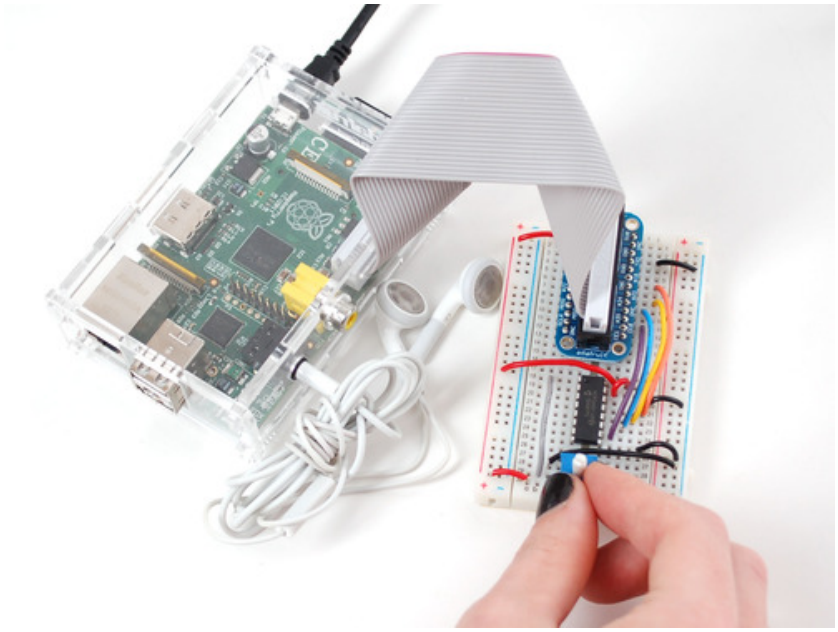
adafruit learning system

Analog Inputs for Raspberry Pi Using the MCP3008

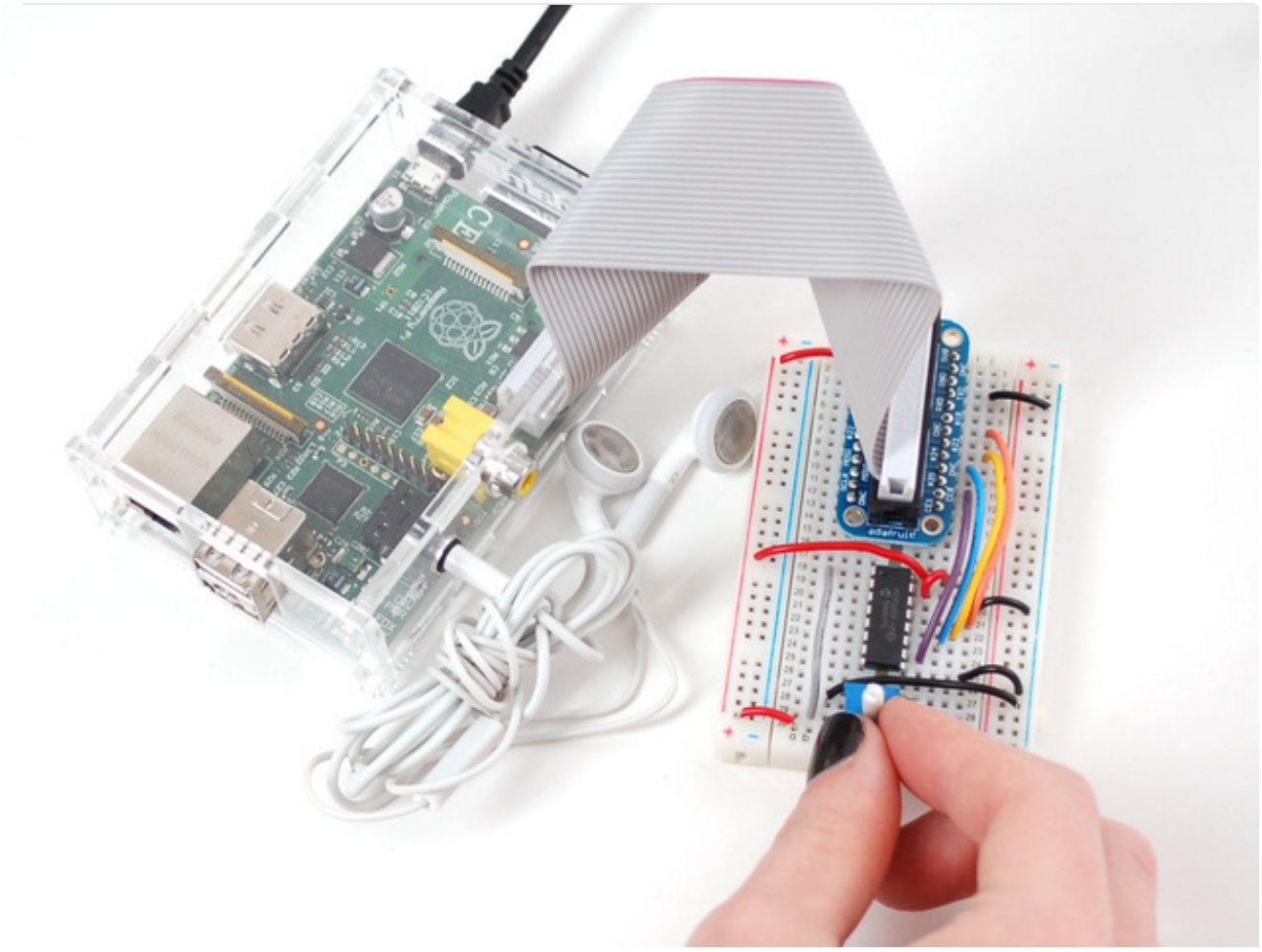
Created by Mikey Sklar



Last updated on 2015-06-18 01:10:10 PM EDT

Guide Contents

Overview



Teaching the Raspberry Pi how to read analog inputs is easier than you think! The Pi does not include a hardware analog-to-digital converter (http://adafru.it/eYp), but an external ADC (such as the MCP3008 (http://adafru.it/856)) can be used, along with some bit banged SPI code in Python to read external analog devices.

Here is a short list of some analog inputs that could be used with this setup:

- potentiometer (http://adafru.it/356)
- photocell (http://adafru.it/161)
- force sensitive resistor (FSR) (http://adafru.it/166)
- temperature sensor (http://adafru.it/165)
