SCUTTLE Self Balancing Robot - Software Installation Guide

Step 1: we will insert the Raspberry Pi's SD card into our Computer.

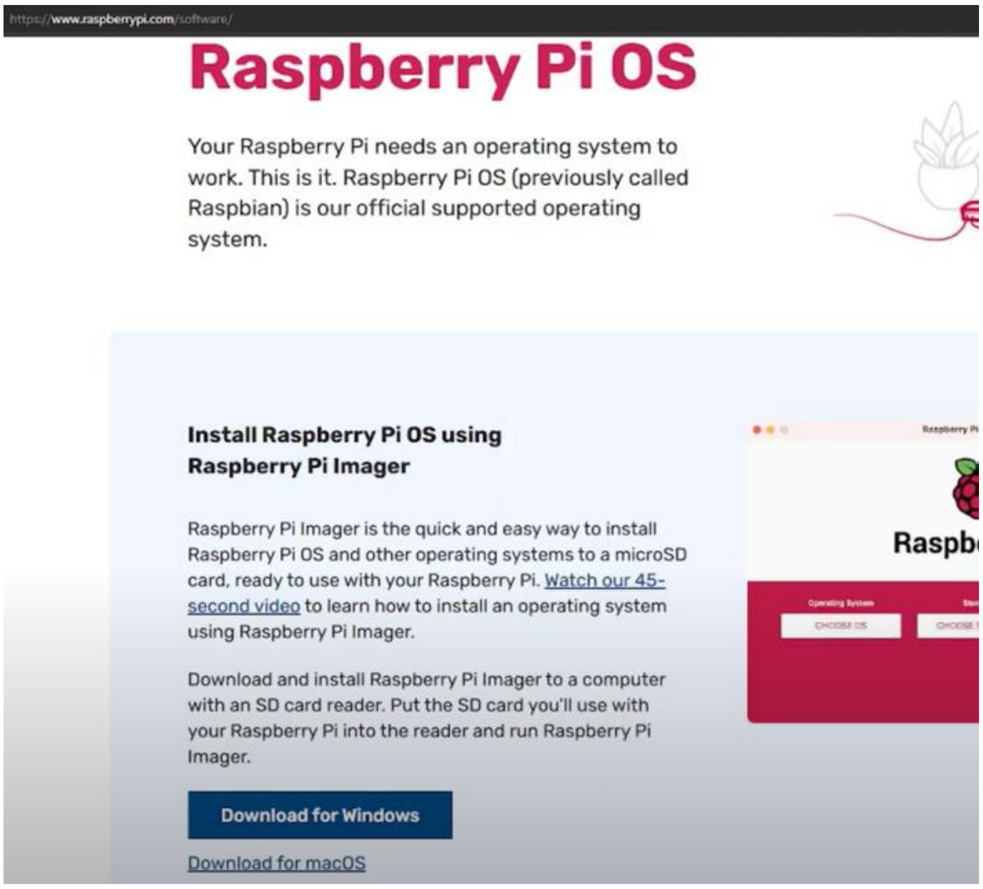
An SD card of at least 4 GB will be sufficient.

Step 2: we are downloading the Lite version of the Raspbian keyboard system on Rasberry Pi's web pages, installing the Raspberry Pi on the SD card, entering the necessary options and information.

a. First, we open the Rasberry Pi's website and select "software".

https://www.raspberrypi.com/software/

b. From this section, we download the Raspberry Pi Imager program.



https://www.raspberrypi.com/software/

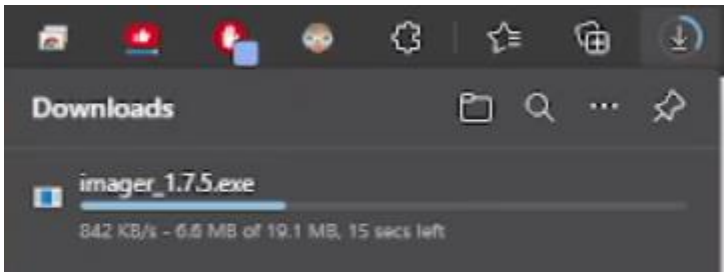
Raspberry Pi OS

Your Raspberry Pi needs an operating system to work. This is it. Raspberry Pi OS (previously called Raspbian) is our official supported operating system.

Install Raspberry Pi OS using Raspberry Pi Imager

Raspberry Pi Imager is the quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card, ready to use with your Raspberry Pi. Watch our 45-second video to learn how to install an operating system using Raspberry Pi Imager.

Download and install Raspberry Pi Imager to a computer with an SD card reader. Put the SD card you'll use with your Raspberry Pi into the reader and run Raspberry Pi Imager.

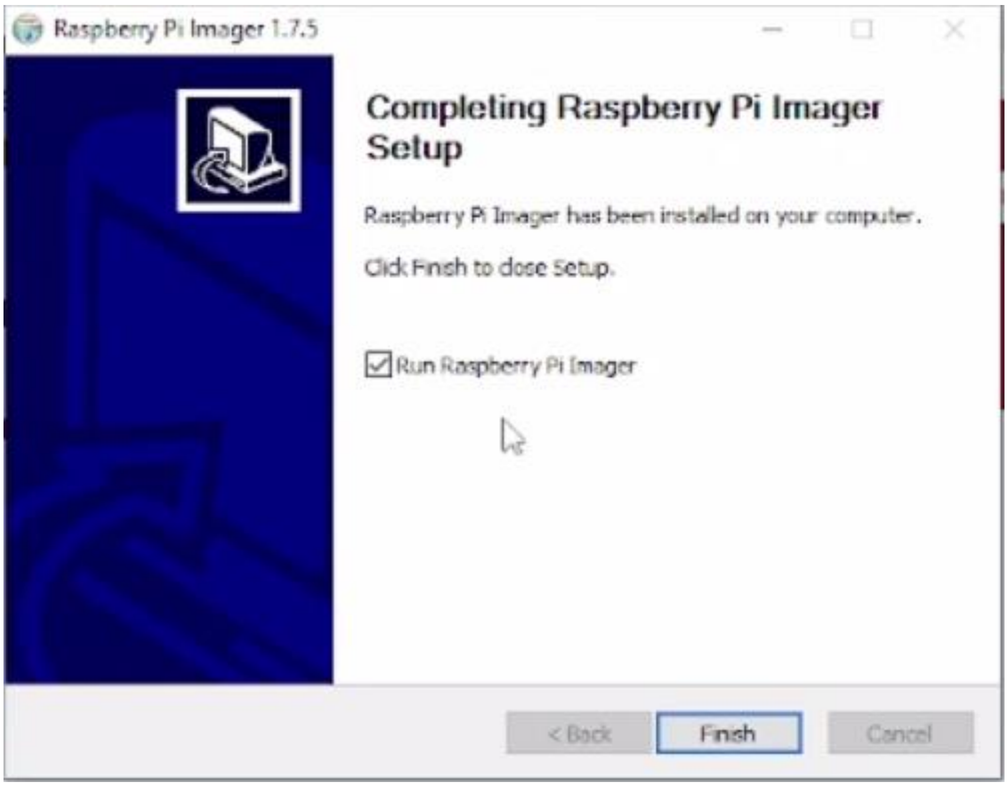


Downloads

imager_1.7.5.exe

842 KB/s - 6.6 MB of 19.1 MB, 15 secs left

c. Then we run this “Imager_x.x.x.exe” file and start the installation.


