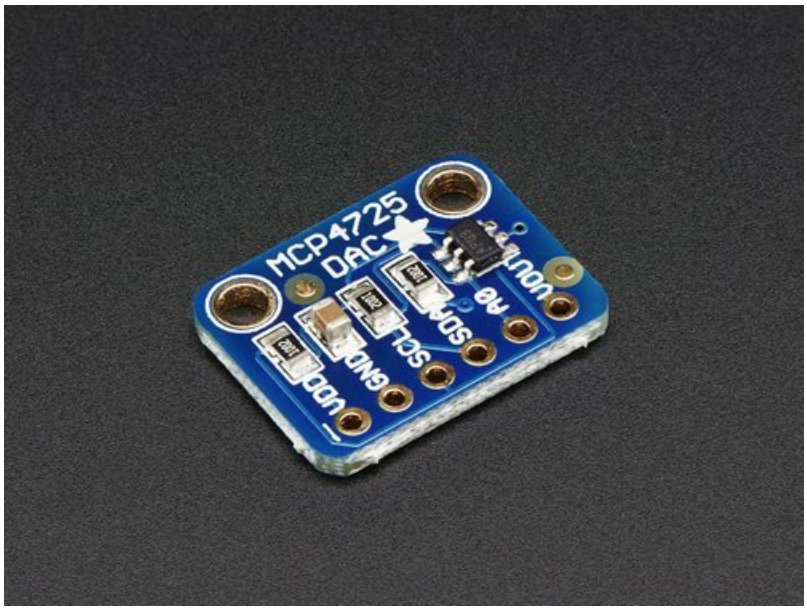
adafruit learning system

MCP4725 12-Bit DAC Tutorial

Created by lady ada

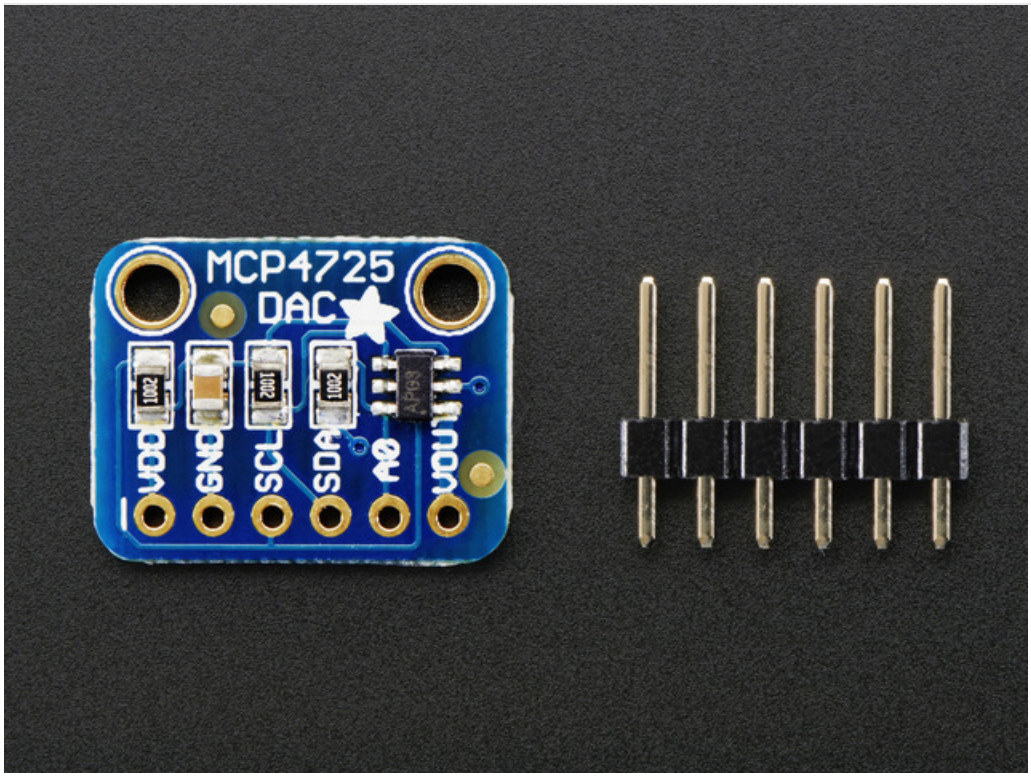


Last updated on 2018-03-05 10:51:16 PM UTC

Guide Contents

Guide Contents 2
Overview 3
Wiring 4
Arduino Code 7
  Using the library 7
Increasing the speed 8
CircuitPython Code 9
Usage 10
Download 13
Files 13
Schematic & Fabrication Print 13

Overview



Your microcontroller probably has an ADC (analog -> digital converter) but does it have a DAC (digital -> analog converter)??? Now it can! This breakout board features the easy-to-use MCP4725 12-bit DAC. Control it via I2C and send it the value you want it to output, and the VOUT pin will have it. Great for audio / analog projects, such as when you can't use PWM but need a sine wave or adjustable bias point.

