Vivarium Pi Installation Guide:

A Complete Guide for Beginners

Equipment

The equipment that you will need for this Vivarium-Raspberry Pi Project is listed below.

Keep in mind the breadboard, and the breadboard wire is optional, but I highly recommend investing in these as they may work working circuitry a lot easier.

Recommended

- Raspberry Pi
- Micro SD Card or a SD Card if you're still using an old version of the Pi.
- 6 x DS18B20 Temperature Sensor
- 2 x DHT11 Humidity Sensor
- 2 x DHT22 Humidity Sensor
- 2 x Hx711 1kg Weight Sensor
- 2 x MQ4 Gas Sensor (Methane, CNG)
- 10k ohm resistors (Brown, Black, Orange, Gold)
- 4.7k Resistors
- GPIO breakout kit
- 16 Channel Relay
- Breadboard
- Breadboard Wire

Optional

- Raspberry Pi Case
- USB Keyboard
- USB Mouse
- Ethernet Cord or Wi-Fi dongle (If you want networking, keep in mind the Raspberry Pi 3 has inbuilt Wi-Fi.)

The Steps on how to Install Raspian

The process of installing Raspbian is super easy and won't take you long to get it up and going. There are two different flavors of Raspbian you can install which I will briefly just mention below.

Raspbian: This is the full version of the operating system and is the version that most people will use. It comes preinstalled with an extensive range of apps and is very easy to get going with. It also comes with the desktop software preinstalled.

Raspbian Lite: This is the light version of the Raspbian operating system and contains the bare basics to get going. If you know exactly what you want to do, then this may be the better version to install.

Raspbian Lite doesn't have the desktop software pre-installed so you will only be able to use the command line. This version is ideal if you're trying to save on system resources and know you will unlikely need the desktop graphical user interface (GUI). It makes it great if you want a headless server. (e.g. Connecting the Pi to a screen will just give you the terminal).

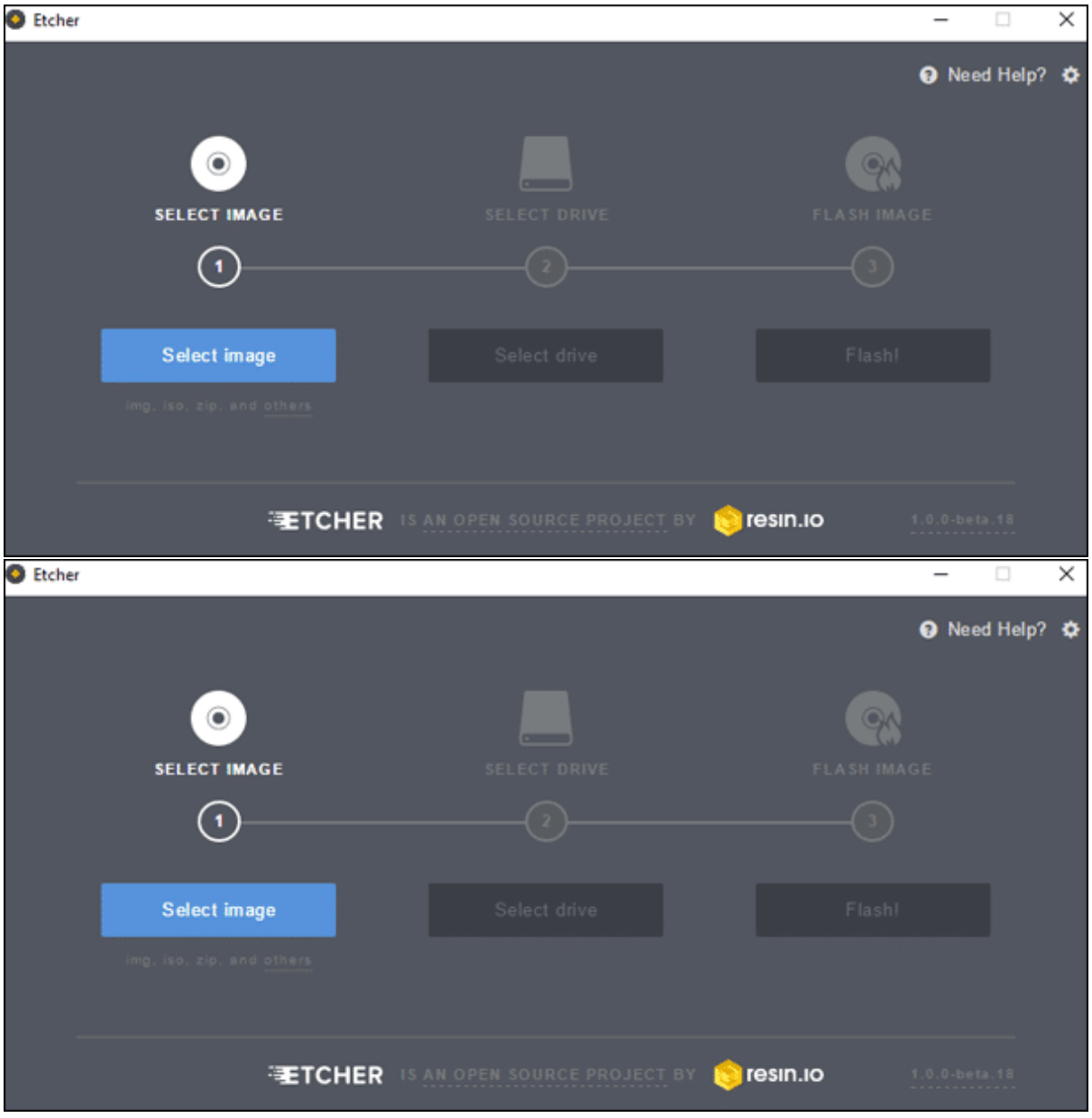
1. First, download the version of Raspbian you wish to install, you can find both the direct download links and torrents here.

2. Once downloaded, unzip the zip file, so you're left with just a *.img* file. If you're having trouble unzipping the file you may need to install something like **7zip**.

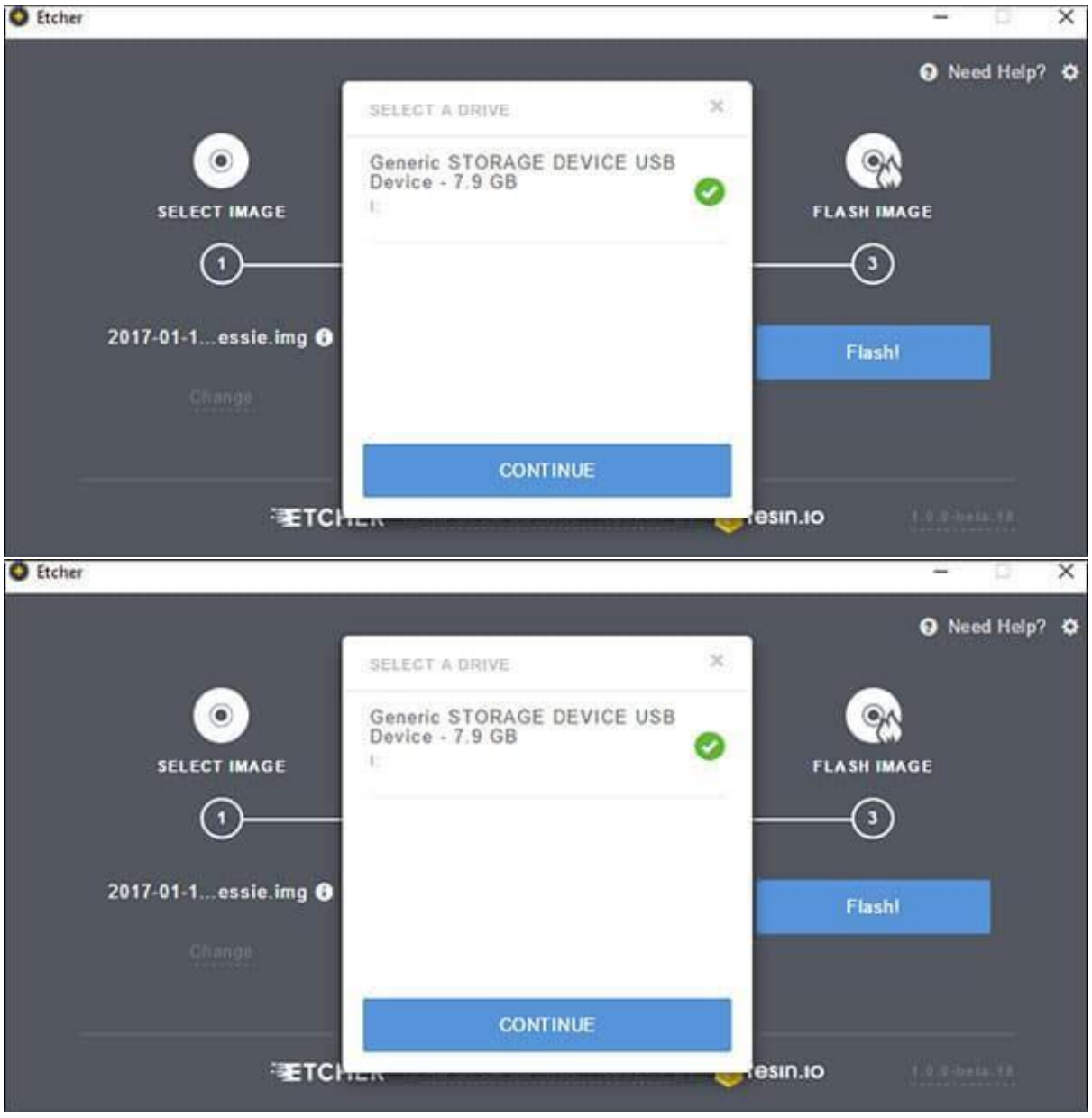
3. Now you will need to download & install the software that will write the image to the SD card. A good one to use is called Etcher, and it works on all platforms (Linux, Mac, and Windows). You're able to download it from over at their website here.

4. Once you have downloaded Etcher, follow the prompts to install it.

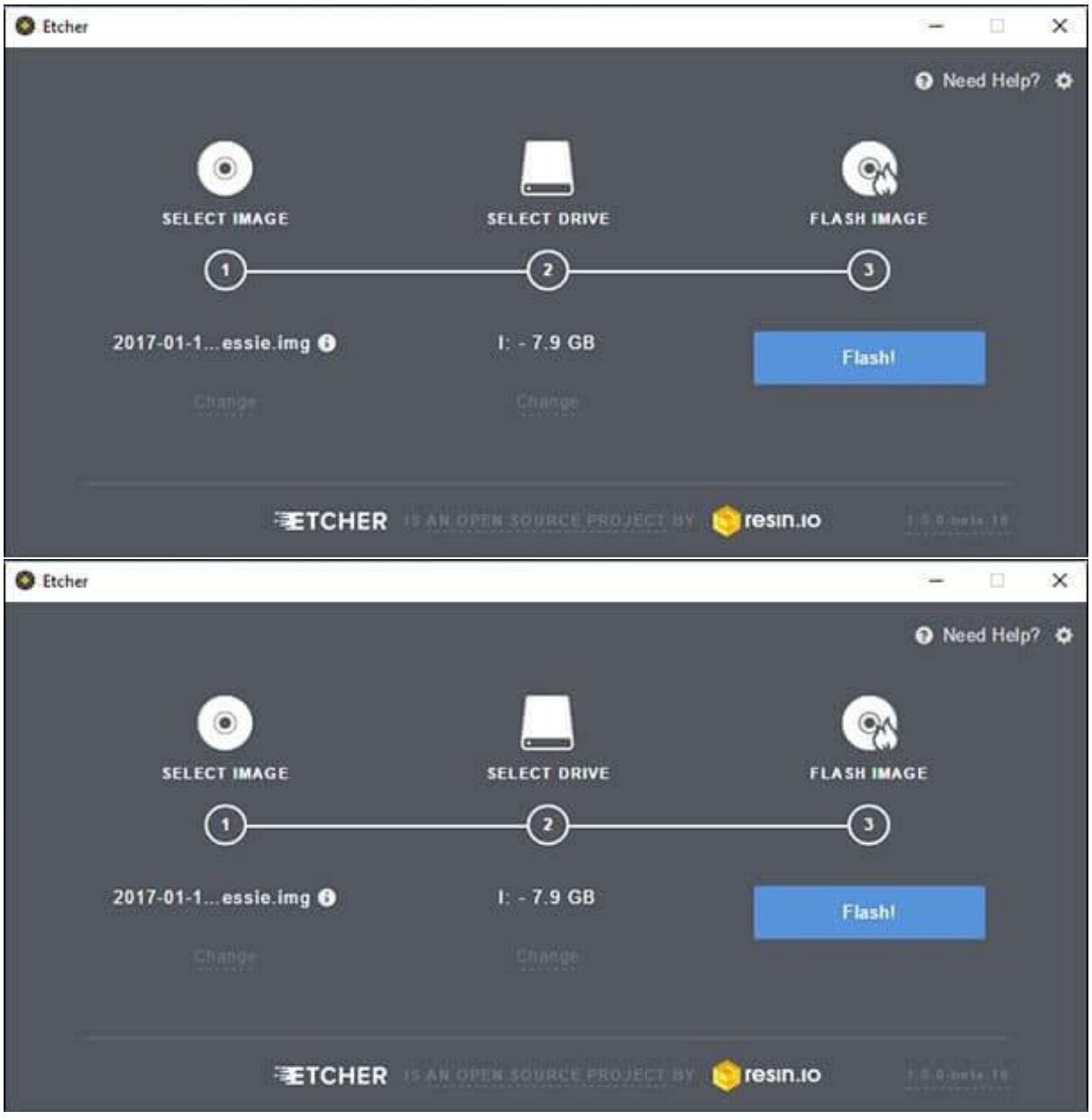
5. Open Etcher and select the chosen Raspbian image.