Raspberry Pi Imager 1.7.5

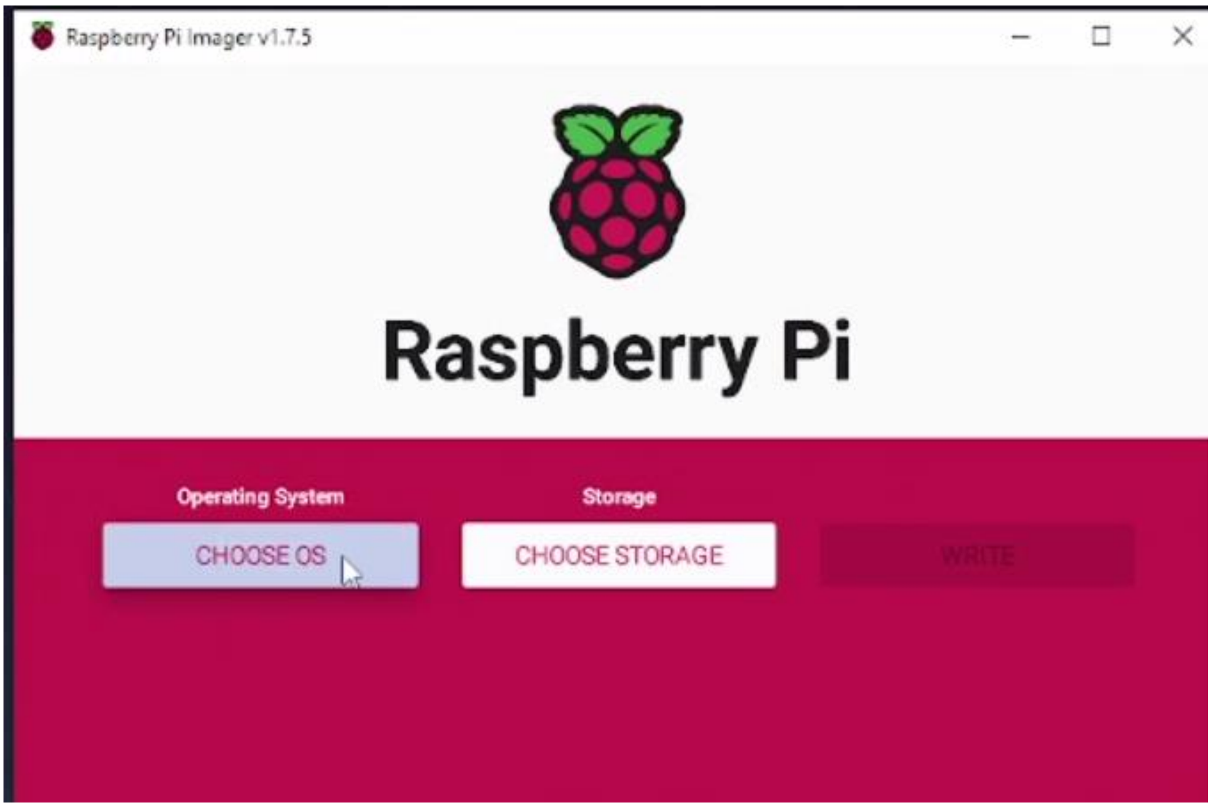
Completing Raspberry Pi Imager Setup

Raspberry Pi Imager has been installed on your computer.

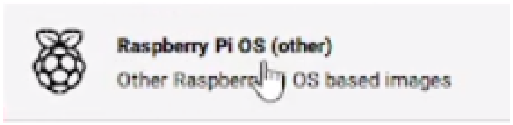
Click Finish to close Setup.

☑ Run Raspberry Pi Imager

d. After the installation is complete, we run the "Raspberry Pi Imager X.X.X" file. we push the “Finish” button while “Run Raspberry Pi Imager” is selected, it will start the program.

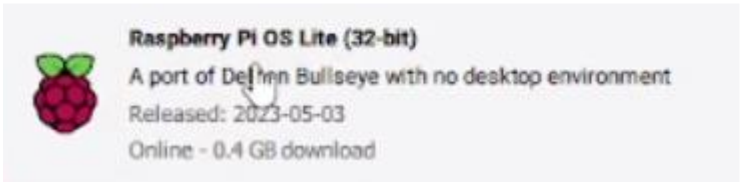


e. First, we open the window where we will select the operating system by pressing the "Choose OS" button. Here we first select the “Raspberry Pi OS (other)” option.



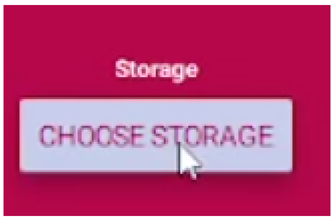
Raspberry Pi OS (other)
Other Raspberry Pi OS based images

f. Then we choose the operating system "Raspberry Pi OS Lite (32-bit)". You can also choose the “Raspberry Pi OS (32-bit)” operating system if you wish.

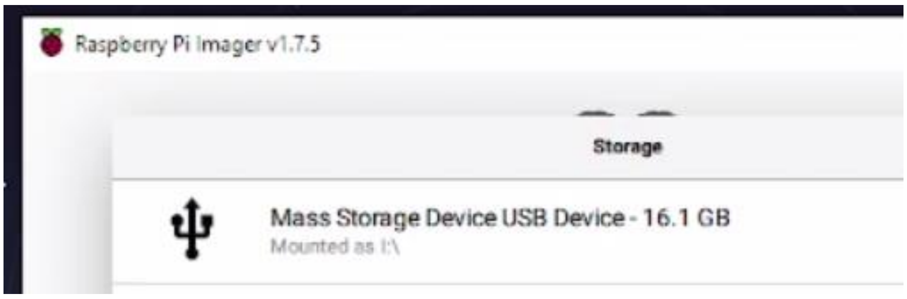


Raspberry Pi OS Lite (32-bit)
A port of Debian Bullseye with no desktop environment
Released: 2023-05-03
Online - 0.4 GB download

g. Then we press the "Choose Storage" button to choose where the operating system will be installed.

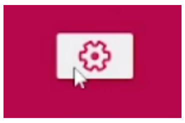


h. We select our SD card.



Raspberry Pi Imager v1.7.5
Storage
Mass Storage Device USB Device - 16.1 GB
Mounted as I:\

i. We go to the next process and press the "Settings" button.



j. In the window that opens, we first select the "to always use" option for "image customization options". Thus, we do not need to re-make these settings every time and these changes are saved to the system permanently.

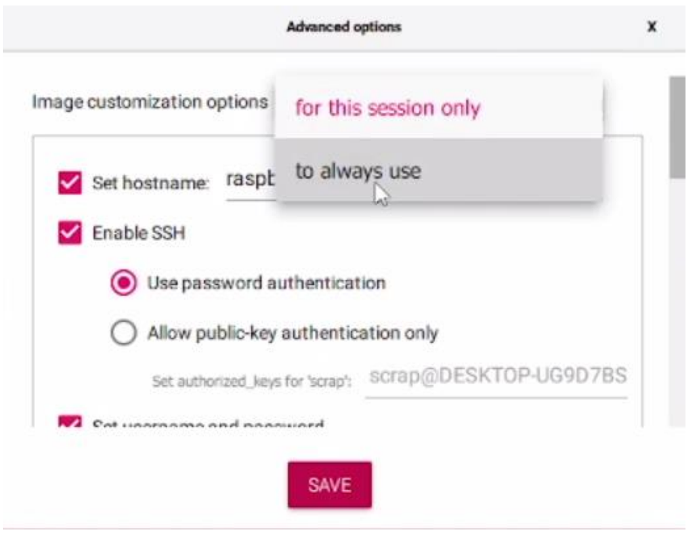

Advanced options

Image customization options

for this session only

to always use

☑ Set hostname: raspb

☑ Enable SSH

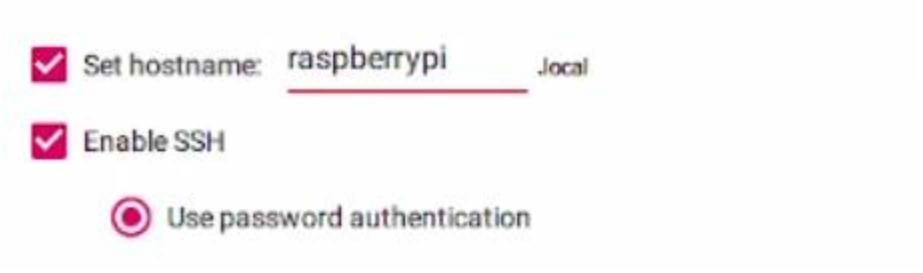
● Use password authentication

○ Allow public-key authentication only

Set authorized_keys for 'scrap': scrap@DESKTOP-UG9D7BS

SAVE

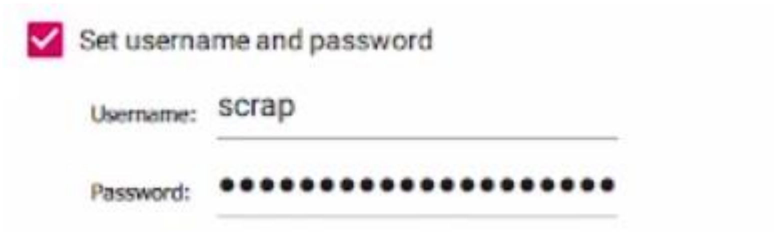
k. Other settings are as follows:


☑ Set hostname: raspberrypi .local

☑ Enable SSH

● Use password authentication

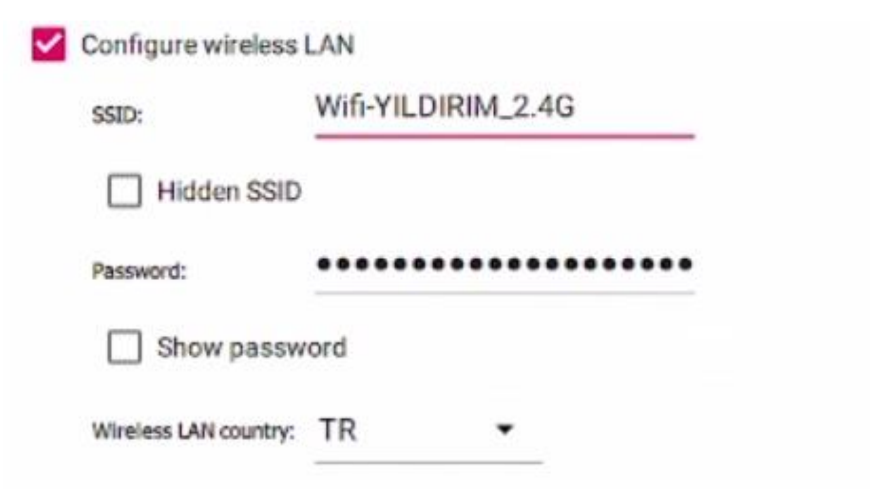
l. In this section, you set a user name and password.