- 2-axis joystick (http://adafru.it/512)

This guide uses a potentiometer to control the volume of a mp3 file being played, but the code can be used as the basis for any kind of analog-input project.

Connecting the Cobbler to a MCP3008

To follow this tutorial you will need

- MCP3008 DIP-package ADC converter chip (http://adafru.it/856)
- 10K trimer (http://adafru.it/356) or panel mount potentiometer (http://adafru.it/562)
- Adafruit Pi Cobbler (http://adafru.it/914) or Pi Cobbler Plus (http://adafru.it/2029)
- Half (http://adafru.it/64) or Full-size breadboard (http://adafru.it/239) (use a full-size one with the Cobbler Plus)
- Breadboarding wires (http://adafru.it/aHz)

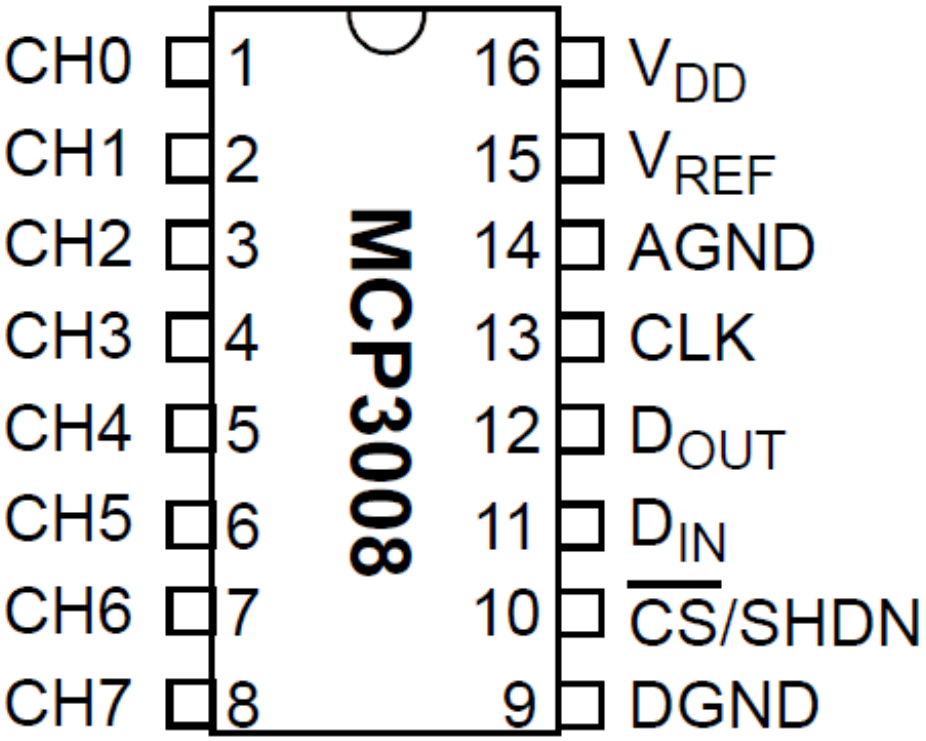
And of course a working Raspberry Pi with an Internet connection.

