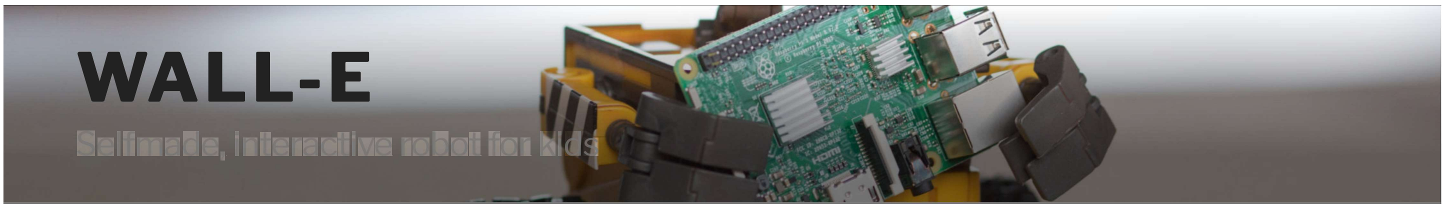


SETTING UP YOUR RASPBERRY PI

Download and install current Raspbian image

Here is a quick summary in case you have a Mac on how to install the image on an empty SD card. You will need to use a Micro-SD card adapter for your Mac. Please note that all content of the SD card will be deleted!

Open the "Terminal" on Mac and enter the follwing **commands**:

```
diskutil list
```

```
diskutil unmountDisk /dev/disk#
1. Replace "#" by the number the list command showed
```

```
cd /users/#user#/desktop
1. Replace"#user#" with your user (Use shell command: id -un to get the name)
2. Unpack the downloaded Raspbian image to desktop and renamed it to "raspbian.in
```

```
sudo dd bs=1m if=raspbian.img of=/dev/rdisk#
1. Replace "#" by the number the list command showed
```

Wait to be finished. You can check the progress by pressing Ctrl+T

Prepare Pi for headless boot and connect to WiFi

1. On your computer create a file called “wpa_supplicant.conf” which includes our **wifi configuration**
2. Open it with a text editor
3. Copy & paste the following lines into it:

```
country=DE
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
 ssid="#WIFI_NAME#"
 psk="#WIFI_PASSWORD#"
 key_mgmt=WPA-PSK
}
```

Replace #WIFI_NAME# and #WIFI_PASSWORD# accordingly and save this file to the root directory of the SD card most likely mounted as “boot”. When Raspbian boots it will automatically copies to the right location (/etc/wpa_supplicant/) and connects to the network.

In order to access the Pi from remote, we need to **activate SSH**

1. Create a file called “ssh” e.g. by using a text editor
2. Copy it to the root directory of the SD card as well

Plug the SD-Card, **power up the Pi** and wait until it is up and running.

Check your wifi router to see which IP address your Pi got. I recommend to configure the router to assign a fixed (reserved) IP address to the Pi. This is done by using the MAC address. Usually routers have an overview to show you all connected devices. Watch out for a hostname “raspberrypi”. In my example the Pi’s IP address is: 192.168.0.3 but can be totally different in your case!

You can now **access the Pi using ssh**. The following commands will be entered in the Terminal (aka Unix shell) for Mac and Linux. On Windows you can use putty or any other SSH client.

```
ssh pi@192.168.0.3

You might get the following warning:
The authenticity of host '192.168.0.3 (192.168.0.3)' can't be established.
ECDSA key fingerprint is SHA256:dr1234af3434....
Are you sure you want to continue connecting (yes/no)?

Enter 'yes'
Enter default Password: "raspberry"
```

Security matters! – a few necessary changes

Set a **new hostname**. This is not only a security issue but makes working with the Pi a bit more individual:

```
sudo nano /etc/hosts
Replace 'raspberrypi' mit 'wall-e'. It should look then like this:
127.0.1.1       wall-e

Close the editor and save changes by pressing:
Ctrl+X
Y + [Enter]
```

```
sudo nano /etc/hostname
Replace the only word in the file 'raspberrypi' with 'wall-e'.

Close the editor and save changes by pressing:
Ctrl+X
Y + [Enter]

sudo hostnamectl set-hostname wall-e
```