☑ Set username and password

Username: scrap

Password: ●●●●●●●●●●●●●●●●●●●●

m. In this section, we determine the Wi-Fi local network you will use for the wireless connection and enter its password.


☑ Configure wireless LAN

SSID: Wifi-YILDIRIM_2.4G

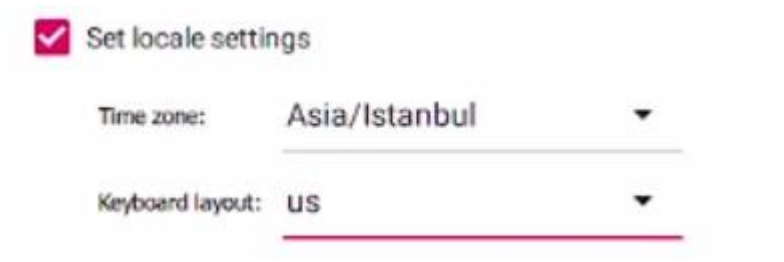
☐ Hidden SSID

Password: ●●●●●●●●●●●●●●●●●●●●

☐ Show password

Wireless LAN country: TR

n. In this section, you choose your time zone and keyboard.



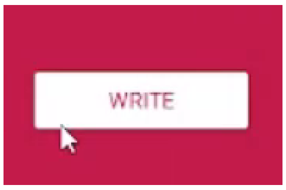
Set locale settings

Time zone: Asia/Istanbul

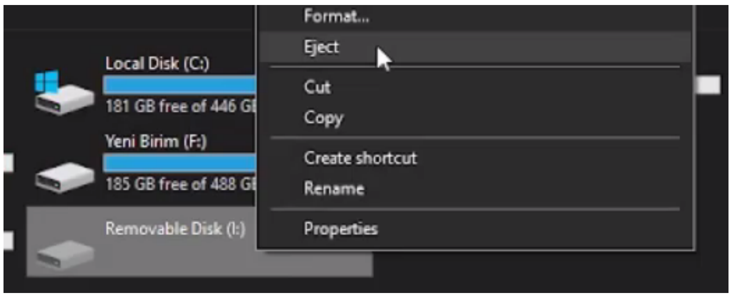
Keyboard layout: us

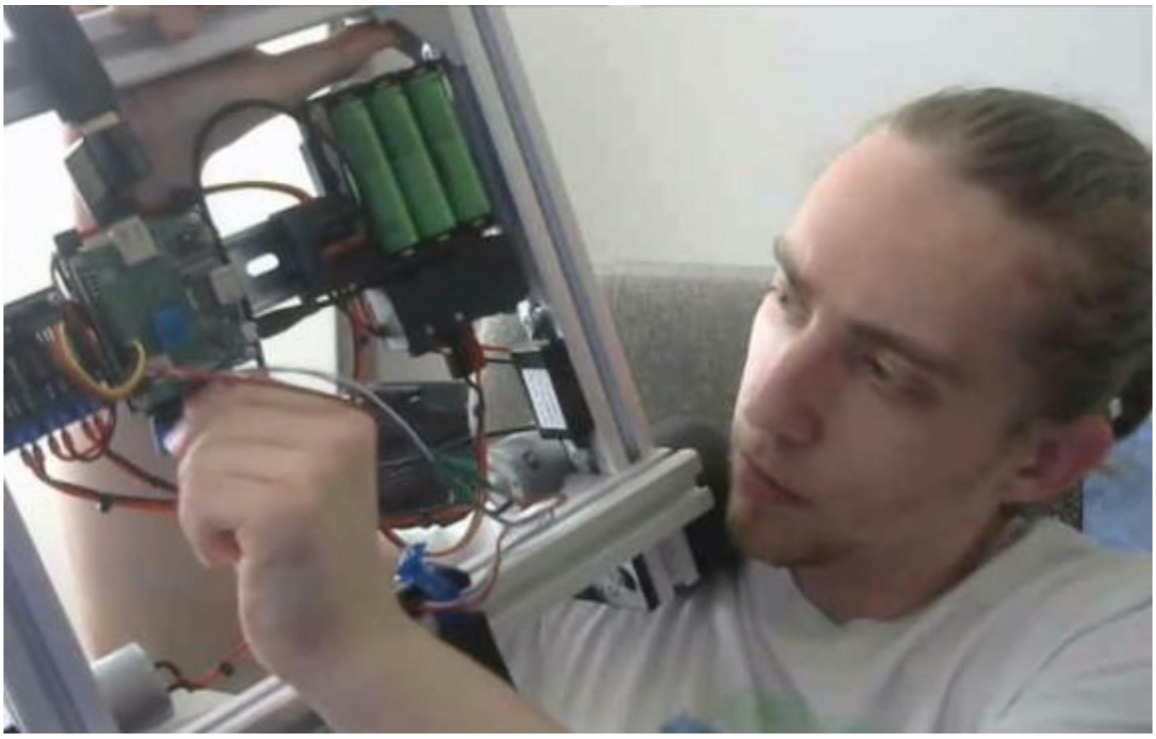
o. Finally, we save these changes by pressing the “write” button. We wait until the process is completed.

p.

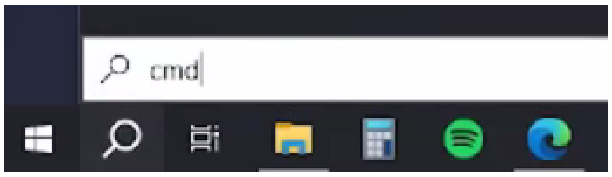


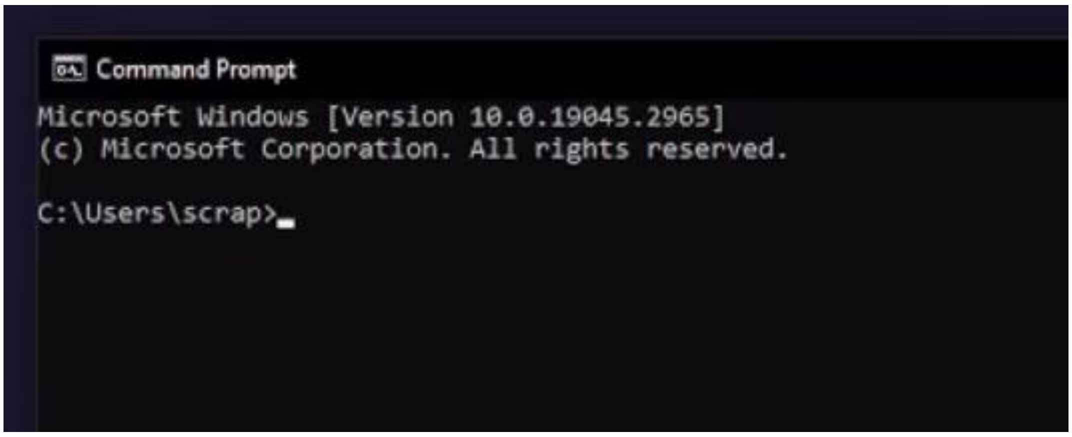
Step 3: I remove the SD card from the Computer and insert into Raspberry Pi. And I am turning the Raspberry Pi on.





Step 4: While Raspberry Pi is booting, we run the "cmd comment prompt" on our computer.

