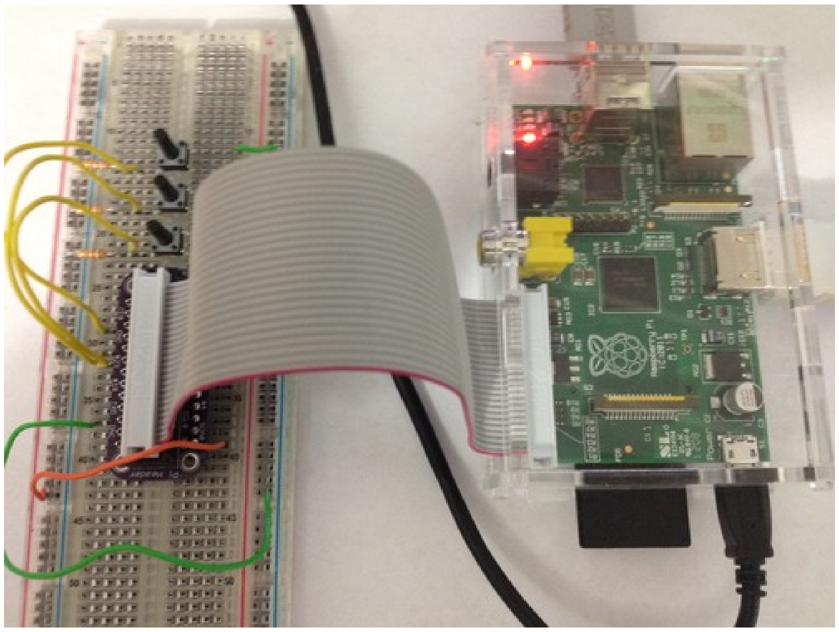
adafruit learning system

Playing sounds and using buttons with Raspberry Pi

Created by Mikey Sklar



Last updated on 2015-04-15 01:30:08 PM EDT

Guide Contents

Guide Contents 2

Overview 3

Install Audio 4

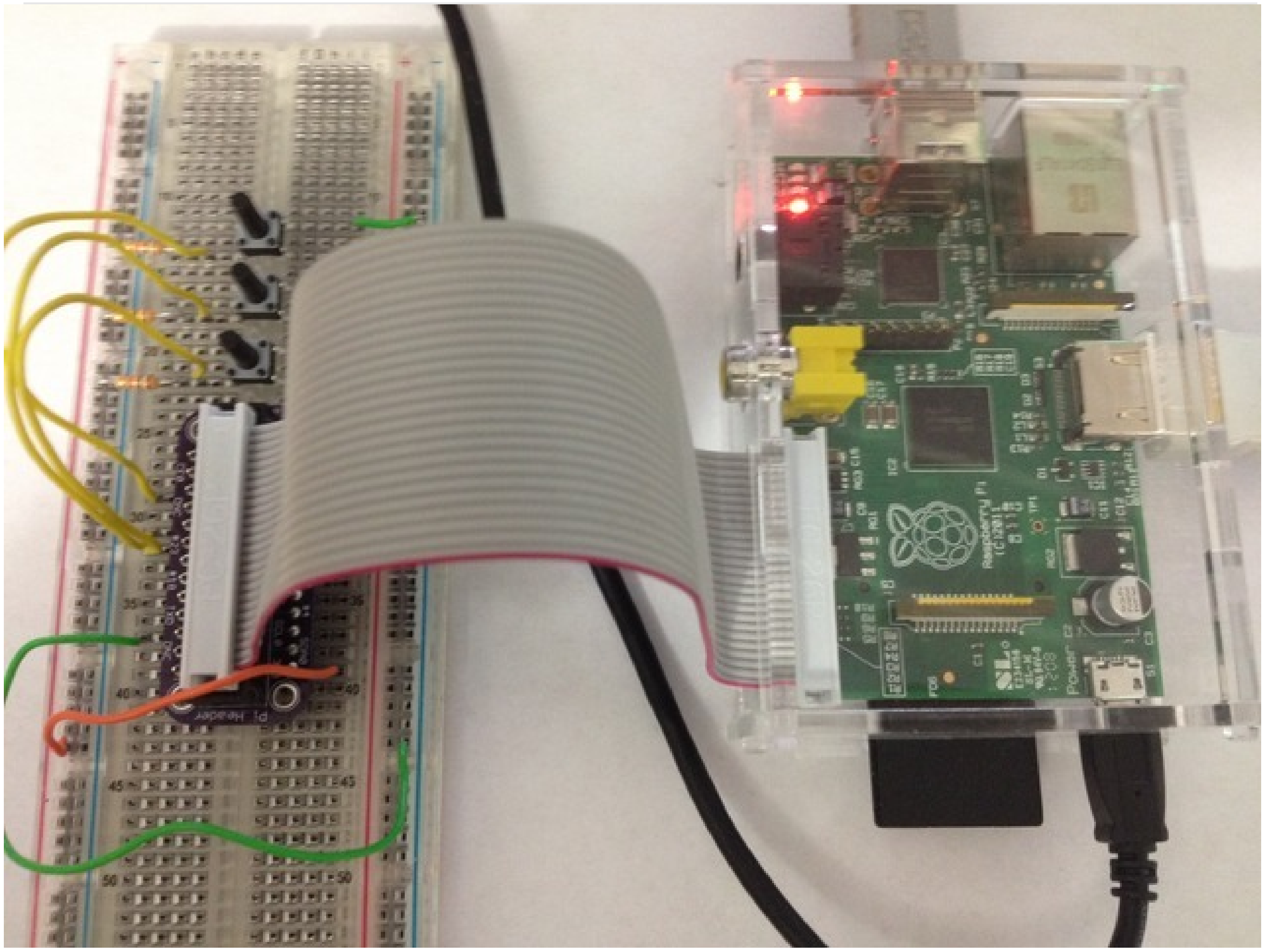
Install Python Module RPi.GPIO 6

Bread Board Setup for Input Buttons 7

Code 10

Fancier Code: A Very Simple Jukebox 12

Overview