We break out the ADDR pin so you can connect two of these DACs on one I2C bus, just tie the ADDR pin of one high to keep it from conflicting. Also included is a 6-pin header, for use in a breadboard. Works with both 3.3V or 5V logic.

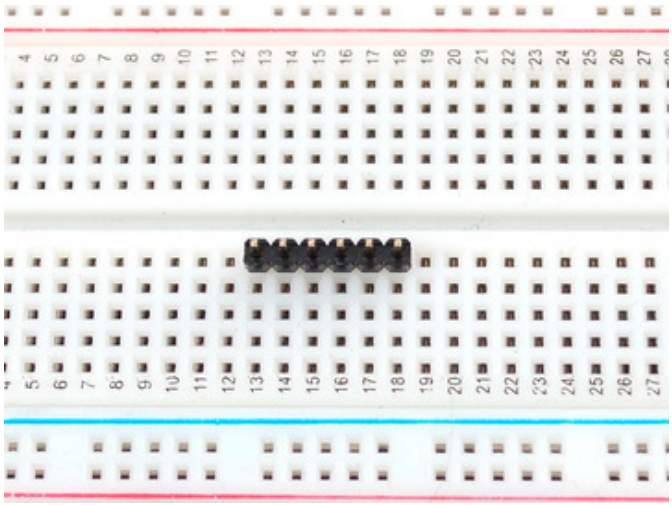
Some nice extras with this chip: for chips that have 3.4Mbps Fast Mode I2C (Arduino's don't) you can update the Vout at ~200 KHz. There's an EEPROM so if you write the output voltage, you can 'store it' so if the device is power cycled it will restore that voltage. The output voltage is rail-to-rail and proportional to the power pin so if you run it from 3.3V, the output range is 0-3.3V. If you run it from 5V the output range is 0-5V.

Available from the Adafruit shop!

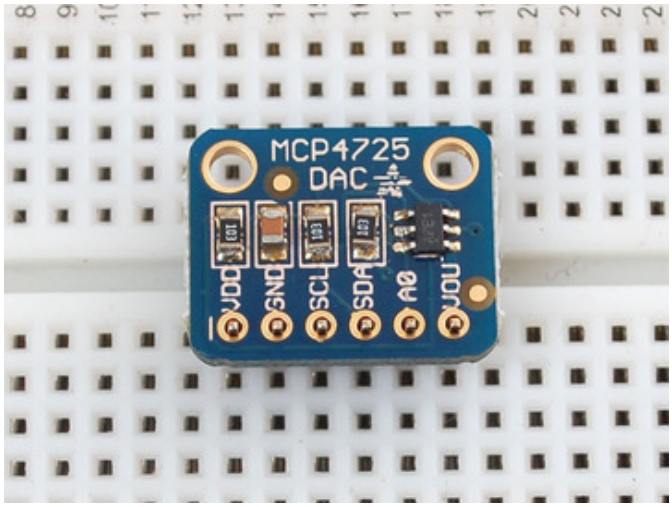
Wiring

Wiring up the MCP4725 breakout PCB is super easy. To start, we'll attach the breakout headers so we can plug it into a breadboard.