Exit the shell and connect again. You will see the new hostname:

```
exit
ssh pi@192.168.0.3
```

Change default password:

```
passwd
1. Enter current password (e.g. rapsberry)
2. Enter and retype new password
3. The shell outputs: 'passwd: password updated successfully'
```

```
sudo reboot
...wait...
ssh pi@192.168.0.3
```

Create new SSH keys because of new hostname changed and because all Pis have the same default keys:

```
sudo rm /etc/ssh/ssh_host_*
sudo dpkg-reconfigure openssh-server
sudo service ssh restart
exit
```

Delete old SSH keys on Mac:

```
nano /Users/FamPa/.ssh/known_hosts
Position cursor on line with IP-address of PI and use Ctrl+K to delete entire lir

Close the editor and save changes by pressing:
Ctrl+X
Y + [Enter]
```

Encrypt your wifi password:
To allow headless boot to wifi we originally entered the password in plain text. For security reasons I highly recommend to encrypt it:

```
ssh pi@192.168.0.3
sudo -i

wpa_passphrase '#WIFI_NAME#' '#WIFI_PASSWORD#' >> /etc/wpa_supplicant/wpa_supplic
1. Replace #WIFI_NAME# and #WIFI_PASSWORD# according to your network you want to

exit
```

Open the file ‘wpa_supplicant.conf’. You will see two network configurations: The first one is the plain text we entered for headless boot.

The second one is the configuration we just injected by the commands above. As you can see psk is encrypted as hash value .

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf

country=DE
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
  ssid="#WIFI_NAME#"
  psk="#WIFI_PASSWORD#"
  key_mgmt=WPA-PSK
}
network={
  ssid="#WIFI_NAME#"
  #psk="#WIFI_PASSWORD#"
  psk=73e083.....
}
```

Delete all text in red with Ctrl+K and close the editor by pressing Ctrl+X and Y + [Enter].

Tweak for stable WiFi connection

To ensure a stable wifi connection, we have to **avoid WiFi drop on Raspberry Pi**:

```
sudo iw dev wlan0 set power_save off
sudo nano /usr/local/bin/checkwifi.sh
```

Copy & paste the following script. This script will be called periodically and checks whether the gateway (typically our router) can be reached. If not, the script will shut down the wifi adapter and restarts it:

```
now="$(date)"

GATEWAY_IP=$(/sbin/ip route | awk '/default/ { print $3 }')

echo "$now: Checking wifi connection with $GATEWAY_IP" >> /usr/local/bin/checkwi

ping -c1 $GATEWAY_IP 1>/dev/null 2>/dev/null
SUCCESS=$?

if [ $SUCCESS -eq 0 ]
then
  echo "$now: all ok!" >> /usr/local/bin/checkwifi.log
```

```
else
  echo "$now: cannot reach gateway, no network connection, restarting wlan0" >> /
  /sbin/ifdown 'wlan0'
  sleep 5
  /sbin/ifup --force 'wlan0'
fi
```

Close the editor by pressing Ctrl+X and Y + [Enter].

Now we need to register this shell script to be called periodically:

```
crontab -e
1. select 2 (nano)
2. add this line at the end:
*/5 * * * * /usr/bin/sudo -H /usr/local/bin/checkwifi.sh >> /dev/null 2>&1
```

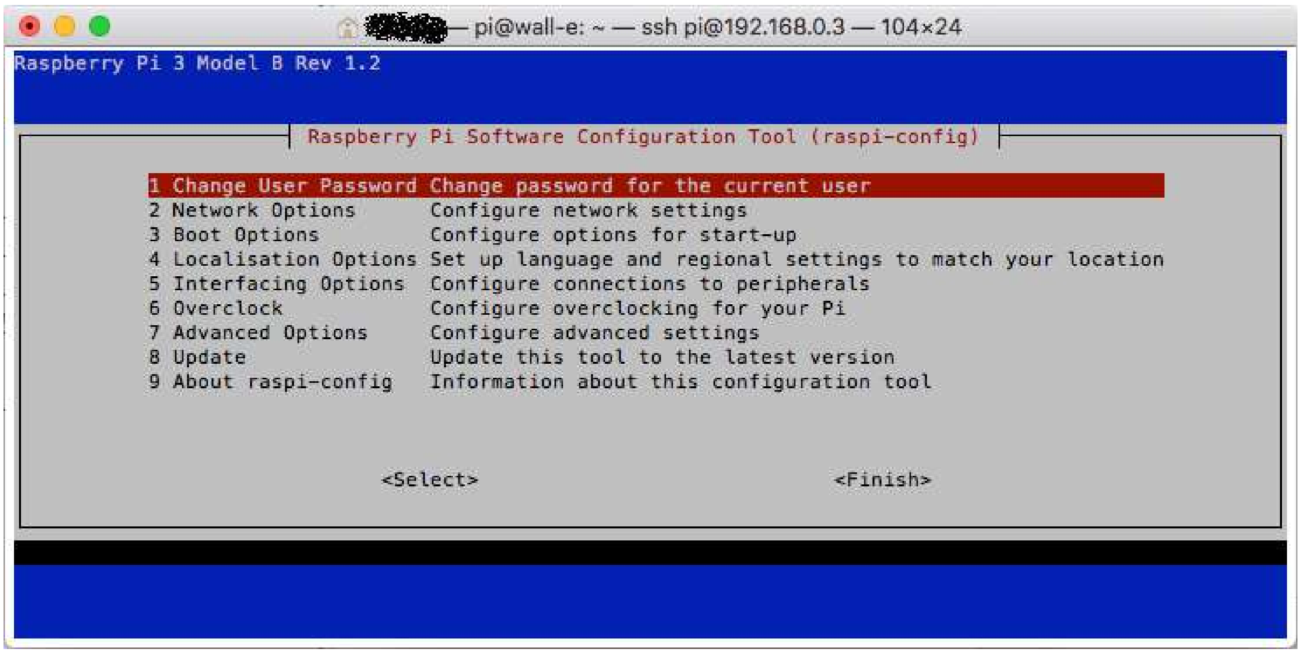
You can check whether the script is working by calling it from the shell:

```
sudo /usr/local/bin/checkwifi.sh
Check content of log file:
sudo cat /usr/local/bin/checkwifi.log
Example:
Sat 30 Jun 16:25:01 UTC 2018: Checking wifi connection with 192.168.0.1
Sat 30 Jun 16:25:01 UTC 2018: all ok!
```

Additional settings using the configuration tool

This settings will be all done using the shell based Raspbian Configuration Tool:

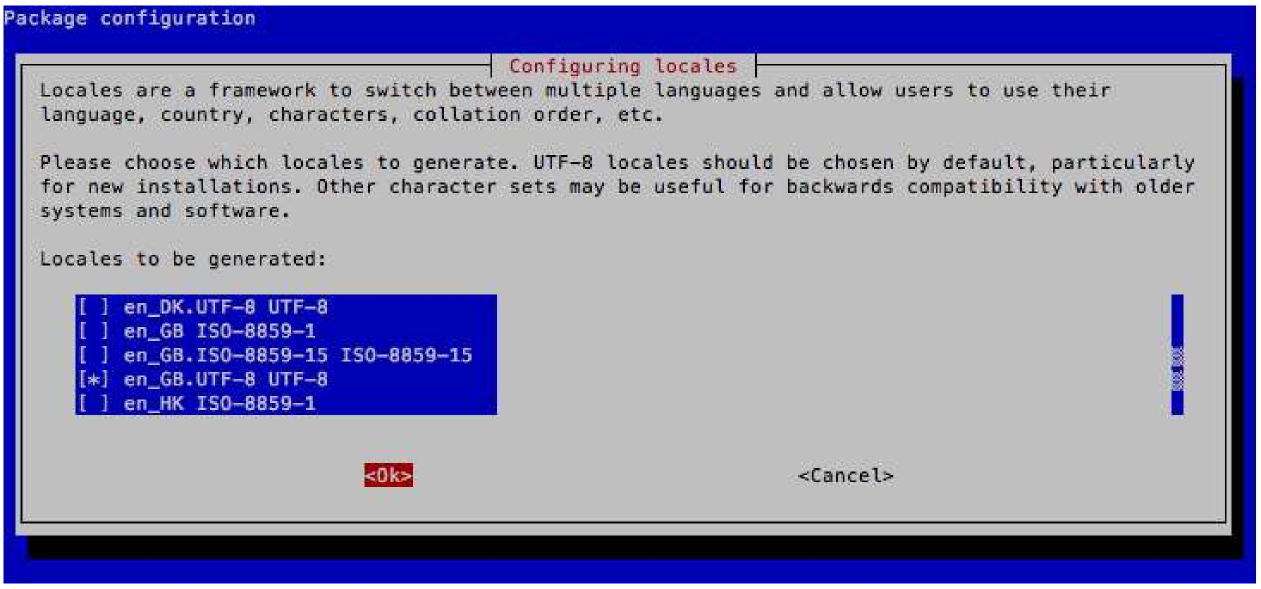
```
sudo raspi-config
```



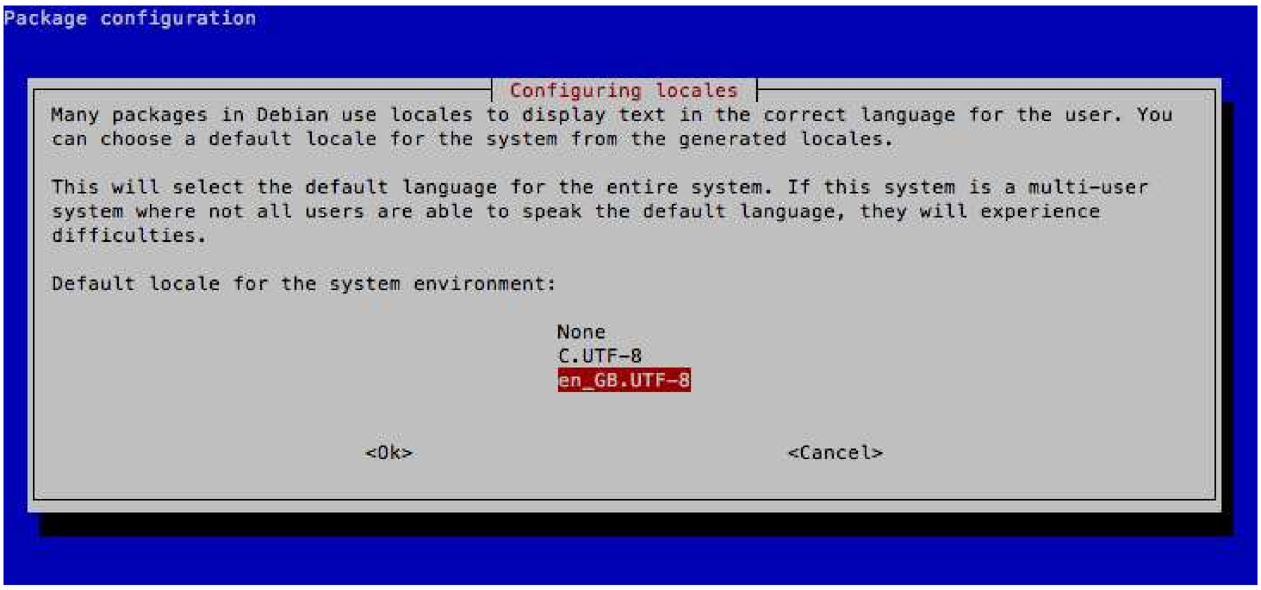
```
[redacted] — pi@wall-e: ~ — ssh pi@192.168.0.3 — 104×24
Raspberry Pi 3 Model B Rev 1.2

              ┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

           1 Change User Password Change password for the current user
           2 Network Options      Configure network settings
           3 Boot Options         Configure options for start-up
           4 Localisation Options Set up language and regional settings to match your location
           5 Interfacing Options  Configure connections to peripherals
           6 Overclock            Configure overclocking for your Pi
           7 Advanced Options     Configure advanced settings
           8 Update               Update this tool to the latest version
           9 About raspi-config   Information about this configuration tool


                        <Select>                             <Finish>
```

Select “**Localisation Options**”:

- Change Locale: choose your UI language and home country



```
Package configuration

                        ┤ Configuring locales ├
  Locales are a framework to switch between multiple languages and allow users to use their
  language, country, characters, collation order, etc.