Why we need an ADC

The Raspberry Pi computer does not have a way to read analog inputs. It's a digital-only computer. Compare this to the Arduino, AVR or PIC microcontrollers that often have 6 or more analog inputs! Analog inputs are handy because many sensors are analog outputs, so we need a way to make the Pi analog-friendly.

We'll do that by wiring up an MCP3008 chip (http://adafru.it/856) to it. The MCP3008 (http://adafru.it/856) acts like a "bridge" between digital and analog. It has 8 analog inputs and the Pi can query it using 4 digital pins. That makes it a perfect addition to the Pi for integrating simple sensors like photocells (http://adafru.it/aHA), FSRs (http://adafru.it/aHC) or potentiometers, thermistors (http://adafru.it/aHD), etc.!

Let's check the datasheet of the MCP3008 chip. (http://adafru.it/aHE) On the first page in the lower right corner there's a pinout diagram showing the names of the pins:



Wiring Diagram

In order to read analog data we need to use the following pins:

VDD (power) and **DGND** (digital ground) to power the MCP3008 chip. We also need four "SPI" data pins: **DOUT** (Data Out from MCP3008), **CLK** (Clock pin), **DIN** (Data In from Raspberry Pi), and /**CS** (Chip Select). Finally of course, a source of analog data. We'll be using the basic 10k trim pot.

The MCP3008 has a few more pins we need to connect: **AGND** (analog ground, used sometimes in precision circuitry, which this is not) connects to **GND**, and **VREF** (analog voltage reference, used for changing the "scale" - we want the full scale, so tie it to **3.3V**).

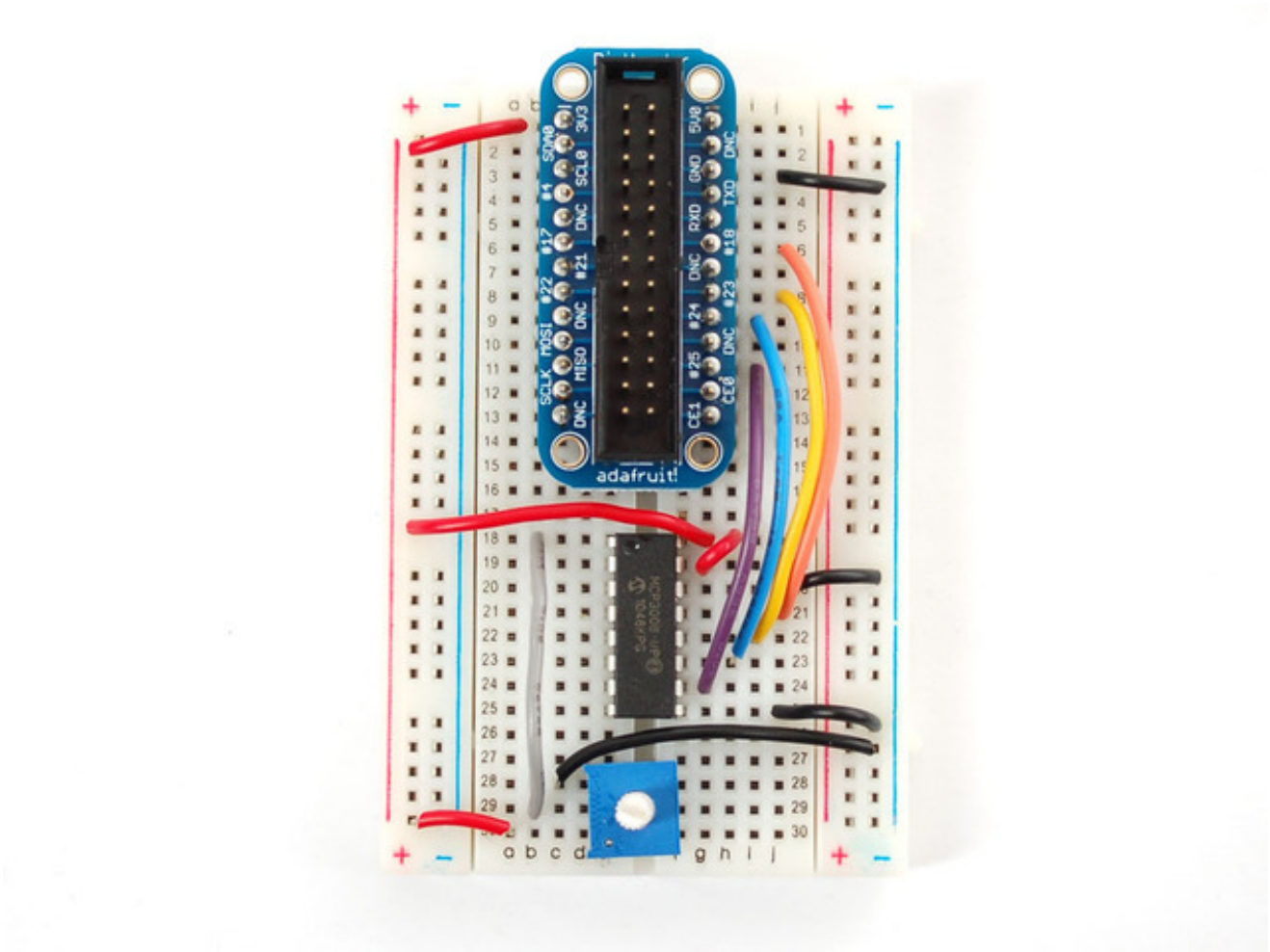
Below is a wiring diagram. Connect the 3.3V cobbler pin to the left + rail and the GND pin to the right - rail. Connect the following pins for the MCP chip

