc. Includes Alexa Mobile Accessories.
- **Alexa Gadget**
  An accessory that connects to a compatible Echo device via Bluetooth. For example, an animatronic that moves a motor in response to sound, or lights up if a notification arrives. Learn More

Will your device use a companion app? *

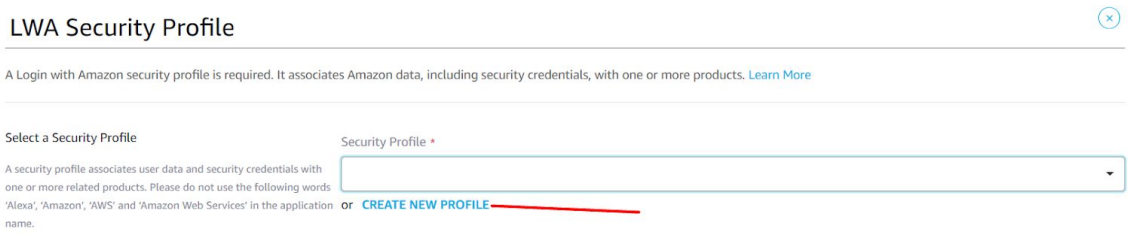
- Yes
- No

Product category *

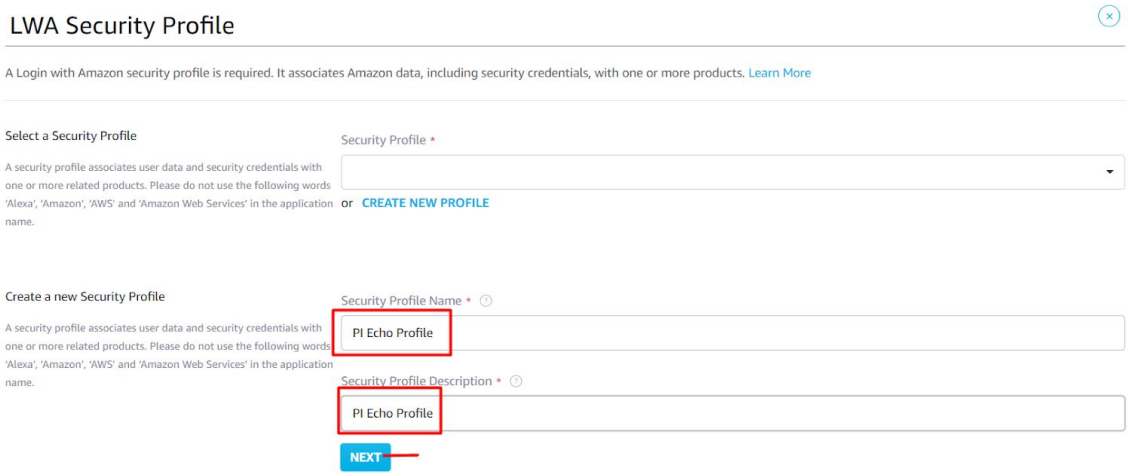
Computing



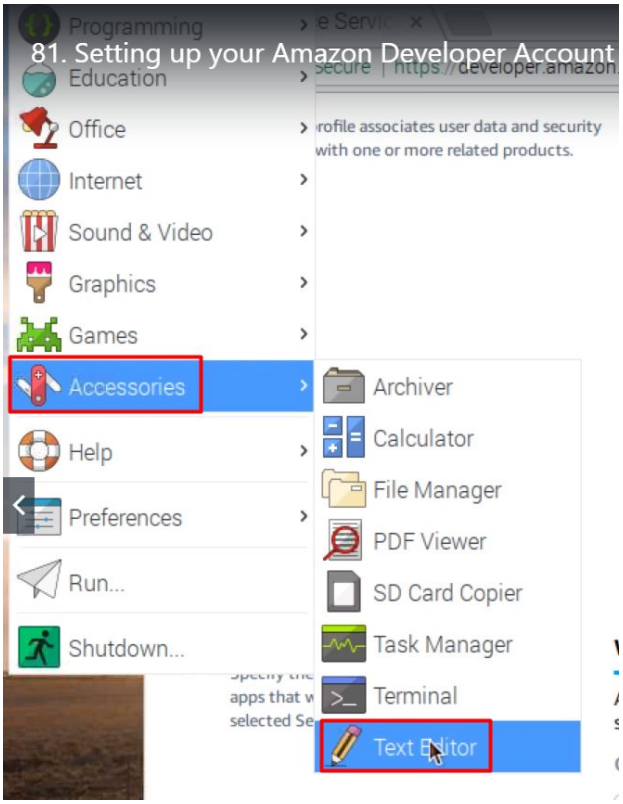
8) NEXT
9) Create a New profile:



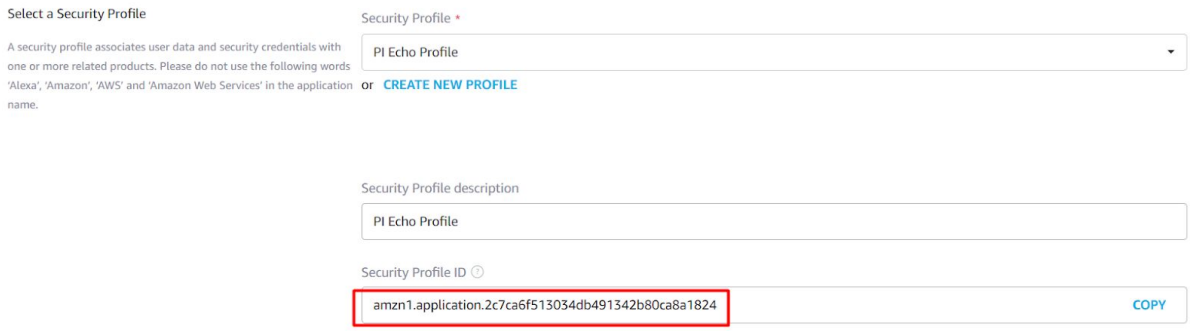
10) Fill the fields below:



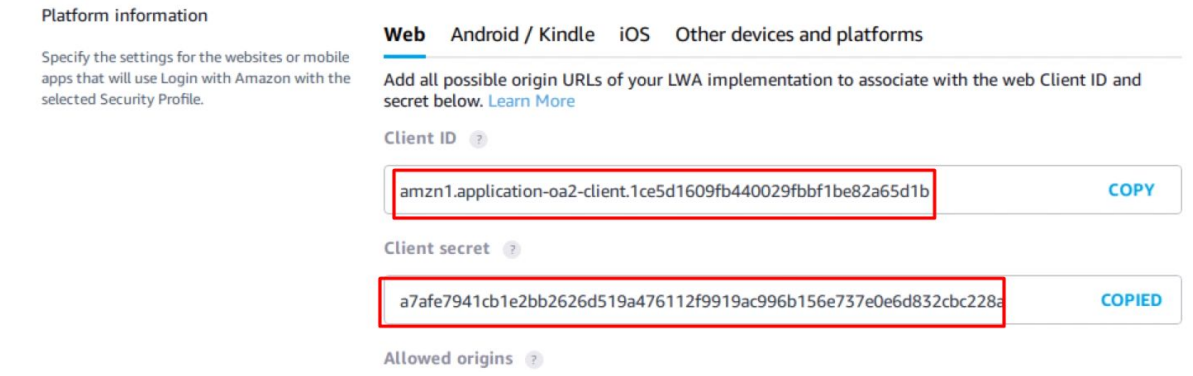
11) Go to the Raspberry Pi Screen=> Accessories=> Text Editor:



12) Copy the information below and paste in the Text Editor:

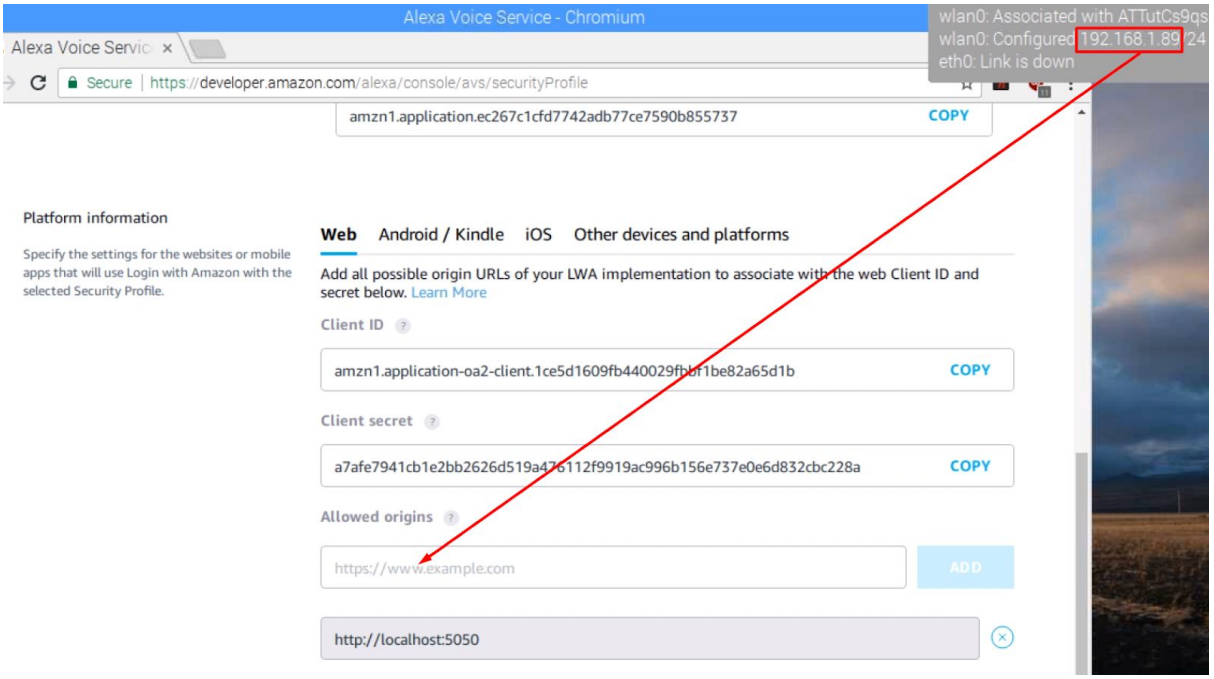


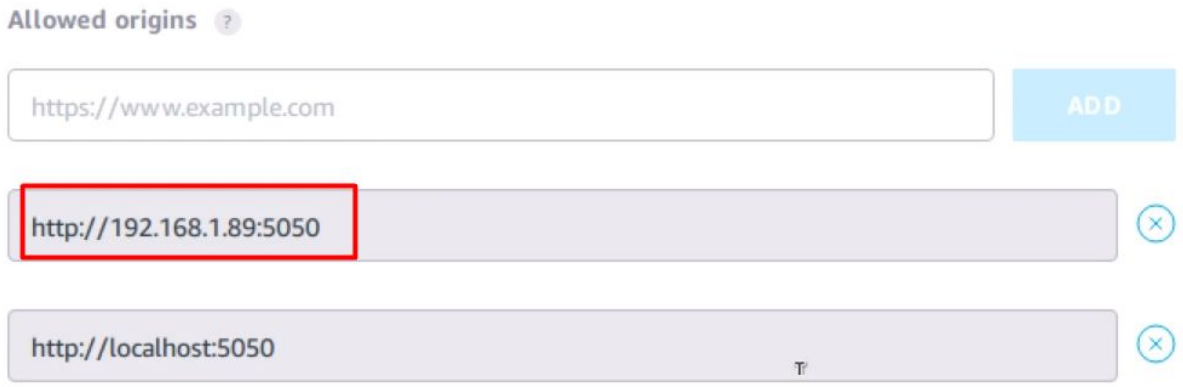
13) Copy the information below and paste in the Text Editor:



14) Add in Allowed Origins: http://localhost:5050

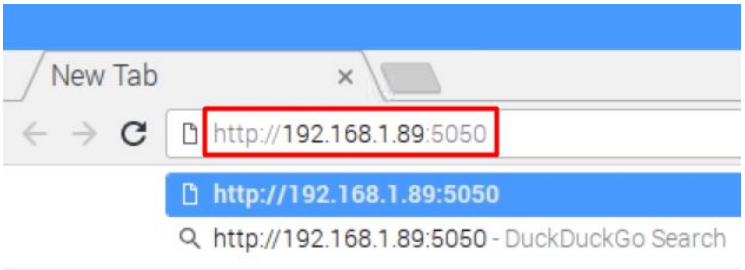
15) Add the IP Address of the Raspberry PI (you can check at the top) in the same field: http://IPADDRESS:5050
Example: http://192.168.1.89:5050





16) Add in the Allowed return URLs: http://localhost:5050/code
17) Add the IP Address of the Raspberry PI (you can check at the top) in the same field: http://IPADDRESS:5050/code
Example: http://192.168.1.89:5050/code

18) Go to the Raspberry Terminal and type: sudo apt-get update
19) sudo apt-get dist upgrade
20) sudo apt-get install git
21) sudo pip install CherryPy==17.4.0
22) cd /opt
23) sudo git clone https://github.com/alexa-pi/AlexaPi.git
24) sudo ./AlexaPi/src/scripts/setup.sh
25) Keep hitting ENTER
26) Airplay Support=> Type: n
27) Enter your Device Type ID=> Pi_Echo
28) Security Profile Description=> Code from Step 12
29) Client ID=> Code from Step 13
30) Client Secret=> Code from Step 13
31) Go to the Raspberry Browser: http://IPADDRESS:5050
Example: http://192.168.1.89:5050



32) Login in your account
33) After the Success Message, close your browser
34) Go back to the terminal and press: CTRL+C

35) sudo systemctl start AlexaPi.service
36) s
37)

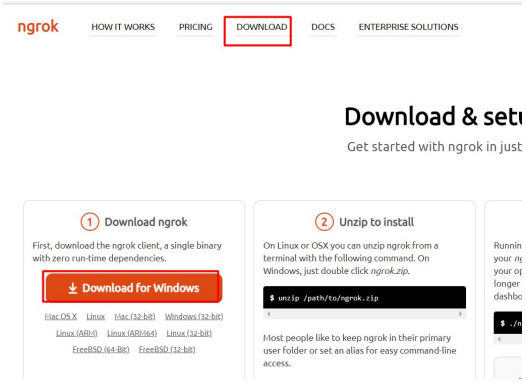
Custom Skill

You need four things: Intents, Utterances, Slots, Dialog and a Skill Handler

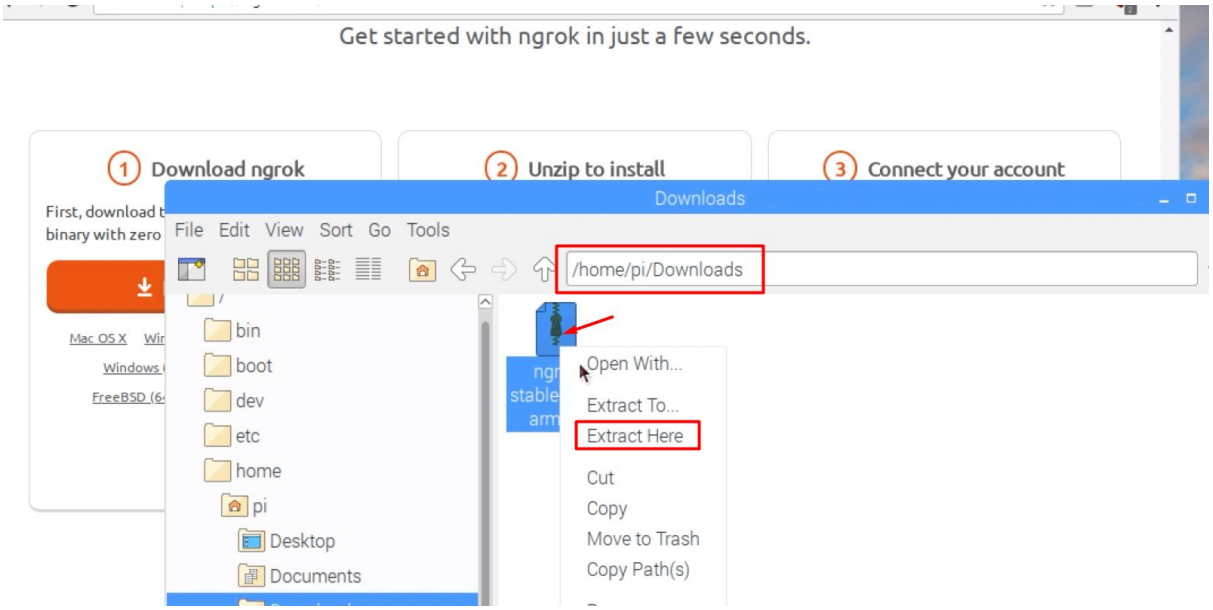
1) Intent: Fulfill something the user want to do. Example: book a trip
2) Utterances: spoken phrases of the user to give orders and execute the intents
3) Slot: Values to fulfill the intent. Example: toppings of the pizza
4) Dialog: multi-turn conversation
5) Skill Handler: code to process the intent