One of the great things about the Raspberry Pi is how everyone starts with the same piece of gear. Since the audio hardware is identical on every unit, it is trivial to load the drivers and play mp3 files.

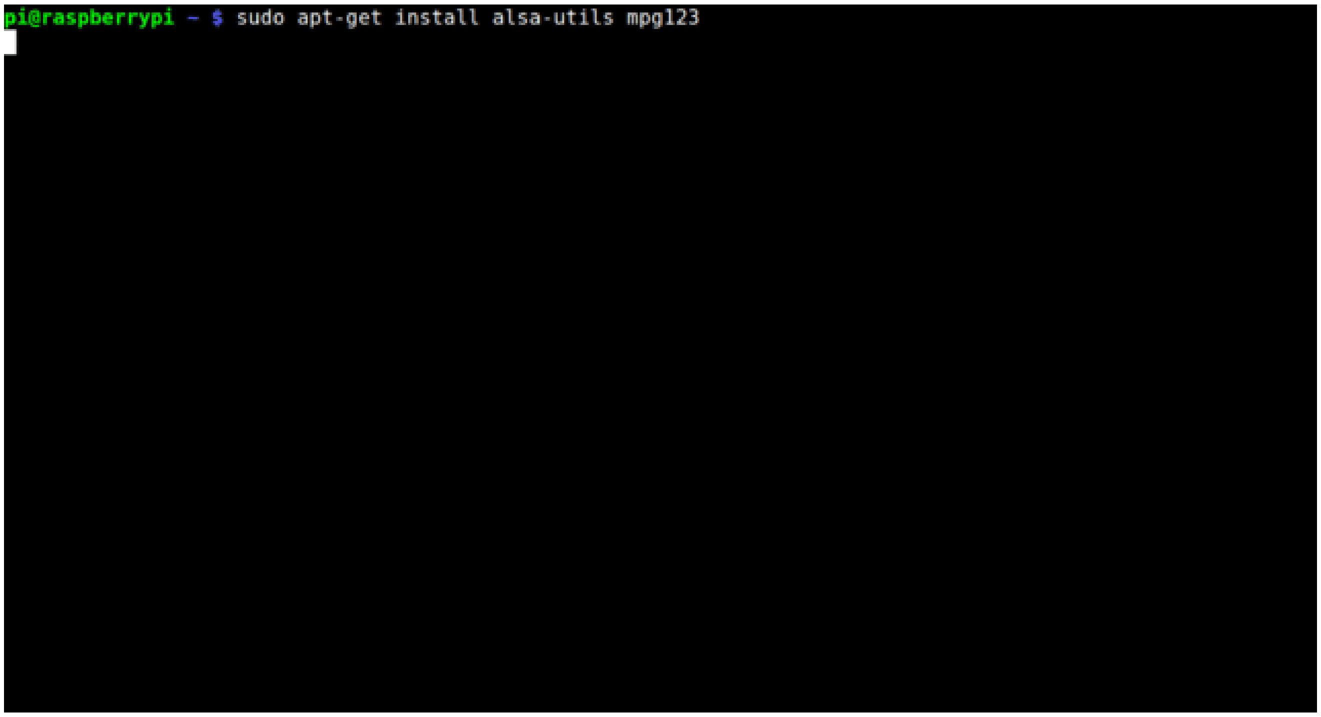
This guide describes how to connect input buttons and play audio files using a Raspberry Pi running Raspbian. We make use of Adafruit's Pi Cobbler and the Python module RPi.GPIO.