- MCP3008 VDD -> 3.3V (red)
- MCP3008 VREF -> 3.3V (red)
- MCP3008 AGND -> GND (black)
- MCP3008 CLK -> #18 (orange)
- MCP3008 DOUT -> #23 (yellow)

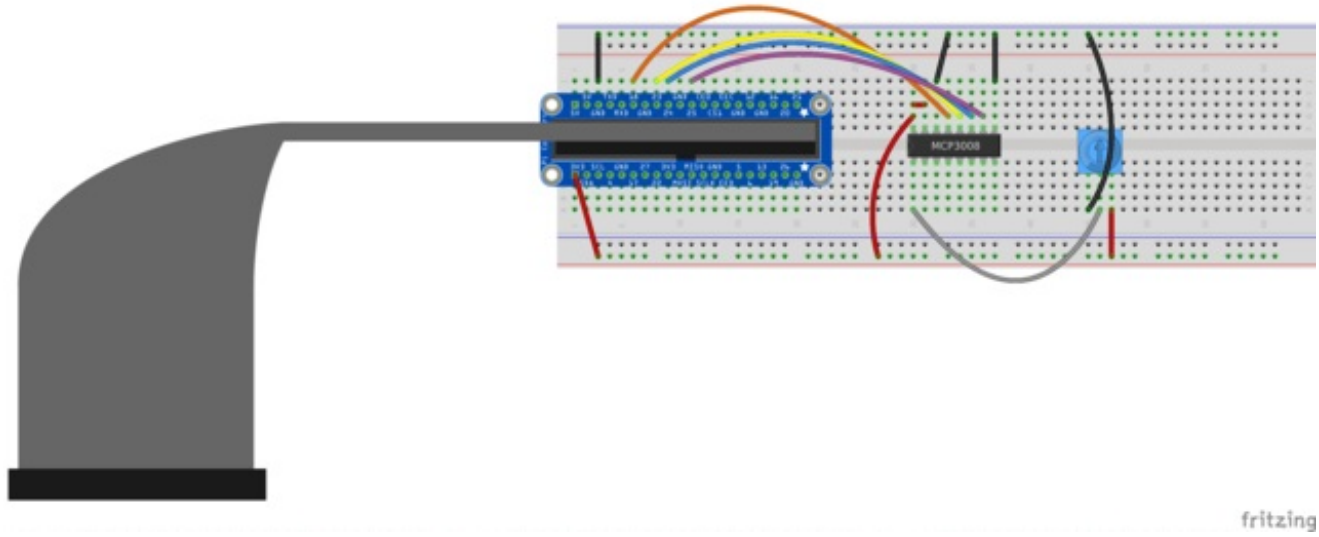
- MCP3008 DIN -> #24 (blue)
- MCP3008 CS -> #25 (violet)
- MCP3008 DGND -> GND (black)