Flash Disk (Library to create Alexa Skills) and Ngrok (Public Availability)

1) Update the Raspberry PI: Click on this link
2) Update pip:
   sudo -H pip3 install --upgrade pip
3) sudo pip3 install flask-ask
4) Go to the Raspberry PI Chromium: ngrok.com
   This website allows to expose a public URL to the Raspberry Pi. This is how the skill is going to be available in Alexa
5) Go to download:



6) Go to the folder where you downloaded and click Extract Here

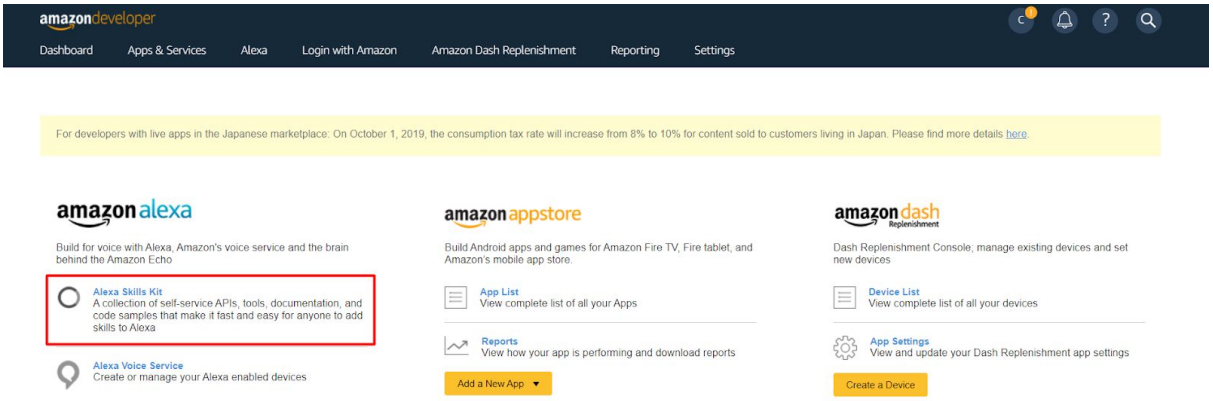


Optional: Get the Code from Git hub

7) cd /Downloads
8) For the Relay: sudo git clone https://github.com/leeassam/alexa-rpi-skills.git
9) For the TV: sudo git clone https://github.com/leeassam/alexa-rpi-tv-control.git

Creating a New Skill

1) Go to: https://developer.amazon.com/dashboard
2) Click in Skills



Project Overview

Hello World Project

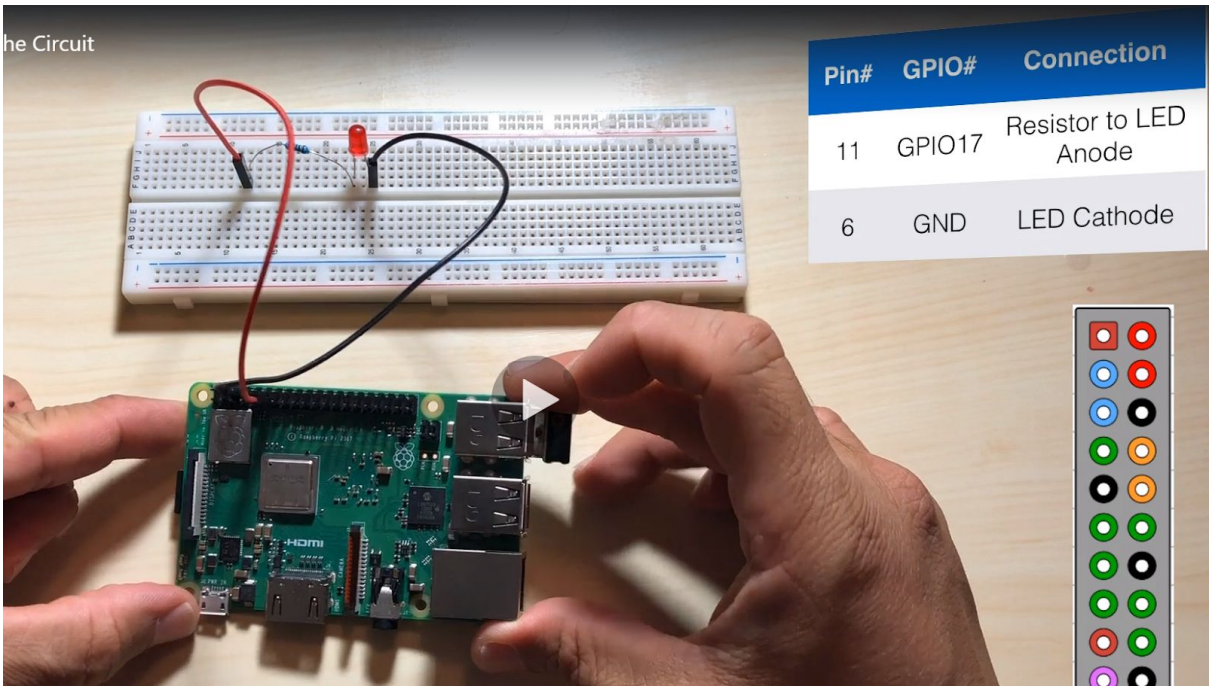
- Amazon Developer Portal Skill Setup
  - Intent
    - HelloWorldIntent
  - Slots
    - firstname
  - Utterances
    - say hello
    - say hi to firstname
    - say hello to firstname
- Skill Handler Code
  - hello-world.py
  - templates.yaml

Off

LED turning ON and OFF Project

Overview of the MyLED Skill

- Amazon Developer Portal Skill Setup
  - Intent
    - OnOffIntent
  - Slots
    - OnOff
  - Utterances
    - switch on/off
    - turn on/off
- Skill Handler Code
  - led-control.py
  - templates.yaml



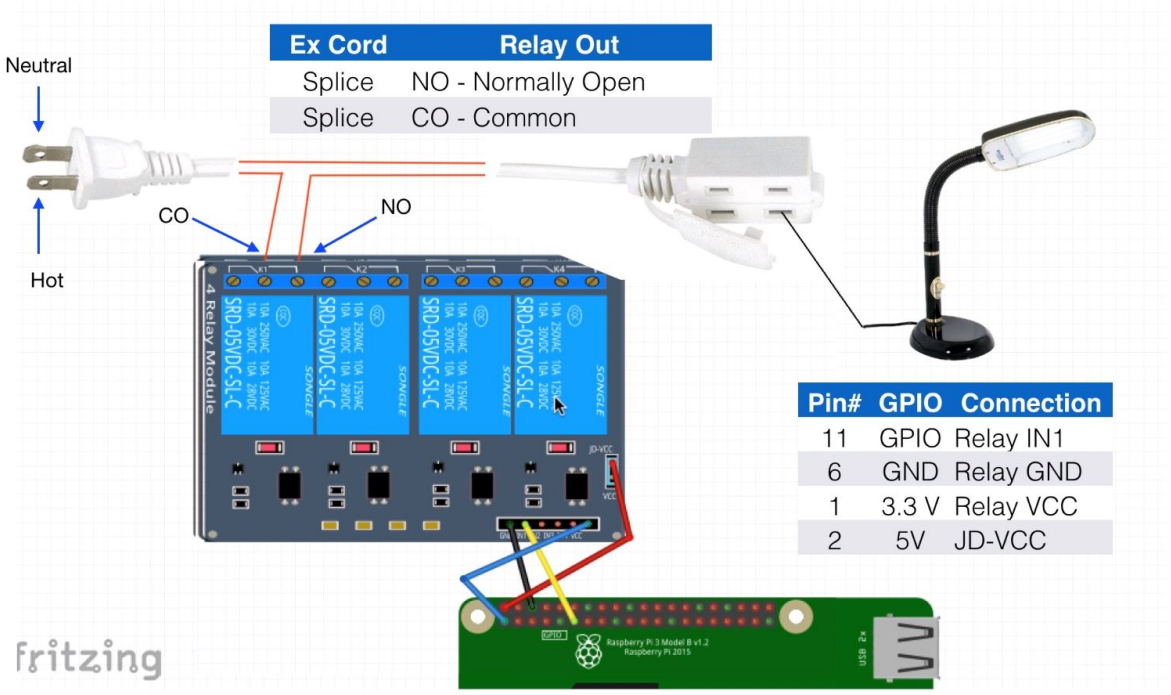
Relay Control Project

Overview of the MyDevice Skill

- Amazon Developer Portal Skill Setup
  - Intent
    - OnOffIntent
  - Slots
    - OnOff
  - Utterances
    - switch on/off
    - turn on/off
- Skill Handler Code
  - relay-control.py
  - templates.yaml

"Alexa, tell my device to turn on"

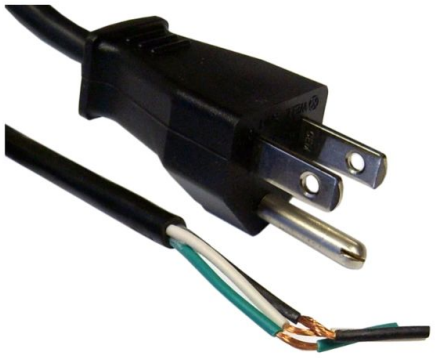
Relay Circuit



Ex Cord	Relay Out
Splice	NO - Normally Open
Splice	CO - Common

Pin#	GPIO	Connection
11	GPIO	Relay IN1
6	GND	Relay GND
1	3.3 V	Relay VCC
2	5V	JD-VCC

Three prong plug

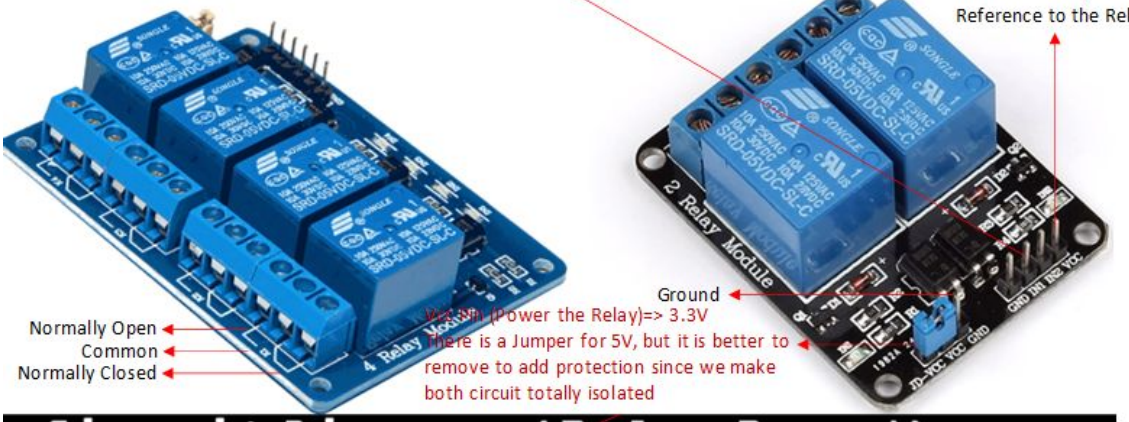


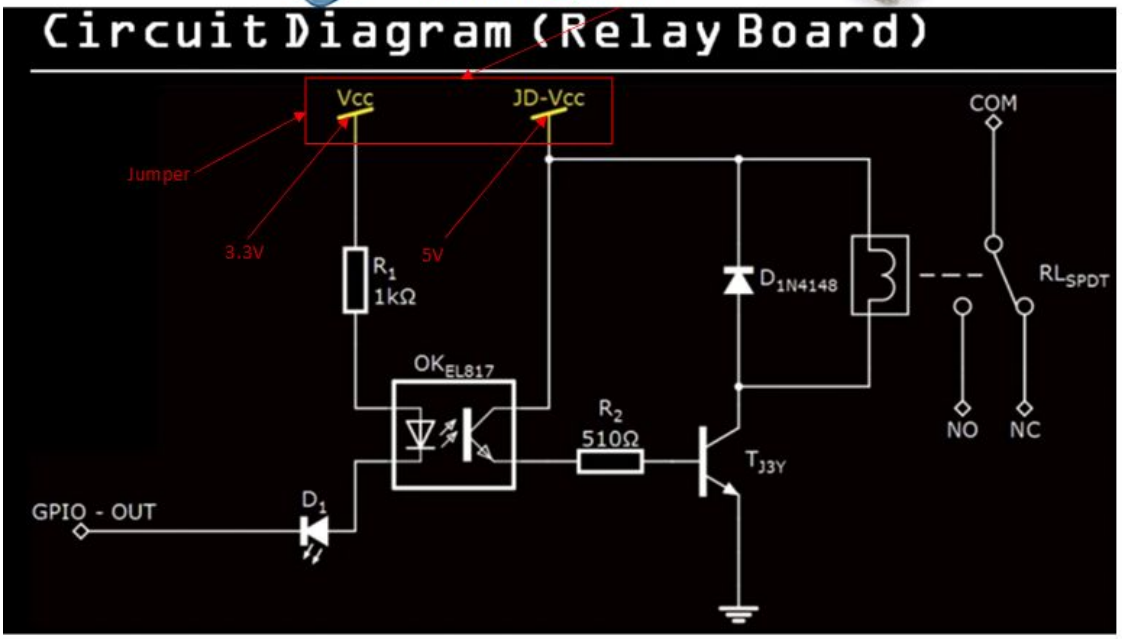
Splice Hot (Black Wire) to Relay

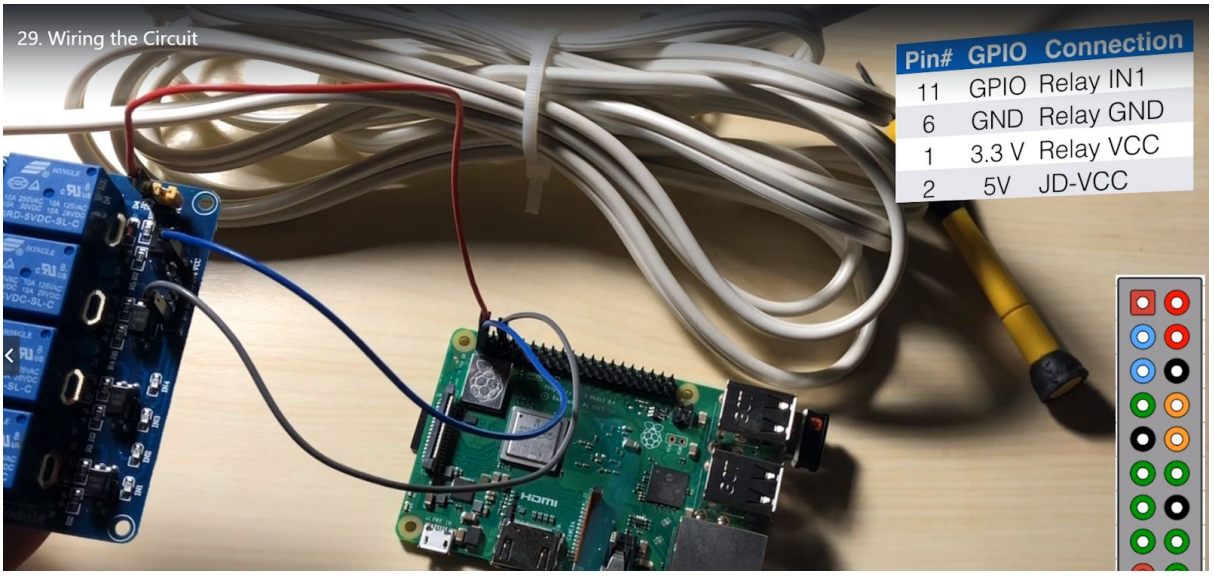
In the US and Canada

Relay Modules

IN1=> Control Relay 1
IN2=> Control Relay 2
Relays are Active Low=>
- When you apply a Low Signal IN1 or IN2=> You energize the Relays and change the Switch position, NO switches to close while NC switches to Open