If you have not already used the Raspberry Pi as a input device this guide will show you how to wire the pull-down resistors, GPIO pins, and buttons.

Install Audio

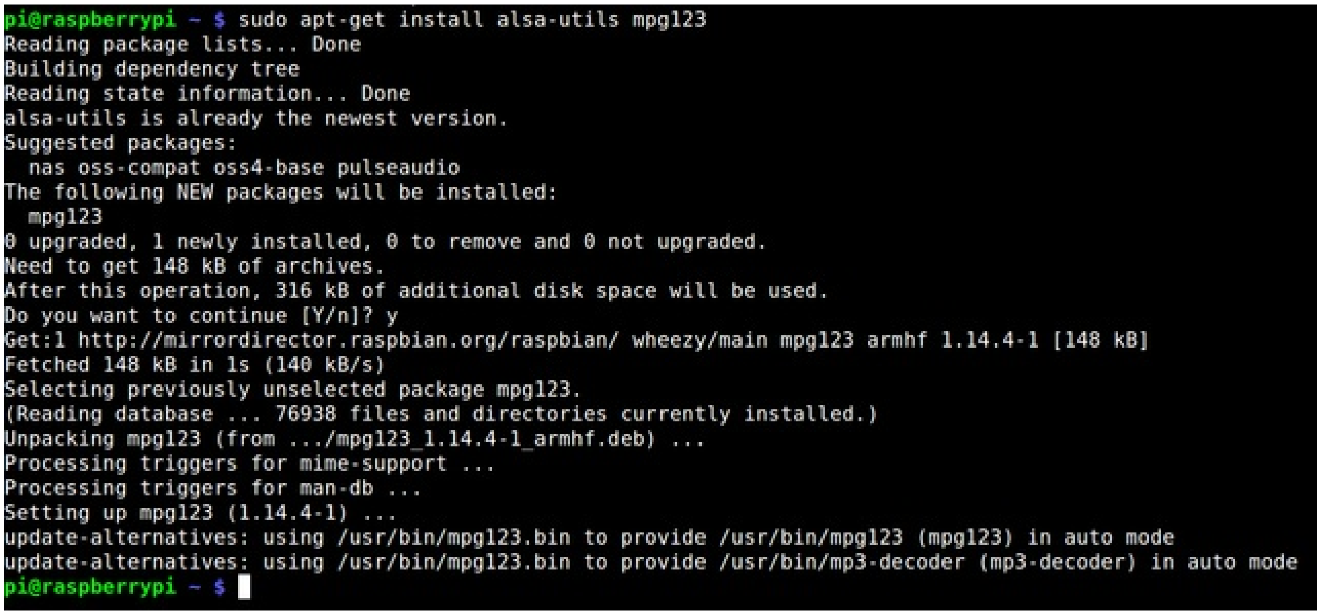
With the Pi connected to the Internet and SSHed in (see our previous tutorial (http://adafru.it/aJ5)), install the Alsa audio drivers and the mpg123 MP3 Player:

```
sudo apt-get update
sudo apt-get install alsa-utils mpg123
```



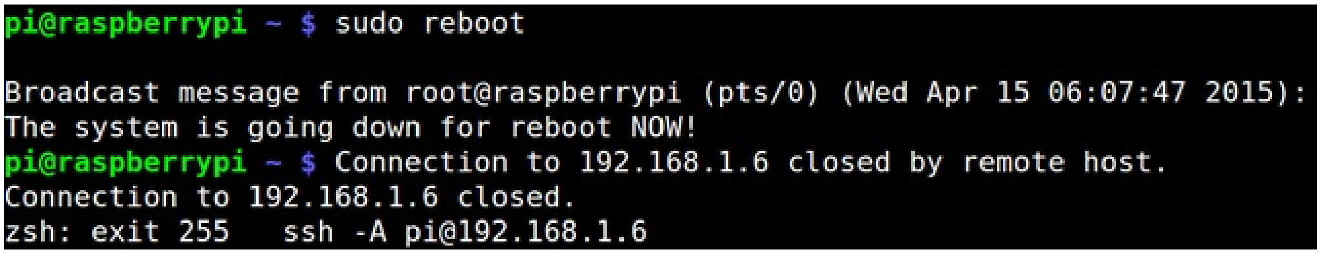
```
pi@raspberrypi ~ $ sudo apt-get install alsa-utils mpg123
```