6. Select the SD card that you wish to install Raspbian on. Double check to make sure it's the correct drive as this will wipe it clean.



7. Once you have confirmed the image and the drive you can then proceed to flash the SD card, select Flash.



8. Once it's finished, you can safely remove the SD card from the computer.

9. Insert into your Raspberry Pi and any other extra cords such as power, mouse, keyboard, and the HDMI cord.

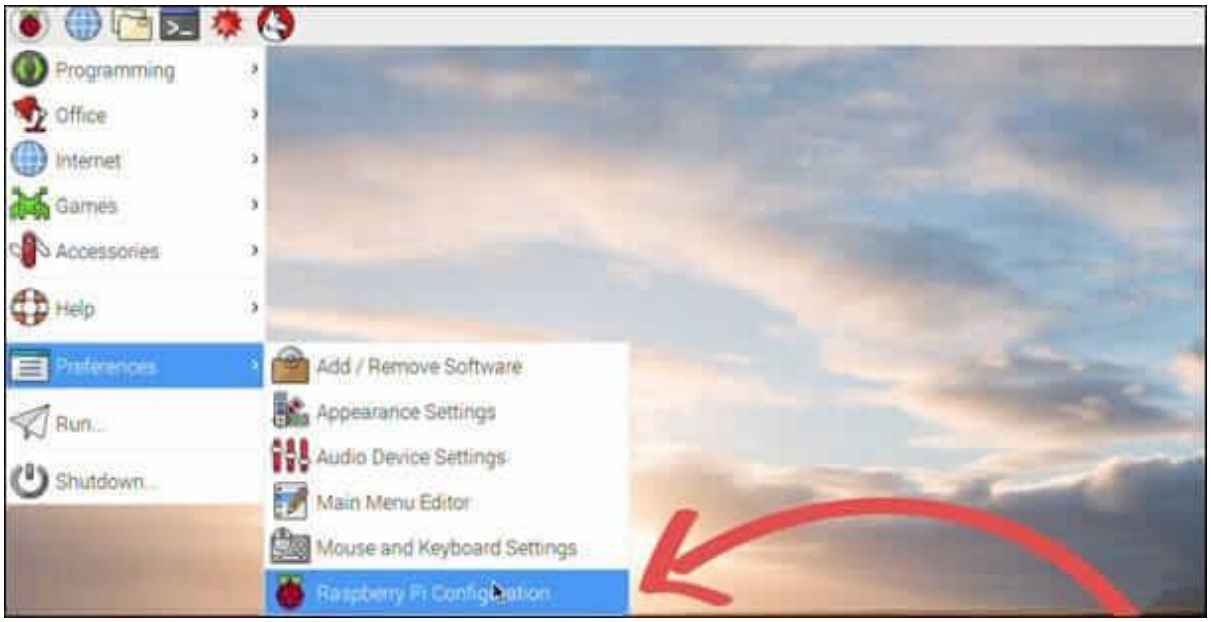
Basic Raspbian Setup

Depending on which version of Raspbian you installed there is minimal set up you will need to do to be able to get going. It's important that when you learn how to install Raspbian that you also learn how to configure it correctly. Below are a few

recommendations of what you should change. Please note that these are my personal recommendations.

Open the configuration menu by going to the **menu icon -> preferences -> Raspberry Pi configuration**. Please note the location of some menu items may change over time. You can also open up a similar menu in the terminal by entering the following command.

```
sudo raspi-config
```



System

Firstly, you will want to expand the file system, so it takes up the entire SD Card. Simply select this in the options, and it will expand the file system. (You can find this option in the terminal by using `sudo raspi-config`)

Since the **default password** is **raspberry** and the **default username** is **pi**, this makes it extremely easy for someone to access your Raspberry Pi. So you should change the default password for the user **pi**, this can easily be changed in the systems tab.

You can also change it in the terminal by using the following command.

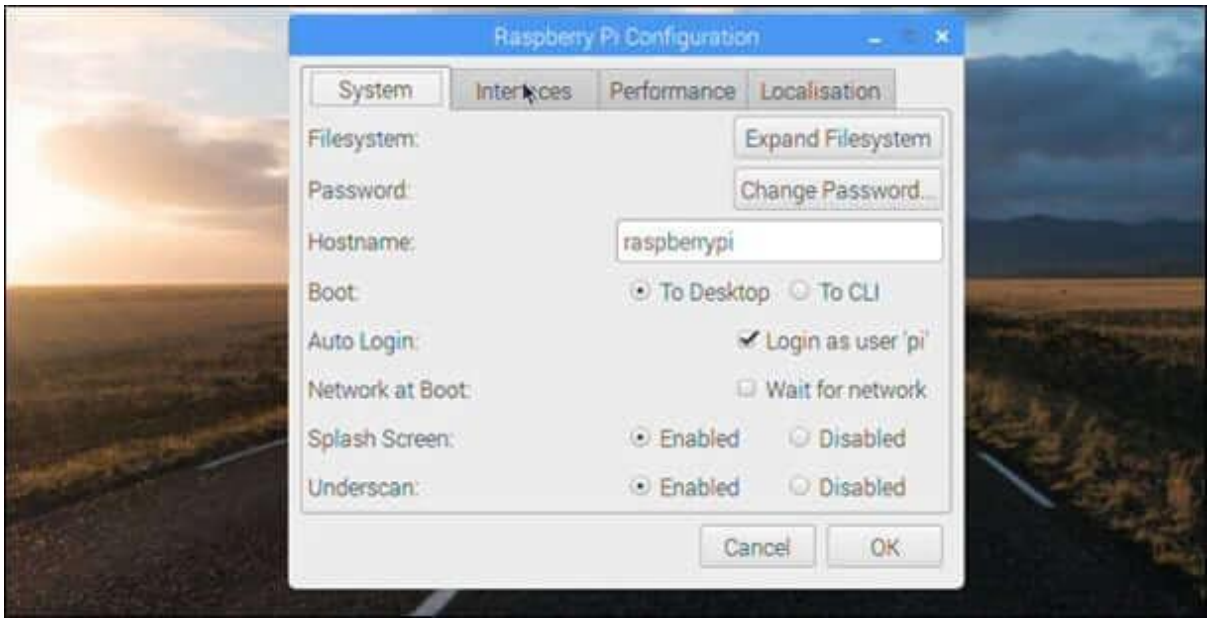
```
passwd
```

If you installed the full version of Raspbian but want to login to the command interface rather than the GUI then you can change this in the settings. You can start the interface by running the following line.

```
startx
```

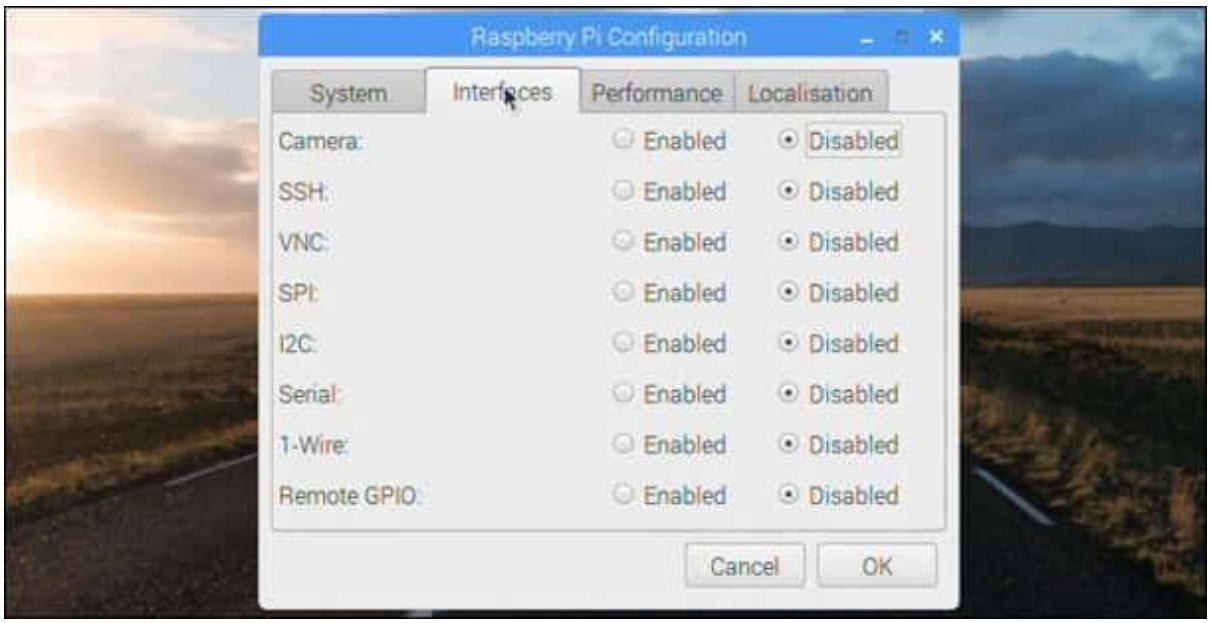
You may also want to turn off auto login if you don't want someone to turn on the pi and have access.

If you have black bars around your screen, then turn off underscan to get rid of them. If this doesn't remove them, then you may need to tinker with your monitor settings directly.



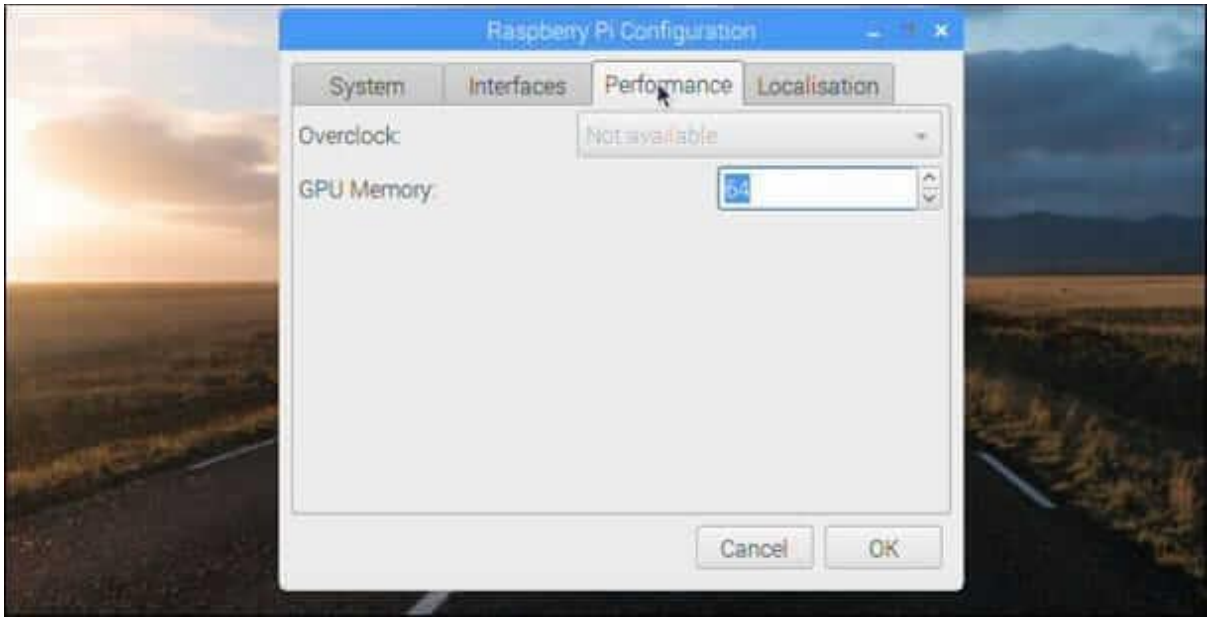
Interfaces

You can keep all these options off unless you know you will need them on. For example, if you're going to be connecting to the Pi using SSH then make sure you switch SSH on.