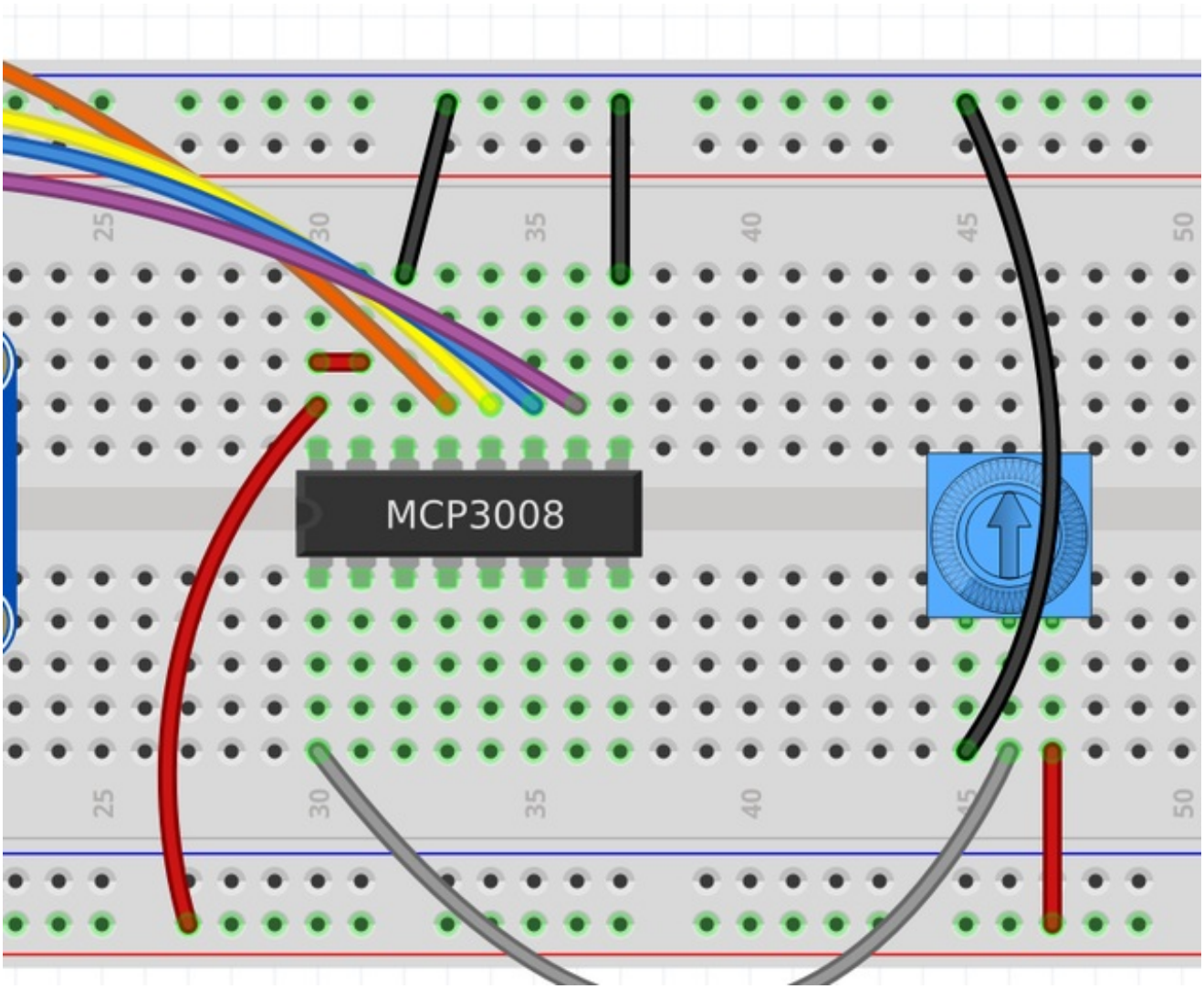
Next connect up the potentiometer. Pin #1 (left) goes to GND (black), #2 (middle) connects to MCP3008 **CH0** (analog input #0) with a gray wire, and #3 (right) connects to 3.3V (red)