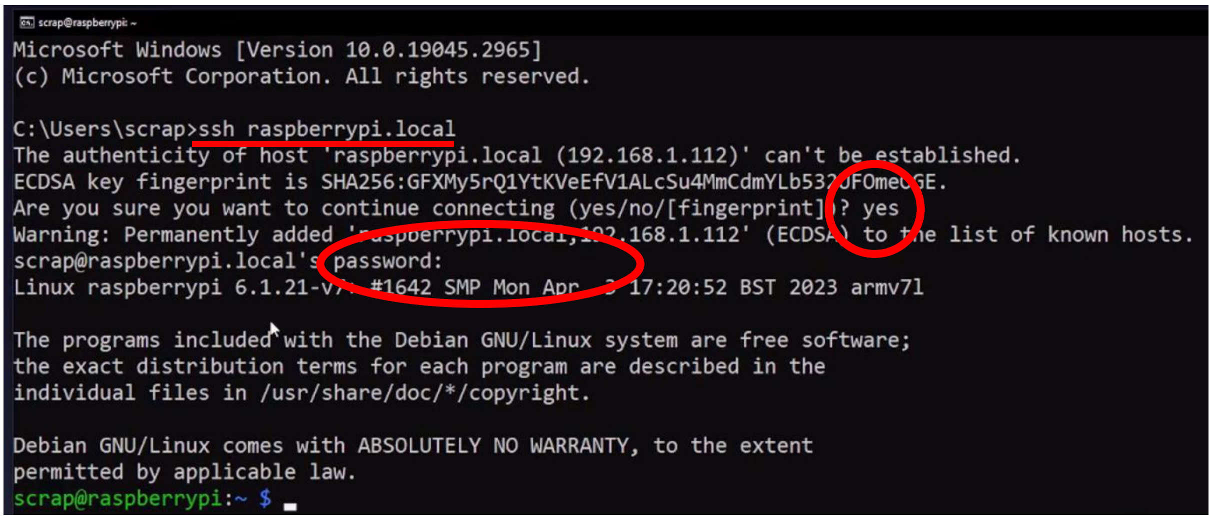




```
Command Prompt
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\scrap>_
```

Step 5: I am connecting to Raspberry Pi from my computer via "cmd comment prompt".

We use this command to connect: “ssh raspberrypi.local”



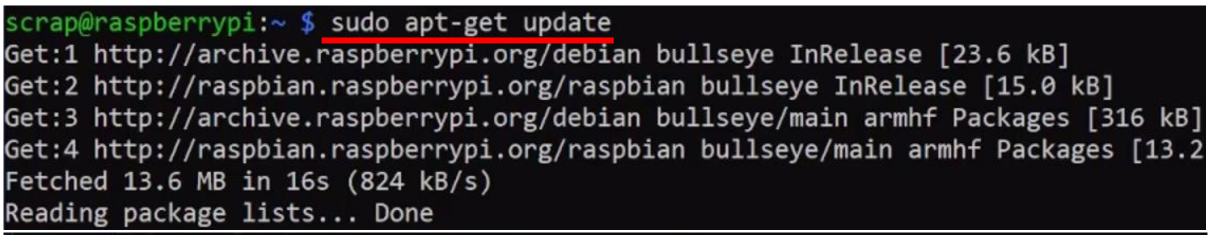
```
scrap@raspberrypi: ~
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\scrap>ssh raspberrypi.local
The authenticity of host 'raspberrypi.local (192.168.1.112)' can't be established.
ECDSA key fingerprint is SHA256:GFXMy5rQ1YtKVeEfV1ALcSu4MmCdmYLb532UFOmeCGE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'raspberrypi.local,192.168.1.112' (ECDSA) to the list of known hosts.
scrap@raspberrypi.local's password:
Linux raspberrypi 6.1.21-v7+ #1642 SMP Mon Apr  3 17:20:52 BST 2023 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
scrap@raspberrypi:~ $ _
```

We made the connection.

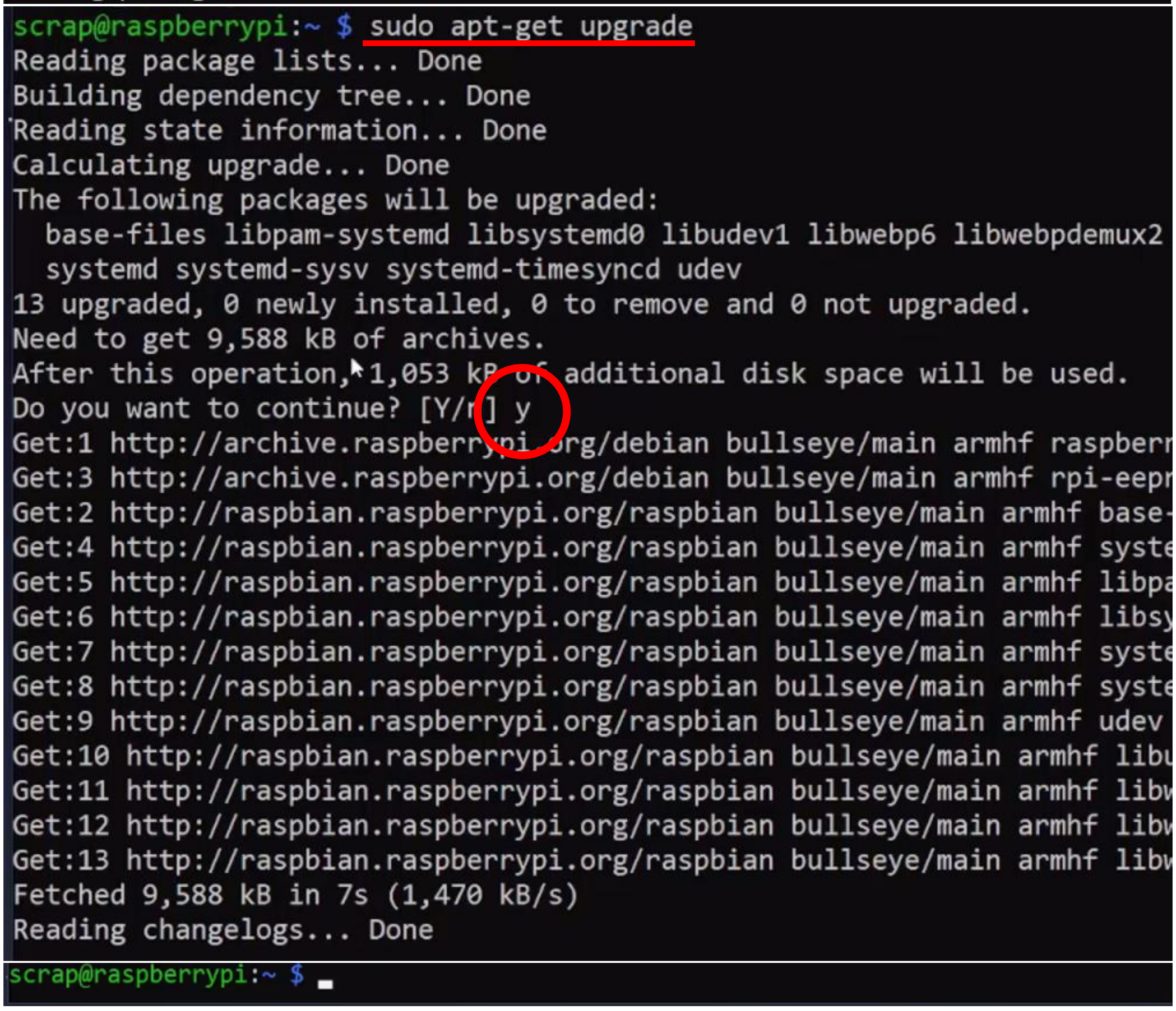
Step 6: We are doing the "Update" and "Upgrade" on Raspberry for those software require update and upgrade.