Performance

You can keep all the options in here as the defaults unless you have been instructed to change them for a specific tutorial.



Localisation

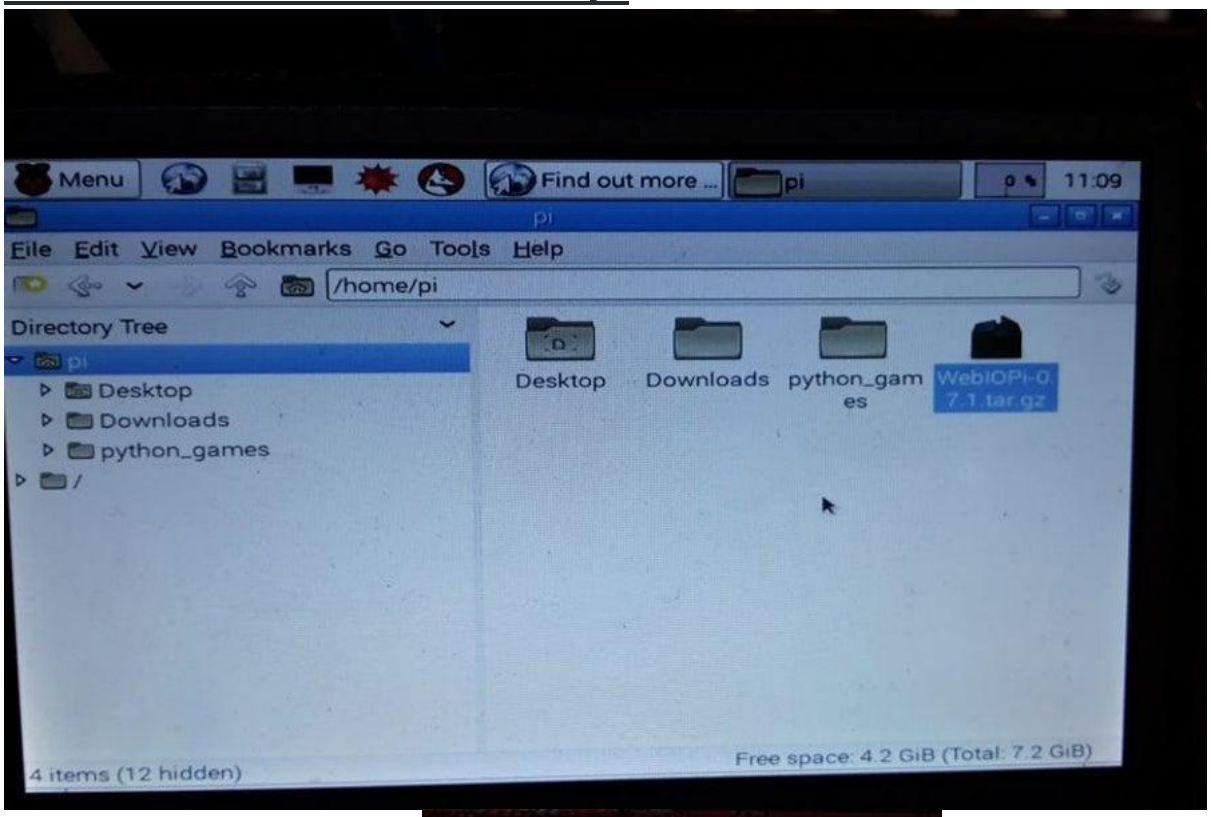
You will probably need to change all the settings in this the localisation menu, simply set these to wherever you live. The most important one you will want to change is the keyboard, there is nothing worse than going to type and having the wrong symbols pop up.

Vivarium Pi Installation Guide:

A Complete Guide for Beginners

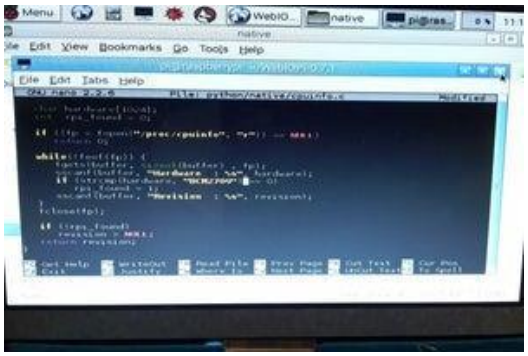
You're now able to start doing a Raspberry Pi project or using it just like you would use a regular computer. Its range of applications makes it easy enough to do most typical computing tasks.

Download & Install Webiopi



Vivarium Pi Installation Guide:

A Complete Guide for Beginners



1. Download file webiopi on http://webiopi.trouch.com/DOWNLOADS.html and adapt x.y.z with the version you download. Put the file to /home/pi

2. Open LXTerminal and write the step

- tar xvzf WebIOPi-x.y.z.tar.gz
- cd WebIOPi-x.y.z

3. For only Raspberry pi 2, must change the following step

- sudo nano python/native/cpuinfo.c change 'BCM2708' to 'BCM2709'
- sudo nano python/native/gpio.c '#define BCM2708_PERI_BASE 0x20000000' to '#define BCM2708_PERI_BASE 0x3f000000'

4. Then install webiopi

- wget https://raw.githubusercontent.com/doublebind/raspi/master/webiopi-pi2bplus.patch
- patch -p1 -i webiopi-pi2bplus.patch
- sudo ./setup.sh
- Setup may take a moment

5. Run your webiopi

- sudo webiopi -d-c /etc/webiopi/config
- You are done, and ready to enjoy WebIOPi! But the server and GPIO state will be lost when you'll stop the script (CTRL-C) or close the terminal

How to Start WebIOPi

Follow the steps below if Raspbian is installed by NOOBS later than 1.4.2.

cd /etc/systemd/system/
sudo wget https://raw.githubusercontent.com/doublebind/raspi/master/webiopi.service
sudo systemctl start webiopi
sudo systemctl enable webiopi

6. Start/Stop WebIOPi

- sudo /etc/init.d/webiopi start
- sudo /etc/init.d/webiopi restart
- sudo /etc/init.d/webiopi stop

Auto start at boot

To setup your system to start webiopi at boot :

```
$ sudo update-rc.d webiopi defaults
```

To remove webiopi start from boot :

```
$ sudo update-rc.d webiopi remove
```

7. Access WebIOPi over local network

- If your Raspberry Pi is connected to your network, you can open a browser to http://raspberrypi:8000/ with any device of your network. Replace raspberrypi by its IP
- Default user is 'webiopi' and password is 'raspberry'

8. Begin the script

- Create a folder somewhere on your P, for instance /home/pi/myproject. This is main folder
- Create another python folder in the previously created. I will store Python script file here
- Create another html folder next to python. We will store HTML and other resources here

9. Configuration

- Edit /etc/webiopi/config
  - Script
    - myproject =/home/pi/myproject/python/script.py
  - Http
    - doc-root=/home/pi/myproject/html
- Remember to start your WebIOPi
  - $sudo /etc/init.d/webiopi start

Python Script, Html File and Connection Raspberry Pi and Relay Module

1. Inside script.py file

- http://sourceforge.net/projects/script-py-file/files/script.py/download

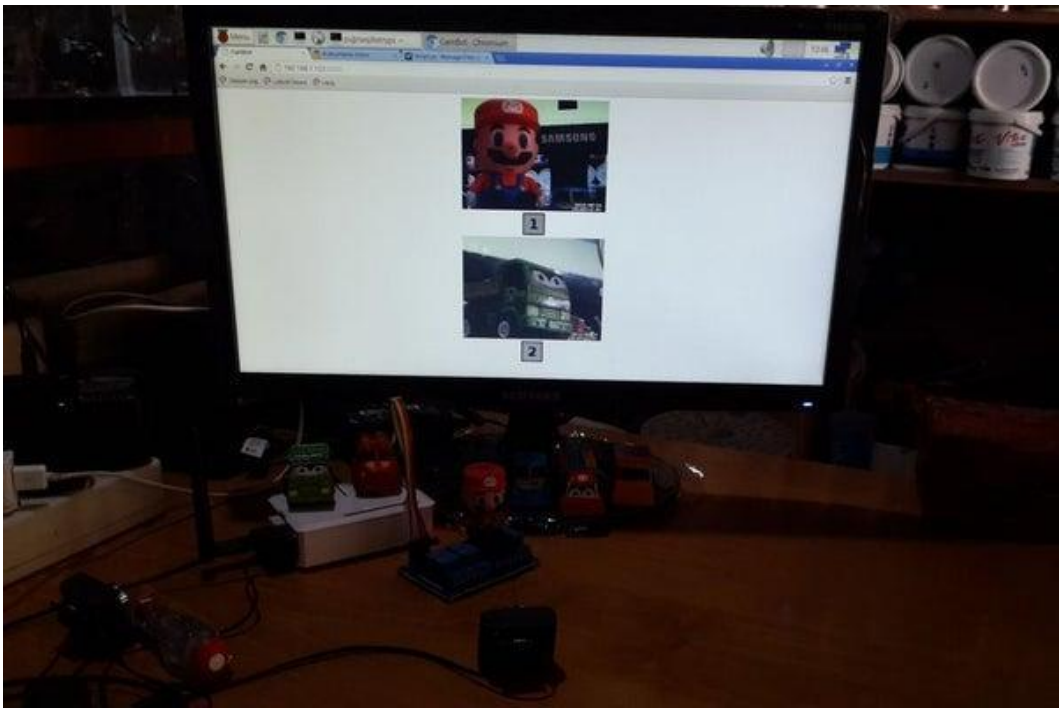
2. inside html file

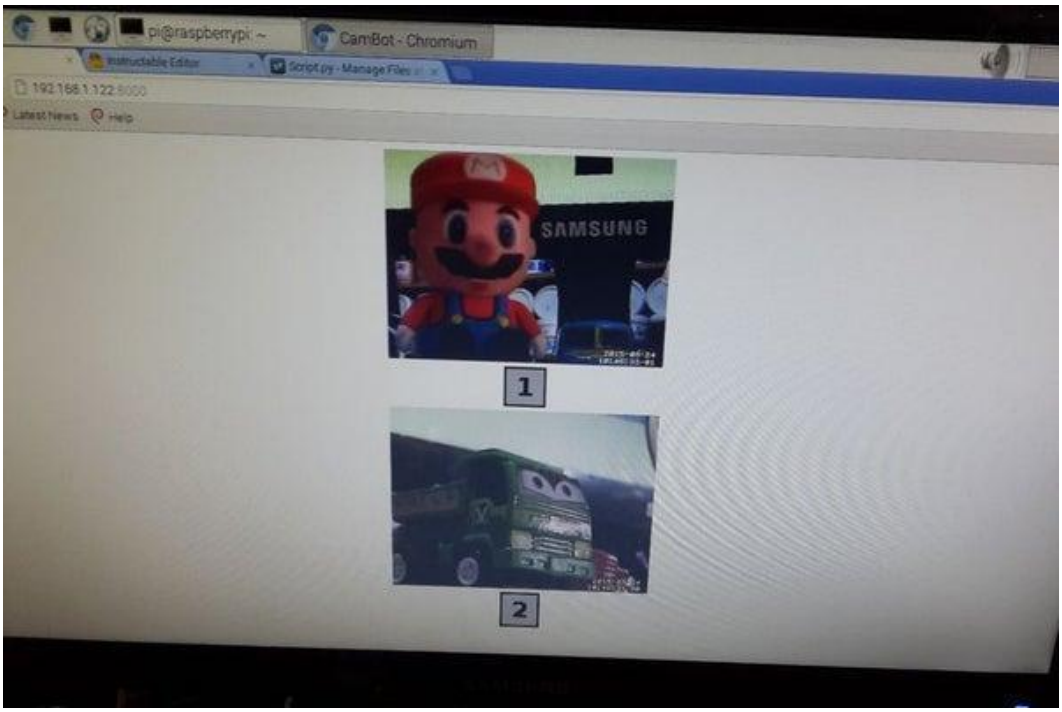
- http://sourceforge.net/projects/script-py-file/files/index.html/download

3. Connection Raspberry Pi and relay module

- Raspberry pi PIN 02 (DC 5v) to relay module (VCC)
- Raspberry pi PIN 06 (Ground) to relay module (GND)
- Raspberry pi PIN 03 (GPIO 02) to relay module (IN1)
- Raspberry pi PIN 05 (GPIO 03) to relay module (IN2)

Test Connection Over 192.168.1.122:8000





There are two camera online at 192.168.1.122:8000, two relay that can be energize by press the button. User name is 'webiopi' password 'raspberry'

WiringPi for Python

1. Get/setup repo

```
git clone --recursive https://github.com/WiringPi/WiringPi-Python.gitcd WiringPi-Python
```

Don't forget the `--recursive`; it is required to also pull in the WiringPi C code from its own repository.