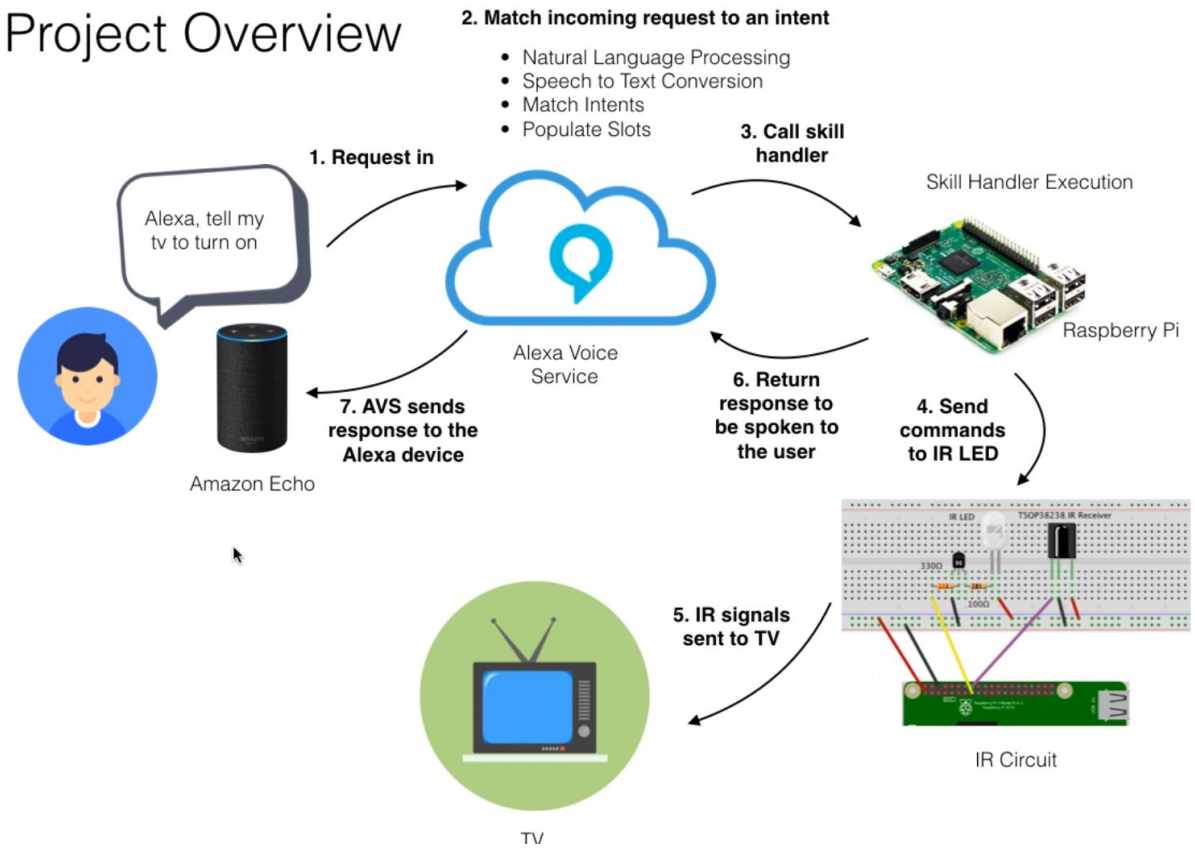




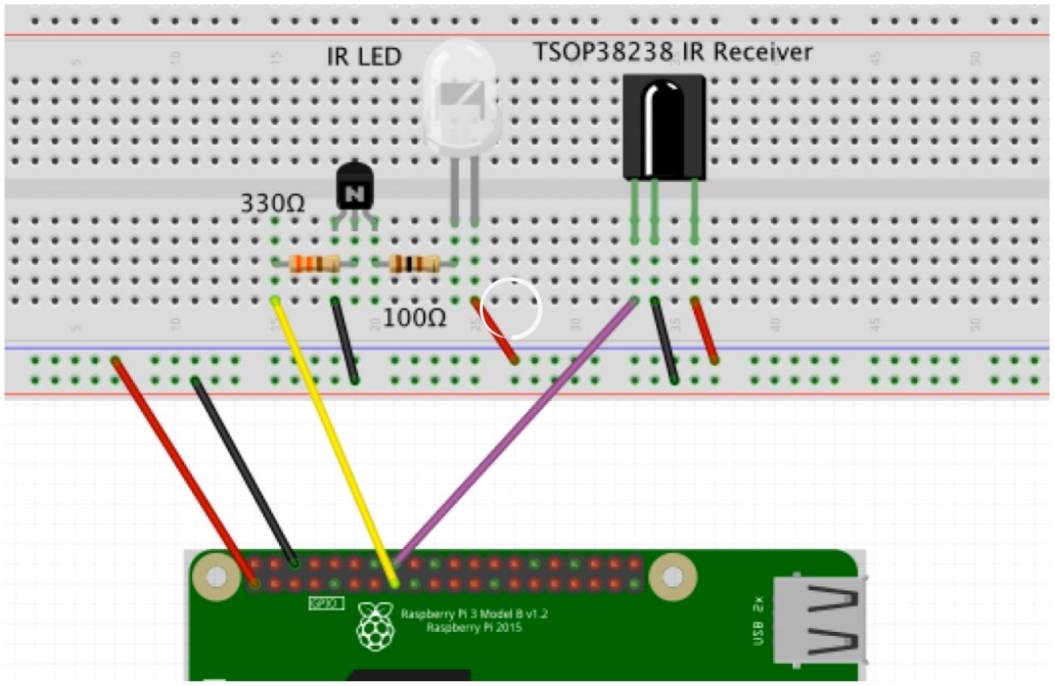


TV Voice Control Project

Material: Breadboard, Infrared LED, Infrared Receiver, NPN Transistor, Connecting Wires, Raspberry Pi and its associated peripherals (SD card, Power Adapter, HDMI cable etc.)



Circuit Diagram



NPN Transistor

Pin	Connection
Base	Resistor to GPIO22
Collector	Resistor to Cathode IR LED
Emitter	GND

Raspberry Pi

Pin#	GPIO#	Connection
1	3.3V	Power Rail
6	GND	Ground Rail
15	GPIO22	Resistor to NPN Base
16	GPIO23	OUT IR Receiver

IR Receiver

Pin	Connection
Vs	3.3V
GND	GND
OUT	GPIO23

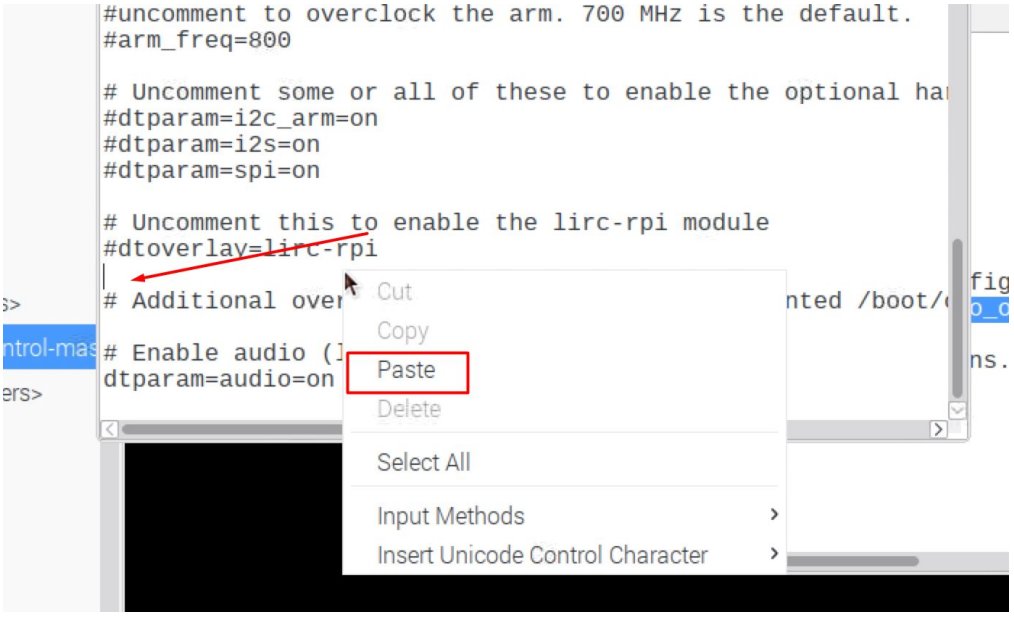
IR LED

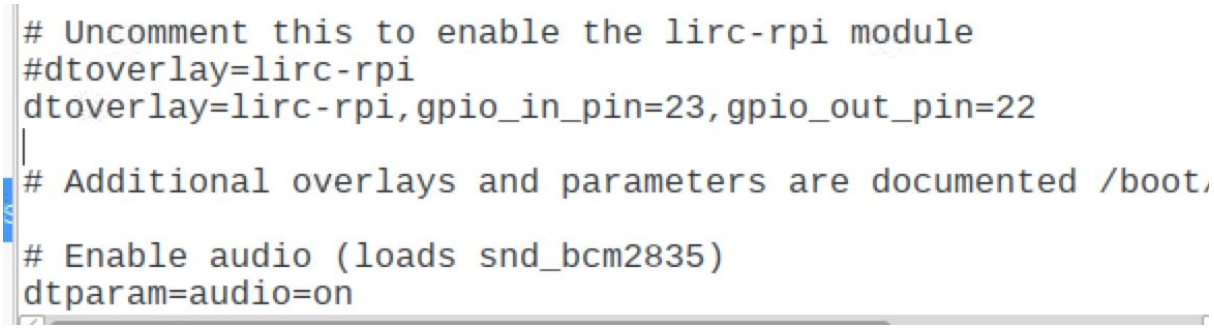
Pin	Connection
Anode	3.3V
Cathode	Resistor to NPN Collector

1) For the TV Voice Control, the LIRC (Linux Infrared Remote Control) library is going to be needed:

https://www.lirc.org/

2) Go to the Terminal and update everything Click on this link)
3) sudo apt-get install lirc
4) Y
5) Get the files from github: https://github.com/leeassam/alexa-rpi-tv-control
6) sudo leafpad /boot/config.txt
7) Go to the config.txt and insert dtoverlay=lirc-rpi,gpio_in_pin=23,gpio_out_pin=22 in the position below:

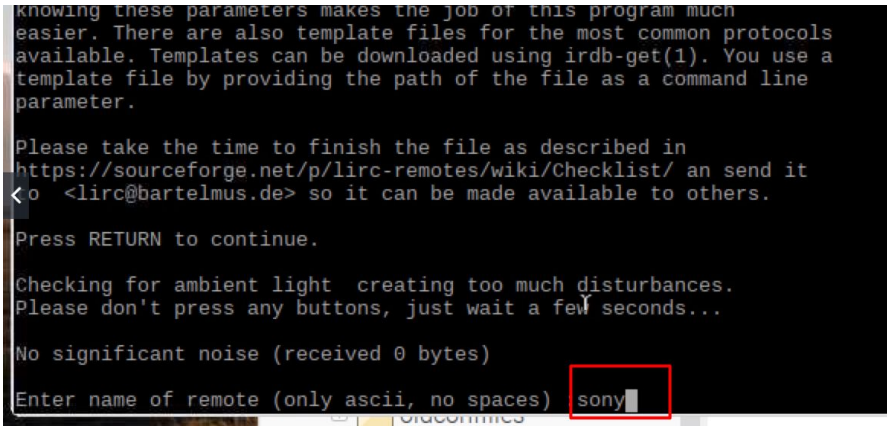




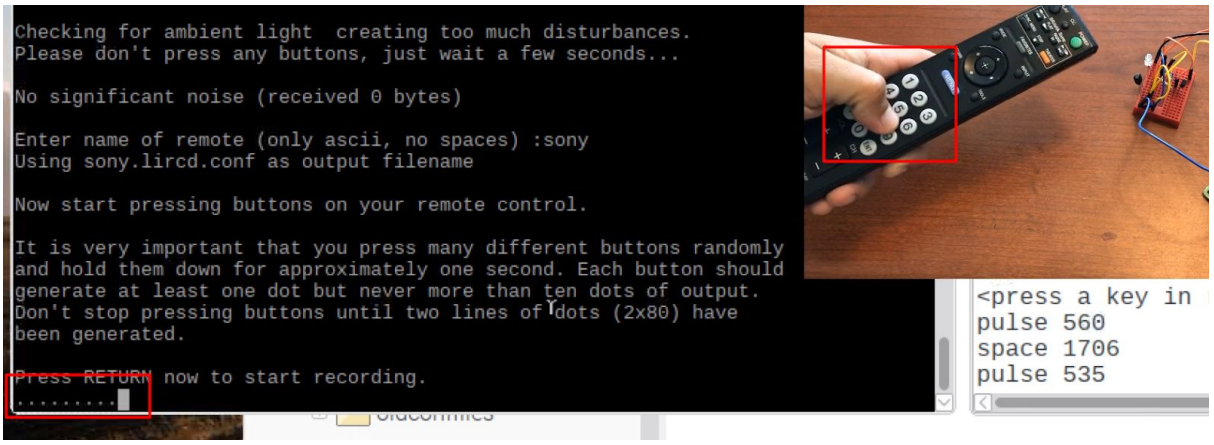
8) sudo leafpad /etc/lirc/lirc_options.conf
9) Change the driver to default and device to /dev/lirc0

```
[lircd]
nodaemon        = False
driver          = default
device          = /dev/lirc0
output          = /var/run/lirc/lirc
pidfile         = /var/run/lirc/lirc
plugindir       = /usr/lib/arm-linux
permission      = 666
allow-simulate  = No
```

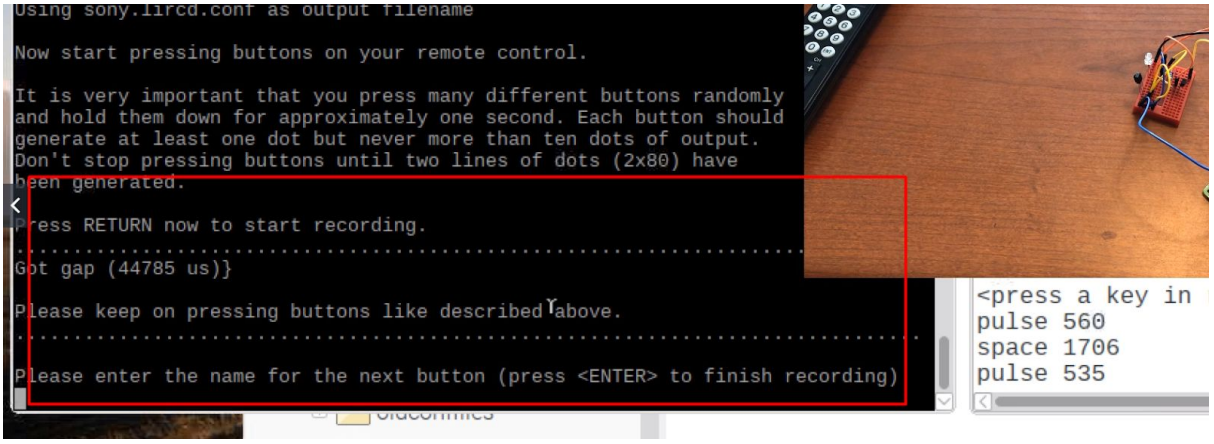
10) sudo reboot
11) To Check the Status of the LIRC: sudo /etc/init.d/lircd status
12) To Start the LIRC: sudo /etc/init.d/lircd start
13) To Stop the LIRC: sudo /etc/init.d/lircd stop
14) Make sure that the LIRC is NOT RUNNING: sudo /etc/init.d/lircd stop
15) mode2 -d dev/lirc0
16) If you get your remote controller from your TV and press a button, this pulse would be recorded.
17) CTRL + C to stop everything
18) sudo irrecord -d /dev/lirc0 ~/lircd.conf
19) ENTER
20) After checking that there is no Noise, insert a name (for example the name of the TV):



21) Press many buttons in your remote controller and see if it is working:



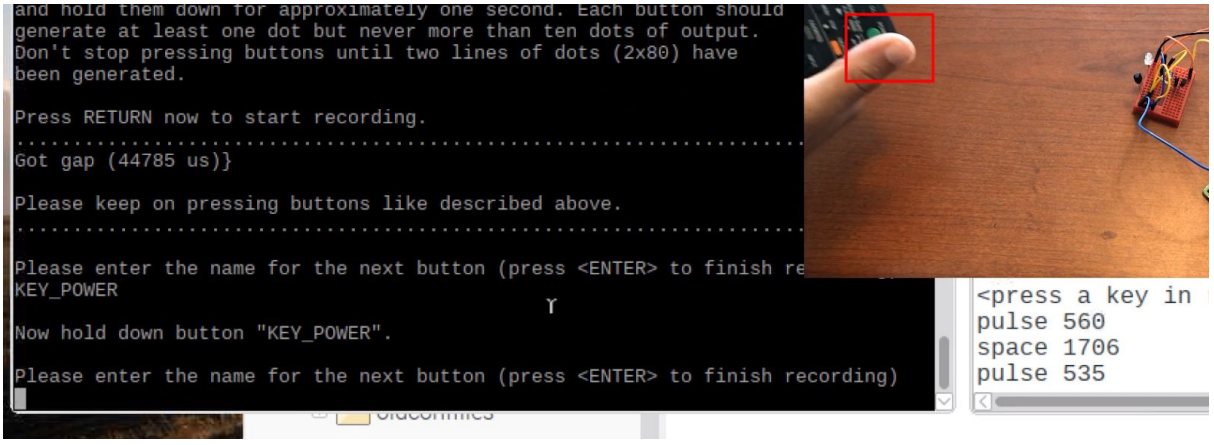
22) Keep pressing buttons until the below appears



23) You can check all the keys available in:
https://gist.github.com/unforgiven512/0c232f4112b63021a8e0df6eedfb2ff3

24) Type: KEY_POWER

25) Hold the Power Key on the Remote Controller to record it



26) Type: KEY_VOLUMEUP

27) Hold the Volume Up Key on the Remote Controller to record it

28) Type: KEY_VOLUMEDOWN

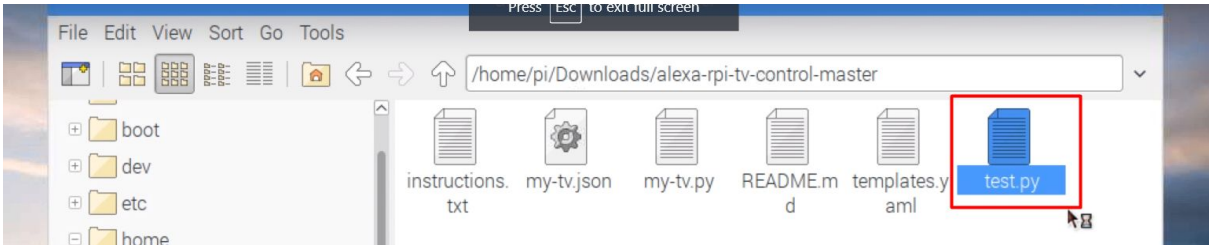
29) Hold the Volume Down Key on the Remote Controller to record it

30) Type: KEY_MUTE

31) Hold the MUTE Key on the Remote Controller to record it

32) Type: KEY_CHANNELUP

33) Hold the Channel Up Key on the Remote Controller to record it
34) Type: KEY_CHANNELDOWN
35) Hold the Channel Down Key on the Remote Controller to record it
36) Put all the Remote Controller Numbers there:
37) Type: KEY_0
38) Hold the 0 Key on the Remote Controller to record it
39) At the end press Enter
40) Press any button as quickly as possible
41) Make a backup of the file: sudo mv /etc/lirc/lircd.conf /etc/lirc/lircd_original.conf
42) ls
43) sudo cp ~/**device-name**.lircd.conf /etc/lirc/lircd.conf
The device name is the one you created in Step 20
44) Go to the Test.py file and change the device name to the same one you chose in Step 20:



Code:

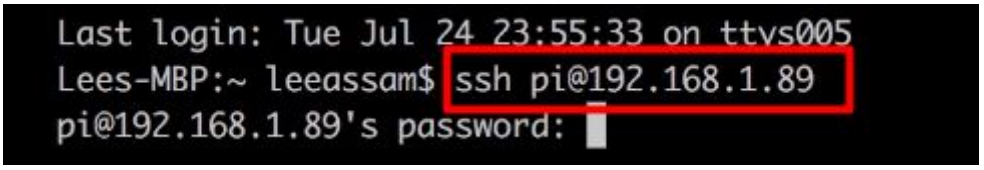
```
import argparse, os, time

'''
-- Program for quickly testing your remote --
#Command line arguments
#Usage: python3 test.py power on
#Usage: python3 test.py volume increase

#Turning your tv on or off
power on
power off

#Adjusting your volume
```

```
volume increase
volume decrease
volume mute
volume unmute

#Switching to a specific channel
channel 35

#Channel Surfing
surf up
surf down

'''

#Specify the name of your TV that was configured in the lircd.conf file
device = "sony"

#getting the command line arguments
parser = argparse.ArgumentParser()
parser.add_argument("control", help="What you want to control [volume | power | channel |
surf]")
parser.add_argument("action", help="How you want to adjust the control [increase |
decrease]")
args = parser.parse_args()

#Extracting the arguments
control = args.control
action = args.action

#Debug
print(control)
print(action)

#sends a repeating command
#see the irsend command in lirc documentation http://www.lirc.org/html/irsend.html
#command : The IR command to execute
#duration : The duration in seconds to sleep/pause while the command is being repeated
def sendRepeatCommand(command, duration):
    os.system("irsend SEND_START {} {} ; sleep {}".format(device, command, duration))
    os.system("irsend SEND_STOP {} {}".format(device, command))

#adjust power
if control == "power":
    os.system("irsend --count=2 SEND_ONCE %s KEY_POWER KEY_POWER" % device)

#adjust volume
```

```
elif control == "volume":
    if action == "increase":
        sendRepeatCommand("KEY_VOLUMEUP", 3)
    elif action == "decrease":
        sendRepeatCommand("KEY_VOLUMEDOWN", 3)
    elif action == "mute" or action == "unmute":
        sendRepeatCommand("KEY_MUTE", 0.1)

#switch to a specific channel
elif control == "channel":
    channelNums = list(str(action))
    for i in channelNums:
        sendRepeatCommand("KEY_" + i, 0.1)
        time.sleep(0.5)

#channel surf
elif control == "surf":
    if action == "up":
        sendRepeatCommand("KEY_CHANNELUP", 0.1);
    elif action == "down":
        sendRepeatCommand("KEY_CHANNELDOWN", 0.1);
    elif action == "stop":
        #exit
        print("Stopping")
```

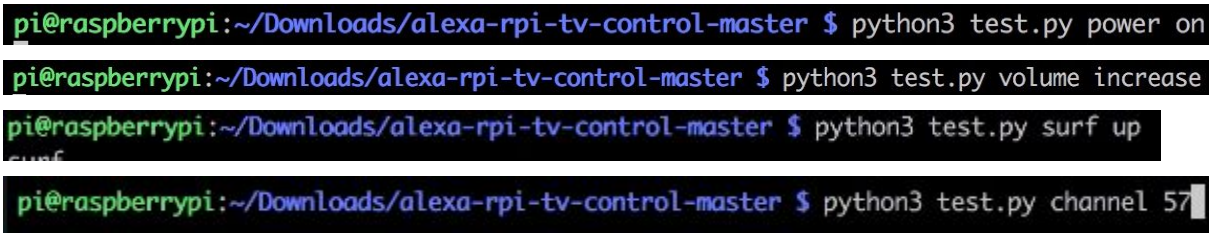
45) Run the Test.py program

46) Access your raspberry pi using ssh



47) Go to the project folder:

48) Select the program and the action

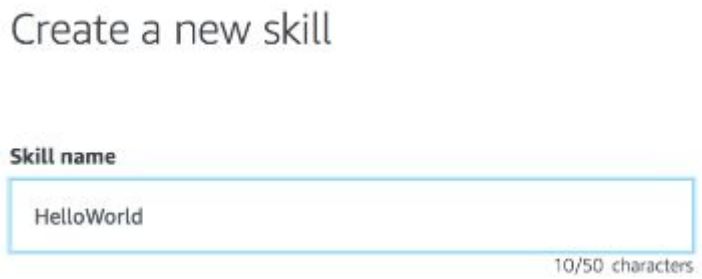


49)

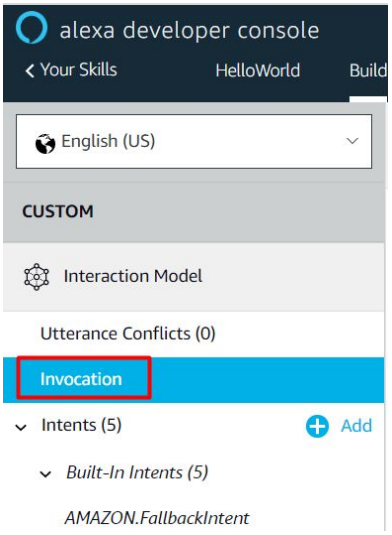
Creating a Skill

3) Create Skill=> Put the Skill Name and choose Custom=>Start from Scratch

For Hello World: HelloWorld
For LED: myLED
For Relay:: myDevice
For TV:: myTV

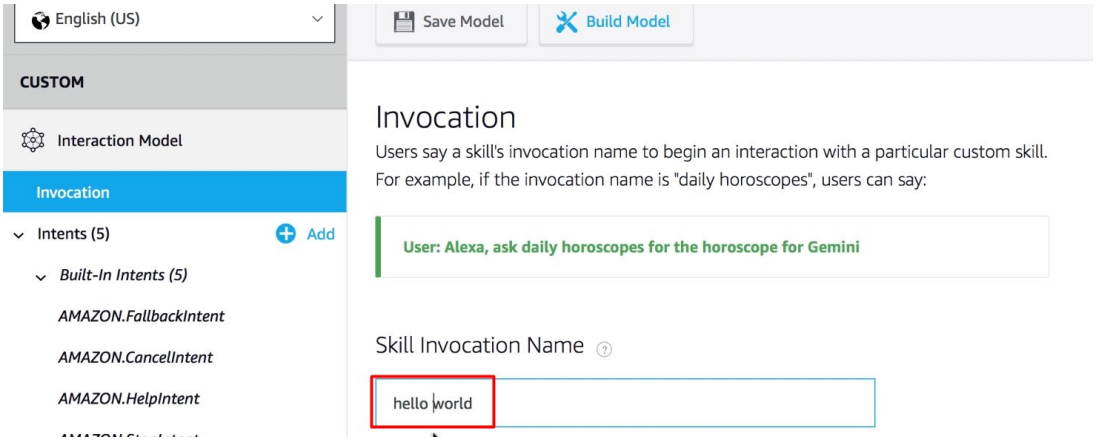


4) Go to Invocation

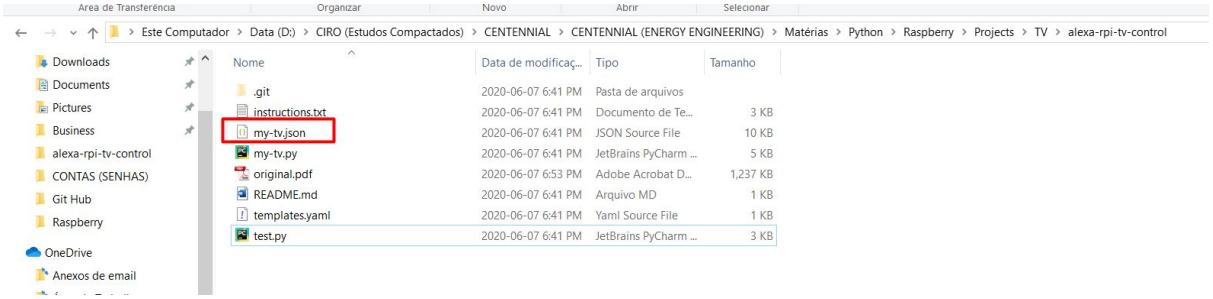


5) The invocation is how we will tell Alexa to get the skill. Choose a Name:

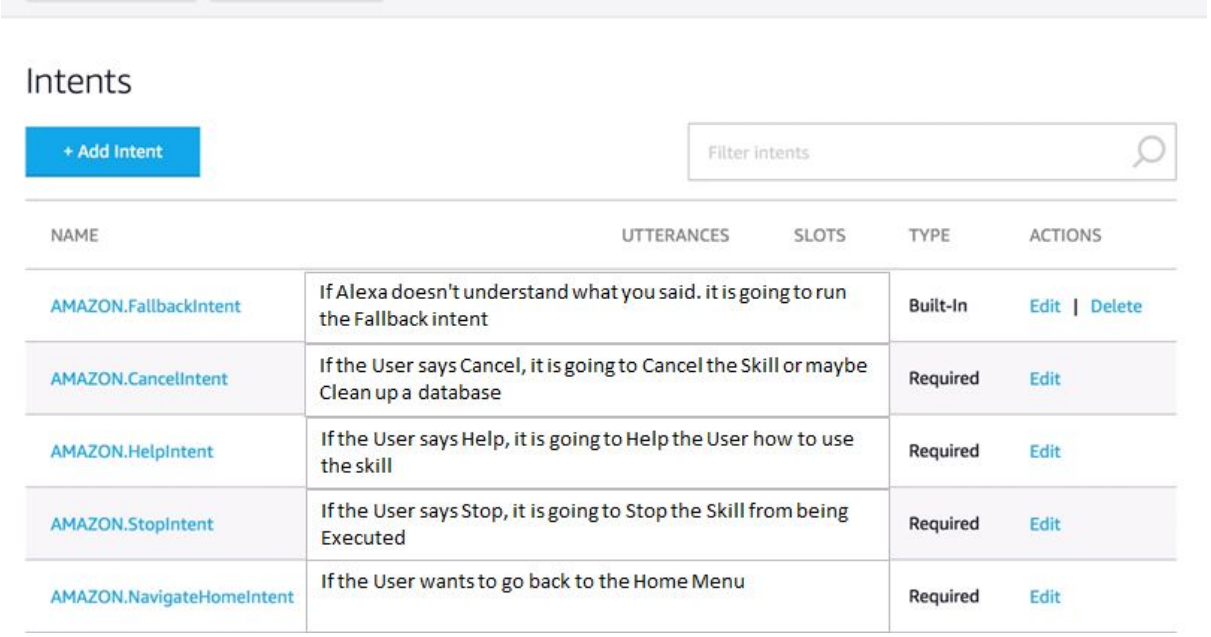
For Hello World: hello world
For LED: my project
For Relay:: my device



For TV: Go to JSON and copy all the contents from the JSON.file
YOU CAN JUMP FROM HERE TO THE STEP 24



6) Go to Intents (The Description of the Intents are shown below)



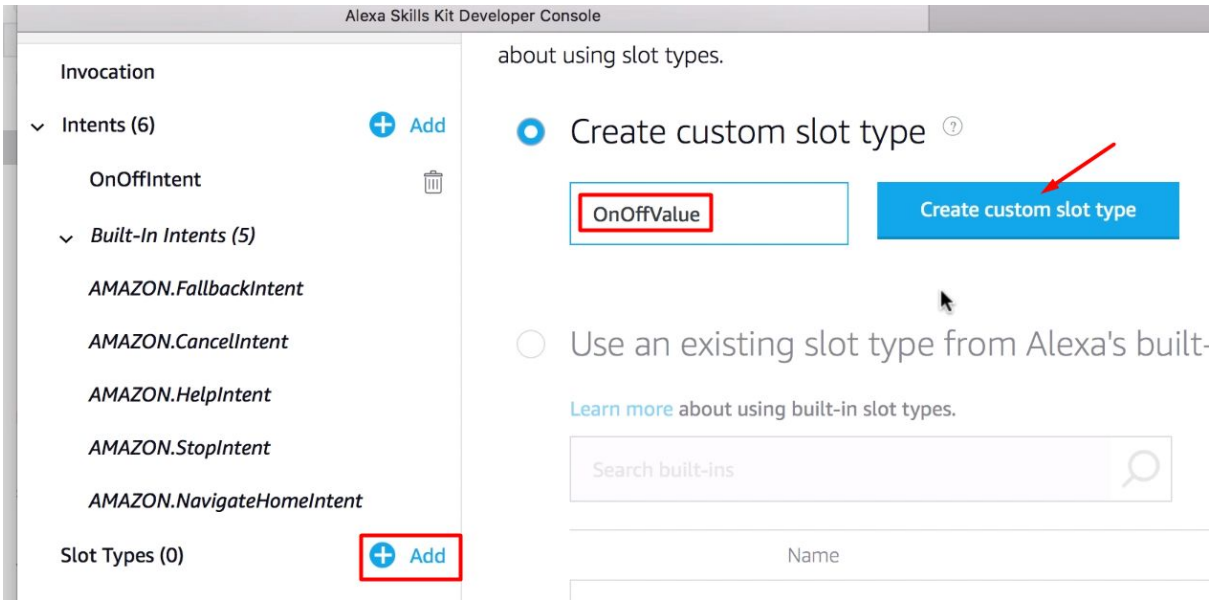
7) Add Intent
8) Put the name of the intent.

