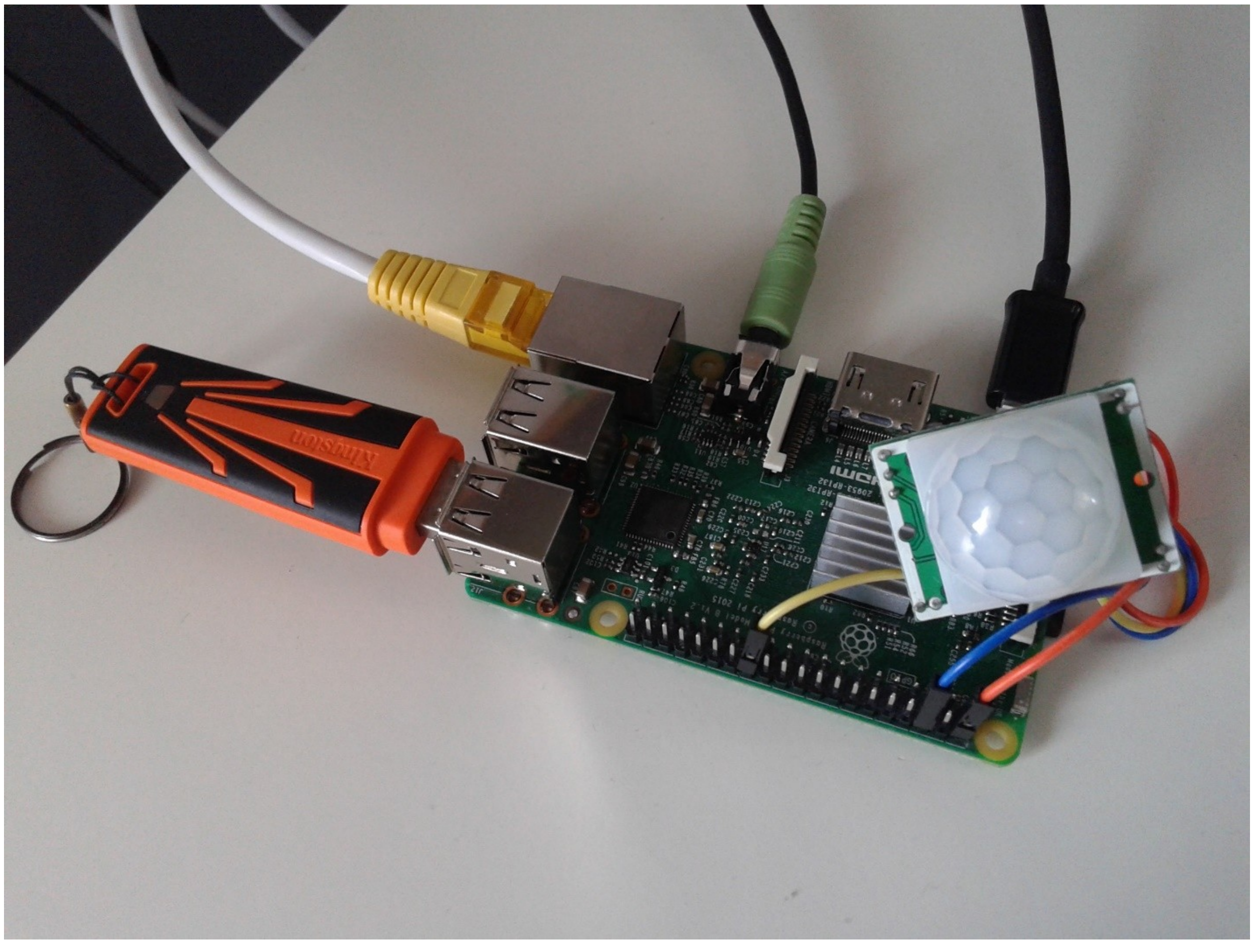
Motion Triggered Music

In this project we are about to use a Raspberry Pi 3 (RaPi3) as a music player. Not an ordinary music player of course but a motion activated one.
If anything moves in the room where the system is about to be installed a sensor sends a signal to the RaPi.
When the RaPi receives a signal it picks randomly a song from a folder and plays it.
We can adjust the path that RaPi picks a song from an external USB flash drive.



