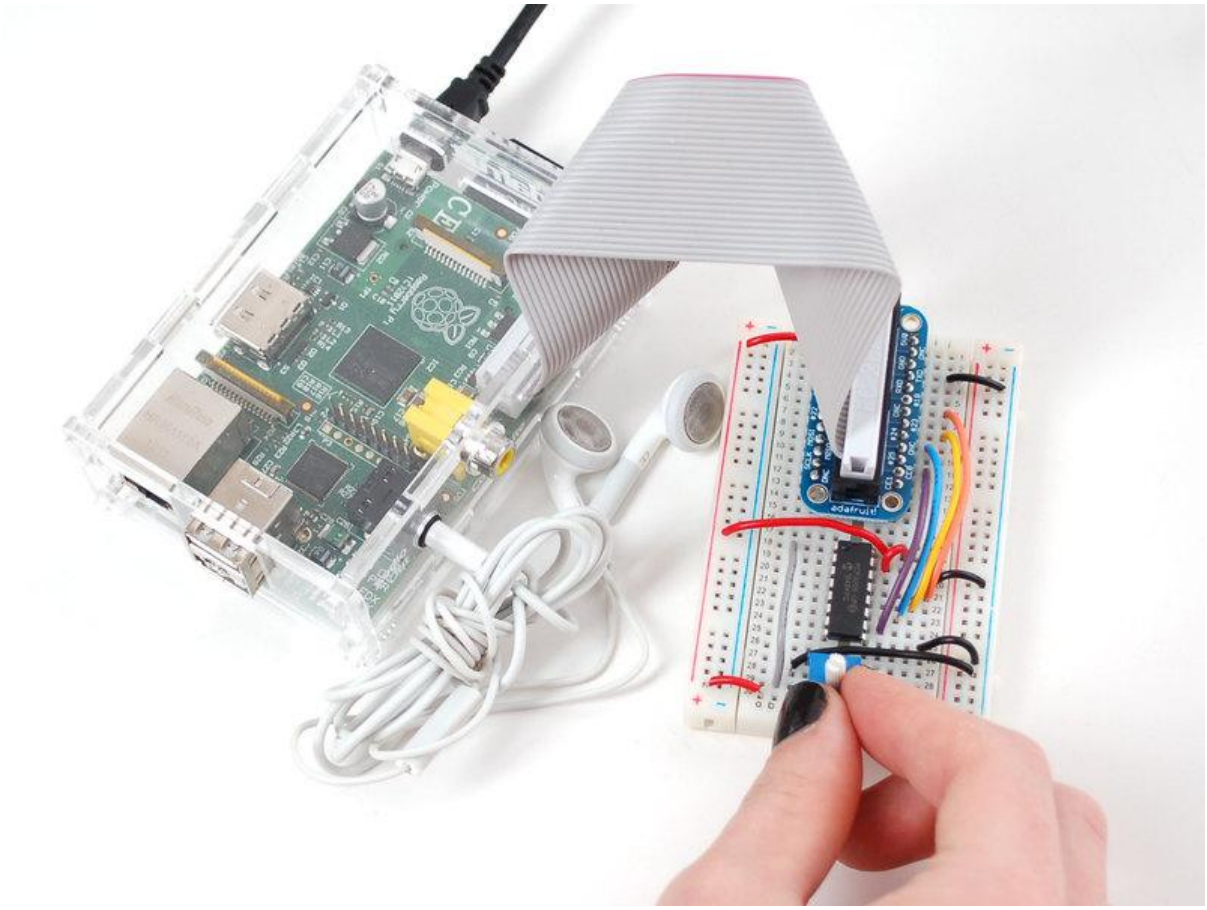
Analog Inputs for Raspberry Pi Using the MCP3008