The commands we use for this are:

“sudo apt-get update” and “sudo apt-get upgrade”



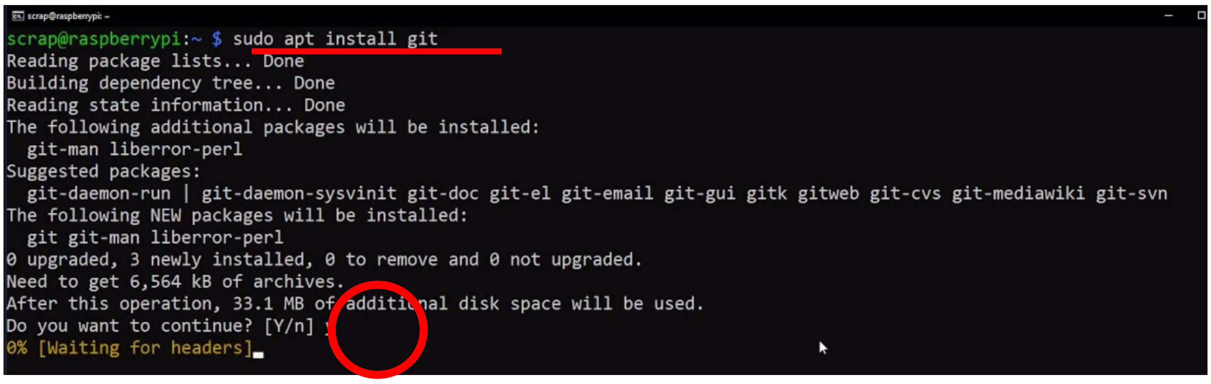
```
scrap@raspberrypi:~ $ sudo apt-get update
Get:1 http://archive.raspberrypi.org/debian bullseye InRelease [23.6 kB]
Get:2 http://raspbian.raspberrypi.org/raspbian bullseye InRelease [15.0 kB]
Get:3 http://archive.raspberrypi.org/debian bullseye/main armhf Packages [316 kB]
Get:4 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf Packages [13.2
Fetched 13.6 MB in 16s (824 kB/s)
Reading package lists... Done
```



```
scrap@raspberrypi:~ $ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  base-files libpam-systemd libsystemd0 libudev1 libwebp6 libwebpdemux2
  systemd systemd-sysv systemd-timesyncd udev
13 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 9,588 kB of archives.
After this operation, 1,053 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.raspberrypi.org/debian bullseye/main armhf raspberr
Get:3 http://archive.raspberrypi.org/debian bullseye/main armhf rpi-eepr
Get:2 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf base-
Get:4 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf syste
Get:5 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf libpa
Get:6 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf libsy
Get:7 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf syste
Get:8 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf syste
Get:9 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf udev
Get:10 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf libu
Get:11 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf libw
Get:12 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf libw
Get:13 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf libw
Fetched 9,588 kB in 7s (1,470 kB/s)
Reading changelogs... Done
scrap@raspberrypi:~ $ _
```

Step 7: We are installing the "GIT" program that allow files to be downloaded from GitHub to Raspberry Pi.

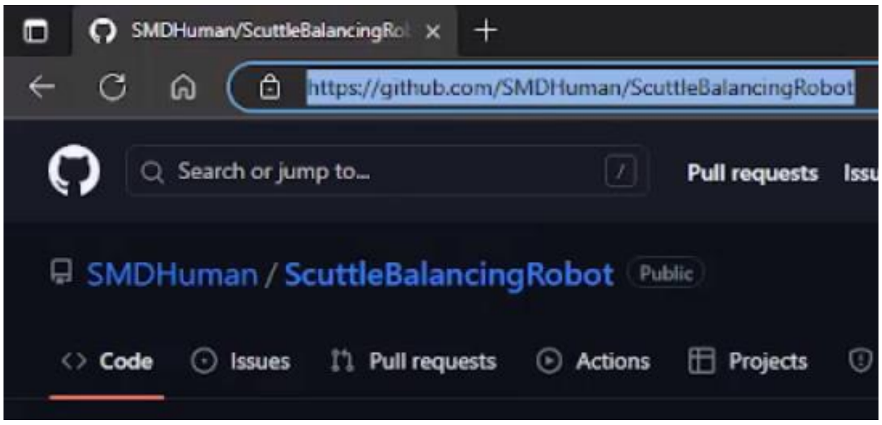
The command we use for this is: "sudo apt install git"

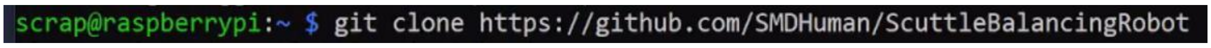


```
scrap@raspberrypi:~ $ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 6,564 kB of archives.
After this operation, 33.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
0% [Waiting for headers]_
```

Step 8: I copy the GitHub link of the project and upload all project files to the Raspberry Pi, including the Python software codes, with the GIT program.

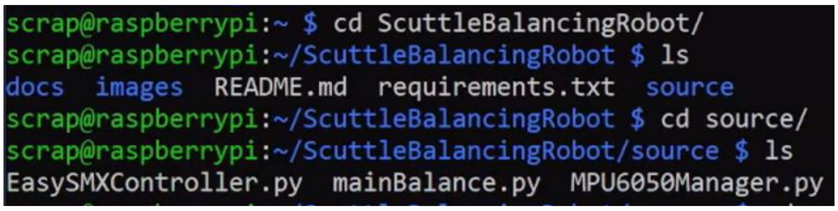
The command we use for this is: "git clone <github project web address link>"


SMDHuman/ScuttleBalancingRo
https://github.com/SMDHuman/ScuttleBalancingRobot
Search or jump to...
Pull requests
Issu
SMDHuman / ScuttleBalancingRobot
Public
Code
Issues
Pull requests
Actions
Projects



```
scrap@raspberrypi:~ $ git clone https://github.com/SMDHuman/ScuttleBalancingRobot
```

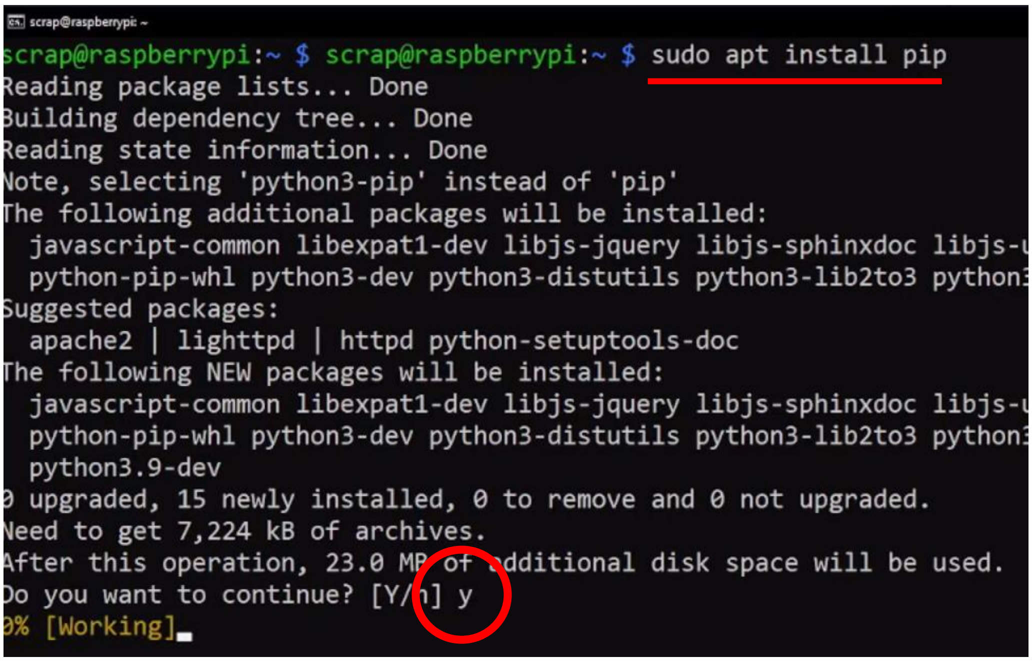
after they are downloaded, we may see all files.



```
scrap@raspberrypi:~ $ cd ScuttleBalancingRobot/
scrap@raspberrypi:~/ScuttleBalancingRobot $ ls
docs  images  README.md  requirements.txt  source
scrap@raspberrypi:~/ScuttleBalancingRobot $ cd source/
scrap@raspberrypi:~/ScuttleBalancingRobot/source $ ls
EasySMXController.py  mainBalance.py  MPU6050Manager.py
```

Step 9: we install the “PIP” program to the Raspberry Pi, which allows us to install the necessary libraries for Python and allows to control.