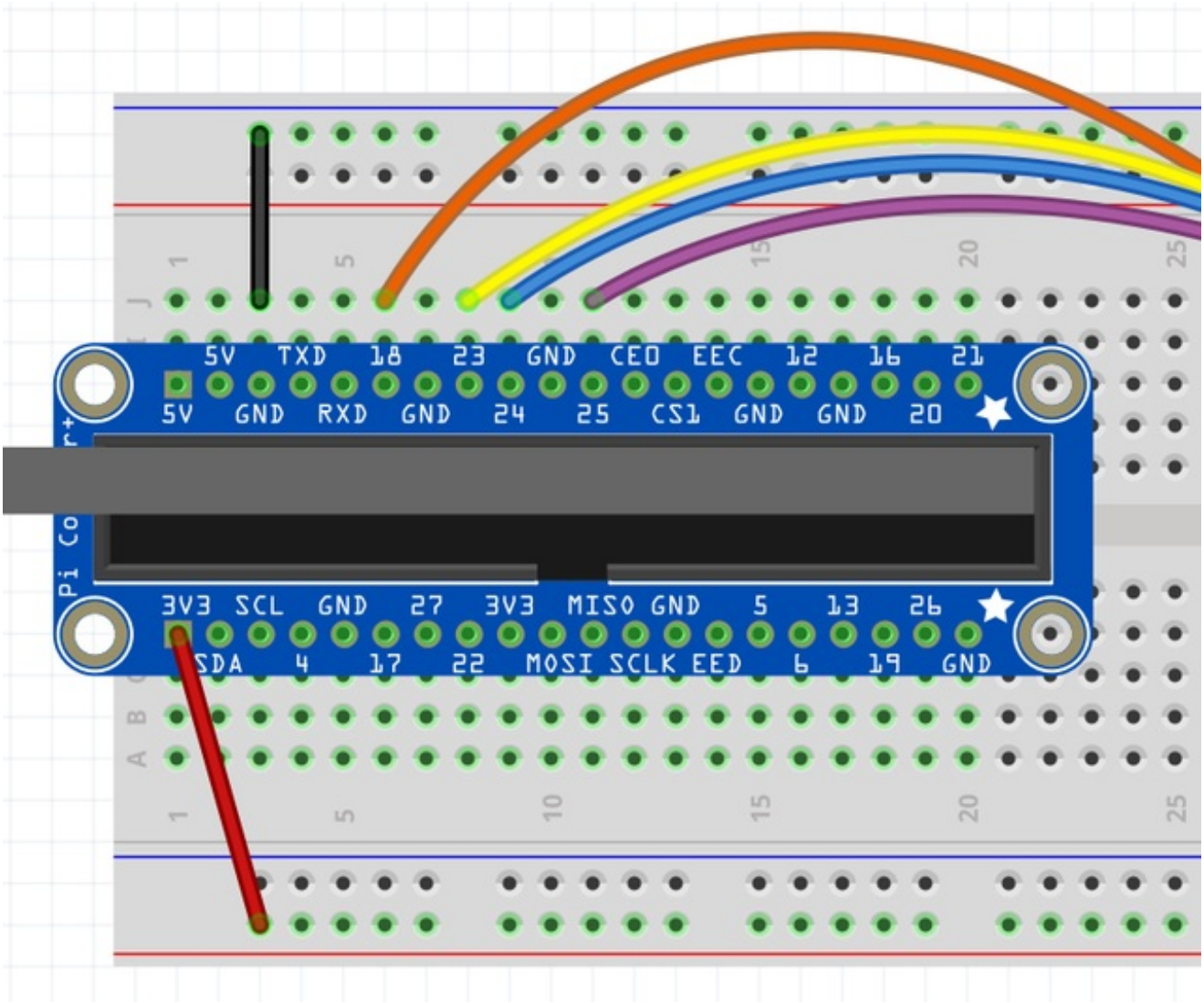
Advanced users may note that the Raspberry Pi does have a hardware SPI interface (the Cobbler pins are labeled MISO/MOSI/SCLK/CE0/CE1). The hardware SPI interface is super fast but not included in all distributions. For that reason we are using a bit banged SPI implementation so the SPI pins can be any of the Raspberry Pi's GPIOs (assuming you update the script).



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


MCP3008
fritzing


MCP3008
25
30
35
40
45
50



Download Sketch

http://adafru.it/fql

Necessary Packages

If you've already worked through Playing Sounds and Using Buttons with the Raspberry Pi (http://adafru.it/eXD), you're probably good to go here. Otherwise, you may need a few things. Open up a terminal, and enter the following commands:

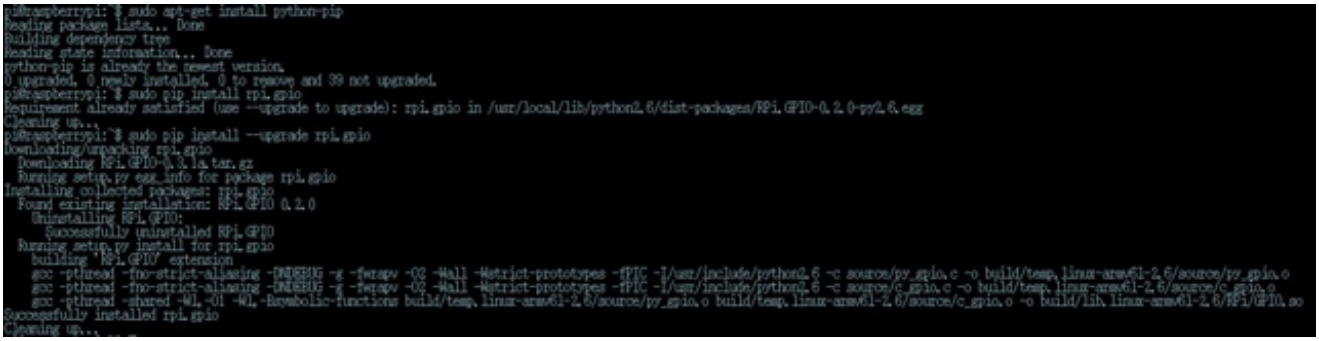
Update Python (2.x) to the latest release:

```
sudo apt-get update
sudo apt-get install python-dev
```

```
pi@raspberrypi: $ sudo apt-get install python-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libpython2.6 python python-minimal python2.6-dev
Suggested packages:
  python-doc python-tk python-profiler
The following NEW packages will be installed:
  libpython2.6 python-dev python2.6-dev
The following packages will be upgraded:
  python python-minimal
2 upgraded, 3 newly installed, 0 to remove and 39 not upgraded.
Need to get 5,655 kB of archives.
After this operation, 14.3 MB of additional disk space will be used.
Do you want to continue [Y/n]?
Get:1 http://ftp.uk.debian.org/debian/ squeeze/main python all 2.6.6-3+squeeze7 [169 kB]
Get:2 http://ftp.uk.debian.org/debian/ squeeze/main python-minimal all 2.6.6-3+squeeze7 [33.8 kB]
Get:3 http://ftp.uk.debian.org/debian/ squeeze/main libpython2.6 armel 2.6.6-8+b1 [985 kB]
Get:4 http://ftp.uk.debian.org/debian/ squeeze/main python2.6-dev armel 2.6.6-8+b1 [4,466 kB]
Get:5 http://ftp.uk.debian.org/debian/ squeeze/main python-dev all 2.6.6-3+squeeze7 [916 B]
Fetched 5,655 kB in 1min 16s (74.0 kB/s)
(Reading database ... 50265 files and directories currently installed.)
Preparing to replace python 2.6.6-3+squeeze6 (using .../python_2.6.6-3+squeeze7_all.deb) ...
Unpacking replacement python ...
Preparing to replace python-minimal 2.6.6-3+squeeze6 (using .../python-minimal_2.6.6-3+squeeze7_all.deb) ...
Unpacking replacement python-minimal ...
Selecting previously deselected package libpython2.6.
Unpacking libpython2.6 (from .../libpython2.6_2.6.6-8+b1_armel.deb) ...
Selecting previously deselected package python2.6-dev.
Unpacking python2.6-dev (from .../python2.6-dev_2.6.6-8+b1_armel.deb) ...
Selecting previously deselected package python-dev.
Unpacking python-dev (from .../python-dev_2.6.6-3+squeeze7_all.deb) ...
Processing triggers for man-db ...
Setting up python-minimal (2.6.6-3+squeeze7) ...
Setting up python (2.6.6-3+squeeze7) ...
Setting up libpython2.6 (2.6.6-8+b1) ...
Setting up python2.6-dev (2.6.6-8+b1) ...
Setting up python-dev (2.6.6-3+squeeze7) ...
```

Install the latest RPi.GPIO module. We will use easy_install to manage the python packages.

```
sudo apt-get install python-setuptools
sudo easy_install rpi.gpio
```



Install the ALSA sound utilities and a mp3 player:

```
$ sudo apt-get install alsa-utils
$ sudo apt-get install mpg321
```

Python Script

This ~100 line python script can be pasted into an editor and saved on your raspberry pi. You can also grab it directly from the pi if it's connected to the Internet by running `git clone git://gist.github.com/3151375.git`

The script is fairly simple. Half of the code (the **readadc** function) is a function that will 'talk' to the MCP3008 chip using four digital pins to 'bit bang' the SPI interface (this is because not all Raspberry Pi's have the hardware SPI function).