You'll probably be prompted to continue. Press **y** for yes.



```
pi@raspberrypi ~ $ sudo apt-get install alsa-utils mpg123
Reading package lists... Done
Building dependency tree
Reading state information... Done
alsa-utils is already the newest version.
Suggested packages:
  nas oss-compat oss4-base pulseaudio
The following NEW packages will be installed:
  mpg123
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 148 kB of archives.
After this operation, 316 kB of additional disk space will be used.
Do you want to continue [Y/n]? y
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main mpg123 armhf 1.14.4-1 [148 kB]
Fetched 148 kB in 1s (140 kB/s)
Selecting previously unselected package mpg123.
(Reading database ... 76938 files and directories currently installed.)
Unpacking mpg123 (from .../mpg123_1.14.4-1_armhf.deb) ...
Processing triggers for mime-support ...
Processing triggers for man-db ...
Setting up mpg123 (1.14.4-1) ...
update-alternatives: using /usr/bin/mpg123.bin to provide /usr/bin/mpg123 (mpg123) in auto mode
update-alternatives: using /usr/bin/mpg123.bin to provide /usr/bin/mp3-decoder (mp3-decoder) in auto mode
pi@raspberrypi ~ $
```

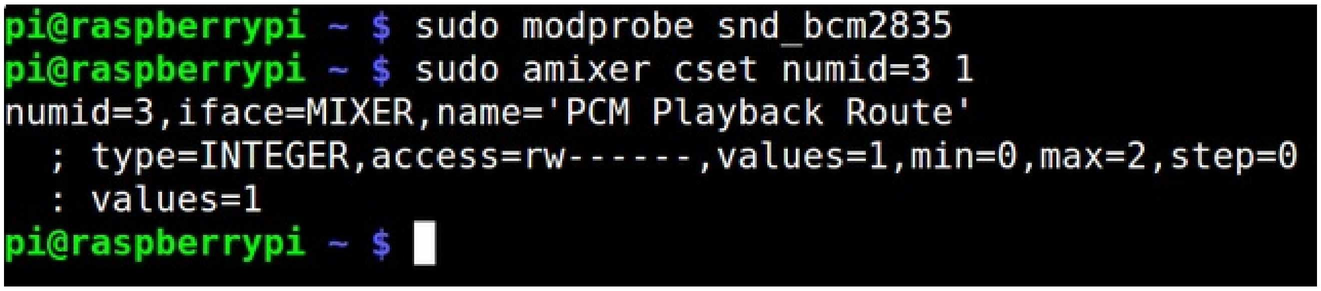
Next, **reboot the Pi** by typing `sudo reboot`. If you're using SSH, you'll have to reconnect after it boots.



```
pi@raspberrypi ~ $ sudo reboot

Broadcast message from root@raspberrypi (pts/0) (Wed Apr 15 06:07:47 2015):
The system is going down for reboot NOW!
pi@raspberrypi ~ $ Connection to 192.168.1.6 closed by remote host.
Connection to 192.168.1.6 closed.
zsh: exit 255    ssh -A pi@192.168.1.6
```

Once you're logged back in, load the sound drivers and do some setup for the 3.5mm jack output:

```
sudo modprobe snd_bcm2835
sudo amixer cset numid=3 1
```



```
pi@raspberrypi ~ $ sudo modprobe snd_bcm2835
pi@raspberrypi ~ $ sudo amixer cset numid=3 1
numid=3,iface=MIXER,name='PCM Playback Route'
  ; type=INTEGER,access=rw------,values=1,min=0,max=2,step=0
  : values=1
pi@raspberrypi ~ $
```