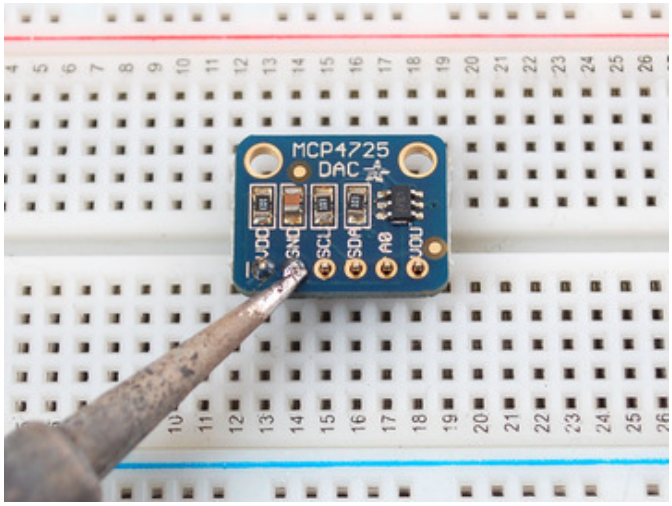
Break off a strip of 6-pins of 0.1" male header and stick the LONG pins down into a breadboard

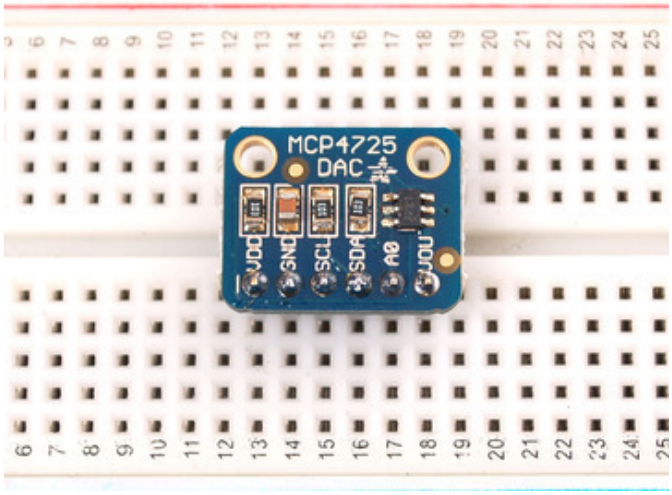


Break off a strip of 6-pins of 0.1" male header and stick the LONG pins down into a breadboard



Place the breakout board on top so the short ends of the header stick up through the pads





Solder each pin using a soldering iron and solder, to make solid connection on each pin.

This part is not optional! You cannot 'press fit' the header on, it must be attached permanently

Now that the header is attached, we can wire it up. We'll demonstrate using an Arduino.