The MCP3008 is a 10-bit ADC. That means it will read a value from 0 to 1023 (2^{10} = 1024 values) where 0 is the same as "ground" and "1023" is the same as "3.3 volts". We don't convert the number to voltage, although its easy to do that by multiplying the number by (3.3 / 1023).

We check to see if the pot was turned more than 5 counts - this keeps us from being too "jittery" and resetting the volume too often.

The raw analog count number is then converted into a volume percentage of 0%-100%. When the trimpot is turned up or down it will print the volume level to `STDOUT` and adjust the audio level of the playing file by telling the mixer to adjust the volume.

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

# Written by Limor "Ladyada" Fried for Adafruit Industries, (c) 2015
# This code is released into the public domain

import time
import os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
DEBUG = 1

# read SPI data from MCP3008 chip, 8 possible adc's (0 thru 7)
def readadc(adcnum, clockpin, mosipin, misopin, cspin):
        if ((adcnum > 7) or (adcnum < 0)):
                return -1
        GPIO.output(cspin, True)

        GPIO.output(clockpin, False)  # start clock low
        GPIO.output(cspin, False)     # bring CS low

        commandout = adcnum
        commandout |= 0x18  # start bit + single-ended bit
        commandout <<= 3    # we only need to send 5 bits here
```

```
        for i in range(5):
                if (commandout & 0x80):
                        GPIO.output(mosipin, True)
                else:
                        GPIO.output(mosipin, False)
                commandout <<= 1
                GPIO.output(clockpin, True)
                GPIO.output(clockpin, False)

        adcout = 0
        # read in one empty bit, one null bit and 10 ADC bits
        for i in range(12):
                GPIO.output(clockpin, True)
                GPIO.output(clockpin, False)
                adcout <<= 1
                if (GPIO.input(misopin)):
                        adcout |= 0x1

        GPIO.output(cspin, True)

        adcout >>= 1       # first bit is 'null' so drop it
        return adcout

# change these as desired - they're the pins connected from the
# SPI port on the ADC to the Cobbler
SPICLK = 18
SPIMISO = 23
SPIMOSI = 24
SPICS = 25

# set up the SPI interface pins
GPIO.setup(SPIMOSI, GPIO.OUT)
GPIO.setup(SPIMISO, GPIO.IN)
GPIO.setup(SPICLK, GPIO.OUT)
GPIO.setup(SPICS, GPIO.OUT)

# 10k trim pot connected to adc #0
potentiometer_adc = 0;

last_read = 0       # this keeps track of the last potentiometer value
tolerance = 5       # to keep from being jittery we'll only change
                # volume when the pot has moved more than 5 'counts'

while True:
        # we'll assume that the pot didn't move
        trim_pot_changed = False
```

```
    # read the analog pin
    trim_pot = readadc(potentiometer_adc, SPICLK, SPIMOSI, SPIMISO, SPICS)
    # how much has it changed since the last read?
    pot_adjust = abs(trim_pot - last_read)

    if DEBUG:
            print "trim_pot:", trim_pot
            print "pot_adjust:", pot_adjust
            print "last_read", last_read

    if ( pot_adjust > tolerance ):
           trim_pot_changed = True

    if DEBUG:
            print "trim_pot_changed", trim_pot_changed

    if ( trim_pot_changed ):
            set_volume = trim_pot / 10.24           # convert 10bit adc0 (0-1024) trim pot read into 0-100 volume
            set_volume = round(set_volume)          # round out decimal value
            set_volume = int(set_volume)            # cast volume as integer

            print 'Volume = {volume}%' .format(volume = set_volume)
            set_vol_cmd = 'sudo amixer cset numid=1 -- {volume}% > /dev/null' .format(volume = set_volume)
            os.system(set_vol_cmd)  # set volume

            if DEBUG:
                    print "set_volume", set_volume
                    print "tri_pot_changed", set_volume

            # save the potentiometer reading for the next loop
            last_read = trim_pot

    # hang out and do nothing for a half second
    time.sleep(0.5)
```

After you have pasted this script into a file, make it executable:

```
chmod +x raspi-adc-pot.py
```

Run It

On every boot, the sound module will need to be loaded and set to output to the 3.5mm audio jack:

```
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 ~ $
```

Next, play a mp3 file:

```
mpg321 <filename>
```

Leave the file playing and open a new terminal window or SSH connection to start the Python script:

```
sudo ./raspi-adc-pot.py
```

Now simply adjust the trim pot and you should hear the audio level change as the mp3 file is playing.