2. Prerequisites

To rebuild the bindings you **must** first have installed `swig`, `python-dev`, and `python-setuptools` (or their `python3`-equivalents). WiringPi should also be installed system-wide for access to the `gpio` tool.

```
sudo apt-get install python-dev python-setuptools swig wiringpi
```

3. Build & install with

```
sudo python setup.py install
```

Or Python 3:

```
sudo python3 setup.py install
```

PiReplay Installation

1. Commands to Install: Git (Required for Raspbian Lite)

```
sudo apt-get updatesudo apt-get install git -y
```

2. Commands to Install: Apache & PHP

```
sudo apt-get install apache2 -ysudo apt-get install php7.0
libapache2-mod-php7.0 -y
```

3. Create the PHP Script

Use the following command to launch the Nano File Editor and generate the gpio.php file:

```
sudo nano /var/www/html/gpio.php
```

Then copy the following into the Nano Editor:

```
<!DOCTYPE html> <html> <body> <?php $output = shell_exec('gpio mode
'.$_GET['pin'].' out'); $output = shell_exec('gpio write
'.$_GET['pin'].' '.$_GET['status']);for ($i = 0; $i < 32; $i++){
$output = shell_exec("gpio read ".$i); echo "Relay
".$i.":".$output."<br>"; } ?> </body> </html>
```

To exit the Nano Editor press `Ctrl+X` together. When it asks if you want to save the file press “Y” for “Yes” and press Enter! to save your gpio.php file.

That’s it… your Pi is now ready to use with PiRelay App to start switching or pulsing stats of your GPIO pins.

Building the Raspberry Pi DS18B20 Circuit

The circuit that we will need to build is pretty straightforward as we only need a resistor and the temperature sensor.

The sensor that I am using in this tutorial is a waterproof version of the DS18B20 sensor. It simply looks like a very long cord with a thick part on one end. If you have just a plain version without wiring and waterproofing, then it looks exactly like a transistor.

This sensor is pretty accurate being within 0.05°C of the actual temperature. It can handle temperatures of up to 125°C (260°F), but it’s recommended you keep it below

100°C (210°F). This device also has an onboard analog to digital converter so we're able to easily hook it up to a digital GPIO pin on the Pi.

Keep in mind not all temperature sensors are the same something like the TMP36will not simply replace the ds18b20 in this tutorial. The TMP36 is an analogue device making it slightly harder to integrate with the Pi. The TMP36 has basically the same problem we had with the light sensor and is because the Raspberry Pi doesn't have any analogue pins.

Putting this circuit together is super simple, I will quickly go through some basic instructions below. If you're having trouble following them, then please refer to the video or diagram.

If you need more information on the GPIO pins, then be sure to check out my guide to getting started with the Raspberry Pi GPIO pins.

1. First, connect the 3v3 pin from the Pi up to the positive rail & a ground pin to the ground rail on the breadboard.

2. Now place the DS18B20 sensor onto the breadboard.

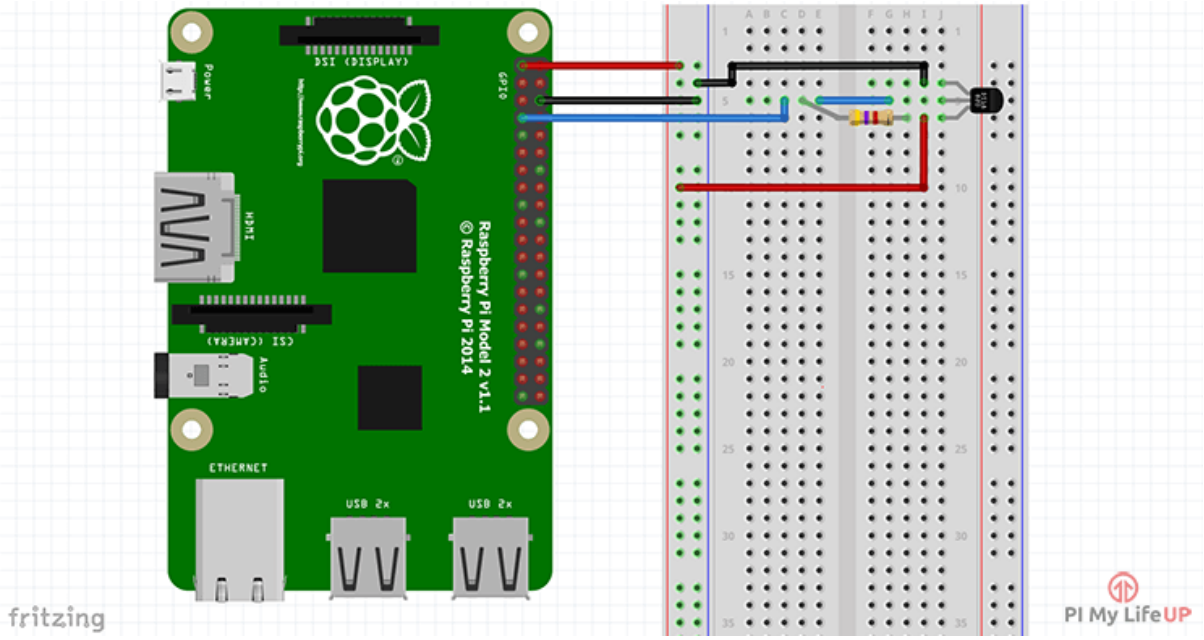
3. Place a 4.7k resistor between the positive lead and the output lead of the sensor.

4. Place a wire from the positive lead to the positive 3v3 rail.

5. Place a wire from the output lead back to pin #4 (Pin #7 if using physical numbering) of the Raspberry Pi.

6. Place a wire from the ground lead to the ground rail.

Once done, your circuit should look similar to the diagram below. Keep in mind some versions of the temperature sensor may have more wires than just three. Refer to the datasheet for more information for each of the wires.



The Raspberry Pi Temperature Sensor Code & Setup

The code for setting up the temperature sensor is a little more complicated than the circuit itself. This complexity is just because of the way we need to handle the data that comes from the sensor.

Before we make the Python script, we first need to setup the Pi so it can read data from the sensor. To do this, we need to add OneWire support.

1. To add support, we first need to open up the boot config file, and this can be done by running the following command:

```
sudo nano /boot/config.txt
```

2. At the bottom of this file enter the following.

```
dtoverlay=w1-gpio
```

3. Once done save & exit by pressing *ctrl x* and then *y*. Now reboot the Pi by running the following command.

```
sudo reboot
```

4. You can skip to downloading the code onto the Pi or follow the next few steps to check that the sensor is actually working.

5. Once the Raspberry Pi has booted back up, we need to run modprobe so we can load the correct modules.

```
sudo modprobe w1-gpiosudo modprobe w1-therm
```

6. Now change into the devices directory and use `ls` to see the folder/files in this directory.

```
cd /sys/bus/w1/devicesls
```

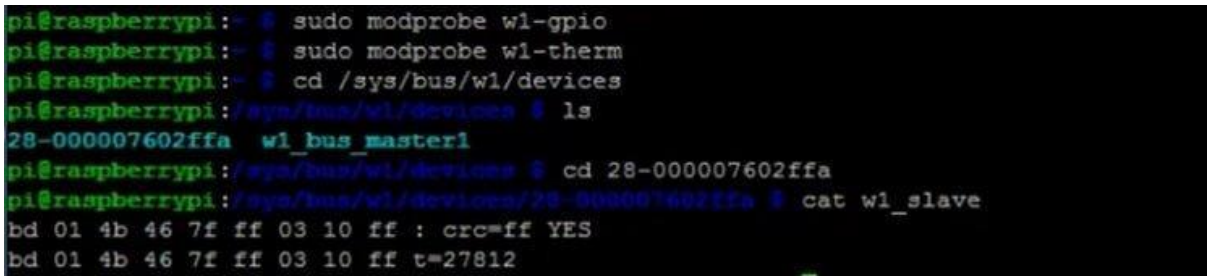
7. Now run the following command, change the numbering after `cd` to what has appeared in your directory by using the `ls` command. (If you have multiple sensors there will be more than one directory)

```
cd 28-000007602ffa
```

8. Now run the following command.

```
cat w1_slave
```

9. This command should output data but as you will notice it is not very user-friendly. The first line should have a YES or NO at the end of it. If it is YES, then a second line with the temperature should appear. This line will look similar to something like t=12323, and you will need to do a bit of math to make this a useable temperature that we can understand easily. For example, Celsius you simply divide by 1000.



Now it's time to move onto the python script.

I'll briefly explain the code below if you want to learn more about it. This code was sourced from the tutorial over at Adafruit. If you want to download it, you can simply download it at the Pi My Life Up GIT page or by using the following command.

```
git clone https://github.com/pimylifeup/temperature_sensor.git
```

To begin the python script, we import three packages, OS, Glob and time.

Next, we run the modprobe commands these are the same as what we used before.

We then declare three different variables that will point to the location of our sensor data. You shouldn't have to change any of these.

```
import osimport globimport time os.system('modprobe w1-
gpio')os.system('modprobe w1-therm') base_dir =
'/sys/bus/w1/devices/'device_folder = glob.glob(base_dir +
'28*')[0]device_file = device_folder + '/w1_slave'
```

In the `read_temp_raw` function we open up the file that contains our temperature output. We read all the lines from this and then return it so the code that has called this function can use it. In this case the `read_temp` function calls this function.

```
def read_temp_raw():    f = open(device_file, 'r')    lines =
f.readlines()    f.close()    return lines
```

In the `read_temp` function we process the data from the `read_temp_raw` function. We first make sure that the first line contains YES. This means there will be a line with a temperature in it.

If there is a temperature we then find the line with `t=` by using `lines[1].find('t=')`. Lines[1] means we're looking at the 2nd element in the array, in this case, the 2nd line. Once we have the line we simply get all the numbers that are after the `t=` this is done here `lines[1][equals_pos+2:]`. `Equals_pos` is the start position of the temperature (t), and we add 2 to the position, so we only get the actual temperature numbers.