We are going to use:
1x Passive Infrared Sensor (or PIR Sensor for short)
1x Raspberry Pi 3
1x Power Supply for the RaPi (AC/DC wall adapter 5V/2A)
1x USB to microUSB cable
1x Ethernet Cable
3x female jumper cables
1x router with an internet connection

Note 1: The router is only needed when we want to control the RaPi without using any peripherals like a monitor, a mouse, a keyboard etc. This way of controlling a RaPi is called headless and it's explained in one of the steps below.

Connecting the hardware

So let's start building our project!
First we have to put the hardware together.
Then we will write a python script that will handle the signals from the PIR Sensor, find an mp3 file and open in with the omxplayer (command line music player)

Step1:
Plug the USB flash drive that contains the music folder into one of the USB ports of the RaPi.
Step2:
Connect the PIR Sensor to the RaPi.

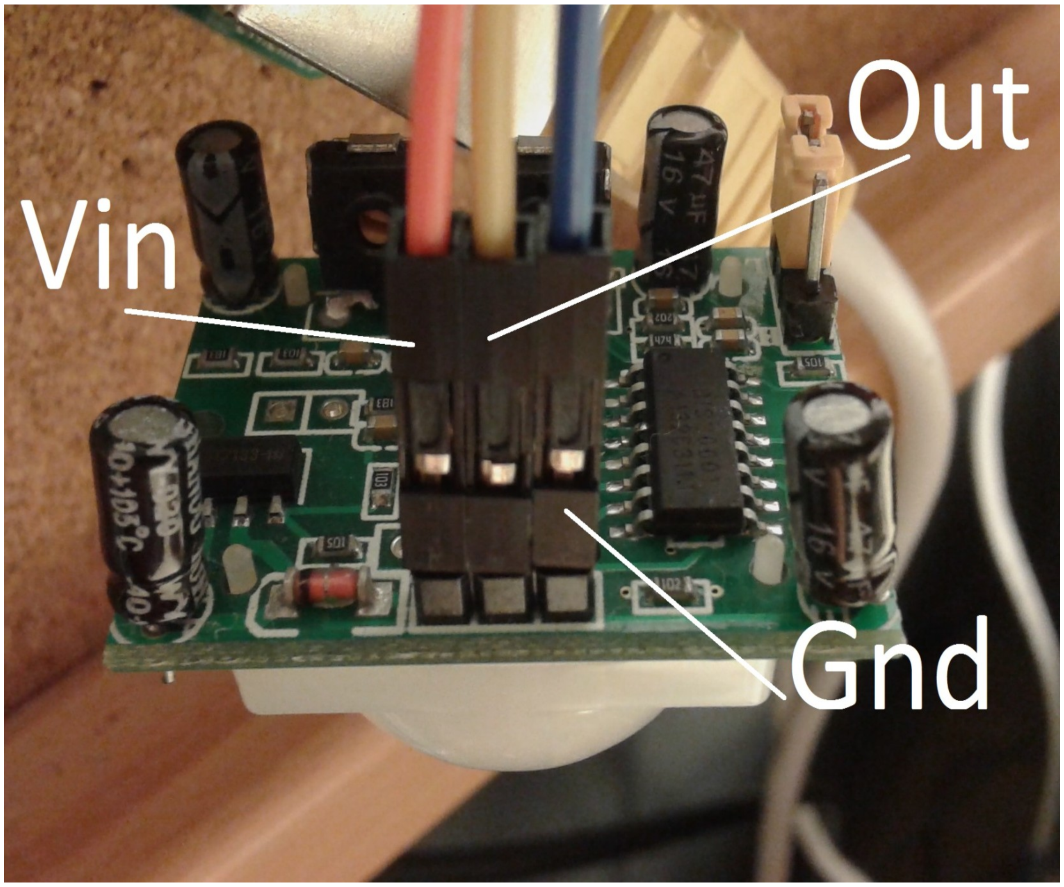


Figure 1: What each pin stands for

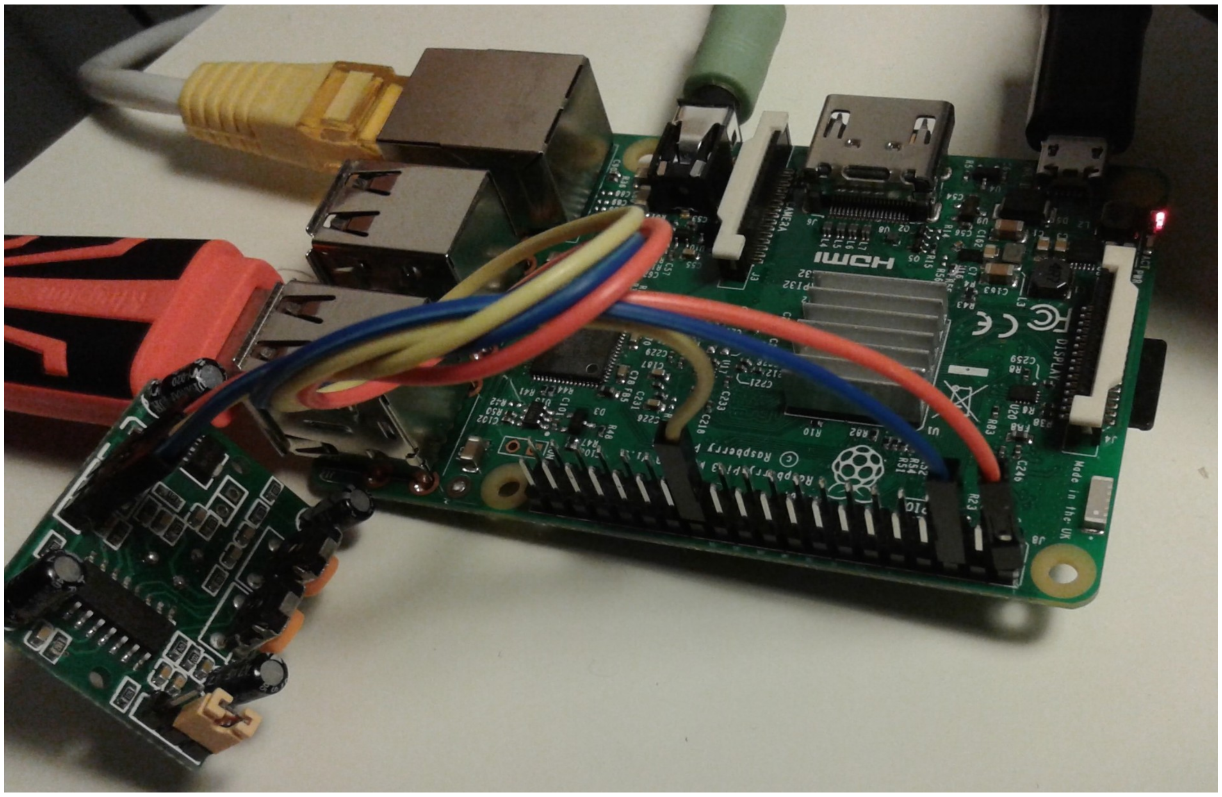
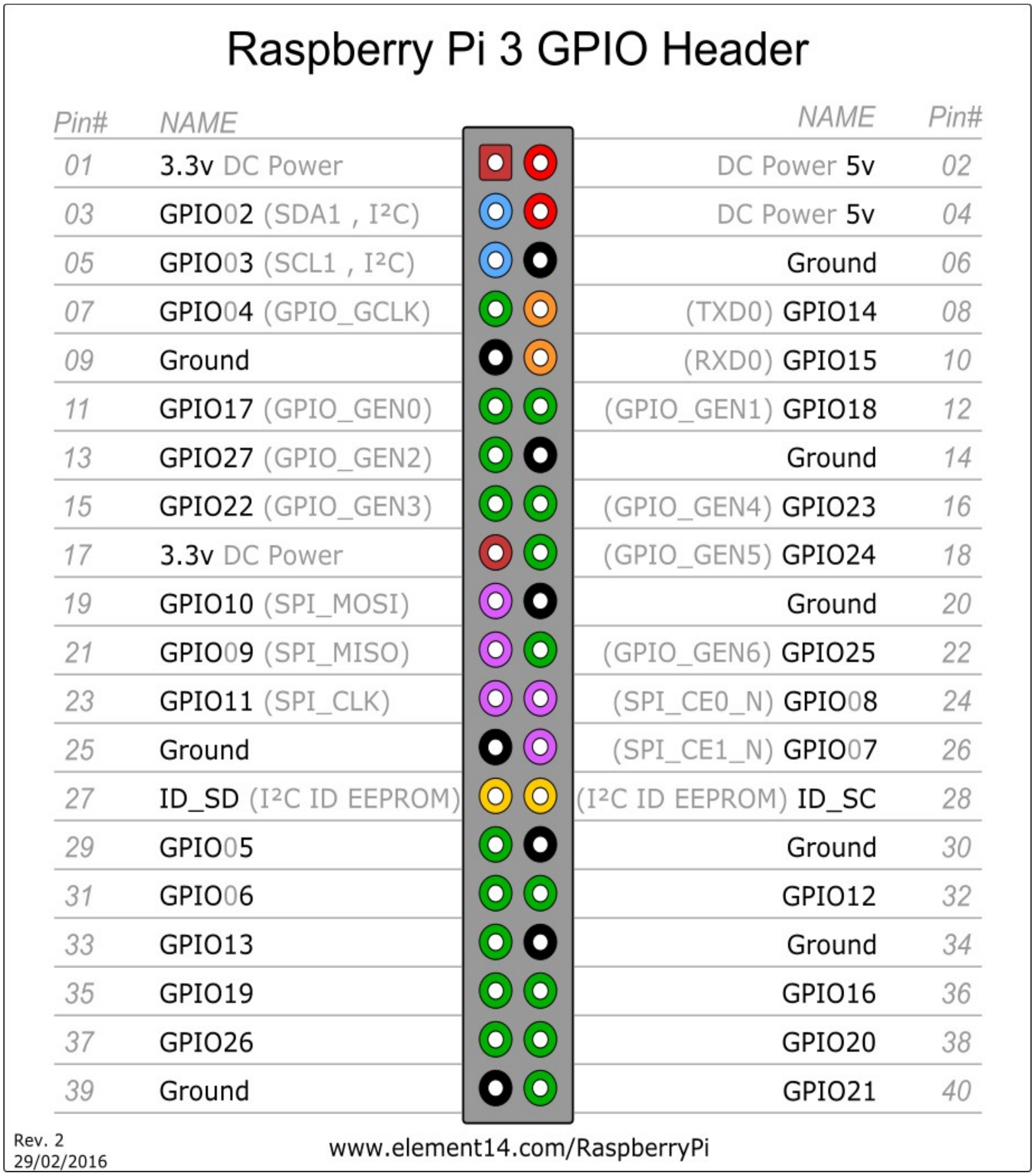