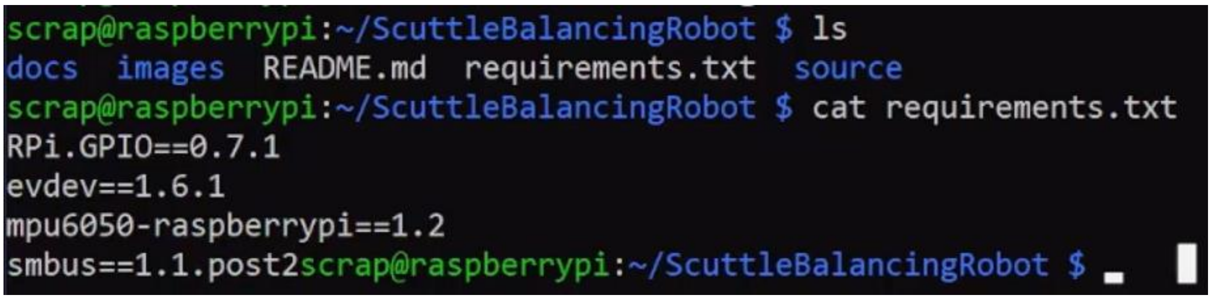
The command I use for this is: "sudo apt install pip"



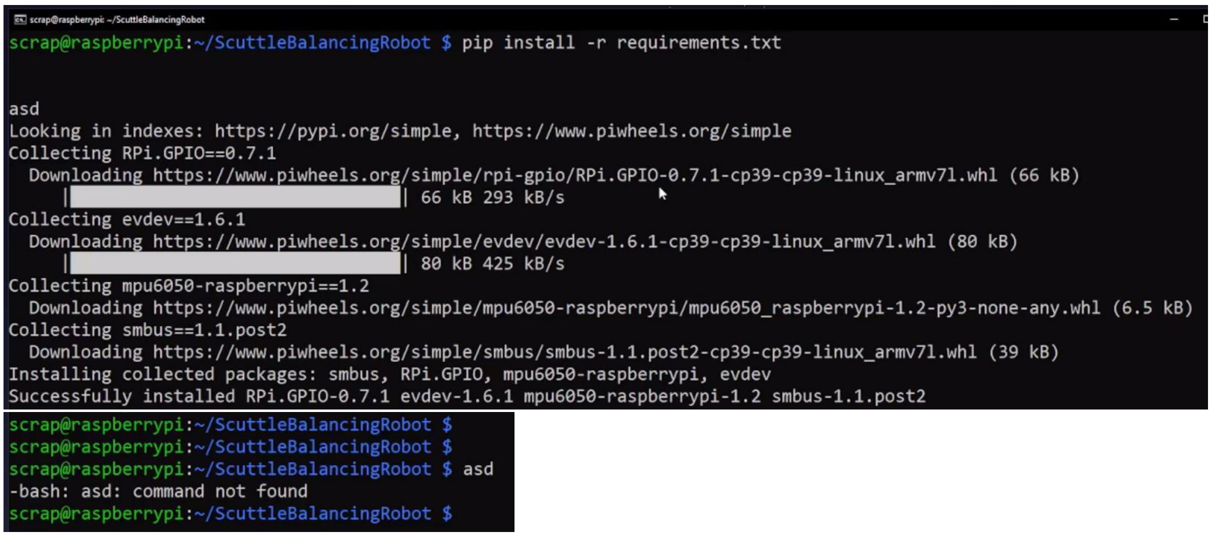
```
scrap@raspberrypi: ~
scrap@raspberrypi:~ $ scrap@raspberrypi:~ $ sudo apt install pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'python3-pip' instead of 'pip'
The following additional packages will be installed:
  javascript-common libexpat1-dev libjs-jquery libjs-sphinxdoc libjs-u
  python-pip-whl python3-dev python3-distutils python3-lib2to3 python3
Suggested packages:
  apache2 | lighttpd | httpd python-setuptools-doc
The following NEW packages will be installed:
  javascript-common libexpat1-dev libjs-jquery libjs-sphinxdoc libjs-u
  python-pip-whl python3-dev python3-distutils python3-lib2to3 python3
  python3.9-dev
0 upgraded, 15 newly installed, 0 to remove and 0 not upgraded.
Need to get 7,224 kB of archives.
After this operation, 23.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
0% [Working]
```

Step 10: we download the required Python libraries which are defined and listed at "Requirement.txt", to Raspberry Pi with PIP application.

The command I use for this is: "pip install – r requirement.txt"



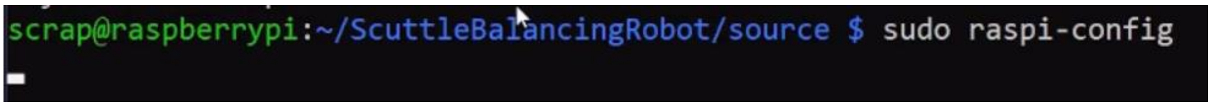
```
scrap@raspberrypi:~/ScuttleBalancingRobot $ ls
docs  images  README.md  requirements.txt  source
scrap@raspberrypi:~/ScuttleBalancingRobot $ cat requirements.txt
RPi.GPIO==0.7.1
evdev==1.6.1
mpu6050-raspberrypi==1.2
smbus==1.1.post2scrap@raspberrypi:~/ScuttleBalancingRobot $ _
```



```
scrap@raspberrypi:~/ScuttleBalancingRobot $ pip install -r requirements.txt


asd
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting RPi.GPIO==0.7.1
  Downloading https://www.piwheels.org/simple/rpi-gpio/RPi.GPIO-0.7.1-cp39-cp39-linux_armv7l.whl (66 kB)
     |████████████████████████████████| 66 kB 293 kB/s
Collecting evdev==1.6.1
  Downloading https://www.piwheels.org/simple/evdev/evdev-1.6.1-cp39-cp39-linux_armv7l.whl (80 kB)
     |████████████████████████████████| 80 kB 425 kB/s
Collecting mpu6050-raspberrypi==1.2
  Downloading https://www.piwheels.org/simple/mpu6050-raspberrypi/mpu6050_raspberrypi-1.2-py3-none-any.whl (6.5 kB)
Collecting smbus==1.1.post2
  Downloading https://www.piwheels.org/simple/smbus/smbus-1.1.post2-cp39-cp39-linux_armv7l.whl (39 kB)
Installing collected packages: smbus, RPi.GPIO, mpu6050-raspberrypi, evdev
Successfully installed RPi.GPIO-0.7.1 evdev-1.6.1 mpu6050-raspberrypi-1.2 smbus-1.1.post2
scrap@raspberrypi:~/ScuttleBalancingRobot $
scrap@raspberrypi:~/ScuttleBalancingRobot $
scrap@raspberrypi:~/ScuttleBalancingRobot $ asd
-bash: asd: command not found
scrap@raspberrypi:~/ScuttleBalancingRobot $
```

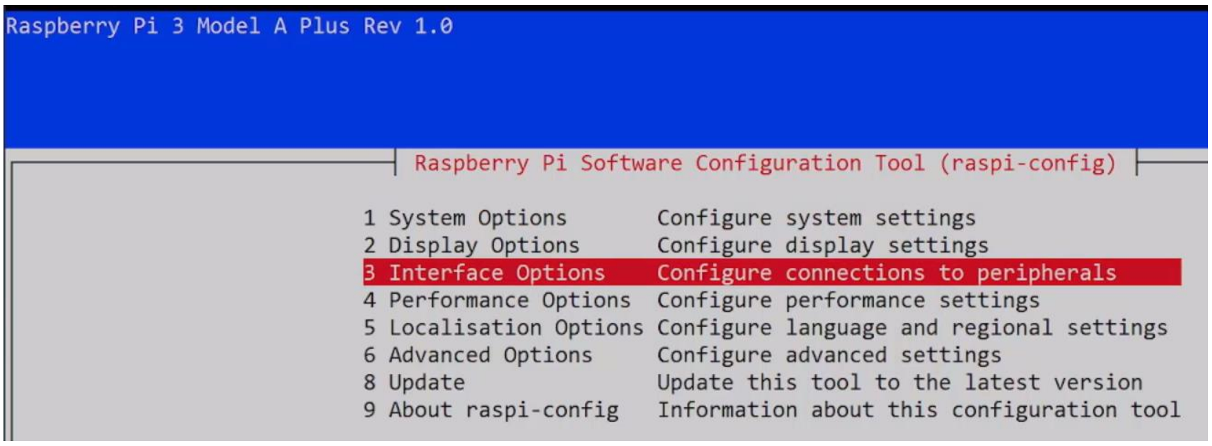
Step 11: By entering the Raspberry Pi's configuration interface, we activate the I2C pins that will allow it to reach the MPU6050

a. The command we use for this is: "sudo raspi-config"



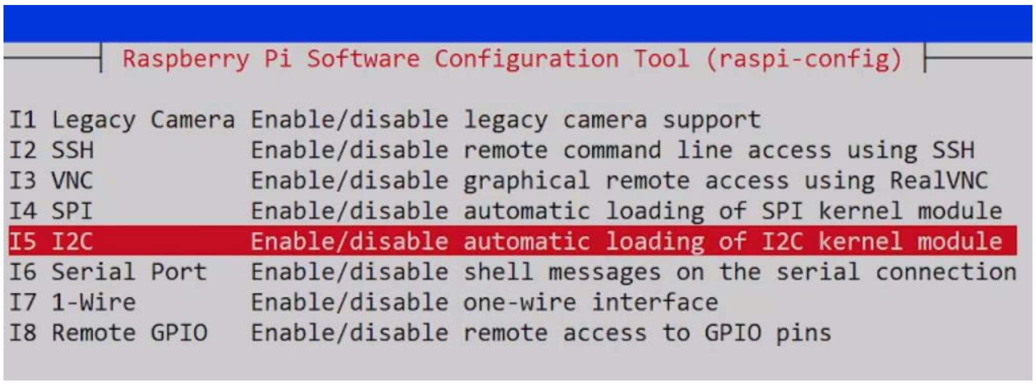
```
scrap@raspberrypi:~/ScuttleBalancingRobot/source $ sudo raspi-config
```

b. We select "3 Interface Options" from the configuration screen.



