**Installing Raspbian on Raspberry Pi for Windows Users.**

* Put SD card into SD reader.
* Download and install SD card formatter: <https://www.sdcard.org/downloads/formatter_4/>
* After installation, run the software. It should automatically detect your SD card and set the format size. Click format.
* Download NOOBS zip from here: <https://www.raspberrypi.org/downloads/noobs/>
* Copy the extracted files onto the SD card that you just formatted, so that this file is at the root directory of the SD card. Please note that in some cases it may extract the files into a folder; if this is the case, then please copy across the files from inside the folder rather than the folder itself.
* After the files are copied, eject the SD card and insert it into the Raspberry Pi’s SD card slot. Boot the RPi. You’ll need an HDMI cable to a TV to view the RPi’s installation. You’ll also need a USB keyboard attached to control the RPi.
* When the OS selection screen pops up, select “Raspbian”. You may also need a mouse for this part. At the top of the screen, select “install” (or press “I” key).
* After the installation is completed, the RPi will reboot. You should see a splash screen. After a few minutes a desktop should appear.

**Configuring Raspberry Pi for our uses**

* In the top right corner, click the raspberry icon. Under preferences, click “Raspberry Pi Configuration”
* In the main tab, change boot to desktop, to boot to CLI. This will boot to the command line and is useful for SSH and will save us memory. When the system boots, to get the desktop just give the command “startx”.
* In the interfaces tab, enable camera and SSH.
* In the performance tab, I put my GPU memory as 128MB. In our application we’re not really using the GPU, but at least 128MB is required for use with the RPi camera.
* In the localization tab, I set all the fields. (locale, timezone, keyboard, wifi country).
* Press Ok, and select yes to reboot the RPi.

**Configuring RPi wireless network access**

* If you have a WiFi dongle for the RPi, or are using the RPi 3, you can configure wireless access according the this site: <https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md>

## Getting WiFi network details

To scan for WiFi networks, use the command sudo iwlist wlan0 scan. This will list all available WiFi networks, along with other useful information. Look out for:

1. 'ESSID:"testing"' is the name of the WiFi network.
2. 'IE: IEEE 802.11i/WPA2 Version 1' is the authentication used. In this case it's WPA2, the newer and more secure wireless standard which replaces WPA. This guide should work for WPA or WPA2, but may not work for WPA2 enterprise. For WEP hex keys, see the last example [here](http://www.freebsd.org/cgi/man.cgi?query=wpa_supplicant.conf&sektion=5&apropos=0&manpath=NetBSD+6.1.5). You'll also need the password for the wireless network. For most home routers, this is found on a sticker on the back of the router. The ESSID (ssid) for the examples below is testing and the password (psk) is testingPassword.

## Adding the network details to the Raspberry Pi

Open the wpa-supplicant configuration file in nano:

sudo nano /etc/wpa\_supplicant/wpa\_supplicant.conf

Go to the bottom of the file and add the following:

network={

ssid="HomeOneSSID"

psk="passwordOne"

priority=1

id\_str="homeOne"

}

network={

ssid="HomeTwoSSID"

psk="passwordTwo"

priority=2

id\_str="homeTwo"

}

Where the ssid and psk are replaced with the appropriate values for you. If you only have one network, then you only need one network block.

Now save the file by pressing Ctrl+X, then Y, then finally press Enter.

Reconfigure the interface with wpa\_cli -i wlan0 reconfigure.

You can verify whether it has successfully connected using ifconfig wlan0. If the inet addr field has an address beside it, the Raspberry Pi has connected to the network. If not, check that your password and ESSID are correct.

If your RPi has connected to your WiFi, you can now move to controlling the RPi via SSH.

**Logging into the RPi via SSH**

* On your RPi, type ifconfig. Under wlan0, the inet addr is the address of the RPi on your network.
* This will be the address you use to login via ssh. This could also be obtained via a network scan, but I think this is easier. The login username is pi, and the password is raspberry.
* Once logged in, cd into the Documents folder.
* Clone our fancy-robot repo into the folder.
* Type “git status” to make sure it worked.