Install Python Module RPi.GPIO

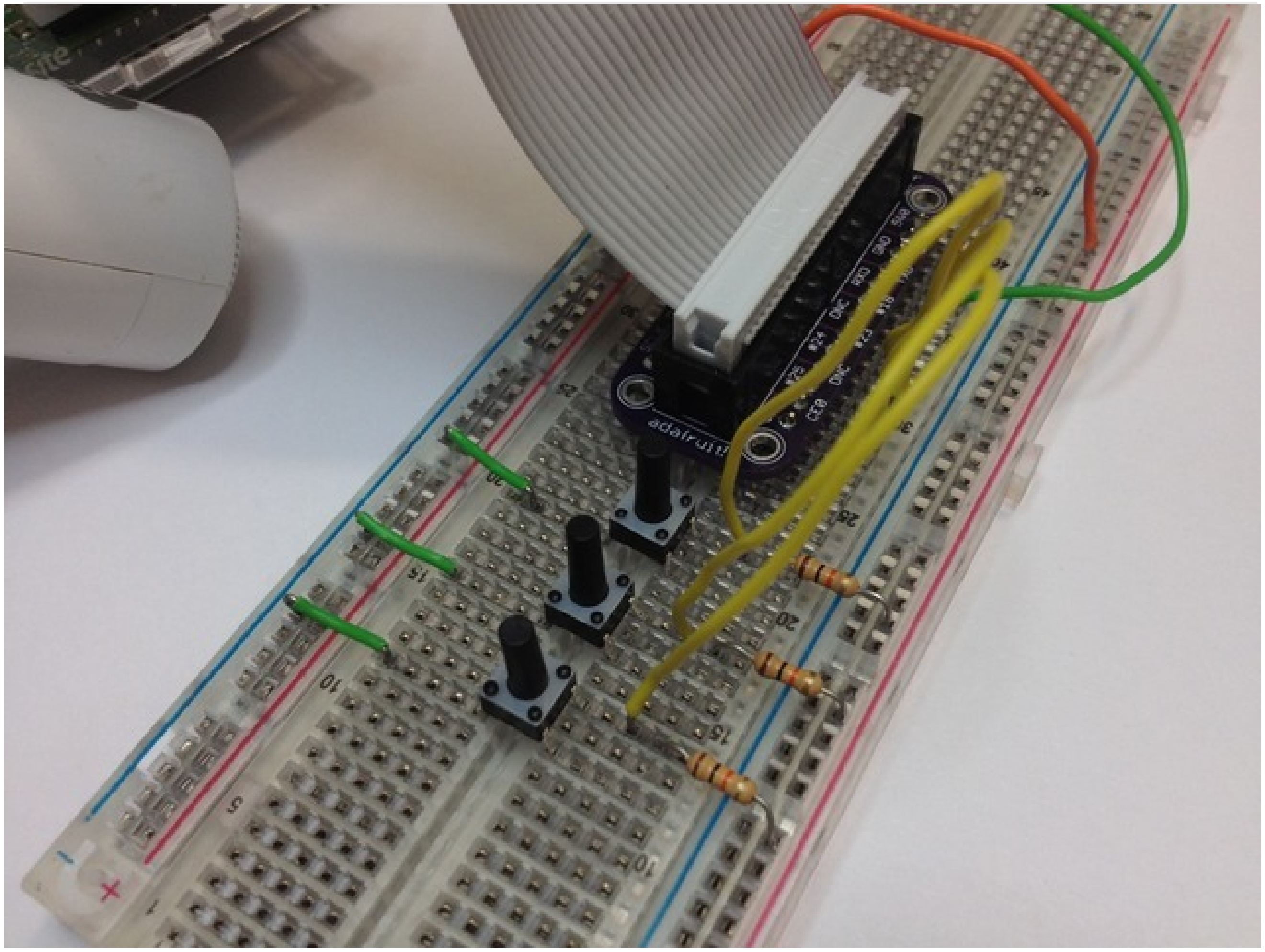
The RPi.GPIO python module offers easy access to the general purpose IO pins on the Raspberry Pi.

To get the latest version of this, you can take a little diversion to follow the instructions in Adafruit's Raspberry Pi Lesson 4. GPIO Setup (http://adafru.it/eXn).