For Hello World: HelloWorldIntent
For LED and Relay: OnOffIntent

Creating a Slot Type (Some projects don't need like Hello World that already has a built-in firstname Slot Type)

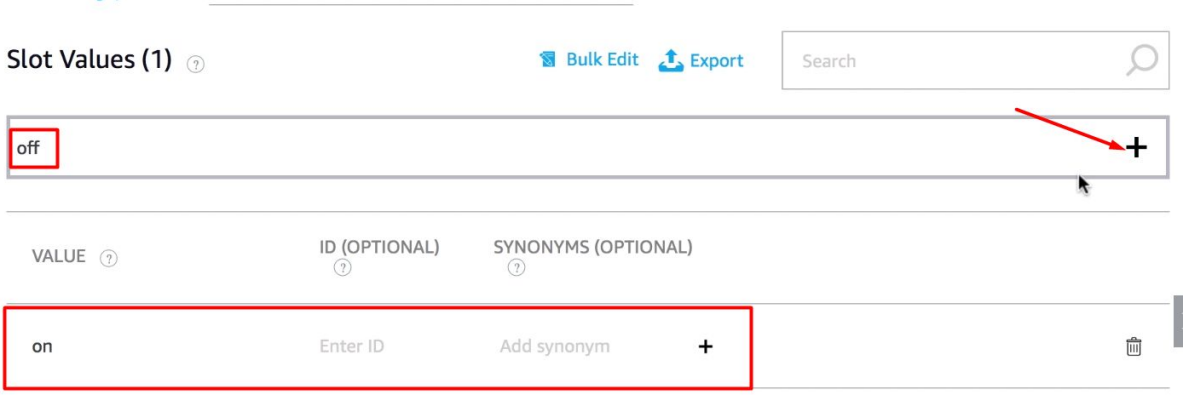
9) Create the Slot Type name:

For LED and Relay: OnOffValue

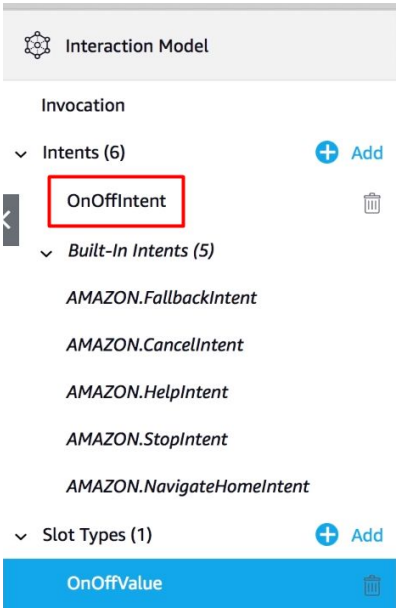


10) Create the Slot Values:

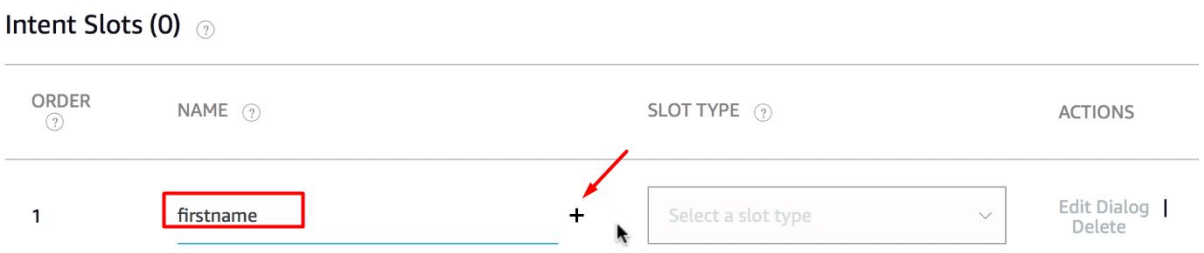
For LED and Relay: On / Off



11) Go back to the Intent



12) Scroll down and Create the Slots

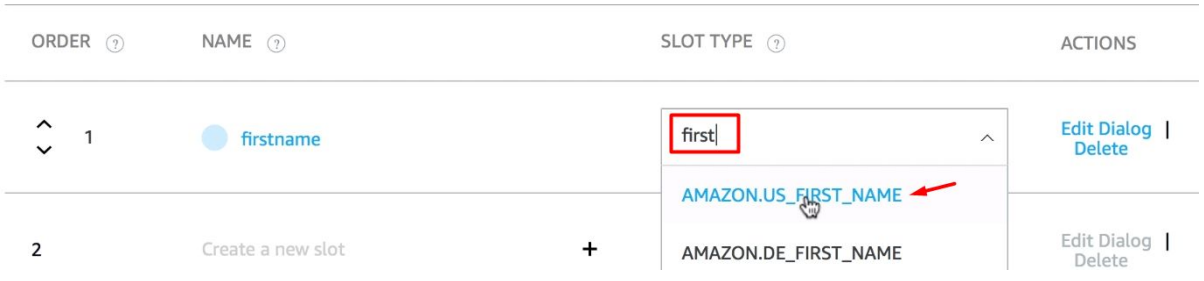


Linking the Slots with the Intent

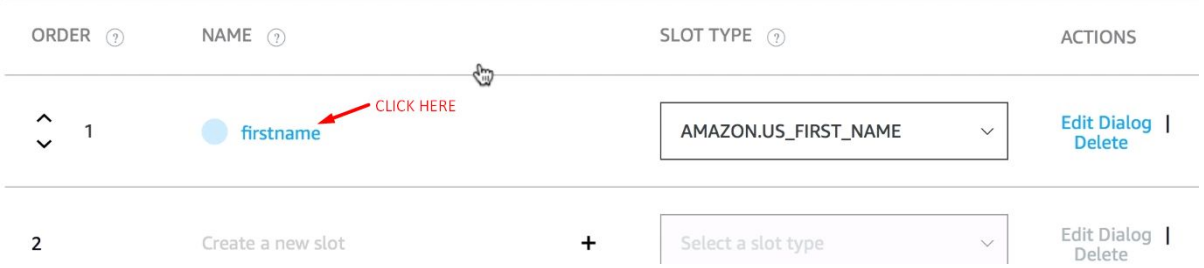
13) Put a Name / Slot Type:

For Hello World: firstname / AMAZON.US_FIRST_NAME

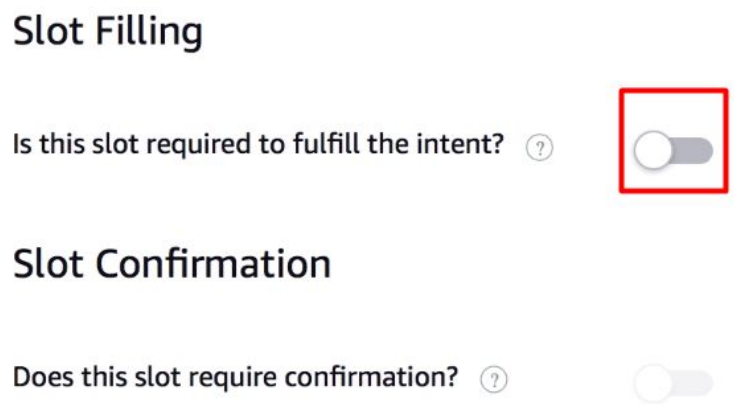
For LED and Relay: OnOff / OnOffValue



14) Click in the Name:



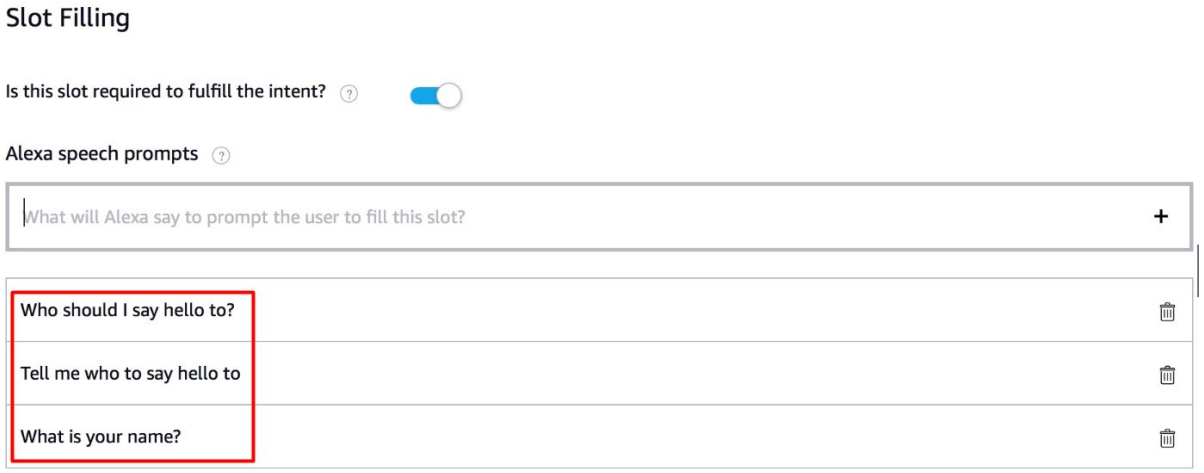
15) Turn on the Slot Requirement if, that slot is required for the Skill



16) Put multiple ways that Alexa can ask for that information:

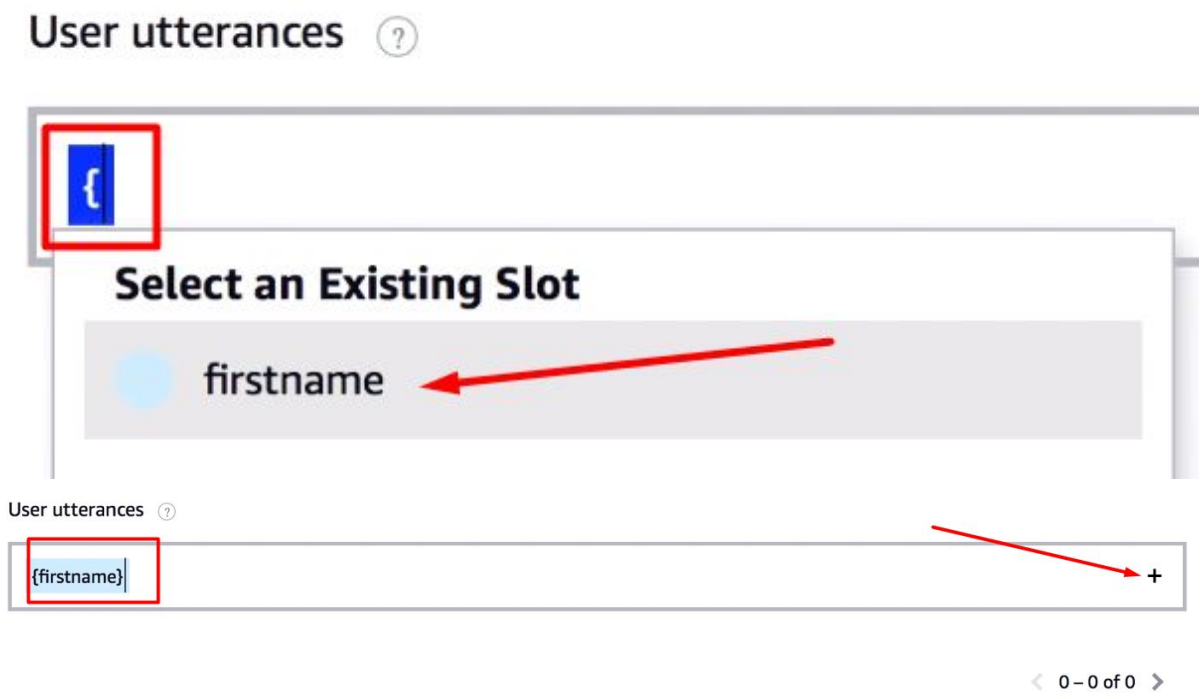
For Hello World: What is your name?
For LED: Do you want to turn your LED on or off?
For Relay: Do you want to turn your device on or off?

17) You can add multiple ways.



18) Scroll down and go to Utterances=> Type { so you will have access to the Slot Name
The Utterance is mentioning to Alexa that the user response is going to be the

For Hello World: {firstname}
For LED and Relay: {OnOff}



19) Go back to the previous Screen:

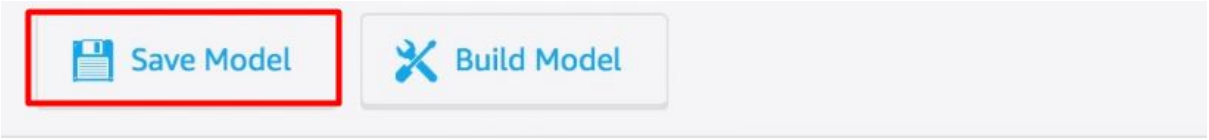
20) Go Sample Utterances:

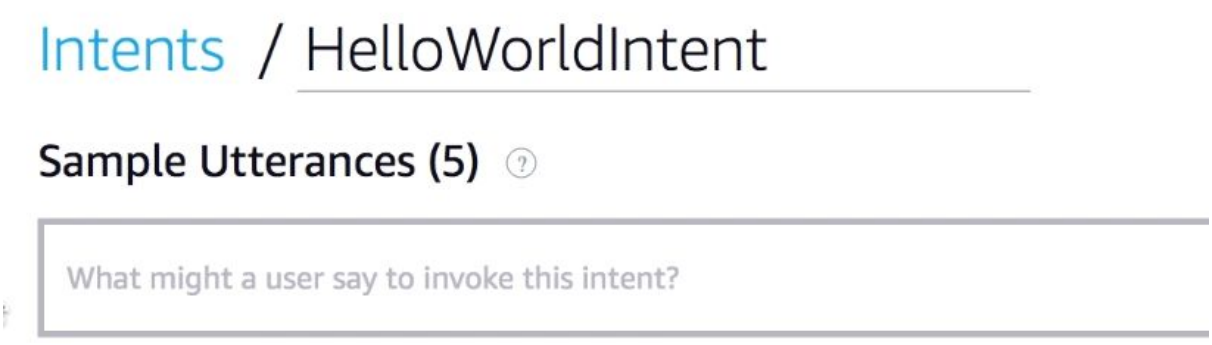
Configure in Alexa how the User is going to request to use the Skill

For Hello World: say hi to {firstname} / say hi

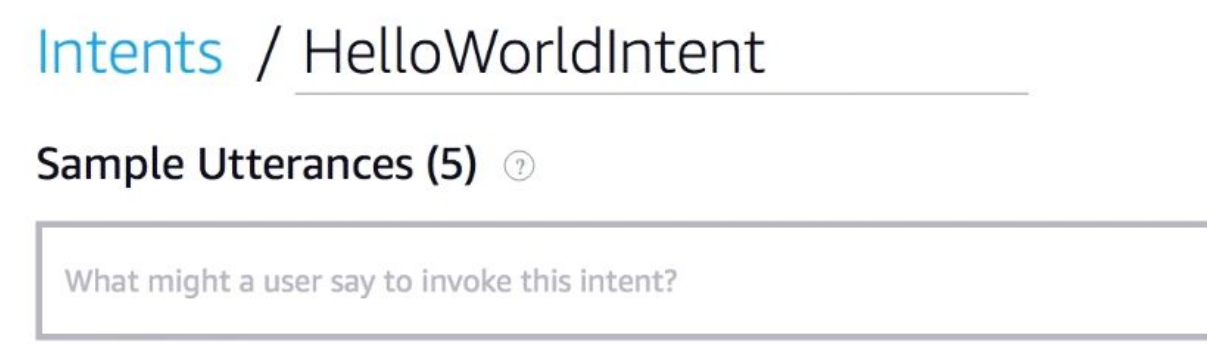
For LED and Relay: switch {OnOff} / {OnOff} / turn {OnOff}