We then convert the number into both a Celsius and Fahrenheit temperature. We return both of these to the code that called this function. This function is the print located in the while loop.

```
def read_temp():    lines = read_temp_raw()    while
lines[0].strip()[-3:] != 'YES':        time.sleep(0.2)        lines
= read_temp_raw()    equals_pos = lines[1].find('t=')    if
equals_pos != -1:        temp_string = lines[1][equals_pos+2:]
temp_c = float(temp_string) / 1000.0        temp_f = temp_c * 9.0 /
5.0 + 32.0        return temp_c, temp_f
```

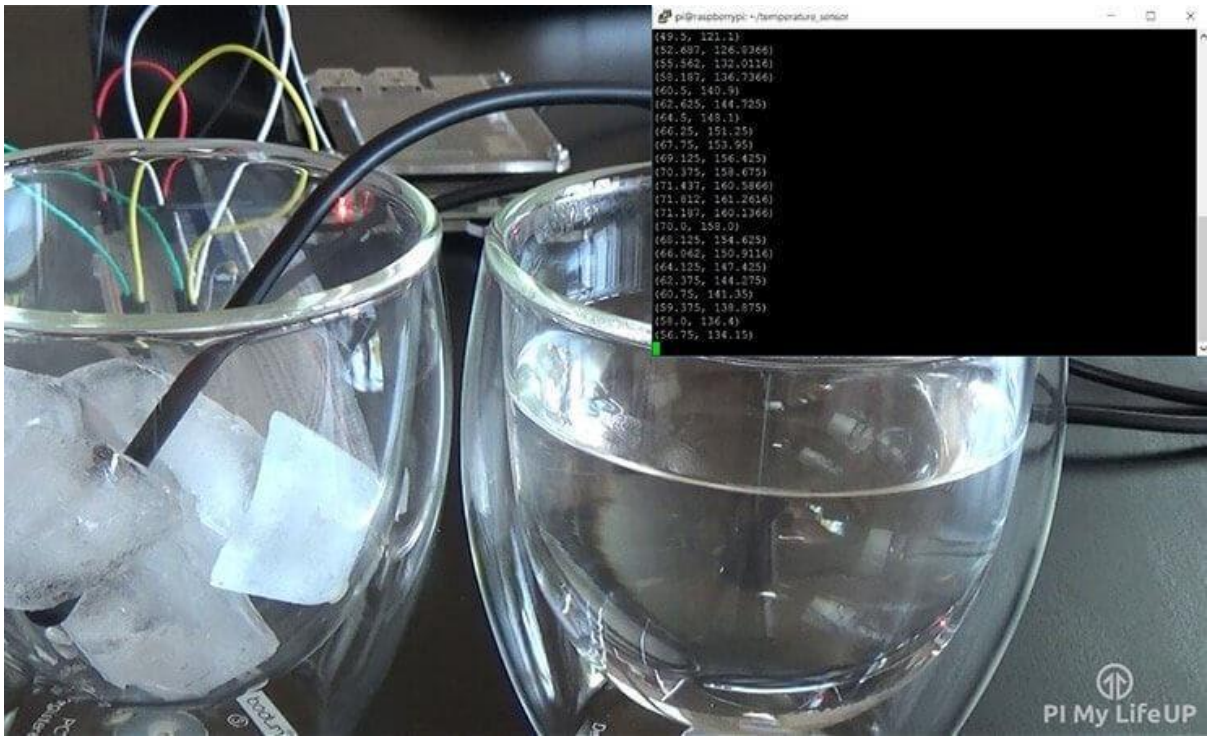
The while loop is always true so it will run forever until the program is interrupted by an error or the user cancelling the script. It simply calls the read_temp within the print function. The print function allows us to see the output on our screen. The script is then put to sleep for 1 second every time it has read the sensor.

```
while True:print(read_temp())time.sleep(1)
```

Once you have either downloaded or finished writing up the code and you also have set up the circuit correctly, we can now call the Python script. To call the python script simply run the following command.

```
sudo python thermometer_sensor.py
```

You should now have an output of temperatures in both Fahrenheit and Celsius. You can alter this just to display your preferred temperature scale.



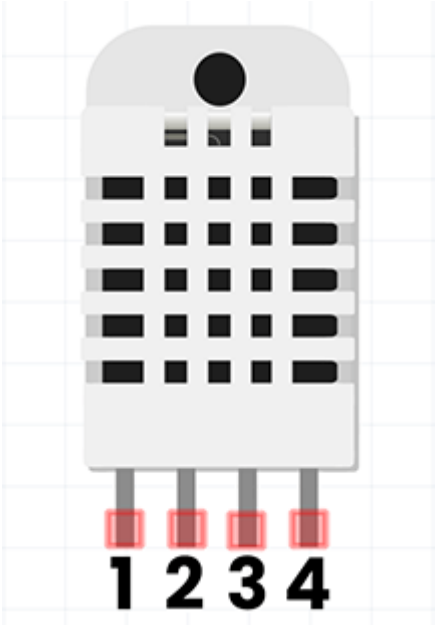
That's everything you need to know for getting your Raspberry Pi ds18b20 temperature sensor up and running.

DHT22 Pinout

To make things easier when assembling the humidity sensor circuit we have included the pinout of the DHT22 sensor.

This diagram should help you work out where each pin needs to go on the Raspberry Pi.

- **Pin 1** is **VCC** (Power Supply)
- **Pin 2** is **DATA** (The data signal)
- **Pin 3** is **NULL** (Do not connect)
- **Pin 4** is **GND** (Ground)

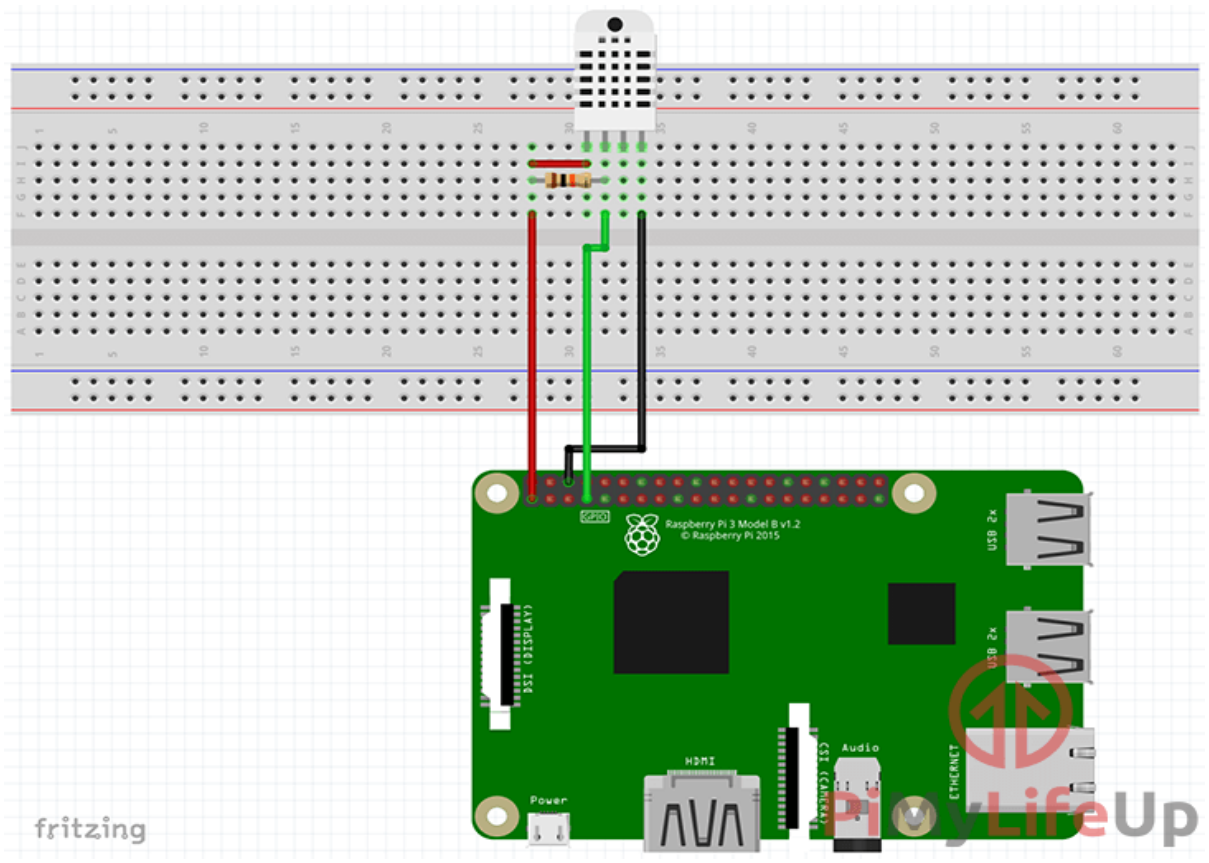


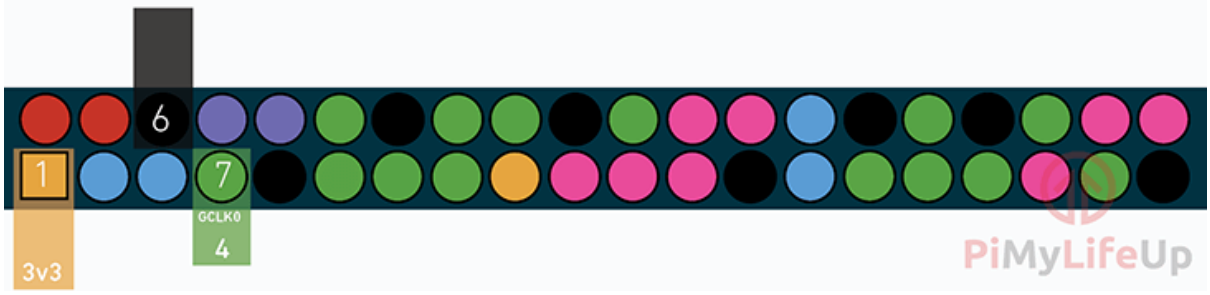
Raspberry Pi DHT22 Circuit

Here we have included a couple of ways you can put the Raspberry Pi Humidity sensor circuit together.

You can either rely on our two diagrams below to see what pins need to go to what or use our written steps below.

- Place a **10k resistor** between **Pin 1** and **Pin 2** of the DHT22
- Wire **Pin 1** of the DHT22 to **Physical Pin 1** (**3v3**) on the Pi
- Wire **Pin 2** of the DHT22 to **Physical Pin 7** (**GPIO4**) on the Pi
- Wire **Pin 4** of the DHT22 to **Physical Pin 6** (**GND**) on the Pi





Preparing the Raspberry Pi to talk with the Humidity Sensor

1. Before we get started with programming a script for the Raspberry Pi humidity sensor, we must first ensure that we have the latest updates on our Raspberry Pi.

We can do this by running the following two commands to update both the package list and installed packages.

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

2. With our package list now up to date, we need to go ahead and install both **python 3** and **pip**. We will be using both of these packages to interact with our humidity sensor.

Install both **python3-dev** and **python3-pip** by running the command below.

```
sudo apt-get install python3-dev python3-pip
```

3. Before we go ahead and install the DHT library, we should first run the following command to ensure we have the latest versions of the **setuptools**, **wheel** and **pip**python packages.

```
sudo python3 -m pip install --upgrade pip setuptools wheel
```

4. Now using pip, we will go ahead and install Adafruit's DHT library to the Raspberry Pi.

We will be using this Python library to interact with our DHT22 Humidity/Temperature sensor.

As a bonus, the library also supports the DHT11 and AM2302 humidity/temperature sensors making it a great library to learn how to utilize.

Run the following command to install the DHT library to your Raspberry Pi.

```
sudo pip3 install Adafruit_DHT
```

5. In the next section of our Raspberry Pi humidity sensor tutorial, we will show you how you can put this library to use and utilize it within a python script to talk with the DHT22 sensor.

Programming for the Raspberry Pi humidity sensor

1. Now that we have installed the Adafruit DHT library to our Raspberry Pi we can now proceed to program with it.

First, we must make a Python script in which we will store all our code. To begin writing this file, you can enter the following command into your Raspberry Pi.

```
nano ~/humidity.py
```

2. Within this file, we need to enter the following lines of code. We will explain each section of the code as we go along.

```
import Adafruit_DHT
```

This line imports the "**Adafruit_DHT**" library that we obtained using **pip** in the previous section.

We will be using this library to talk and interact with the DHT22 sensor. The library allows us to very easily retrieve the temperature and humidity from the sensor with a few lines of python code.

```
DHT_SENSOR = Adafruit_DHT.DHT22
```

On this line, we define a constant for the sensor that we want to utilize for the DHT Library. The Adafruit DHT library has three choices that you can choose from depending on the sensor used.

For this tutorial, we are making use of the **DHT22**, so we defined our "**DHT_SENSOR**" constant as "**Adafruit_DHT.DHT22**".