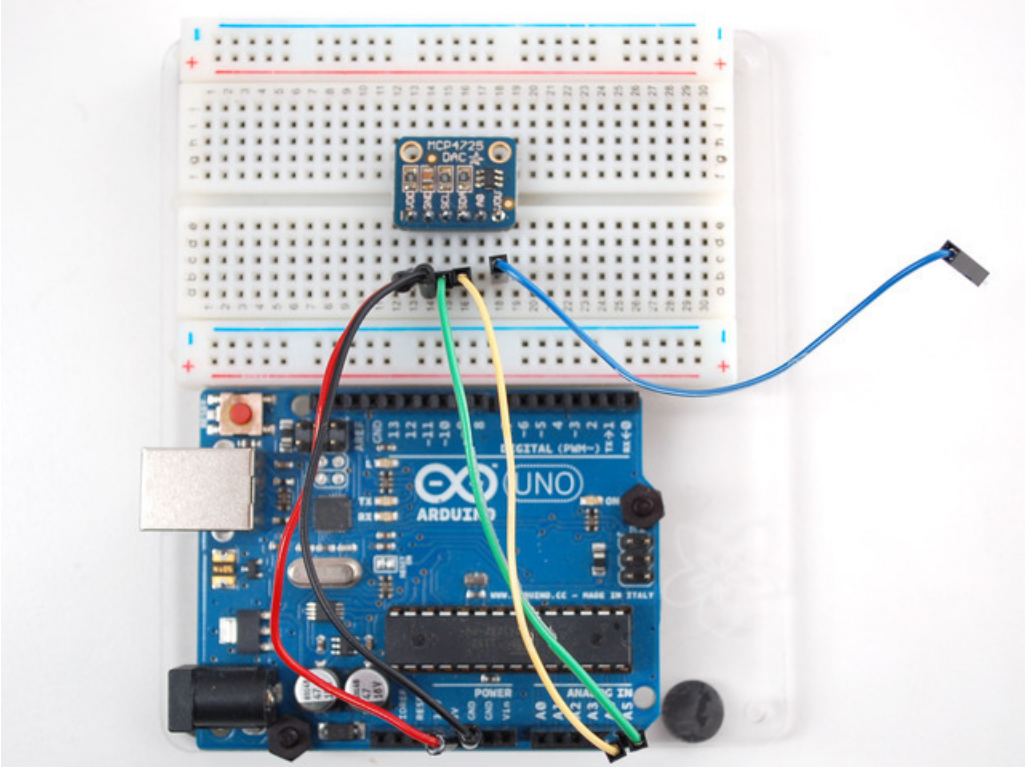
FIrst, connect **VDD** (power) to a 3-5V power supply, and **GND** to ground.

The DAC uses I2C, a two-pin interface that can have up to 127 unique sensors attached (each must have a different **ADDRESS**).

- **SDA** to I2C Data (on the Uno, this is **A4** on the Mega it is **20** and on the Leonardo digital **2**)
- **SCL** to I2C Clock(on the Uno, this is **A5** on the Mega it is **21** and on the Leonardo digital **3**)

There's two other pins remaining.

- **A0** allow you to change the I2C address. By default (nothing attached to A0) the address is hex **0x62**. If A0 is connected to **VDD** the address is **0x63**. This lets you have two DAC boards connected to the same SDA/SCL I2C bus pins.
- **VOUT** is the voltage out from the DAC! The voltage will range from 0V (when the DAC value is 0) to VDD (when the DAC 'value' is the max 12-bit number: 0xFFF)



Arduino Code

Next up, download the Adafruit MCP4725 library. This library does all of the interfacing, so you can just "set and forget" the DAC output. It also has some examples to get you started

The library is available on GitHub. You can download it by clicking the button below.