21) Save the Model

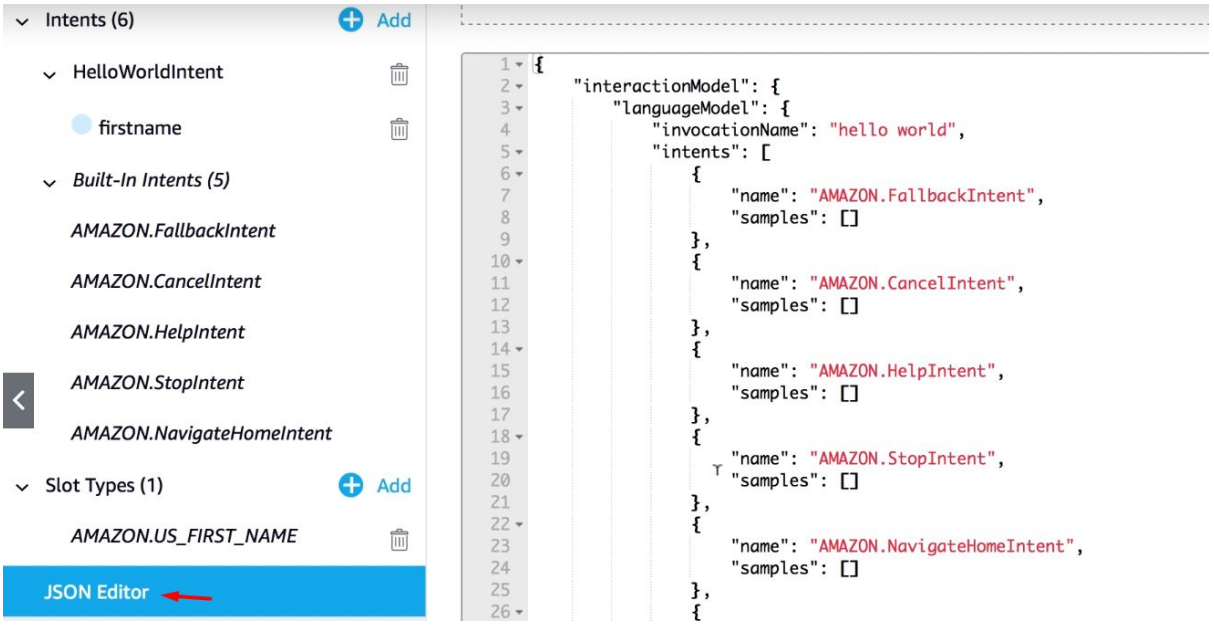




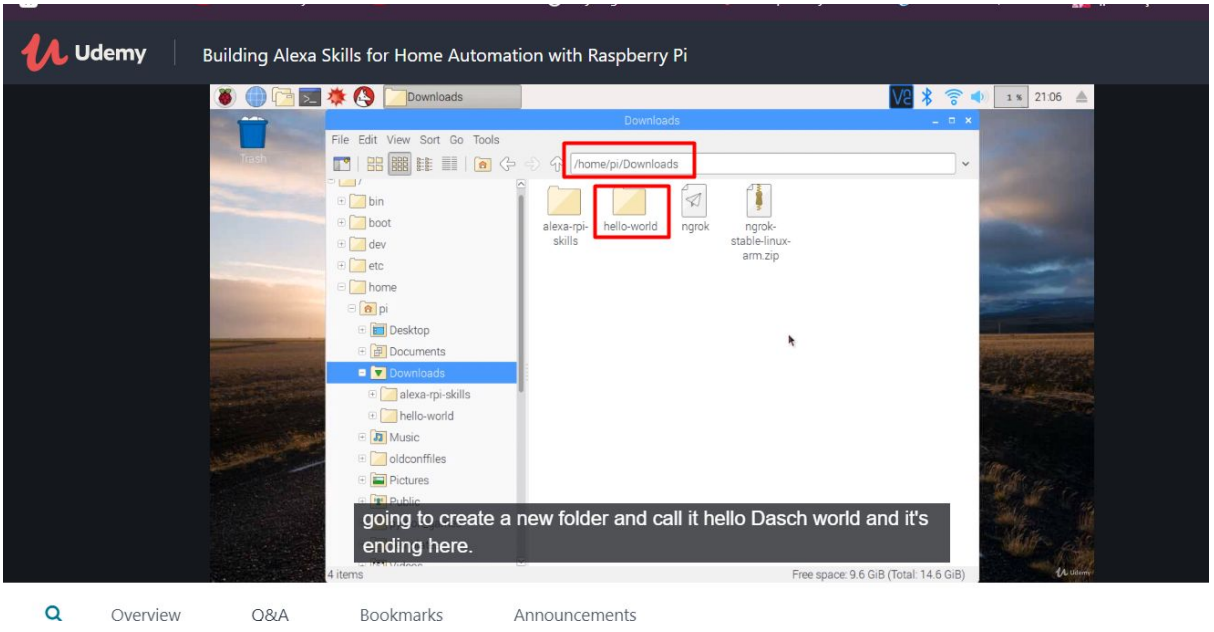
22) Build the Model



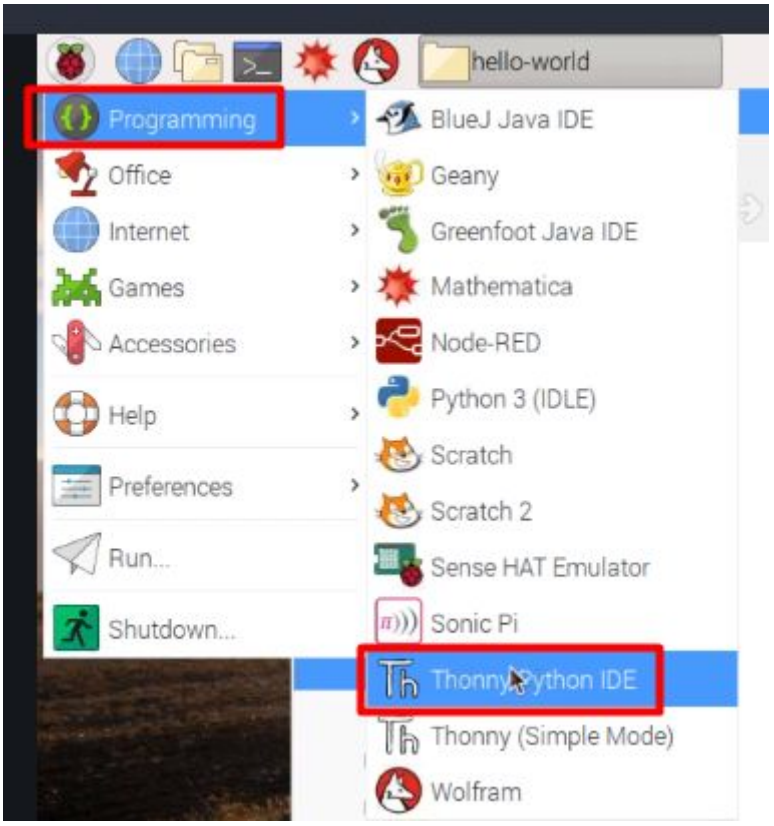
23) The JSON gives all the code for this Skill. So the only thing the person needs to do is to copy the skill there. To copy: CTRL+A and CTRL+C



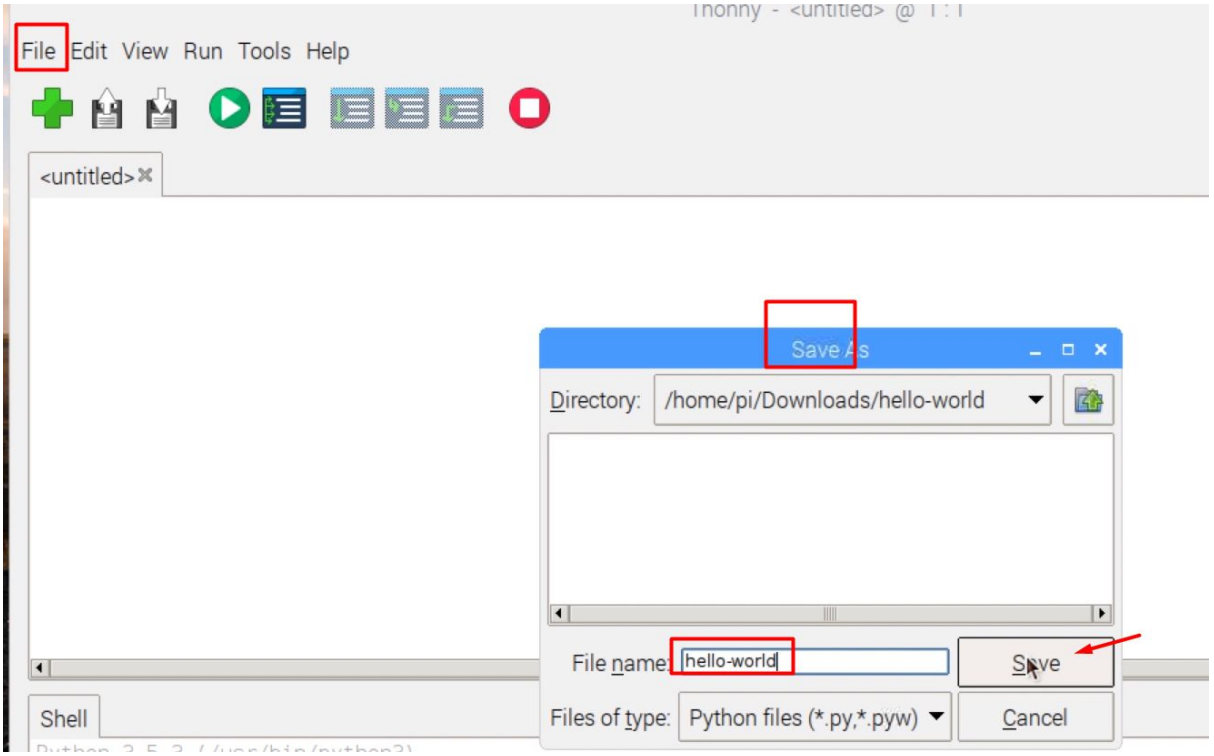
24) Go to the Terminal
25) Go to Downloads
    a) Two files are going to be created one for when the user calls the skill and the python file is going to map the skill in Alexa
    b) Then Alexa is going to call the end point which would be the Python program in the Raspberry PI
    c) First File: The Raspberry PI is going to receive the message and send a response back to the user
    d) Second File: It is going to store all the strings that Python uses when interacting with the User
26) Create a folder



27) Go to programming=> ThonnyPython IDE

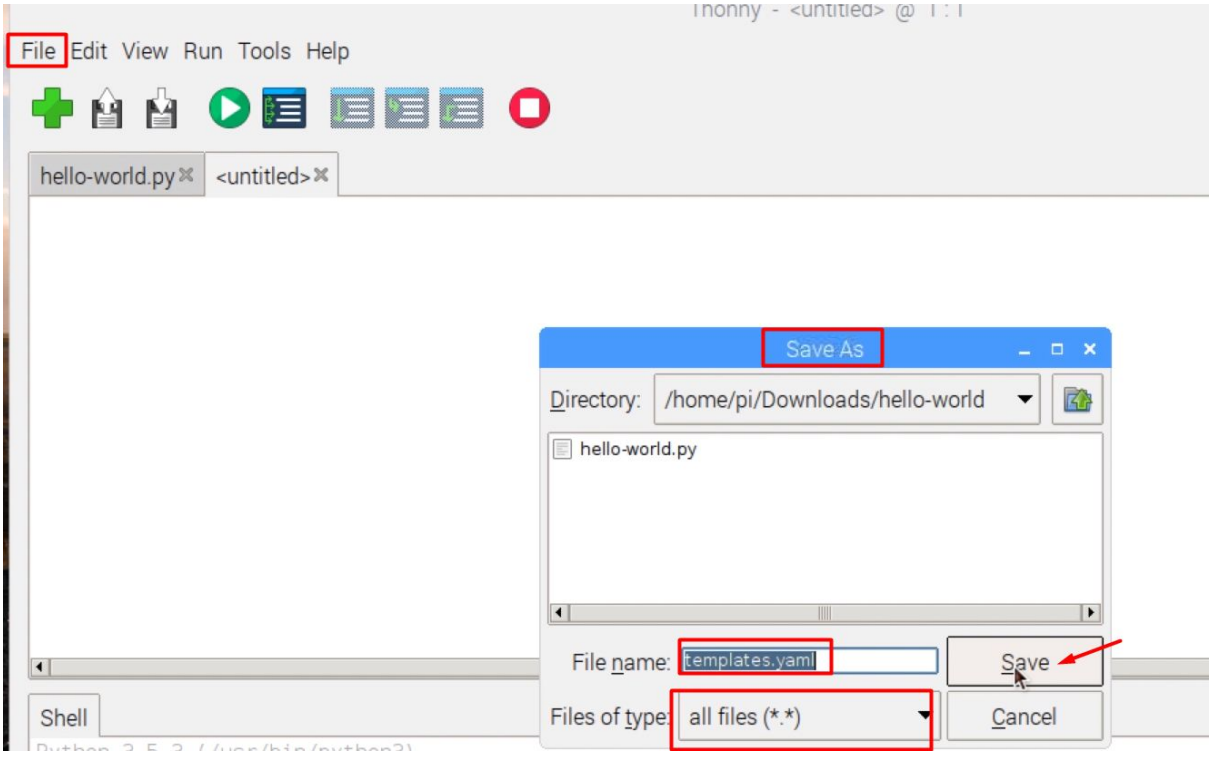


28) Save one File as py



29) Create another file: File=> New

30) The second file WILL NOT BE A PYTHON ONE. The name of it will be templates.yaml



31) Open the templates.yaml file and edit it with the strings:

a) Hello World Example:

welcome_text: Welcome to the Hello World Alexa Skill. You can say your name , or, how are you

hello: Hello {{ firstname }}. Great job on running your first Alexa skill.

ask_name: What is your name

ask_name_repromp: Can you tell me your name

b) LED Example:

This is for the LED Example:

welcome_text: Welcome to the LED control skill. You can say, tell my project to turn on, or, tell my project to turn off

command: Turning your LED {{ onOffCommand }}

command_reprompt: Do you want to turn your LED on, or, off

c) Relay Controlling Device Example:

welcome_text: Welcome to the Device control skill. You can say, tell my device to turn on, or, tell my device to turn off

command: Turning your device {{ onOffCommand }}

command_reprompt: Do you want to turn your device on, or, off

d) TV Control Example:

power_on: Turning your tv on

power_off: Turning your tv off

volume_increase: Increasing volume

volume_decrease: Decreasing volume

volume_mute: Mute volume

volume_unmute: Unmute volume

change_channel: Channel changed to {{ channel }}

change_channel_up: Channel up

change_channel_down: Channel down

stopping : Stopping

finished : Finished

channel_done : Say stop when you have reached a channel you like

welcome : Welcome to the voice control app for your tv, you can say turn on, turn off, increase volume, or switch to channel 25

fallback_message: Sorry. I did not understand your tv command

clarify_volume_command: Do you want to increase, decrease or mute the volume

clarify_channel: Which channel number would you like to see

e) s

32) Open the hello-world.py

Hello World Example

```
#This is for the Hello World Example:
from flask import Flask, render _template #The render_template is going to be used to connect with the templates.yaml file
from flask_ask import Ask, statement, question #Statement closes the interaction with the user while the question keep the conversation going for example in the case of alexa not understanding the question

app= Flask(__name__)
ask= Ask(app, '/')
```

```
@ask.launch #Decorator to map the intent to the function
 def launch():
        welcome_text=render_template('welcome_text') #The welcome_text is a
string from template.yaml
        return question(welcome_text)

 @ask.intent('AMAZON.FallbackIntent') #This name has to be exactly the name in
comparison to the skill created in Amazon Developer. This is used when Alexa didn't
understand what was asked for
 def fallback():
        reprompt_text = render_template('ask_name_reprompt') #The
ask_name_reprompt is a string from template.yaml
        return question(reprompt_text)

 @ask.intent('HelloWorldIntent')#This name has to be exactly the name in
comparison to the skill created in Amazon Developer.
 def hello(firstname):
        if firstname is None: #Checks if the User gave the First Name or not
                ask_name_text = render_template('ask_name') #The ask_name is a
string from template.yaml
                return question(ask_name_text) #Alexa asks for the name if it was not
given
        response_text = render_template('hello',firstname=firstname) #Alexa is going
to run the hello string if a name was given from the User
#firstname = firstname is the sentence that links the firstname that was given by the
user to the Alexa skill
        return statement(response_text).simple_card('Hello', response_text) #Returns
the response_text with voice and shows in Amazon Echo screen the simple_card of
Hello

if __name__ == '__main__':
        app.run(debug=True) #If there is a bug, we will be able to see in the terminal
```