If you are following this guide but are using either the **DHT11** or the **AM2302** you can use either "**Adafruit_DHT.DH11**" or "**Adafruit_DHT_AM2302**" respectively.

```
DHT_PIN = 4
```

Next, we define a constant called "**DHT_PIN**" this is to store the number of the GPIO pin that the DHT22 is plugged into on the Raspberry Pi.

The number we use refers to the BCM numbering of the GPIO pins. In our tutorials case, this means we need to use the number "**4**" as we utilized the **GPIO4** pin for our data connection.

If you utilized a different GPIO pin, then make sure you update this number accordingly.

```
while True:    humidity, temperature =
Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)
```

Next, we start up an infinite loop by using the "**while TRUE:**" code. This loop will continually run until a user kills the Python script.

At the start of every loop, we utilize the Adafruit DHT library to retrieve both the humidity and temperature from the DHT22 sensor, specifying both our **DHT_SENSOR** and **DHT_PIN** constant.

The "**read_retry**" function will continually try to retrieve the data from the humidity sensor.

```
   if humidity is not None and temperature is not None:
print("Temp={0:0.1f}*C  Humidity={1:0.1f}%".format(temperature,
humidity))    else:        print("Failed to retrieve data from
humidity sensor")
```

Finally, we check to ensure that the DHT library returned data that is useful to us.

If either the "**humidity**" or "**temperature**" variables are "**None**" then we return a message telling the end user that we failed to retrieve data from our humidity sensor.

If we did have values in both of those variables, then we print out a message showing the end user both the temperature and the humidity that we just retrieved from our humidity sensor.

3. Once you have finished entering the code, it should look like what we have displayed below.

```
import Adafruit_DHTDHT_SENSOR = Adafruit_DHT.DHT22DHT_PIN = 4while
True:    humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR,
DHT_PIN)    if humidity is not None and temperature is not None:
print("Temp={0:0.1f}*C  Humidity={1:0.1f}%".format(temperature,
humidity))    else:        print("Failed to retrieve data from
humidity sensor")
```

4. When you are happy that all the code you entered is correct you can then save the file by pressing **CTRL + X**, then **Y** followed by **ENTER**.

5. With the script now saved, let's go ahead and run it so that we can start to see the data from the humidity sensor.

You can run this script by running the following command.

```
python3 ~/humidity.py
```

6. From this command, you should start seeing results like what we have below. The temperature and the humidity should be displayed through the command line.

```
Temp=25.6*C  Humidity=51.8%Temp=25.2*C  Humidity=45.4%Temp=25.1*C
Humidity=45.4%Temp=25.0*C  Humidity=45.3%
```

7. In the next section of the tutorial, we will show you how you can modify this script so that it logs the data to a file. The file will be able to be read by a program such as Microsoft Excel or Google Spreadsheets.

Logging the Data from the Raspberry Pi Humidity Sensor

1. Let's start by creating a new script. This script will be somewhat similar to our previous script but with a few changes made so that we can write to a file.

Start by running the following command to begin writing our new script.

```
nano ~/humidity_logger.py
```

2. Within this file enter the following lines. We will only be explaining the new additions to the script in this section.

```
import osimport timeimport Adafruit_DHT
```

Here we add the import for both the "**os**" and "**time**" libraries. We utilize the "**os**" library so we can check if our CSV file exists before we try to write to it.

We use the "**time**" library so that we can timestamp each new row with the current date and time.

```
DHT_SENSOR = Adafruit_DHT.DHT22DHT_PIN = 4
```

There is no difference with these two lines, keep them as you had in the previous script.

```
try:    f = &open('/home/pi/humidity.csv', 'a+')    if
os.stat('/home/pi/humidity.csv').st_size == 0:
f.write('Date,Time,Temperature,Humidity\r\n')except:    pass
```

Here we wrap everything within a try statement to handle any potential errors.

We open up a handle to our "**/home/pi/humidity.csv**" file with the "**a+**" tag applied. The **a+ tag** means that any data written to the file will be appended to the end of it.

Next, we use the "**os**" library to see if we have ever written to this file before. If the size returns as 0 we then write an initial line to the file. This line will contain our column headers so you can understand the data easier.

```
while True:    humidity, temperature =
Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)
```

Here we have our while loop again. We utilize the DHT Library to read data from our DHT22 humidity sensor again.

This function will store the data it reads from the sensors into our "**humidity**" and "**temperature**" variables.

```
   if humidity is not None and temperature is not None:
f.write('{0},{1},{2:0.1f}*C,{3:0.1f}%\r\n'.format(time.strftime('%m/
%d/%y'), time.strftime('%H:%M'), temperature, humidity))      else:
print("Failed to retrieve data from humidity sensor")
time.sleep(30)
```

Our big change to this section of code is that instead of printing the temperature and humidity to the console, we write it to our "**humidity.csv**" file. We do this by using the file handle we opened earlier.

We format the text written to the file to include both the **current date**, **current time**, our formatted **temperature** and **humidity**.

At the end of the loop, we sleep the script for 30 seconds, if you want to make it sleep longer or poll more often then modify this number.

3. Once you have finished entering the code, it should look like what we have displayed below.

```
import os

import time

import Adafruit_DHTDHT_SENSOR = Adafruit_DHT.DHT22DHT_PIN = 4try:
f = open('/home/pi/humidity.csv', 'a+')    if
os.stat('/home/pi/humidity.csv').st_size == 0:
f.write('Date,Time,Temperature,Humidity\r\n')except:    passwhile
True:    humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR,
DHT_PIN)    if humidity is not None and temperature is not None:
f.write('{0},{1},{2:0.1f}*C,{3:0.1f}%\r\n'.format(time.strftime('%m/
%d/%y'), time.strftime('%H:%M'), temperature, humidity))    else:
print("Failed to retrieve data from humidity sensor")
time.sleep(30)
```

4. Once done, save the file by pressing **CTRL + X**, then **Y** followed by **ENTER**.

5. Now like our last script lets go ahead and now run it by entering the following command.

```
python3 ~/humidity_logger.py
```

6. Once you exit out of the script, you will now be able to check out the "**humidity.csv**" file to see all the results that you have logged over time.

If you run the following command, you should see the contents of the file.

```
nano ~/humidity.csv
```

7. Below is an example of the contents that you should see within this file.

```
Date,Time,Temperature,Humidity05/04/19,05:47,22.3*C,50.6%05/04/19,05
:48,22.2*C,50.4%05/04/19,05:48,22.2*C,50.6%05/04/19,05:49,22.2*C,50.
4%05/04/19,05:49,22.2*C,50.4%05/04/19,05:50,22.2*C,50.3%05/04/19,05:
50,22.2*C,50.3%05/04/19,05:51,22.2*C,50.2%
```

Once you have verified that the file is successfully being written, then you are finally done with programming your Raspberry Pi humidity sensor.

If you need a waterproof temperature sensor, then be sure to check out our DS18B20 temperature sensor tutorial. It covers everything you need to know, and it works perfectly alongside this sensor.

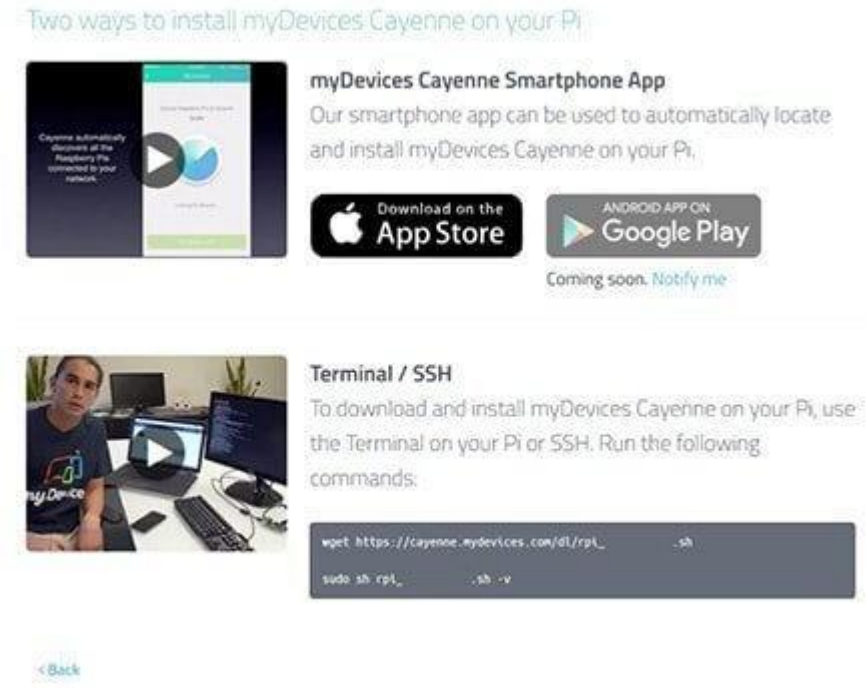
I hope that everything has worked correctly for you and you're not collecting the humidity and temperature without issues. If you want to leave some feedback, then please don't hesitate to leave a comment below.

Installing Raspberry Pi Cayenne

The process of installing Cayenne onto the Raspberry Pi is pretty simple and shouldn't take you too long to get it up and running. You will need to make sure you have Raspbian installed on your Pi. If you're unsure how to do this, then be sure to check out my tutorial on setting up Raspbian.

1. Firstly, head over to myDevices Cayenne and sign up for a free account.

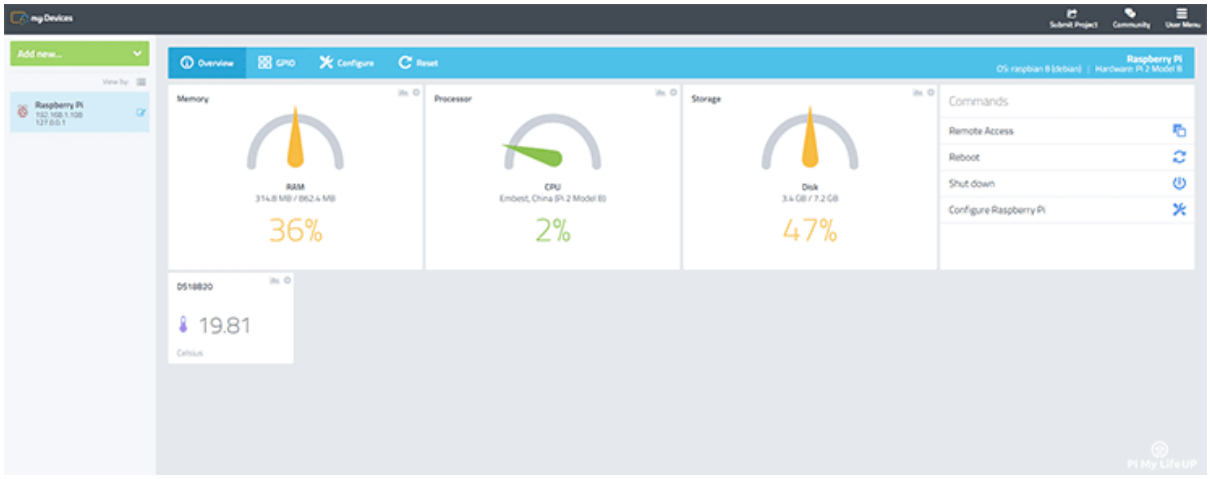
2. Once you're signed up, you will need to register/connect the Pi up to the account you just created. To do this just copy the two command lines shown after you sign up. Enter these into the terminal for your Pi. (These files are unique for every new install)



Alternatively, you can download the app, and it can automatically locate & install Cayenne onto your Pi. (Keep in mind SSH needs to be enabled)

3. It will take a few minutes to install onto your Pi depending on how fast your internet connection is. The web browser or app should automatically update with information on the installation process.