Download Adafruit_MCP4725 Library

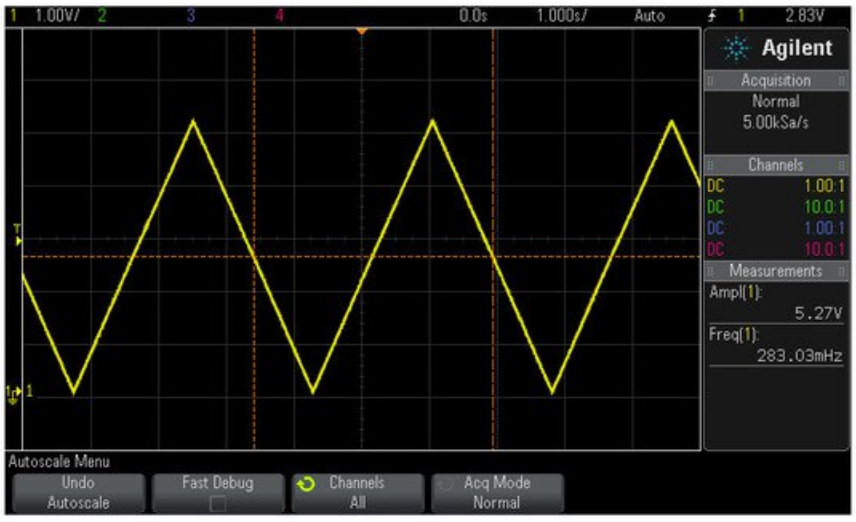
https://adafru.it/cDA

Rename the uncompressed folder **Adafruit_MCP4725.** Check that the **Adafruit_MCP4725** folder contains **Adafruit_MCP4725.cpp** and **Adafruit_MCP4725.h**

Place the **Adafruit_MCP4725** library folder your ***sketchbookfolder*/libraries/** folder. You may need to create the **libraries** subfolder if its your first library. You can figure out your ***sketchbookfolder*** by opening up the Preferences tab in the Arduino IDE.

Restart the IDE.

Open up the **File→Examples→Adafruit_MCP4725→trianglewave** sketch and upload it to the Arduino. Then connect your oscilloscope (or an LED + resistor if you don't have access to an oscilloscope)



We also have a sine wave version showing how to use a lookup table to create a more complex waveform.

Using the library

The library is very simple, so you can adapt it very quickly.

First, be sure to call **begin(addr)** where **addr** is the i2c address (default is 0x62, if A0 is connected to VCC its 0x63). Then call **setVoltage(value, storeflag)** to set the DAC output. **value** should range from 0 to 0x0FFF. **storeflag** indicates to the DAC whether it should store the value in EEPROM so that next time it starts, it'll have that same value output. You shouldn't set the flag to true unless you require it as it will take longer to do, and you could wear out the EEPROM if you write it over 20,000 times.

Increasing the speed

One thing thats a little annoying about the Arduino Wire library in this case is it is set for 100KHz transfer speed. In the MCP4725 library we update the speed to 400KHz by setting the TWBR