LED On and Off Example

```
#This is for the LED Example:
from flask import Flask, render_template
from flask_ask import Ask, statement, question #Statement closes the interaction
with the user while the question keep the conversation going for example in the case
of alexa not understanding the question
import RPi.GPIO as GPIO #use GPIO library

#initializing GPIO
GPIO.setwarnings(False)
ledPin = 11 #pin 11 is connected to the led anode (+ve pin)
```

```
GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location
GPIO.setup(ledPin, GPIO.OUT)   # Set ledPin's mode as output
GPIO.output(ledPin, GPIO.LOW) # Set ledPin low to turn off the led

#initializing flask ask
app = Flask(__name__)
ask = Ask(app, '/')

@ask.launch
def launch():
    welcome_text = render_template('welcome_text')
    return question(welcome_text)

@ask.intent('AMAZON.FallbackIntent')
def fallback():
    reprompt_text = render_template('command_reprompt')
    return question(reprompt_text)

@ask.intent('OnOffIntent')
def control(OnOff):
    command = OnOff
    if command is None:
        #no command was given
        reprompt_text = render_template('command_reprompt')
        return question(reprompt_text)
    elif command == "on" or command == "off":
        if command == "off":
            #turn OFF
            GPIO.output(ledPin, GPIO.LOW)
        else:
            #turn ON
            GPIO.output(ledPin, GPIO.HIGH)
        response_text = render_template('command', onOffCommand=command)
#onOffCommand is in the template.yaml and allows alexa to say what she is doing
with the LED
        return statement(response_text).simple_card('Command', response_text)
#Returns the response_text with voice and shows in Amazon Echo screen the
simple_card of Led ON or Off
    else:
        #a valid command was not given. In case the user says something that it is not
On or Off
        reprompt_text = render_template('command_reprompt')
        return question(reprompt_text)

if __name__ == '__main__':
    app.run(debug=True)
```

Relay Switching On and Off Example

```
#Testing the Relay Switching On and Off
import RPi.GPIO as GPIO #use GPIO library
import time #use time library

GPIO.setwarnings(False)
outputPin = 11 #pin 11 is connected to the relay board
GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location
GPIO.setup(outputPin, GPIO.OUT)   # Set pin's mode as output
GPIO.output(outputPin, GPIO.HIGH) # Set outputPin high to turn off the device

while True: # Continue looping indefinitely
  GPIO.output(outputPin, GPIO.LOW)  # Turn device on
  time.sleep(2) # Pause for 2 seconds
  GPIO.output(outputPin, GPIO.HIGH) # Turn device off
  time.sleep(2) # Pause for 2 seconds
```

Relay Controlling Device Example

```
#Relay Control System (WHOLE PROGRAM)
from flask import Flask, render_template
from flask_ask import Ask, statement, question
import RPi.GPIO as GPIO #use GPIO library

#initializing GPIO
GPIO.setwarnings(False)
outputPin = 11 #pin 11 is connected to the relay board
GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location
GPIO.setup(outputPin, GPIO.OUT)   # Set pin's mode as output
GPIO.output(outputPin, GPIO.HIGH) # Set outputPin  high to turn off the device

#initializing flask ask
app = Flask(__name__)
ask = Ask(app, '/')

@ask.launch
def launch():
    welcome_text = render_template('welcome_text')
    return question(welcome_text)

@ask.intent('AMAZON.FallbackIntent')
def fallback():
    reprompt_text = render_template('command_reprompt')
    return question(reprompt_text)
```

```
@ask.intent('OnOffIntent')
def control(OnOff):
    command = OnOff
    if command is None:
        #no command was given
        reprompt_text = render_template('command_reprompt')
        return question(reprompt_text)
    elif command == "on" or command == "off":
        if command == "off":
            #turn OFF
            GPIO.output(outputPin, GPIO.HIGH)
        else:
            #turn ON
            GPIO.output(outputPin, GPIO.LOW)
        response_text = render_template('command', onOffCommand=command)
        return statement(response_text).simple_card('Command', response_text)
    else:
        #a valid command was not given
        reprompt_text = render_template('command_reprompt')
        return question(reprompt_text)

if __name__ == '__main__':
    app.run(debug=True)
```