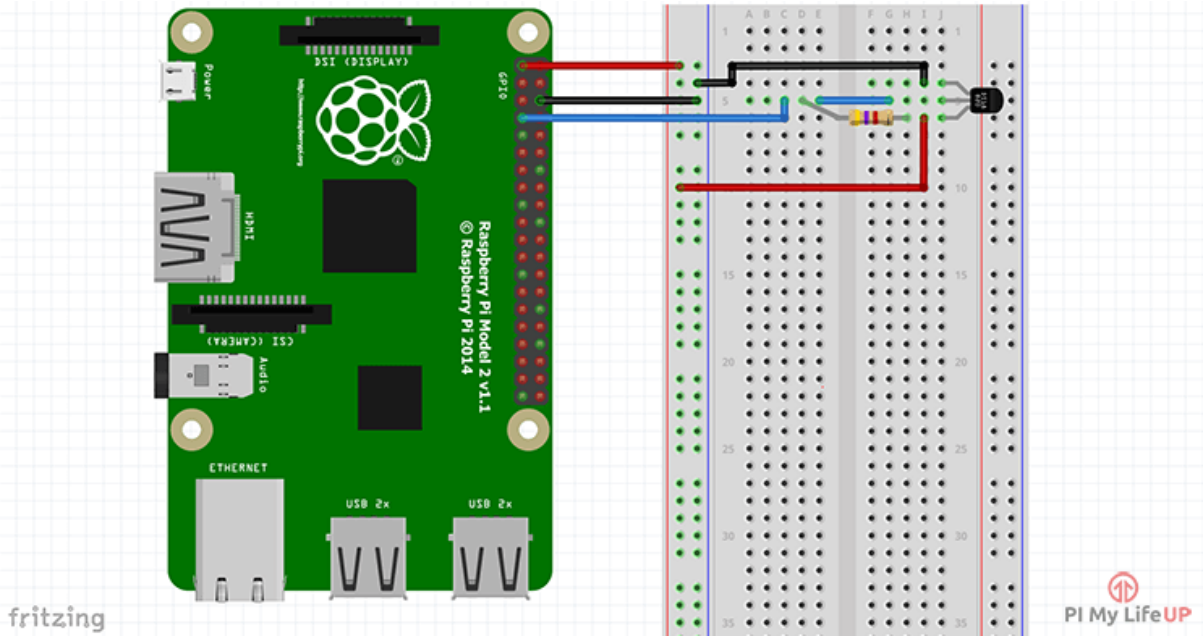
4. Once installed the dashboard will display and should look like something below.



Setting up your first sensor

In this example, I am going to set up a temperature sensor. The sensor that I use is called DS18B20 and is the same sensor I did a tutorial on not that long ago. Now if you remember the process of setting this up was a little more in-depth but using cayenne makes it super simple.

All you need to do is set up the circuit and have it connected up to the Pi. If you need the full tutorial on how to do this, then be sure to check out my Raspberry Pi DS18B20 tutorial or if you know what you're doing just check out the circuit diagram below. I also added an LED that is connected to pin #17 with a 100-ohm resistor to the ground rail.



Now when I set this up the sensor, it was automatically detected and added to my dashboard. Which is pretty cool, however, if it didn't add automatically then you will need to add it manually. To add it manually just, do the following.

1. Go to *add new* in the upper left corner of the dashboard.

2. Select *device* from the drop-down box.

3. Find the device, in this case. It is a DS18B20 temperature sensor.

4. Add all the details of the device. In this case, you will need the slave address for the sensor. To get the slave address enter the following into the Pi's terminal.

```
cd /sys/bus/w1/devices
ls
```

5. The slave address will look similar to this `28-000007602ffa`. Simply copy & paste this into the slave field within Cayenne dashboard.

6. Once entered select *add sensor*.

7. The sensor should now display on the dashboard.

8. If you need to customize your sensor press the cog and it will come up with some options.

9. You can also see stats/graphs. For example, the temperature sensor can plot data in real time and will keep historical data too.

If you also want to add an LED that you can turn on & off via the dashboard follow the next few instructions.

1. Now let's add one more device. Except this one will be an LED.

2. So go back to *add new device*.

3. Now search for *digital output* and select it.

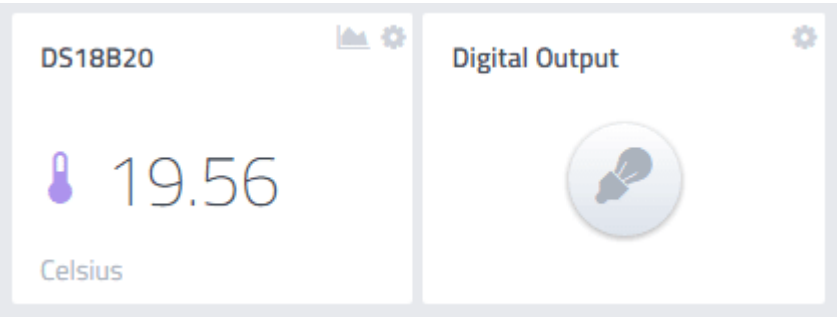
4. For this device select your Pi, widget type is *button*, *icon* can be whatever you want, and then select *integrated GPIO*. Finally, *channel* is the pin/channel that our LED is connected. For this example, it is *pin #17*. (This is the GPIO numbering of the pins).

5. Now press the add sensor button.

6. You can now turn the GPIO pin high & low from the dashboard and also use it in a trigger.

7. We're now ready to set up our first trigger.

You should now have two devices on your dashboard that should look something like this.



If you're completely new to setting up sensors and using the GPIO pins, then it might be worthwhile checking out my guide for the GPIO pins on the Raspberry Pi.

Setting up your first trigger

Triggers in Cayenne are a way to have your Pi react to a certain change on the Pi itself or through a sensor attached to it. This trigger could be something as simple as a temperature exceeding a certain value or even just your Pi going offline. As you could imagine this can be quite powerful in creating smart devices that react to the surroundings. For example, If the room gets too cold, then turn a heater on so the room will heat.

The process of adding a trigger is super simple, and I will go through the basics of getting a couple of triggers set up.

1. Go to *add new* in the upper left corner of the dashboard.

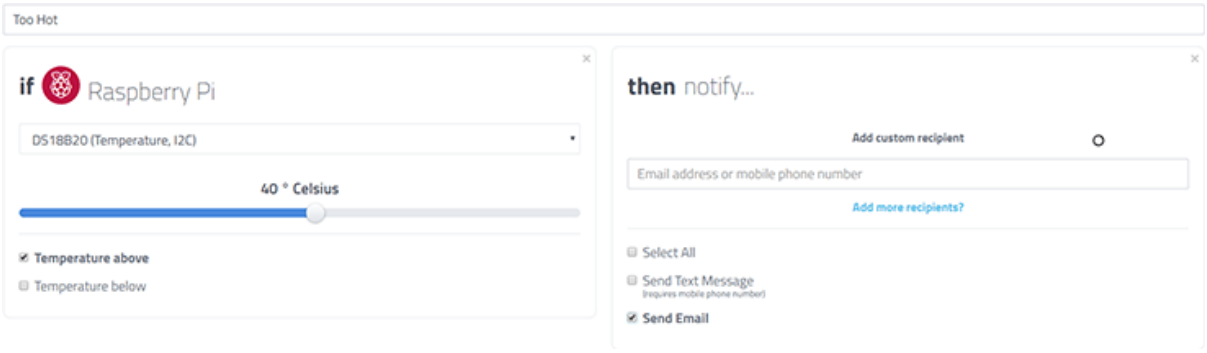
2. Select *trigger* from the down box.

3. First name your trigger, I will call mine "too hot"

4. Now drag and drop your Raspberry Pi from the far left corner into the *if* box.

5. Underneath this select, the temperature sensor and have the checkbox next to "*temperature above*" selected. (If device options don't display simply refresh the page)

6. Now in the *then* box select notification and add an email address or your phone number for a text message (You can add both). Make sure you tick the checkboxes as well.



7. Now click "save trigger".

8. It should now be saved and will send you an alert whenever the temperature sensor is over 40 degrees Celsius.

9. You can also drag the Raspberry Pi into the *then* box and have it do many things including controlling output devices. For example, in the circuit, I have an LED that will turn on when the temperature exceeds 40 degrees Celsius.

10. To make the LED trigger click on *new trigger* located in the upper part of the page. Name this trigger, *activate LED*.

11. Now drag the Pi into the *if* box and then select the temperature sensor again with 40 degrees above Celsius.

12. Next drag the Raspberry Pi into the *then* box. Select our digital output and have the on checkbox ticked.

13. Now click on save trigger.

14. Finally, now whenever our temperature sensor connected to the Pi reports a temperature of 40 degrees Celsius or higher, it will send an email and turn on the LED. You will also need to add another trigger to turn the LED off when it drops back below 40, but I will leave that for now and move on to events.

Events

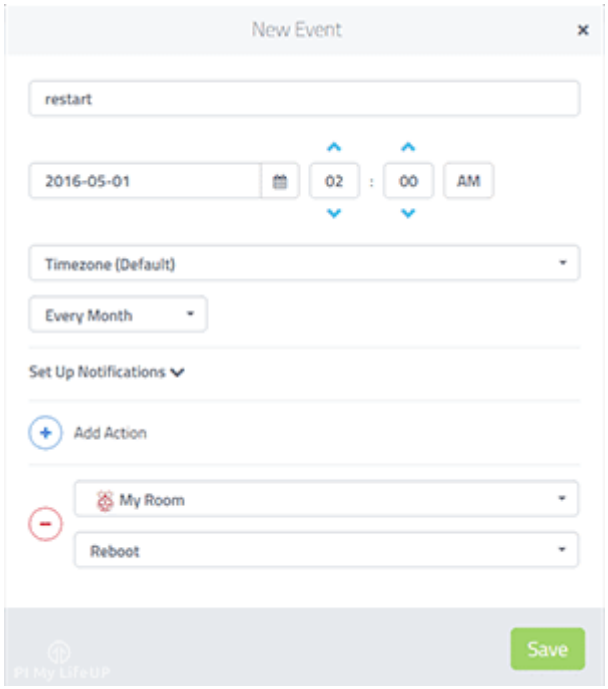
Events in Raspberry Pi Cayenne is somewhat similar to triggers, but they are time dependent rather than relying on a change in a sensor or the device itself. Setting up an event is pretty easy, I will just quickly go through how to setup your Pi to restart once a month.

1. Go to *add new* in the upper left corner of the dashboard.

2. Select *event* from the down box.

3. You should now see a screen with a calendar, and a popup called new event.

4. Enter the details of your event. For example, mine is called restart monthly and will happen on the first of every month at 2 am. Below is an example of the screen.



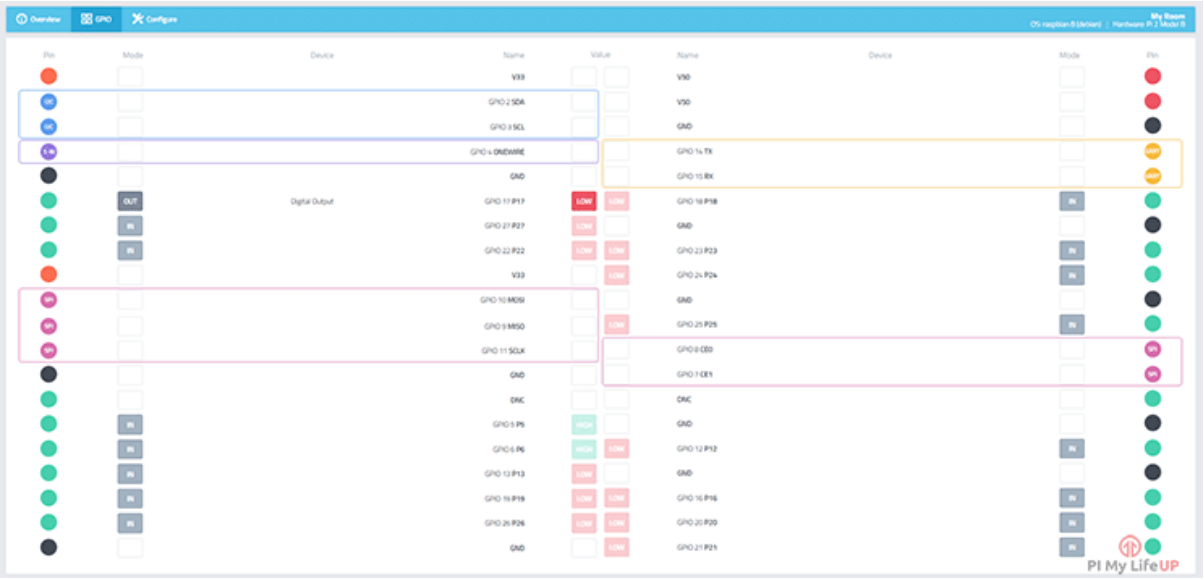
5. Once done, click on save.