Figure 2: A way to connect the sensor with the RaPi. In this photo I have chosen to connect the output of the sensor to GPIO07 (PIN #26 as shown in Figure 3)



Raspberry Pi 3 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I^{2}C)	DC Power 5v	04
05	GPIO03 (SCL1 , I^{2}C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I^{2}C ID EEPROM)	(I^{2}C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Rev. 2
29/02/2016

www.element14.com/RaspberryPi

Figure 3: A full map of the Raspberry 3 pins

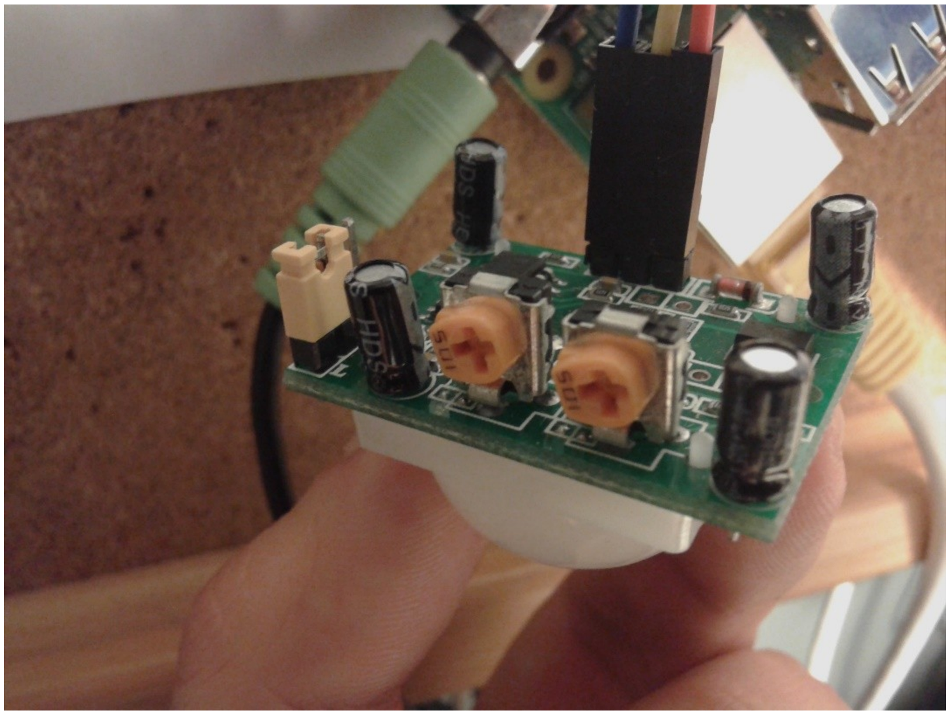


Figure 4: After some experimenting I concluded to the setup above. For more details about the job that each potentiometer does and what to yellow jumper cable does visit: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/testing-a-pir

Step3:
Connect the RaPi to the router via the Ethernet cable.