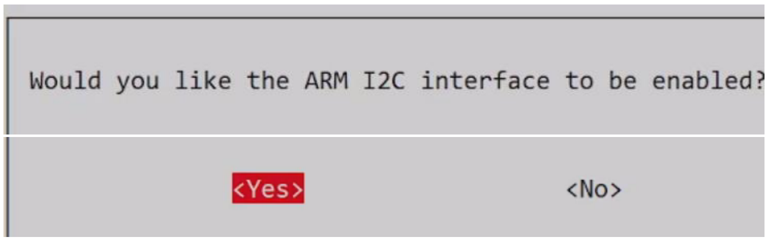
```
Raspberry Pi 3 Model A Plus Rev 1.0

        Raspberry Pi Software Configuration Tool (raspi-config)

   1 System Options       Configure system settings
   2 Display Options      Configure display settings
   3 Interface Options    Configure connections to peripherals
   4 Performance Options  Configure performance settings
   5 Localisation Options Configure language and regional settings
   6 Advanced Options     Configure advanced settings
   8 Update               Update this tool to the latest version
   9 About raspi-config   Information about this configuration tool
```

c. Here we select “I5 I2C Enable/disable”.



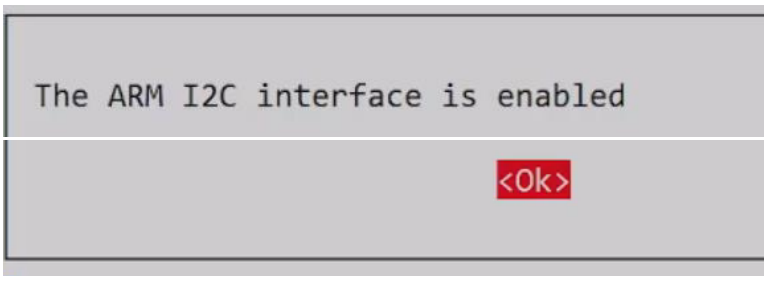
```
Raspberry Pi Software Configuration Tool (raspi-config)

I1 Legacy Camera Enable/disable legacy camera support
I2 SSH           Enable/disable remote command line access using SSH
I3 VNC           Enable/disable graphical remote access using RealVNC
I4 SPI           Enable/disable automatic loading of SPI kernel module
I5 I2C           Enable/disable automatic loading of I2C kernel module
I6 Serial Port   Enable/disable shell messages on the serial connection
I7 1-Wire        Enable/disable one-wire interface
I8 Remote GPIO   Enable/disable remote access to GPIO pins
```

d. we press the "Yes" button to enable the "ARM I2C interface".



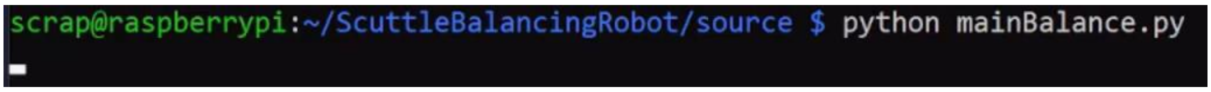
```
Would you like the ARM I2C interface to be enabled?

          <Yes>                    <No>
```

e. “ARM I2C” is enabled.



```
The ARM I2C interface is enabled

                         <Ok>
```

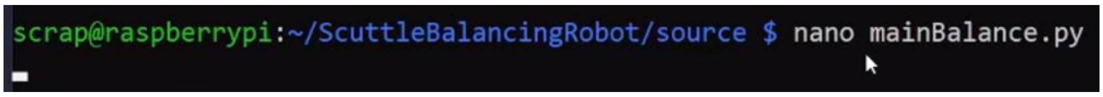
Step 12: we run the “mainBalance.py” Python software file. And the Balance Program starts to work. The command we use for this is: “python mainBalance.py”



```
scrap@raspberrypi:~/ScuttleBalancingRobot/source $ python mainBalance.py
```

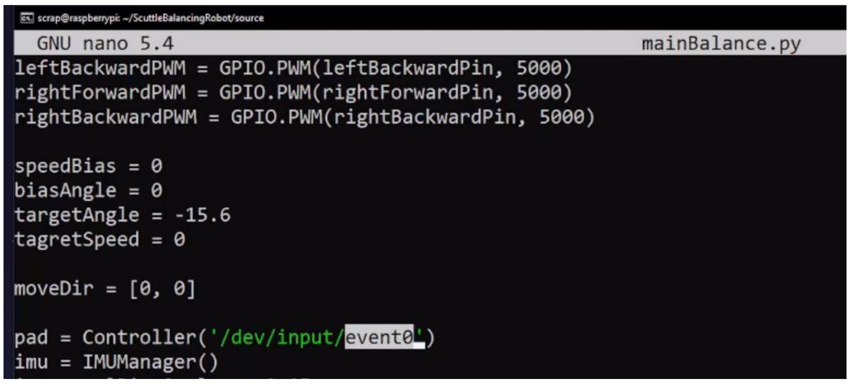
Step 13: Problem 1: Here I am explaining the solution of the problem if you do not have the "Dongle" that allows us to communicate with the remote. I also explain how to see the name of the remote controller when it is found by the Raspberry Pi and how to update the code for that name.

a. First, we remove the "Dongle" from our robot.
b. We open the “mainBalance.py” file by using the “nano mainBalance.py” command.



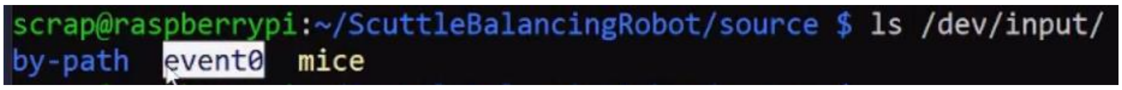
```
scrap@raspberrypi:~/ScuttleBalancingRobot/source $ nano mainBalance.py
```

c. Here Controller is seen as "event0".



```
scrap@raspberrypi: ~/ScuttleBalancingRobot/source
  GNU nano 5.4                                       mainBalance.py
leftBackwardPWM = GPIO.PWM(leftBackwardPin, 5000)
rightForwardPWM = GPIO.PWM(rightForwardPin, 5000)
rightBackwardPWM = GPIO.PWM(rightBackwardPin, 5000)

speedBias = 0
biasAngle = 0
targetAngle = -15.6
tagretSpeed = 0

moveDir = [0, 0]

pad = Controller('/dev/input/event0')
imu = IMUManager()
```

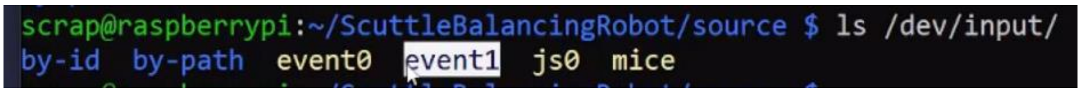
d. Only "event0" can be seen with the command "ls /dev/input".



```
scrap@raspberrypi:~/ScuttleBalancingRobot/source $ ls /dev/input/
by-path  event0  mice
```

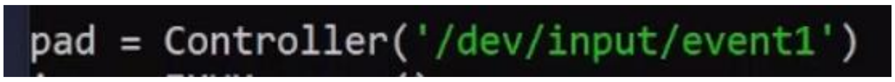
e. we plug the "dongle" back into the Raspberry Pi.

f. When we check with the "ls /dev/input" command, we can see that the new control controller has been added to the system as "event1".