If you're comfortable, you can probably skip the lesson above and issue the following command in a terminal window:

```
sudo apt-get install python-dev python-rpi.gpio
```

Bread Board Setup for Input Buttons

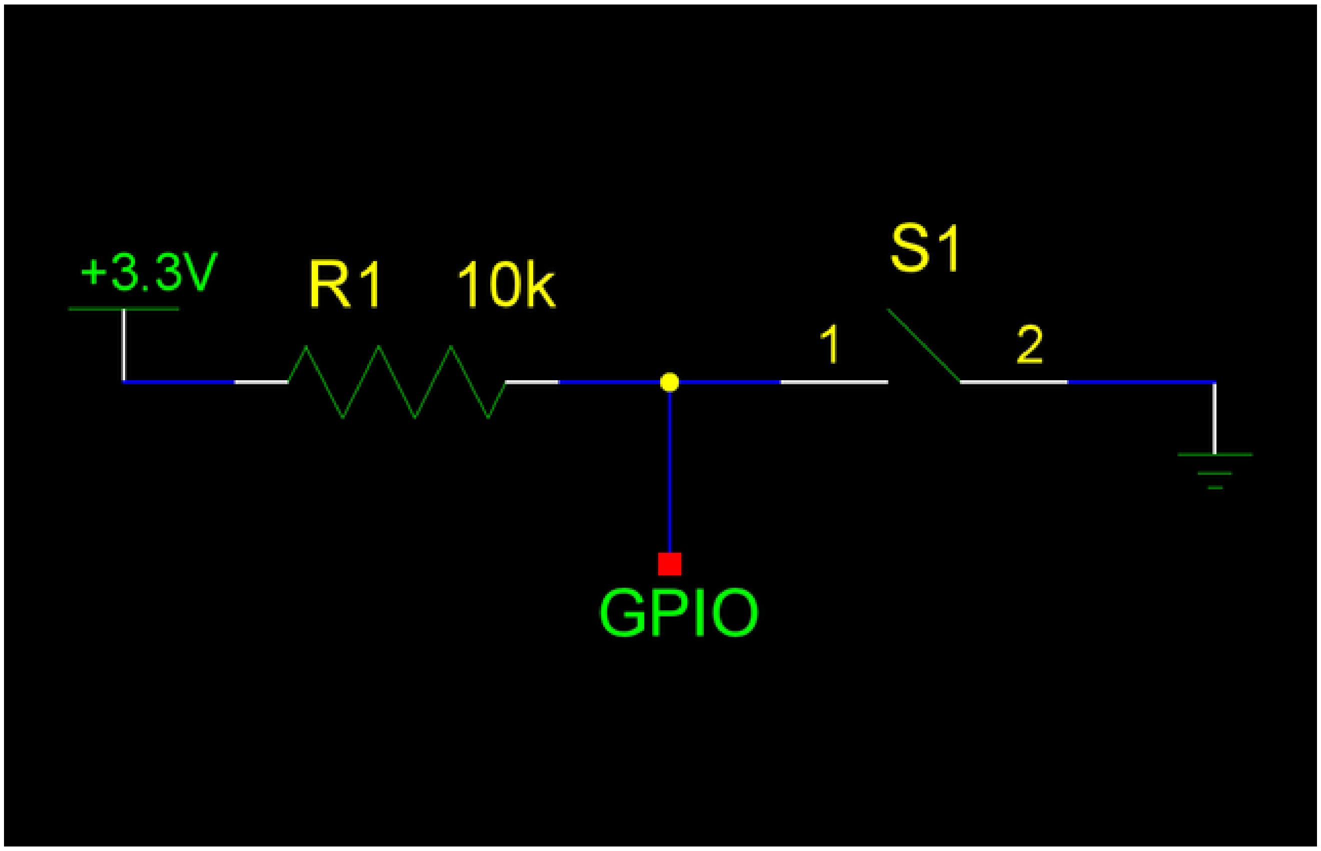


Important things to note:

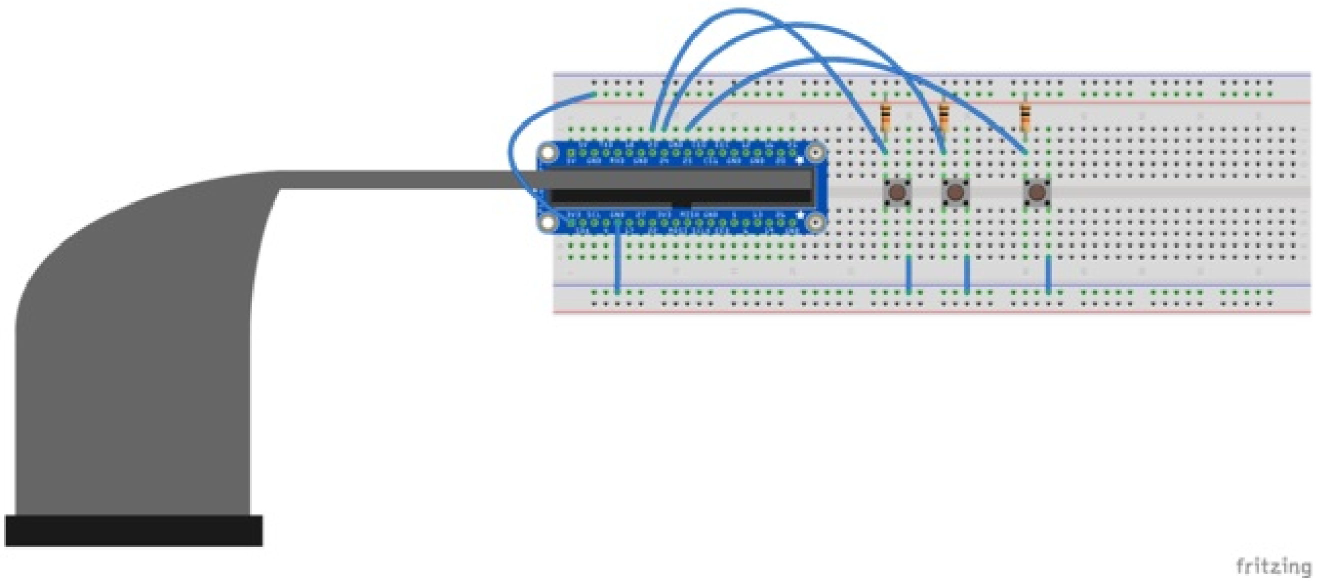
- The Adafruit Pi Cobbler (http://adafru.it/914) is being used; for the A+/B+/Pi 2, the Pi Cobbler Plus (http://adafru.it/2029) should be used instead
- The Adafruit Clear Full sized breadboard (http://adafru.it/239) is being used
- (3) 10k pull-up resistors
- (3) Momentary push-button switches (http://adafru.it/367)
- GPIO pins 23, 24, and 25
- Cobbler 3.3v rail is tied to positive breadboard
- Cobbler gnd rail is tied to negative breadboard

Each button connection looks like:

3.3v --> 10k Pull-up Resistor --> GPIO --> Button --> GND



Here's a Fritzing (http://adafru.it/eXu) sketch of the layout for the Cobbler Plus for a Pi 1 Model B+ or Pi 2 (click for a bigger image):



If you've got Fritzing installed, you can download the sketch here:

Download Sketch

http://adafru.it/eXv

Code

Now for a bit of Python:

- Use a text editor like Nano to paste this code into a file named `raspi-audio-button.py`
- Download or copy several mp3 files to the Pi and place them into the same directory as the `raspi-audio-button.py` script
- Change the filenames in the code to match the files you've just copied
- Make the file executable with `chmod`

```
nano raspi-audio-button.py
```

```python
#!/usr/bin/env python

import os
from time import sleep

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN)
GPIO.setup(24, GPIO.IN)
GPIO.setup(25, GPIO.IN)

while True:
    if (GPIO.input(23) == False):
        os.system('mpg123 -q binary-language-moisture-evaporators.mp3 &')

    if (GPIO.input(24) == False):
        os.system('mpg123 -q power-converters.mp3 &')

    if (GPIO.input(25)== False):
        os.system('mpg123 -q vader.mp3 &')

    sleep(0.1);
```

```
$ chmod +x raspi-audio-button.py
```

Make sure you have speakers or headphones hooked up to the 3.5mm jack, and run the Python program as an administrator using `sudo`:

```
$ sudo python raspi-audio-button.py
```

Now you should be able to trigger each mp3 by hitting the corresponding button.

A handful of things worth noticing here:

- `os.system('command')` will run `command` just like if you typed it at the prompt.
- The `&` tells the shell to run the command in the background. This way you can actually play more than one file at once.
- The `-q` option to `mpg123` suppresses diagnostic messages. You can remove it if you'd like to see song titles.
- The `sleep(0.1)` call is necessary to avoid spawning tons of `mpg123` calls from a single button press.

Fancier Code: A Very Simple Jukebox

Our first Python script uses every button to do pretty much the same thing: Each runs a command to play a different mp3.