6. You should now be able to see your event on the calendar. Simply click on it if you wish to edit it.

As you could imagine events can be pretty powerful so it would be worth looking into these more. An excellent example of using events will be if you need something to run or turn on. A good example is something such as lights requiring to be turned on at a specific time.

GPIO Panel

The GPIO panel within Raspberry Pi Cayenne allows you to control and alter the pins on the Pi. For example, you can turn a pin from being input to an output and vice versa. You can also turn output pins both low & high.



As you can see it also makes a great reference sheet if you need to refer back and see which pins are the ones you need. You can also see the devices that are currently assigned to specific pins. You're also able to see the current status of a pin. (e.g. Input or output & low or high)

Future possibilities

Cayenne is still relatively new and has so much more coming in the future. It is already jam-packed full of features. I have read of future integrations with the Pi camera and other devices. This software will open up to some pretty cool projects. As I mentioned earlier, I will be looking at doing more with Cayenne in the near future as home automation has been something I have wanted to cover for some time now.

I hope you have been able to get myDevices Cayenne installed onto your Raspberry Pi without any issues. If you haven't already signed up, then be sure to go sign up and register a device over at their website. If you do come across anything that I should've mentioned, have feedback or anything else then please feel free to leave a comment on our forums.

Cayenne DHT Plugin

Getting Started

1. Installation

From the command line run the following commands to install this plugin.

```
sudo apt-get install python3-pip
```

```
cd /etc/myDevices/pluginssudo
```

```
git clone https://github.com/myDevicesIoT/cayenne-plugin-dht.git
cd cayenne-plugin-dht
sudo pip3 install .
```

2. Setting the sensor type and pin

Specify the sensor type and pin you are using by modifying `init_args` under `DHT Temperature` in the `cayenne_dht.plugin` file. For a DHT11 use `11` for the sensor argument, for a DHT22 use `22` and for a AM2302 use `2302`. Set the pin argument to the GPIO pin number your sensor is connected to. For example, a DHT11 on pin 17 would use the following:

```
init_args={"sensor": 11, "pin": 17}
```

My Code is as follows:

```
[Lenny's Feeding Platform Temp]
enabled=true
channel=0
module=cayenne_dht
class=CayenneDHT
init_args={"sensor": 11, "pin": 4}
read=get_temperature

[Lenny's Feeding Platform Humidity]
inherit=Lenny's Feeding Platform Temp
enabled=true
channel=1
read=get_humidity

[Lenny's Middle Temp]
enabled=true
channel=4
module=cayenne_dht
class=CayenneDHT
init_args={"sensor": 22, "pin": 18}
read=get_temperature

[Lenny's Middle Humidity]
inherit=Lenny's Middle Temp
enabled=true
channel=6
read=get_humidity

[Bud's Feeding Platform Temp]
```

```
enabled=true
channel=18
module=cayenne_dht
class=CayenneDHT
init_args={"sensor": 11, "pin": 26}
read=get_temperature

[Bud's Feeding Platform Humidity]
inherit=Bud's Feeding Platform Temp
enabled=true
channel=19
read=get_humidity

[Bud's Middle Temp]
enabled=true
channel=8
module=cayenne_dht
class=CayenneDHT
init_args={"sensor": 22, "pin": 20}
read=get_temperature

[Bud's Middle Humidity]
inherit=Bud's Feeding Platform Temp
enabled=true
channel=9
read=get_humidity
```

3. Restarting the agent

Restart the agent so it can load the plugin.

```
sudo service myDevices restart
```

Temporary widgets for the plugin should now show up in the Cayenne Dashboard. You can make them permanent by clicking the plus sign.

NOTE: If the temporary widgets do not show up try refreshing the Cayenne Dashboard or restarting the agent again using `sudo service myDevices restart`.

Setting up a Raspberry Pi Spotify Connect Device

1. Our first task is to ensure everything is up to date with our operating system. It will make installing the Spotify software to our Raspberry Pi a little easier.

To do this, we will need to run the following two commands. The first will **update**the package list. The second command **upgrades** the installed packages.

```
sudo apt update
```

```
sudo apt upgrade
```

2. Next, we need to make sure that both "**curl**" and "**apt-transport-https**" packages are installed to our Raspberry Pi.

Without these, we won't be able to set up the raspotify repository on Raspbian.

Run the following command to install our needed packages.

```
sudo apt install -y apt-transport-https curl
```

3. With the packages we need installed we can go ahead and add the "raspotify" GPG key and its repository.

Without the GPG key, the apt package manager won't be able to verify the files it's retrieving from the repository.

You can do these two things by running the following two commands.

```
curl -sSL https://dtcooper.github.io/raspotify/key.asc | sudo apt-key add -v -
```

```
echo 'deb https://dtcooper.github.io/raspotify raspotify main' | sudo tee /etc/apt/sources.list.d/raspotify.list
```

4. Now that we have the raspotify repository added to our Raspberry Pi, we can now go ahead and install the raspotify package.

This package will handle turning our Raspberry Pi into a Spotify Connect device. We can install the package by running the two commands below.

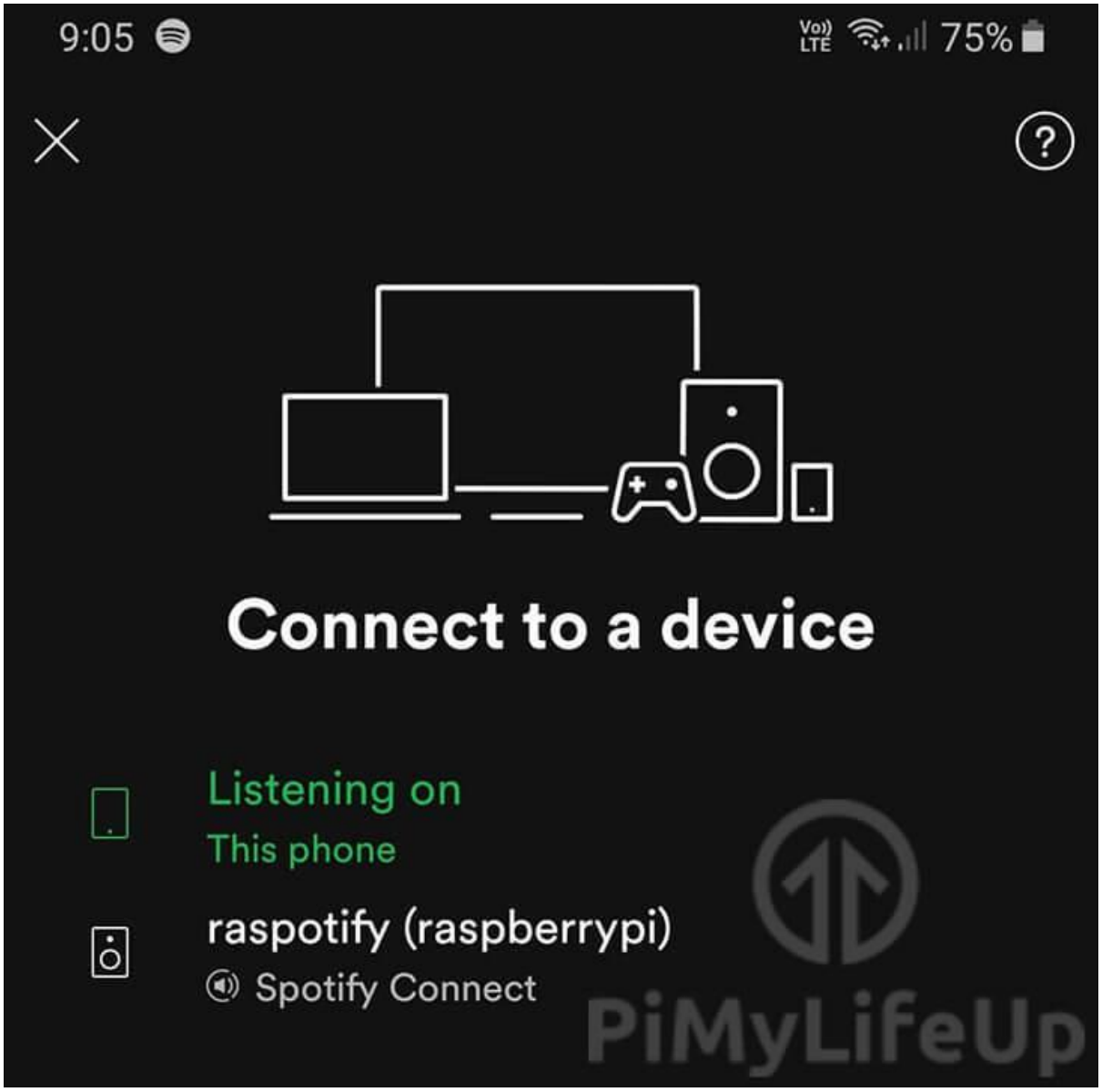
We need to rerun the update due to us adding the **raspotify** repository in the previous step. Without an update, the package manager won't know what's contained in that repository.

```
sudo apt-get update
```

```
sudo apt-get install raspotify
```

5. With the "**raspotify**" software now installed, the software should start automatically and be ready for connections.

You should now be able to connect to your Raspberry Pi's Spotify software. You can do this by opening up the Spotify app on your chosen device and selecting it from the "**Connect to a device**" menu.



Configuring the Spotify Connect Software

1. While the Raspotify software works perfectly fine right out of the box, you can make changes to its configuration.

You can use this configuration file to adjust things such as the bitrate or the device name.

Run the command below to begin modifying the "**raspotify**" config file.

```
sudo nano /etc/default/raspotify
```

2. Within this file, you will see multiple different options that you can configure yourself. We will go into a couple of these and explain what you can use them for.

```
DEVICE_NAME="raspotify"
```

This option defines the name for your Spotify connect service on the Raspberry Pi. You can set this to something to easier identify where the Raspberry Pi is or what the Pi is connected to.

```
BITRATE="160"
```

With this option, you can specify the bitrate that you would like the device to utilize.

You can pick three different values for this option, **96** for **low**-quality, **160** for **medium**-quality or **320** for **high**-quality audio.

```
OPTIONS="--username <USERNAME> --password <PASSWORD>"
```

The "**OPTIONS=**" option allows you to specify additional options to pass through to the Spotify Connect software.

In our example, we will be passing in both a Spotify device username (**<USERNAME>**) and password (**<PASSWORD>**). By specifying these, you will be able to access the Raspberry Pi Spotify service outside the local network.

Please note that the username and password options are not what you use to log back into the account normally. You will need to retrieve these details by going to the "Set device password" screen on your Spotify account.

If the link above directs to your profile homepage and there is no option to set device password then simply enter your normal Spotify Username and password.

3. If you have made any changes, you can save the file by pressing **CTRL + X** then **Y** followed by **ENTER**.

4. Once you have made any changes to the raspotify configuration file, you will need to first restart the "**raspotify**" service by running the command below.

```
sudo systemctl restart raspotify
```

Hopefully, by now, you will now have successfully set up your headless Raspberry Pi Spotify device. It's an excellent way to have a barebones audio player that can use any speaker that supports USB or a 3.5mm connection.

References

All Information and sources were gathered online from the following websites. Some of the links were not as straightforward so it needed some playing around with to get it working.

- https://pimylifeup.com/
  - Raspian
  - SSH
  - DS18B20
  - DHT22
  - Cayenne
  - Spotify
- http://www.knight-of-pi.org/installing-python3-6-on-a-raspberry-pi/
  - Python 3
- https://github.com/
  - Various
- https://github.com/myDevicesIoT/cayenne-plugin-dht
  - Cayennne DHT.plugin
- https://www.instructables.com/
  - Various
- https://pirelay.jasonfindlay.com/
  - PiRelay
- https://community.smartthings.com/t/release-raspberry-pi-relay-controller-webiopi/36362/174
  - Samsung Smartthings Raspberry-pi-relay-controller-webiopi

Vivarium Pi Installation Guide:

A Complete Guide for Beginners

Table of Contents

Vivarium Pi Installation Guide:

A Complete Guide for Beginners

```
wget WebIOPi-x.y.z.tar.gz
tar xvzf WebIOPi-x.y.z.tar.gz
cd WebIOPi-x.y.z
sed -i 's/ python3//' setup.sh
sudo ./setup.sh
sudo webiopi-passwd
sudo service webiopi restart


curl -X POST webiopi:localhost:8000/macros/temperature
```