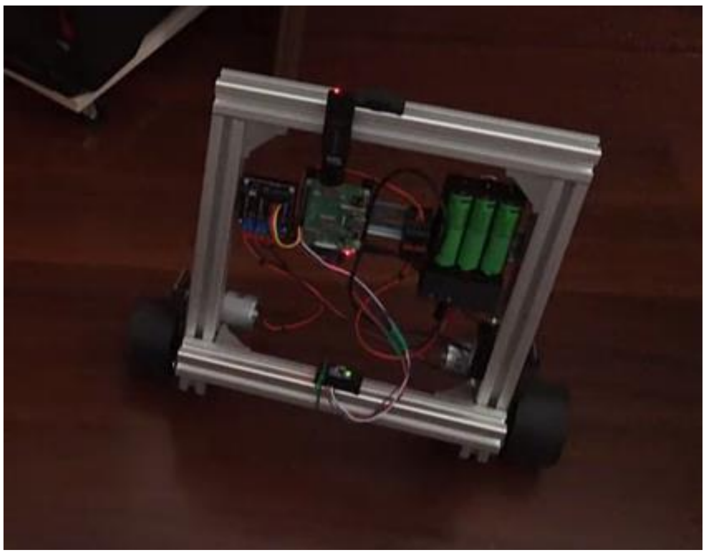
```
scrap@raspberrypi:~/ScuttleBalancingRobot/source $ ls /dev/input/
by-id  by-path  event0  event1  js0  mice
```

g. When this line is corrected in the "mainBalance.py" file in this way, the Controller will be seen by the system. "pad = Controller ('/dev/input/event1')"

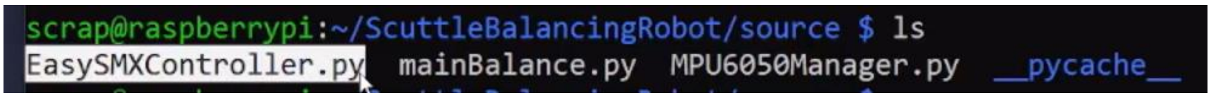


```
pad = Controller('/dev/input/event1')
```

Step 14: My robot works without any problem.



Step 15: Problem 2: In case of a problem with the brand of the controller, this code file should be modified accordingly.

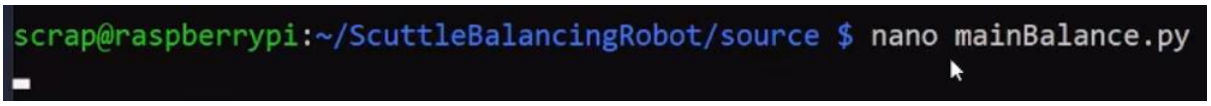


```
scrap@raspberrypi:~/ScuttleBalancingRobot/source $ ls
EasySMXController.py  mainBalance.py  MPU6050Manager.py  __pycache__
```

Necessary adjustments are made in the "EasySMXController.py" file and a new different brand controller can be introduced to the system.

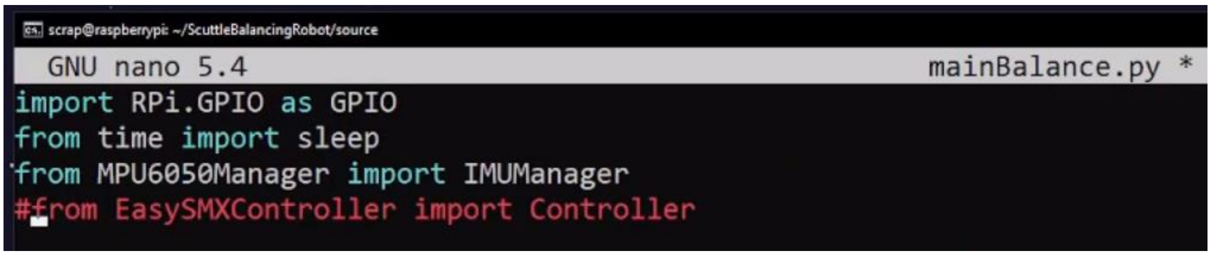
Step 16: If we want to disable the remote controller, these are the modifications needs to be done in the code here.

a. Firstly . We open the "mainBalance.py" file with the "nano mainBalance.py" command.



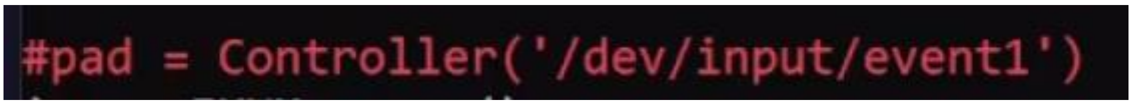
```
scrap@raspberrypi:~/ScuttleBalancingRobot/source $ nano mainBalance.py
```

b. We put "#" at the beginning of the line "from EasySMXConroller import Controller" to disable the line



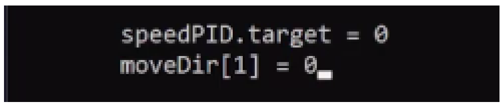
```
scrap@raspberrypi: ~/ScuttleBalancingRobot/source
  GNU nano 5.4                                        mainBalance.py *
import RPi.GPIO as GPIO
from time import sleep
from MPU6050Manager import IMUManager
#from EasySMXController import Controller
```

c. We put "#" at the beginning of "pad= Controller('/dev/input/event1')'" line to disable it.



```
#pad = Controller('/dev/input/event1')
```

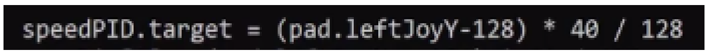
d. We assign 0 to the variables "speedPID.target" and "moveDir[1]"



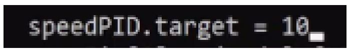
```
speedPID.target = 0
moveDir[1] = 0
```

Step 17: We can make our robot move at constant speed or stands still, by doing these changes in the code as it is given below.

"speedPID.target" is a variable where we control the speed of the robot. If we set a value of 0 to this variable, the robot stands still and does not move.



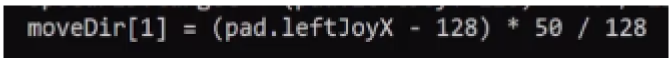
```
speedPID.target = (pad.leftJoyY-128) * 40 / 128
```



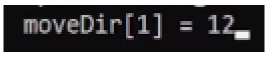
```
speedPID.target = 10
```

Step 18: We can make our robot to turn to the right or to the left, or make it stands still without turning, by doing these changes in the code as it is given below.

"speedPID.target" is a variable where I control the robot's right and left turns. If I assign a value of 0 to this variable, the robot stays in place and does not turn left or right.



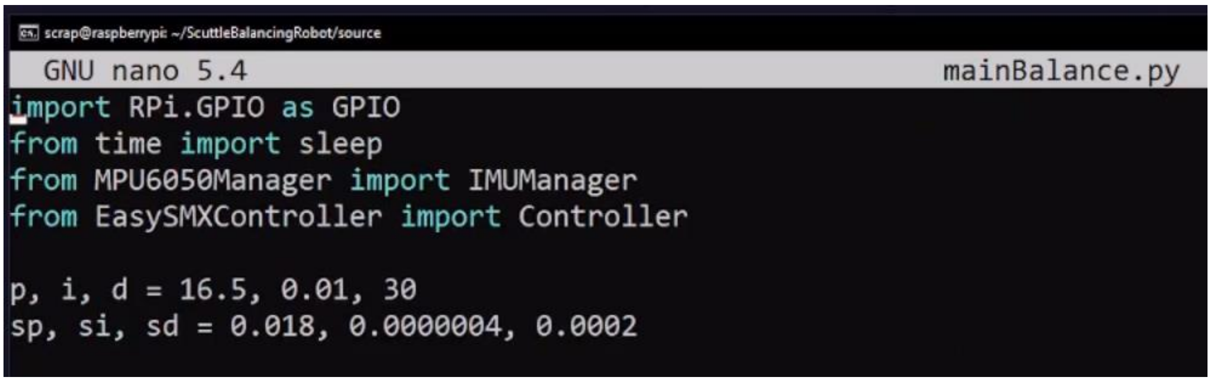
```
moveDir[1] = (pad.leftJoyX - 128) * 50 / 128
```



```
moveDir[1] = 12
```

Step 19: We can do the PID tuning in the code as follows.

When we need to fine tune the PID values, we can find the most successful values by changing the values of the p,i,d and sp,si,sd variables.



```
scrap@raspberrypi: ~/ScuttleBalancingRobot/source
  GNU nano 5.4                                      mainBalance.py
import RPi.GPIO as GPIO
from time import sleep
from MPU6050Manager import IMUManager
from EasySMXController import Controller

p, i, d = 16.5, 0.01, 30
sp, si, sd = 0.018, 0.0000004, 0.0002
```

Step 20: Congratulations. Now you can keep your SCUTTLE robot balanced on two wheels. I would like to keep this document up to date with your suggestions. Please send the deficiencies you see or the areas that need improvement to my e-mail address reductalimenitis@gmail.com , I will definitely work on them. Enjoy.