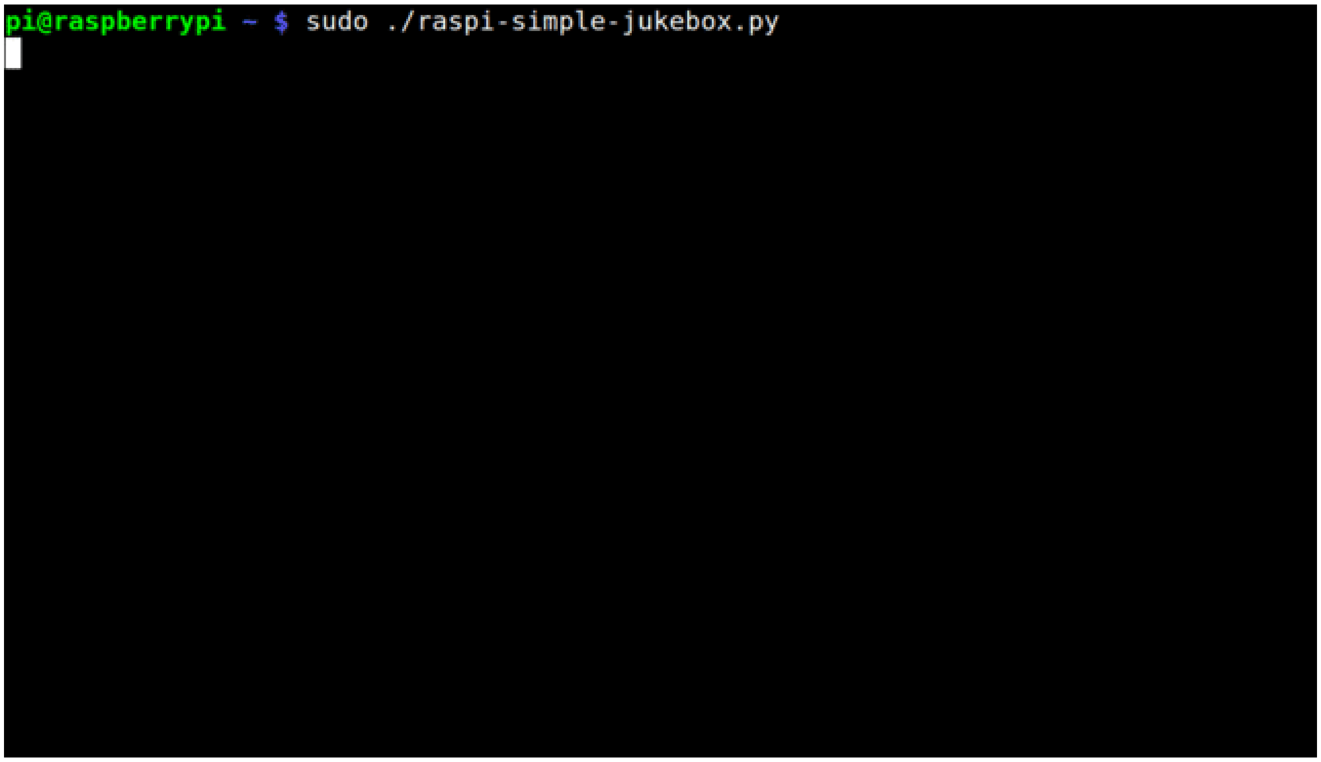
What if you want to use the 3-button scheme to control *lots* of mp3s?

Try the following script:

```
#!/usr/bin/env python

from os import listdir
import subprocess
from time import sleep

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN)
GPIO.setup(24, GPIO.IN)
GPIO.setup(25, GPIO.IN)

mp3_files = [ f for f in listdir('.') if f[-4:] == '.mp3' ]

if not (len(mp3_files) > 0):
    print "No mp3 files found!"

print '--- Available mp3 files ---'
print mp3_files
print '--- Press button #1 to select mp3, button #2 to play current. ---'

index = 0
while True:
    if (GPIO.input(23) == False):
        index += 1
        if index >= len(mp3_files):
            index = 0
        print "--- " + mp3_files[index] + " ---"

    if (GPIO.input(24) == False):
        subprocess.Popen(['mpg123', mp3_files[index]])
        print '--- Playing ' + mp3_files[index] + ' ---'
        print '--- Press button #3 to clear playing mp3s. ---'
        sleep(1)

    if (GPIO.input(25) == False):
        subprocess.call(['killall', 'mpg123'])
        print '--- Cleared all existing mp3s. ---'

    sleep(0.1);
```

Here, we get a list of mp3 files in the current directory, and then set up the following control scheme:

- **button #1** (GPIO 23) clicks through the list of available mp3s
- **button #2** (GPIO 24) plays the currently selected mp3
- **button #3** (GPIO 25) runs a command that kills any existing mp3 player processes, stopping whatever mp3s are playing right now



```
pi@raspberrypi ~ $ sudo ./raspi-simple-jukebox.py
```