Created by Michael Sklar

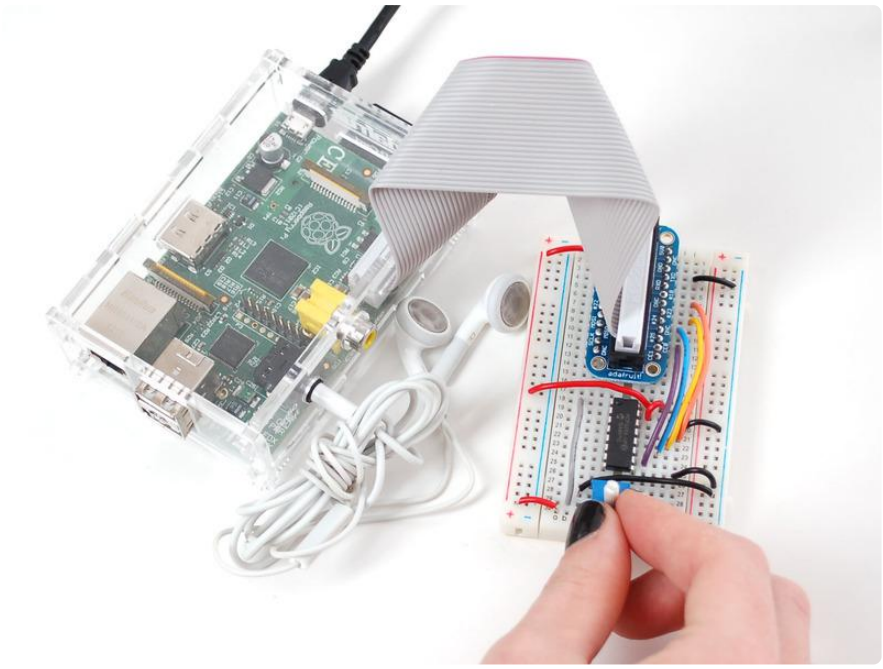


https://learn.adafruit.com/reading-a-analog-in-and-controlling-audio-volume-with-the-raspberry-pi

Last updated on 2022-12-01 01:50:33 PM EST

Table of 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 (), but an external ADC (such as the MCP3008 (http://adafru.it/856)) can be used, along with some 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 an audio tone, 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 T-Cobbler Plus () or for an older 26-pin Pi an Adafruit Pi Cobbler (http://adafru.it/914)
- Full-size breadboard (http://adafru.it/239)
- Breadboarding wires ()

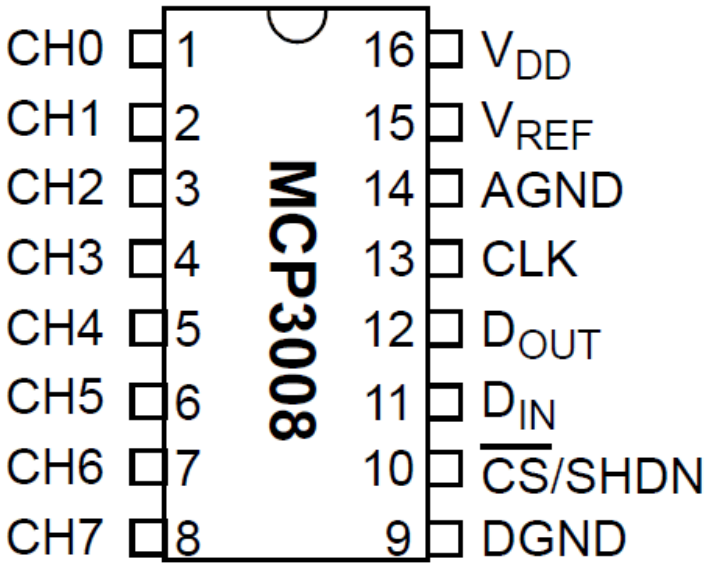
And of course a working Raspberry Pi.

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 (), FSRs () or potentiometers, thermistors (), etc.!