Step4:
Connect the RaPi to the power supply via the USB to microUSB cable.

Step 5:
Make sure that everything is in place as they should and plug the power supply to the socket.

HEADLESS CONTROL
(*or controlling RaPi without peripherals*)

Now let's have a look on how to control headlessly the RaPi.

1) Open the "Advanced IP Scanner as an admin"
2) Press the scan button
3) Find the raspberrypilan IP Address
4) Open PUTTY
5) Create a connection using RaPi's IP
6) Open the Remote Control Computer
7) Enter RaPi's username & Password

DONE!
You can now control RaPi with a GUI

<u>FOR LINUX USERS ONLY :</u>

If we want to connect to RaPi's terminal from a Linux OS we can do it by:

1) Install the arp-scan : `sudo apt-get install arp-scan`
2) Find the IP address of the RaPi, run the command :
   `sudo arp-scan –interface=eth0 –localnet`

(If we have connected the RaPi with an ethernet cable to the router)

Then we run : `ssh pi@192.168.1.77`

"We're on the RaPi's terminal ladies and gentlemen"

Before starting to write the actual code of our project, let's have a quick look on some basic linux terminal commands

man : This command brings up the online Unix manual. Use it on each of the commands to show how they work.

Example: *man pwd* (You will see the manual for the pwd command)

pwd : Shows what directory (folder) you are in.

cd : Changes directory

Example: cd muondata (Moves down from your current directory into the *muondata* sub-directory)

cd .. : Moves up one directory

You can also **move directly into directories**
Example: *cd /home/particle/muondata*

cd~ : Takes you back to your home directory

ls : Lists files

ps aux : shows all the running processes

kill <PID> : whereas <PID> is the ID of the process you want to kill as shown in the ps aux command

Of course there is a ton more commands but we are not going to use them at this project.

TIME TO CODE

Step1:
We will start coding by writing a python script that reads and prints the GPIO pin values where the output of the PIR Sensor is connected.

Step2:
Then we will make the code understand the transitions between motion and absence of motion and vice versa. This will prove useful because our event (music playing) is triggered by a transition from stillness to movement.

Step3:
Now we will make the raspberry play music when motion is detected in the room.

Step4:
Time to create a method that picks randomly an mp3 file from a folder and plays it.

Step5:
Obviously we don't want the same song to be played twice (at least not before the playlist wears out) so we are going to create a method that moves the played mp3 file to another folder. In the end there will be 2 folders: *music* and *played. Music* will contain the playlist, the songs that are about to be played. *Played*, well, this is self explanatory.

Step6:
What if the playlist wears out? We must create a mechanism that refreshes the music folder when it gets empty. This can be divided to two parts: the control part and a method that swaps the names of two folders. This way, the contents of *played* folder will become the contents of the *music* folder. Playlist restarted!

Step7:
Experience shows that the str() method "gets confused" with certain characters like " & (') etc.".
We will create a method that deletes those chars from the .mp3 file names.

To make your script run you enter the command:

```
Sudo python scriptname.py
```

In order to have a greater versatility we can make RaPi run our script on startup.

Step1:

Go to Desktop and create an empty file, name it *launcher*. Enter the following:

```
#!/bin/sh
# launcher.sh

Cd /
Cd home/pi/…….    (this is the path where the python script is located)
Sudo python <nameoftheproject>.py
Cd /
```

Step2:

Make this script executable. Open the terminal.

Enter : `cd /home/pi/Desktop`

We need to make the launcher script an executable, which we do with this command: `chmod 755 launcher`

Now test it, by typing in: `sh launcher`

This should run your Python code.

Step3:

Navigate back to your project's directory and create a folder. Name it *logs*.

Step3:

Open the terminal and run the command: `sudo crontab –e`

Step4:

At the end of the file add the following line:

```
@reboot sleep 30 && sh /home/pi/Desktop/launcher >/home/pi/EESTEC/logs/cronlog 2>&1
```