**Remote file transfer**

* **I recommend the remote-ftp package in atom.**
* In atom, go to “Packages->settings view->open”.
* Under Install, search “remote”
* Install remote-ftp.
* Make sure to commit and push all your git changes, as we won’t be using git on this computer anymore.
* If you have a fancy-robot project open in atom, right click it and click remove project.
* Create a new folder, maybe on your desktop, and name it “Temporary remote files”
* In Atom, click “File->Add project folder” and click the new folder you made.
* Select “Packages->Remote FTP->Create sftp config file”.
* Change host to the address you’re using with ssh, and make sure port is 22. If you left yours default, user is pi and pass is raspberry. I left the rest of the files alone. Make sure to save the file.
* With your ssh terminal connected, in atom press CTRL-AlT-O (simultaneously). This will open a “Remote” tab near your project view. Click this, and click “Connect”. If you’ve configured everything properly, you should get a green pop-up reporting the successful connection, and your RPi’s files will appear in the remote tab.
* Your git repo in Documents is located at “home->pi->Documents->fancy-robot”.
* To open a file, just double-click it. Whenever you save your file, it will automatically transfer to the RPi. You can check this with your ssh command line. Make sure to save **very** often, as unsaved edits may be lost if there’s connection errors.

**Installing our required software packages**

* First, we will need to make sure everything is up to date. This can be done from the command line via this command: “sudo apt-get update && sudo apt-get upgrade”. This may take a while. You may need to periodically approve portions of the process.
* Next, we should install Python 3 and OpenCV.
* This is quite the process, so give it some time (probably 2 hours)
  + First, get rid of the wolfram engine on the RPi to free space, since we don’t need it.
    - sudo apt-get purge wolfram-engine
  + We also don’t need libreoffice, so we can get rid of that
    - sudo apt-get purge libreoffice\*
    - sudo apt-get clean
    - sudo apt-get autoremove
  + Install cmake
    - sudo apt-get install build-essential cmake pkg-config
  + Image I/O packages
    - sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
  + Video I/O packages
    - sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
    - sudo apt-get install libxvidcore-dev libx264-dev
  + GTK development library
    - sudo apt-get install libgtk2.0-dev libgtk-3-dev
  + OpenCV matrix optimizations
    - sudo apt-get install libatlas-base-dev gfortran
  + Install latest python (it may say it’s already installed)
    - sudo apt-get install python2.7-dev python3-dev
  + Download OpenCV source
    - cd ~
    - wget -O opencv.zip <https://github.com/Itseez/opencv/archive/3.3.0.zip>
    - unzip opencv.zip
    - rm opencv.zip
    - wget -O opencv\_contrib.zip <https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip>
    - unzip opencv\_contrib.zip
    - rm opencv\_contrib.zip
  + Install python package manager pip
    - wget <https://bootstrap.pypa.io/get-pip.py>
    - sudo python get-pip.py
    - sudo python3 get-pip.py
  + Install numpy, a math and matrix library for python
    - pip install numpy
  + Setup cmake
    - cd ~/opencv-3.3.0/
    - mkdir build
    - cd build
    - cmake -D CMAKE\_BUILD\_TYPE=RELEASE \

 -D CMAKE\_INSTALL\_PREFIX=/usr/local \

-D INSTALL\_PYTHON\_EXAMPLES=ON \

 -D OPENCV\_EXTRA\_MODULES\_PATH=~/opencv\_contrib-3.3.0/modules \

 -D BUILD\_EXAMPLES=ON ..

* + - Note, you press enter after each “\” in the above command
    - Hopefully this command should go without error…
    - Edit the /etc/dphys-swapfile and set CONF\_SWAPSIZE=1024.
    - sudo /etc/init.d/dphys-swapfile stop
    - sudo /etc/init.d/dphys-swapfile start
  + Compile OpenCV
    - make -j4 (note this may take 1-3 hrs, depending)
    - sudo make install
    - sudo ldconfig
  + Rename .so file
    - cd /usr/local/lib/python3.5/site-packages/
    - sudo mv cv2.cpython-35m-arm-linux-gnueabihf.so cv2.so

Note, if this directory doesn’t exist, replace “site-packages” with “dist-packages”.

If the .so file doesn’t exist, use the name of the .so file in the directory, it may be slightly different.

* Verify that OpenCV can be imported:
  + python
  + import cv2
  + cv2.\_\_version\_\_

This should give back ‘3.3.0

* Remove the two temporary directories (only do this if all above steps were good)
  + cd ~
  + rm -rf opencv-3.3.0 opencv\_contrib-3.3.0
* Change back the CONF\_SWAPSIZE
  + Change CONF\_SWAPSIZE in /etc/dphys-swapfile to 100
  + sudo /etc/init.d/dphys-swapfile stop
  + sudo /etc/init.d/dphys-swapfile start

Verify the camera works with raspistill -o output.jpg and look at the picture that is outputted 😊.

* Install picamera library
  + sudo pip install "picamera[array]"
* Install scikit
  + sudo pip install scikit-image
* Install imutils
  + sudo pip install imutils