Let's check the datasheet of the MCP3008 chip. () 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

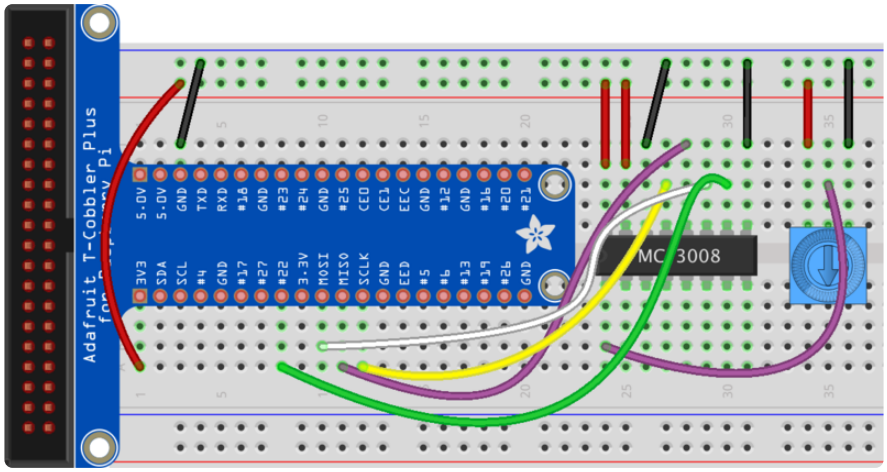
- MCP3008 VDD -> 3.3V (red)
- MCP3008 VREF -> 3.3V (red)
- MCP3008 AGND -> GND (black)
- MCP3008 CLK -> SCLK (yellow)
- MCP3008 DOUT -> MISO (purple)
- MCP3008 DIN -> MOSI (white)
- MCP3008 CS -> #22 (green)
- MCP3008 DGND -> GND (black)

Next connect up the potentiometer.

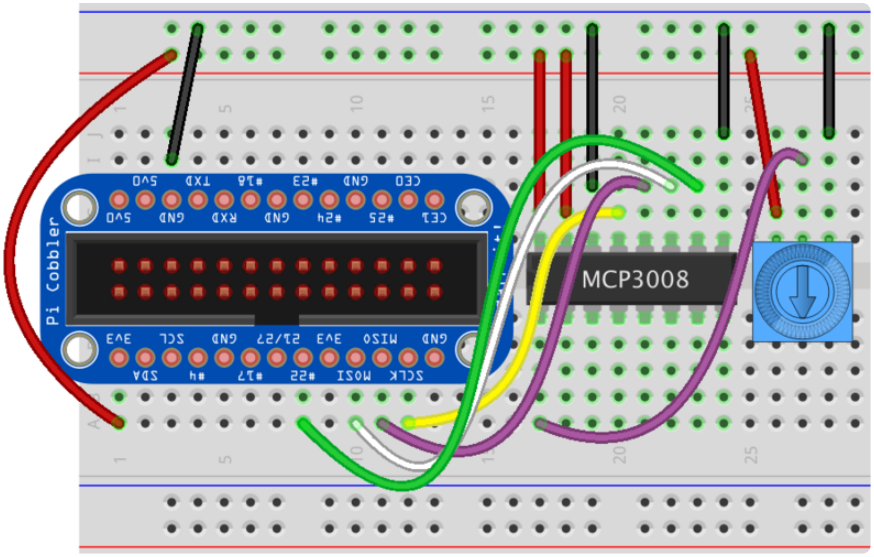
- Pin #1 (left) goes to 3.3v (red)
- Pin #2 (middle) connects to MCP3008 CH0 (analog input #0) with a purple wire
- Pin #3 (right) connects to GND (black)

Below we provide to wiring diagrams that will work with all versions of Raspberry Pi released so far (except the compute node which has no header). The first diagram is for the most recent Pi v3 and Pi Zero models which have a 40-pin GPIO header. The second is for the first two generations of Raspberry Pi which had a smaller 26-pin header. In both cases we are using the same GPIOs so the code will not be any different.