TV Control Example

```
Usage:
Alexa, tell my tv to turn on/off
Alexa, tell my tv to increase/decrease/mute the volume
Alexa, tell my tv to switch to channel 95
Alexa, tell my tv to channel surf up/down

Created by: Lee Assam
www.powerlearningacademy.com

Last Modified: 2018-07-20
'''
from flask import Flask, render_template
from flask_ask import Ask, statement, question, request, session, convert_errors
import os, time

#sends a repeating command
#see the irsend command in lirc documentation http://www.lirc.org/html/irsend.html
#command : The IR command to execute
#duration : The duration in seconds to sleep/pause while the command is being repeated
def sendRepeatCommand(command, duration):
```

```
    os.system("irsend SEND_START {} {} ; sleep {}".format(device, command, duration))
    os.system("irsend SEND_STOP {} {}".format(device, command))

#initialize the app
app = Flask(__name__)
ask = Ask(app, '/')

#name of your TV
device = "sony"

@ask.launch
def start_skill():
    welcome_message = render_template('welcome')
    return question(welcome_message)

@ask.intent('AMAZON.HelpIntent')
def help():
    return start_skill()

@ask.intent('AMAZON.FallbackIntent')
def fallback():
    fallback_message = render_template('fallback_message')
    return statement(fallback_message)

@ask.session_ended
def session_ended():
    return "{}", 200

#onOffCommand : on | off
@ask.intent('PowerIntent')
def power(onOffCommand):
    os.system("irsend --count=2 SEND_ONCE %s KEY_POWER KEY_POWER" % device)
    text = render_template('power_on') if onOffCommand == "on" else
render_template('power_off')
    return statement(text).simple_card('Status', text)

#volumeCommand : increase | decrease | mute |unmute
@ask.intent('VolumeIntent')
def volume(volumeCommand):
    clarifyText = render_template('clarify_volume_command')
    #volume adjustment not specified
    if volumeCommand is None:
        return question(clarifyText)
    else:
        #determine what was asked
        if volumeCommand == "increase":
```

```
            sendRepeatCommand("KEY_VOLUMEUP", 3)
            text = render_template('volume_increase')
        elif volumeCommand == "decrease":
            sendRepeatCommand("KEY_VOLUMEDOWN", 3)
            text = render_template('volume_decrease')
        elif volumeCommand == "mute" or volumeCommand == "unmute":
            sendRepeatCommand("KEY_MUTE", 0.1)
            text = render_template('volume_mute') if volumeCommand == "mute" else
render_template('volume_unmute')
        else:
            #recevied a command that we did not expect
            return question(clarifyText)
        return statement(text).simple_card('Status', text)

#channelNumber : integer
@ask.intent('GotoChannelIntent', convert={'channelNumber': int})
def gotoChannel(channelNumber):
    if channelNumber is None:
        channel_question = render_template('channel_question')
        return question (channel_question)
    else:
        channelNums = list(str(channelNumber))
        #loop over each number and send the ir signal for each number key
        for i in channelNums:
            sendRepeatCommand("KEY_" + i, 0.1)
            time.sleep(0.5)
        text = render_template('change_channel', channel=channelNumber)
        return statement(text).simple_card('Status', text)

#direction: up | down | stop
@ask.intent('ChannelSurfIntent')
def channelSurf(direction):
    if direction == "up":
        sendRepeatCommand("KEY_CHANNELUP", 0.1);
        text = render_template('change_channel_up')
        return question(text).reprompt(render_template('channel_done')).simple_card('Status',
text)
    elif direction == "down":
        sendRepeatCommand("KEY_CHANNELDOWN", 0.1);
        text = render_template('change_channel_down')
        return question(text).reprompt(render_template('channel_done')).simple_card('Status',
text)
    elif direction == "stop":
        return statement(render_template('finished'))
    else:
        return question("Did you want to move up, down or stop")
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

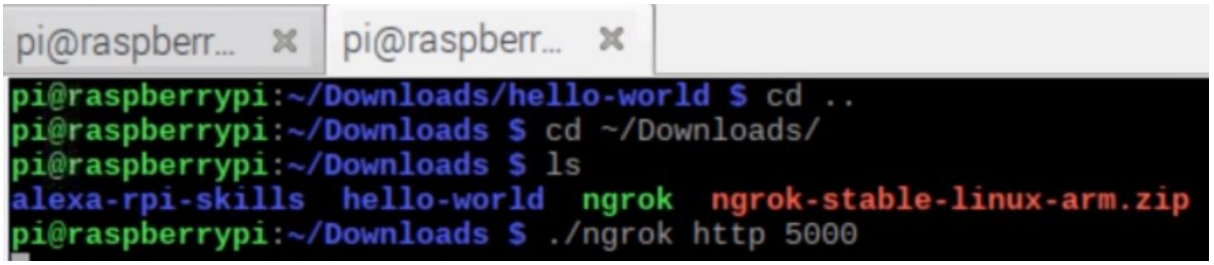
33) Go to the Terminal and then go to the folder the files are.
    Example:
    cd “Downloads/hello-world”
34) Type: python3 nameofthefile.py
    Example:
    python3 hello-world.py
35) The above will run in localhost, so you have to: Open another Terminal
36) Go to the folder you put the ngrok: cd “Downloads”
37) The ngrok will make the hello-world public:
    This is going to create a https endpoint
    ./ngrok http 5000

In order to the ngrok not expire (url expires), create an account:
Here is the link to setup your ngrok account

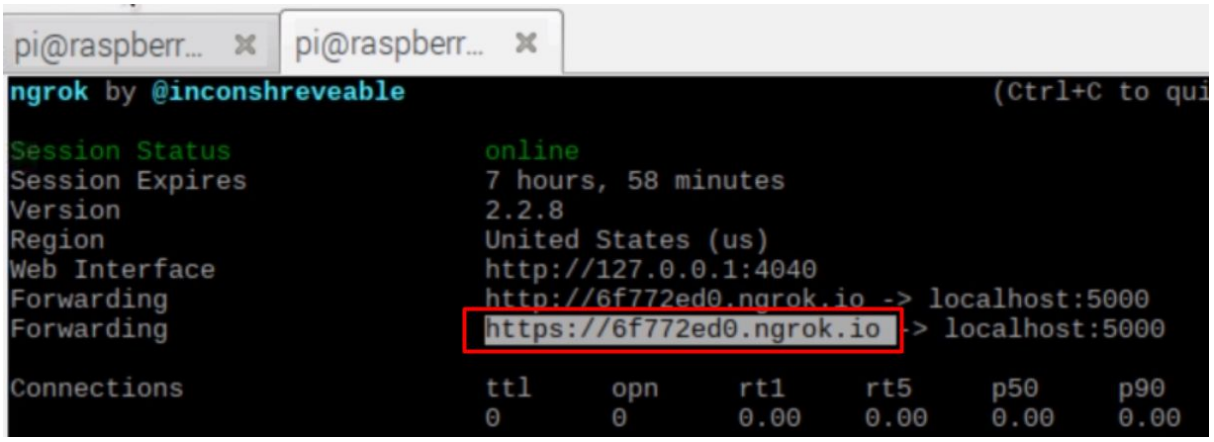
https://dashboard.ngrok.com/user/signup

Once you set up your account, you can setup your authtoken as described in the link below:
https://ngrok.com/docs#authtoken

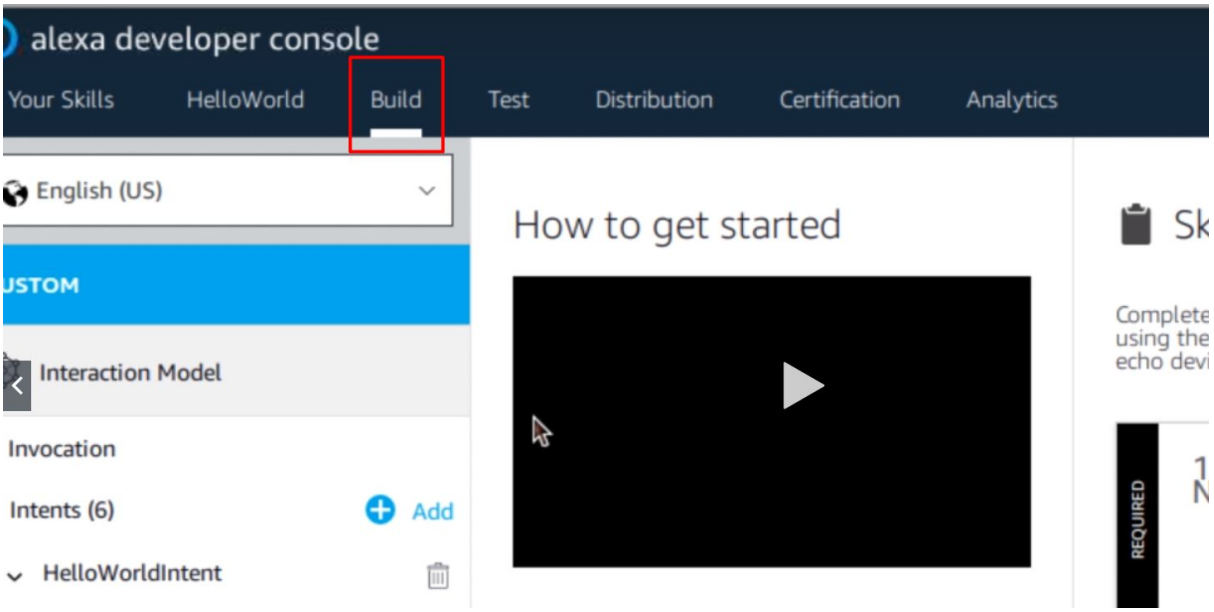
This will enable you to get a dashboard where you can review your https links and tunnels and they will persist longer!



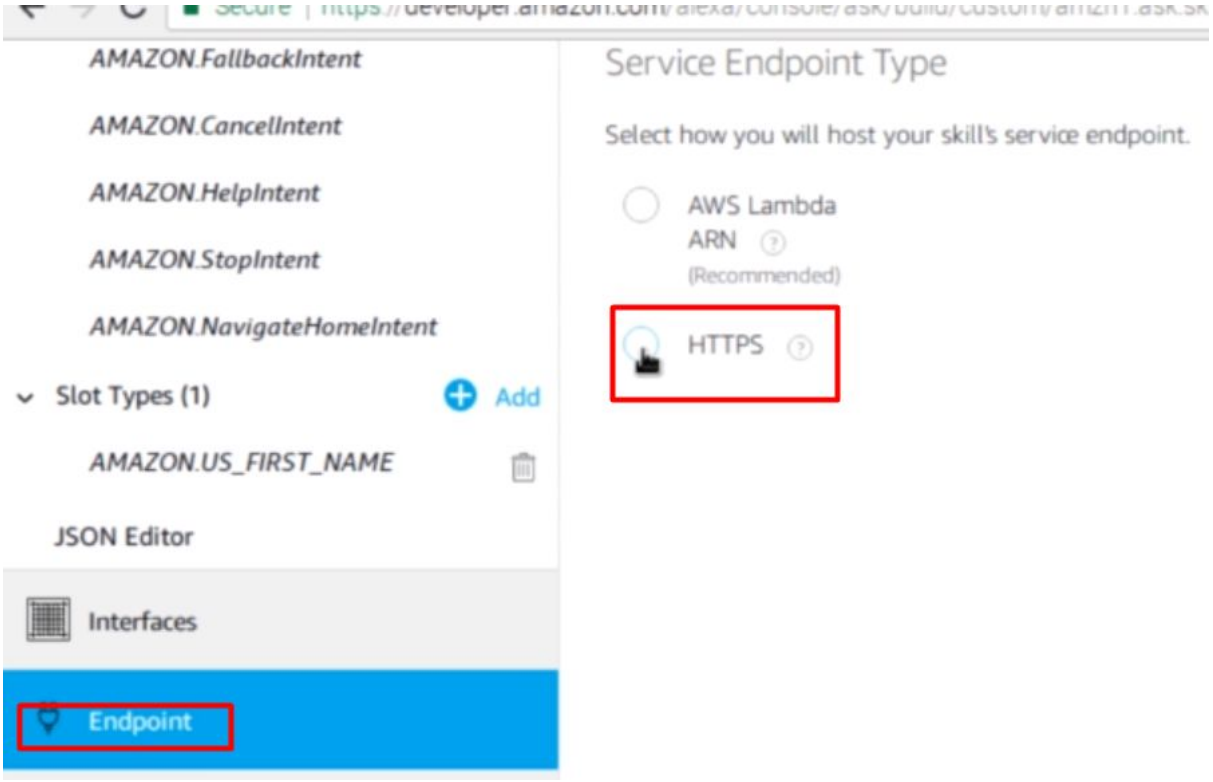
38) Copy the url below:



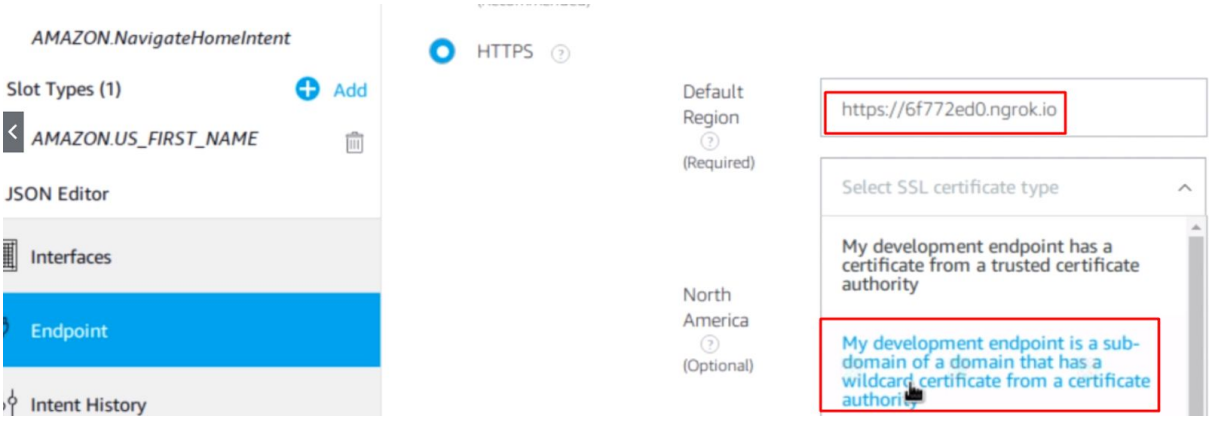
39) Go to the Amazon Developer Website=> Skills=> Click in the Skill Created=> Build



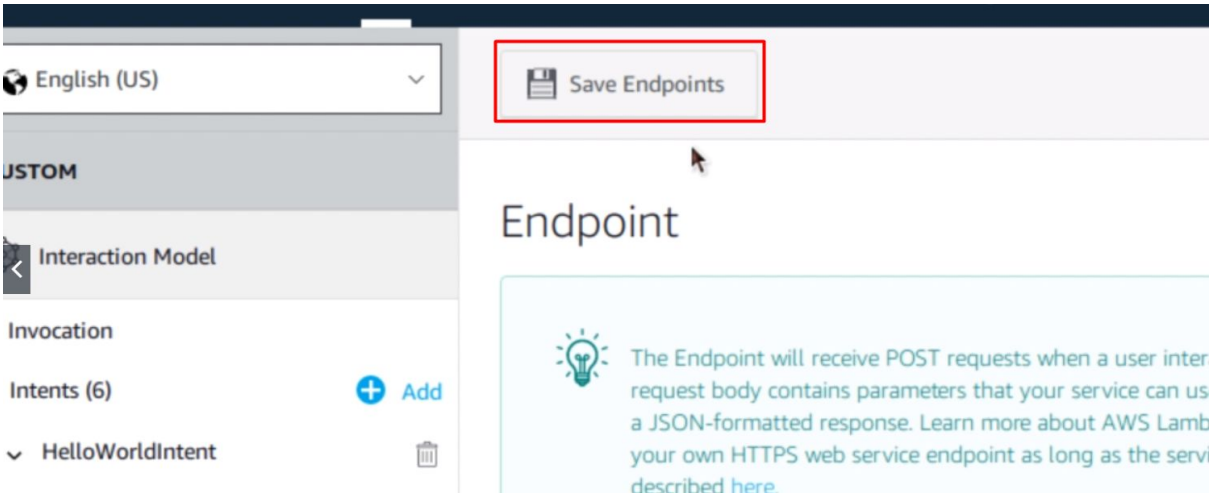
40) Endpoint=> HTTPS



41) Put the link collected in step 35) in Default Region and the Second Option in SSL



42) Save the Endpoint

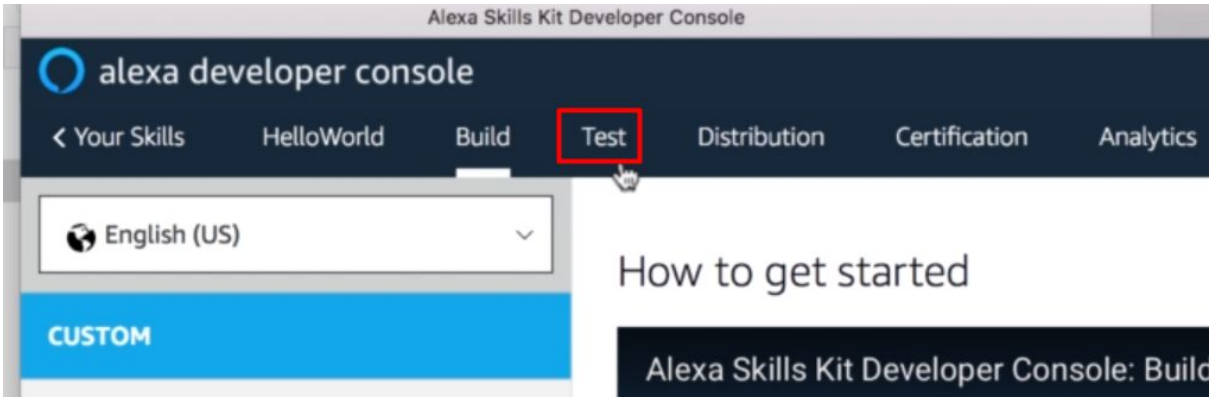


43)

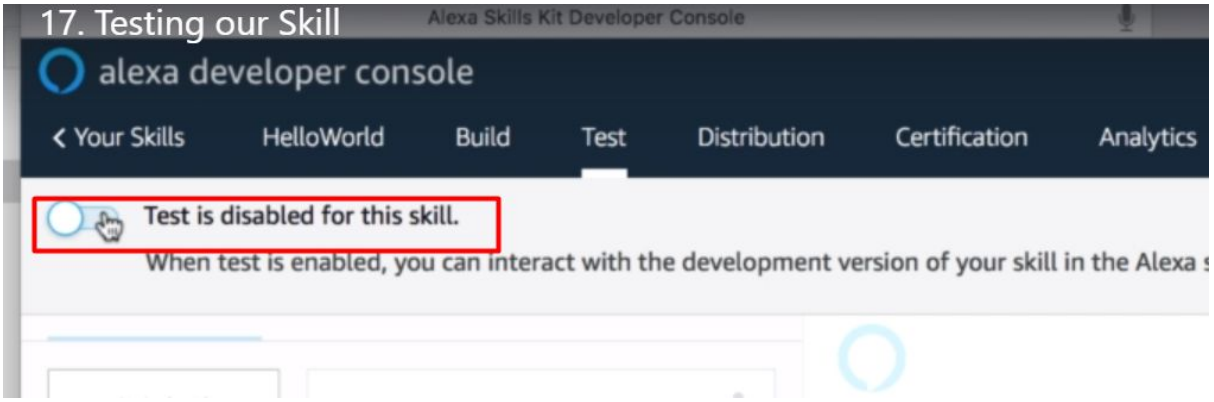
44) If there is a Cryptography Error when you run the Skill Handler. Just install the following package: pip install 'cryptography<2.2'

Testing the New Skill

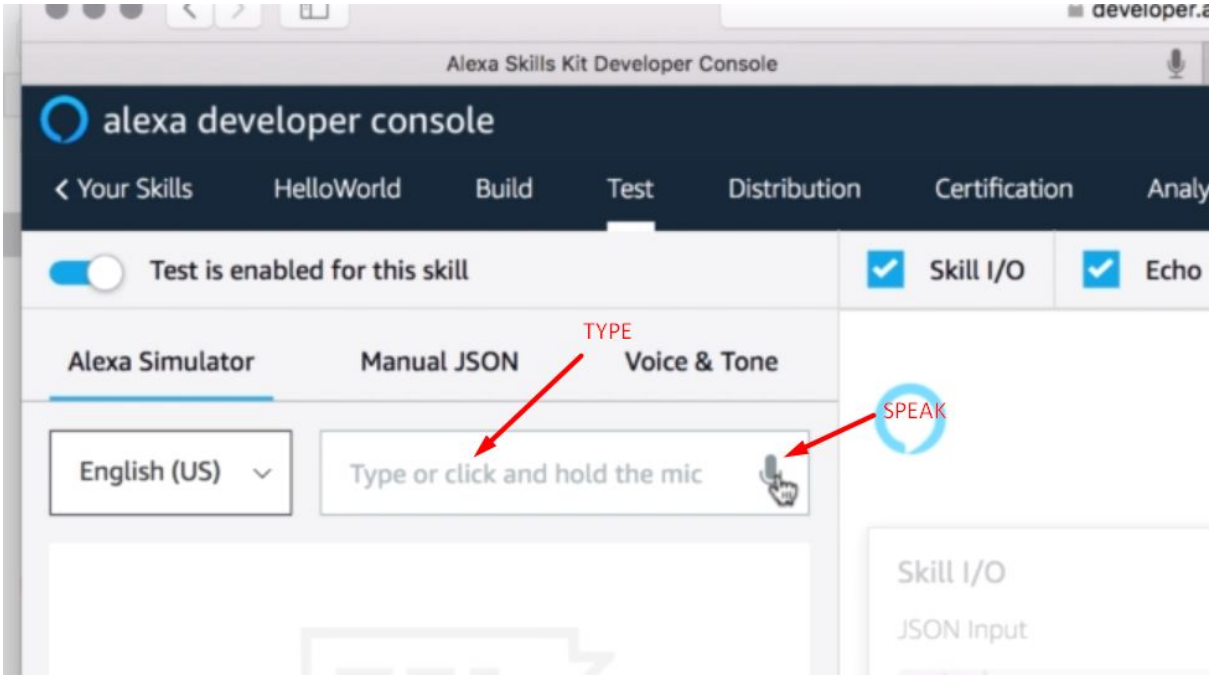
45) Go to the Amazon Developer Website=> Skills=> Click in the Skill Created=> Test



46) Enable the testing of this skill



47) Type or Speak (hold)



48) If speaking: ask skill to say skill to variable

For Hello World: ask hello world to say hello to Lee

For LED: Tell my project to turn on

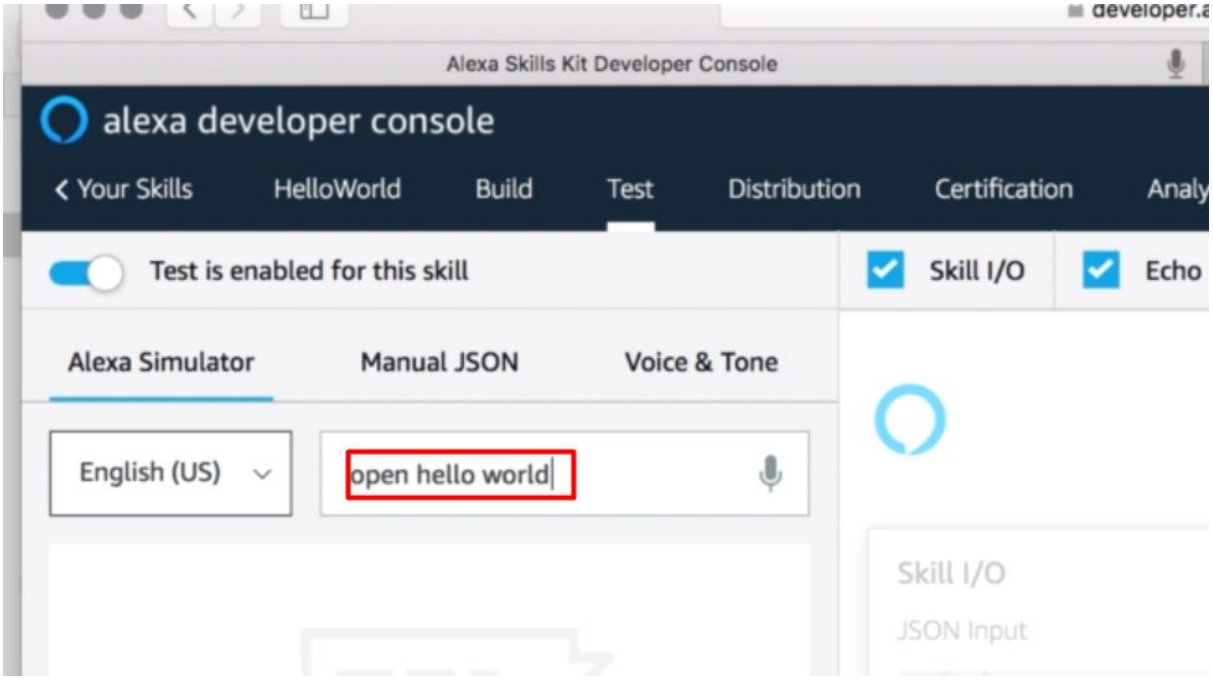
For Relay: Tell my device to turn on

49) If typing: open skill

For Hello World: open hello world

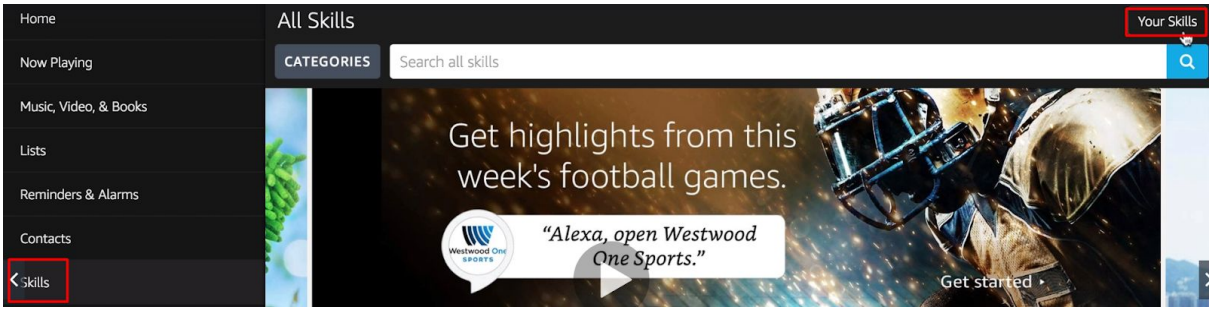
For LED: open myLed

For Relay: Tell my device to turn on



Testing the New Skill in Alexa

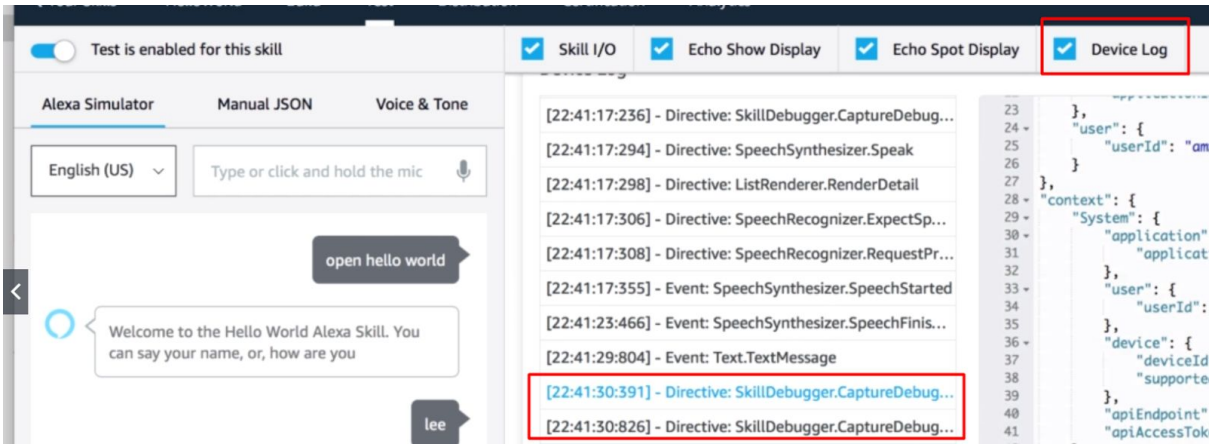
1) Go to:https://alexa.amazon.ca/spa/index.html
2) Skills=> Your Skills



3) DEV Skills
4) Click on the Skill
5) Enable

Debugging the code

1) Check the Device Log=> Scroll Down=> Select the Capture Debug



2) s