  Please choose which locales to generate. UTF-8 locales should be chosen by default, particularly
  for new installations. Other character sets may be useful for backwards compatibility with older
  systems and software.

  Locales to be generated:

       [ ] en_DK.UTF-8 UTF-8
       [ ] en_GB ISO-8859-1
       [ ] en_GB.ISO-8859-15 ISO-8859-15
       [*] en_GB.UTF-8 UTF-8
       [ ] en_HK ISO-8859-1


                     <Ok>                                 <Cancel>
```



```
Package configuration

                         ┤ Configuring locales ├
  Many packages in Debian use locales to display text in the correct language for the user. You
  can choose a default locale for the system from the generated locales.

  This will select the default language for the entire system. If this system is a multi-user
  system where not all users are able to speak the default language, they will experience
  difficulties.

  Default locale for the system environment:

                                   None
                                   C.UTF-8
                                   en_GB.UTF-8


                     <Ok>                                 <Cancel>
```

- Change Timezone: Geographic area and time zone
- Change Keyboard layout: Follow the dialog, e.g. Generic 105-key (Intl),…
- Change WiFi Country

Select “**Interfacing Options**“:

- Camera enable
- VNC enable
- I2C enable

Select “**Advanced Options**“:

- Expand Filesystem

Once back at the main menu select finish and exit raspi-config and confirm reboot.

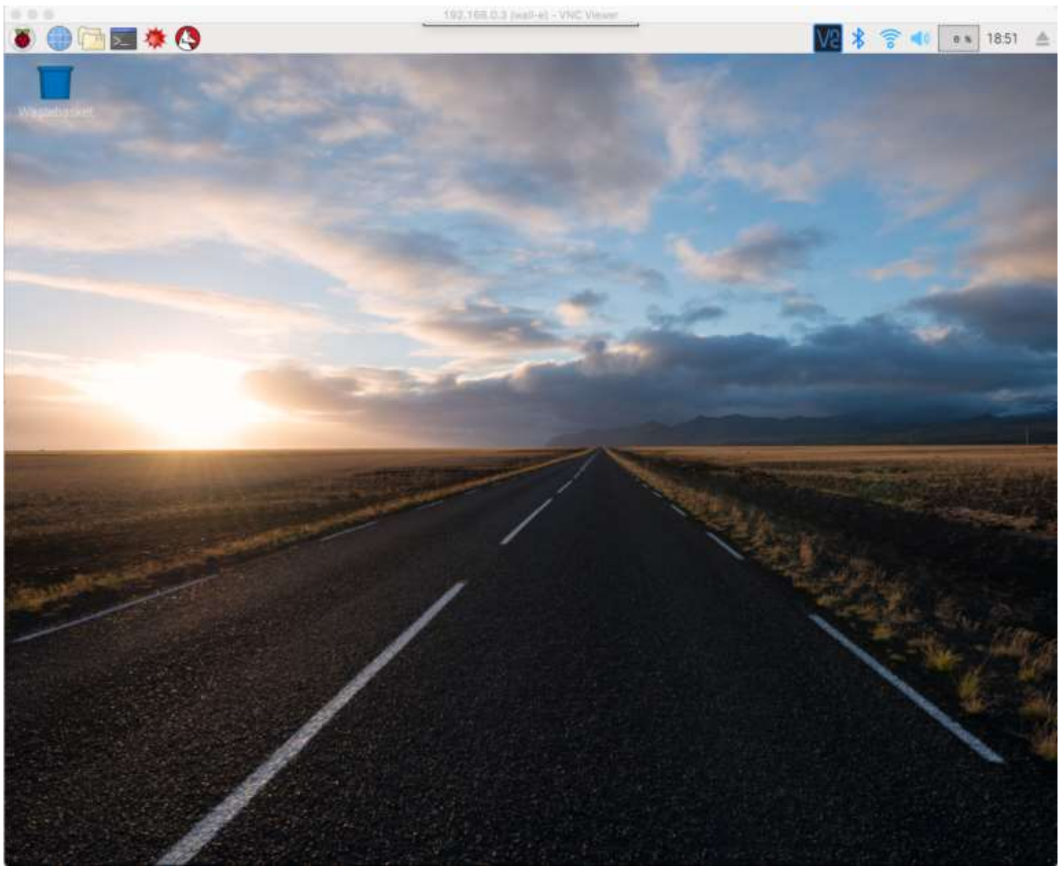
…wait…

Set up VNC server

Wouldn’t it by nice to finally see the see the Desktop of your Pi?

1. Download realvnc for your Mac, Linux or PC: https://www.realvnc.com/en/connect/download/vnc/
2. In the user interface of realvnc enter the IP address of your Pi
3. Use “pi” as username and your password

4. You should see a screen like this:

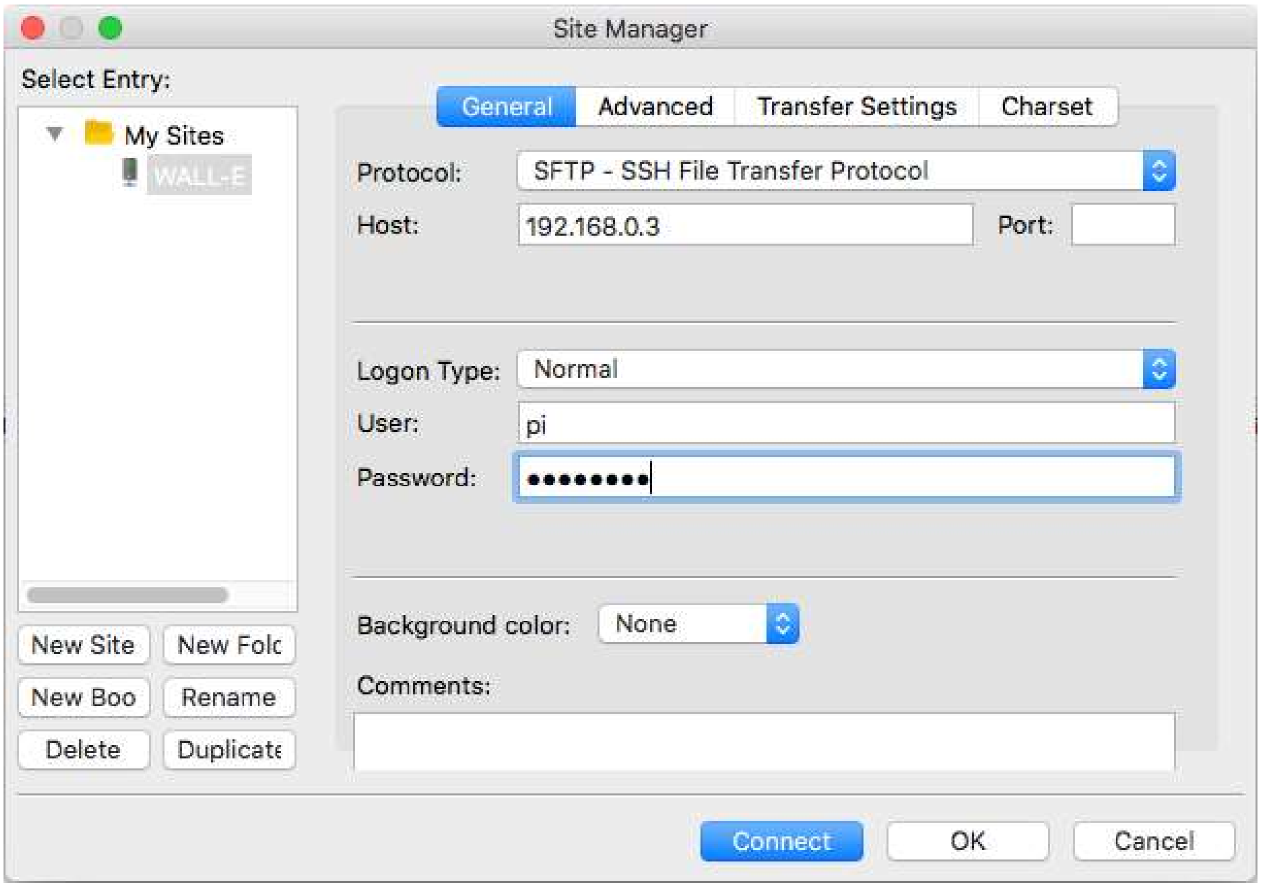


Set up FTP

FTP allows you to easily copy files to and from the Pi.

1. Download filezilla for your Mac, Linux or PC: https://filezilla-project.org/download.php
2. In Filezilla: Navigate in Menu to File / Site Manager
3. Press “New Site”, give it a name e.g. “WALL-E”

4. Select SFTP, IP address, user, password:


Select Entry:
My Sites
WALL-E
Protocol: SFTP - SSH File Transfer Protocol
Host: 192.168.0.3
Port:
Logon Type: Normal
User: pi
Password: •••••••••
Background color: None
Comments:

Press Connect

I²C configuration

Install Python support

Most of the sensors and actors are connected as device on the I²C bus. We are going to access such devices in Python. So we need to install all necessary packages:

```
sudo apt-get install python-smbus python3-smbus python-dev python3-dev
sudo apt-get install i2c-tools
sudo adduser pi i2c
```

If your PCA9685 I²C Servo Board is already connected enter the following command to scan the I²C bus:

```
i2cdetect -y 1

This should be the output in case you use the default address (0x40) of the boarc
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
```

```
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: 40 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- --
```

Install driver for PCA9685 I^{2}C Servo Board for Python3:

```
sudo pip3 install Adafruit-PCA9685
```

Install additional Python packages

```
sudo pip3 install Flask
Necessary for REST communication
sudo pip3 install flask-restful
Extension for Flask to setup up a REST API faster and easier
sudo pip3 install requests
Necessary to handle http requests
```

Install Apache 2 Webserver

With the following commands we install Apache and configure the file system so we are able to actually copy html-files into the root directory of our webpage.

```
1. This command installs the most current version
sudo apt-get install apache2

2. Create a user group 'www-data' (most likely exists already) and add user 'pi'
sudo groupadd www-data
sudo usermod -aG www-data www-data
sudo usermod -aG www-data pi

3. Grant access rights to this group
sudo chown -R pi:www-data /var/www/html/
sudo chmod -R 770 /var/www/html/

4. Remove any existing log file and restart Apache:
```

```
sudo rm /var/log/apache2/error.log
sudo /etc/init.d/apache2 restart
```

To check whether the server is up and running correctly navigate with a browser to the IP address of your Pi (e.g. http://192.168.0.3). You should see a default webpage of Apache2. If not, check the log file:

```
sudo cat /var/log/apache2/error.log
```

In order to check whether you can actually create and copy your on html files into the root directory create on your Mac, Linux or PC a file called "index.html" and add the following text:

```
<!doctype html>
<html>
 <head>
  <meta charset="utf-8">
  <title>WALL-E Test Page</title>
 </head>
 <body>
  <h1>SUCCESS: Apache 2 is loading you own website</h1>
 </body>
</html>
```

Use FTP to upload index.html to the directory "/var/www/html" and navigate again with a browser to the IP address of your Pi.

Autostart of e.g. python scripts at boot-up of Pi

```
sudo nano /etc/rc.local
Before the command line "exit 0" add the following line:

python3 /home/pi/wall-e/main.py &
Important: DO NOT forget "&" otherwise it will not run in a parallel process and
```

Get latest updates

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get clean

sudo reboot
```

You did it! Your Pi is now ready....