T-Cobbler Plus Wiring 40-Pin Pi (v3, Zero)



Pi Cobbler Wiring 26-Pin Pi (v1, v2)



Necessary Packages

Update Your Pi to the Latest Raspbian

Your Pi will need to be running the latest version of Raspbian. This tutorial was written using Raspbian Stretch (Nov. 2018). Checkout our guide for Preparing an SD Card for your Raspberry Pi () if you have not done so already. After the installation is complete be sure and run the following commands to make sure your installation packages are up to date.

```
$ sudo apt-get update -y
$ sudo apt-get upgrade -y
```

You will also need to enable SPI ()

Install pip3

pip3 is already installed with a full Raspbian installation, but the Raspbian Lite does not include pip3 so it needs to be installed as shown below. We will use pip3 to install the necessary CircuitPython libraries.

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

Install adafruit-blinka

```
$ sudo pip3 install adafruit-blinka
```

Install mcp3008

```
$ sudo pip3 install adafruit-circuitpython-mcp3xxx
```

Python Script

The following code can be downloaded directly to your Raspberry Pi. It will read the trimpot value, translate the reading to a volume range and modify the OS output volume level on your Raspberry Pi.

The remap_range() method is being used to convert the 16-bit analog in range 0 - 65,535 to volume 0-100%.

The Code

```
# SPDX-FileCopyrightText: 2019 Mikey Sklar for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import os
import time
import busio
```

```
import digitalio
import board
import adafruit_mcp3xxx.mcp3008 as MCP
from adafruit_mcp3xxx.analog_in import AnalogIn

# create the spi bus
spi = busio.SPI(clock=board.SCK, MISO=board.MISO, MOSI=board.MOSI)

# create the cs (chip select)
cs = digitalio.DigitalInOut(board.D22)

# create the mcp object
mcp = MCP.MCP3008(spi, cs)

# create an analog input channel on pin 0
chan0 = AnalogIn(mcp, MCP.P0)

print('Raw ADC Value: ', chan0.value)
print('ADC Voltage: ' + str(chan0.voltage) + 'V')

last_read = 0       # this keeps track of the last potentiometer value
tolerance = 250     # to keep from being jittery we'll only change
                    # volume when the pot has moved a significant amount
                    # on a 16-bit ADC

def remap_range(value, left_min, left_max, right_min, right_max):
    # this remaps a value from original (left) range to new (right) range
    # Figure out how 'wide' each range is
    left_span = left_max - left_min
    right_span = right_max - right_min

    # Convert the left range into a 0-1 range (int)
    valueScaled = int(value - left_min) / int(left_span)

    # Convert the 0-1 range into a value in the right range.
    return int(right_min + (valueScaled * right_span))

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

    # read the analog pin
    trim_pot = chan0.value

    # how much has it changed since the last read?
    pot_adjust = abs(trim_pot - last_read)

    if pot_adjust > tolerance:
        trim_pot_changed = True

    if trim_pot_changed:
        # convert 16bit adc0 (0-65535) trim pot read into 0-100 volume level
        set_volume = remap_range(trim_pot, 0, 65535, 0, 100)

        # set OS volume playback volume
        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)

        # 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)
```

Download the Code

Let's put this file right in your home directory for simplicity. The wget command makes things easy.

```
$ wget https://raw.githubusercontent.com/adafruit/Adafruit_Learning_System_Guides/
master/Analog_Inputs_for_Raspberry_Pi_Using_the_MCP3008/code.py
```

Run It

Running the Code

The following command will start our volume control script. Adjusting the trimpot will printout the OS volume level on the screen.

```
$ sudo python3 ./code.py
```

Testing the Volume Control

Opening a second terminal on your Raspberry Pi we can run the "speaker-test" command and listen to the volume change as we turn the trimpot.

```
$ speaker-test -t sine -f 440
```

Controlling MP3 Volume

The mpg123 command can be used to play MP3 files on your Raspberry Pi. We can open an additional terminal and play MP3 files and adjust the volume with the trimpot we have wired up.

```
$ mpg123 &amp;lt;MP3 filename&amp;gt;
```