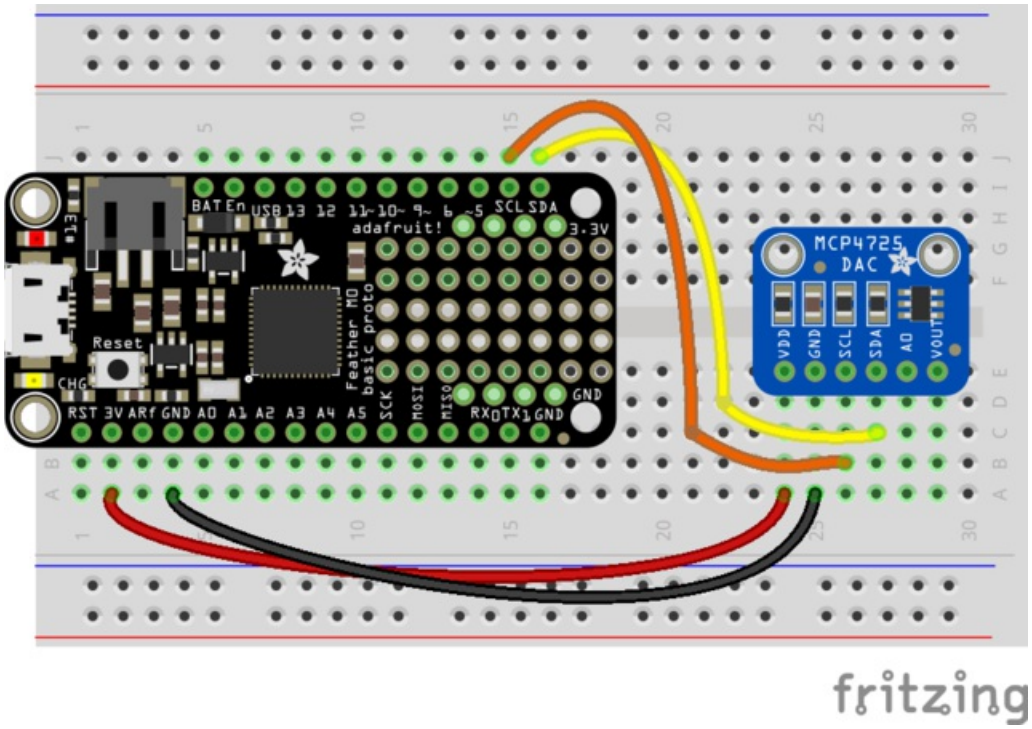
```
TWBR = 12; // 400 khz
```

You can speed this up a bit more, if you'd like, check the ATmega328 datasheet for how to calculate the **TWBR** register.

CircuitPython Code

It's easy to use the MCP4725 digital to analog converter with CircuitPython and the Adafruit CircuitPython MCP4725 module. This module allows you to easily write Python code that controls the output voltage from the DAC.

First wire up a MCP4725 to your board exactly as shown on the previous pages for Arduino using an I2C connection. Here's an example of wiring a Feather M0 to the sensor with I2C:



- **Board 3V** to **sensor VIN**
- **Board GND** to **sensor GND**
- **Board SCL** to **sensor SCL**
- **Board SDA** to **sensor SDA**

Next you'll need to install the Adafruit CircuitPython MCP4725 library on your CircuitPython board. Make sure you are running the latest version of Adafruit CircuitPython for your board before starting..

You'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle. For example the Circuit Playground Express guide has a great page on how to install the library bundle for both express and non-express boards.

Remember for non-express boards like the Trinket M0, Gemma M0, and Feather/Metro M0 basic you'll need to manually install the necessary libraries from the bundle:

- **adafruit_mcp4725.mpy**

You can also download the **adafruit_mcp4725.mpy** from its releases page on Github.

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_mcp4725.mpy** file copied over.

Next connect to the board's serial REPL so you are at the CircuitPython >>> prompt.

Usage

To demonstrate the usage of the DAC we'll initialize it and set the output voltage from the board's Python REPL. Run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import board
import busio
import adafruit_mcp4725
i2c = busio.I2C(board.SCL, board.SDA)
dac = adafruit_mcp4725.MCP4725(i2c)
```

Remember if you're using a board that doesn't support hardware I2C (like the ESP8266) you need to use the **bitbangio** module instead:

```
import board
import bitbangio
import adafruit_mcp4725
i2c = bitbangio.I2C(board.SCL, board.SDA)
dac = adafruit_mcp4725.MCP4725(i2c)
```

Now you can set the output voltage just like controlling a DAC with CircuitPython's built-in AnalogOut class and the **value** property. Simply set this to any 16-bit value (0-65535) and the output of the Vout pin will change to a voltage proportional to 0-3.3V. For example to set the output to 1.65V or about halfway within its range:

```
dac.value = 32767
```

Hook up a multimeter to the **Vout** pin of the board (positive/red lead to **Vout**, ground/black lead to board GND) and you should see about 1.65 volts DC output. Try setting **dac.value** to other numbers like 0 or 65535 to see how the voltage changes.

You can use the MCP4725 instance anywhere you might use the AnalogOut class!

However you might prefer a few other simpler properties to change the output voltage:

- **normalized_value** - Set this to a floating point number between 0 and 1.0. A value of 0 is ground/0V and 1.0 is Vdd or max voltage/3.3V. Anything in-between is a proportional voltage. This is handy for scaling the output value without having to worry about how many bits of resolution it has.
- **raw_value** - Set this to a 12-bit value 0-4095 to control the raw 12-bit output of the DAC. Unlike the **value** property this **raw_value** exposes the true 12-bit resolution of the DAC and is free from quantization errors. If you need the most precise output use the raw_output value for setting voltage.

```
dac.normalized_value = 0.5  # ~1.65V output
dac.raw_output = 2047 # Also ~1.65V output
```

That's all there is to using the MCP4725 DAC with CircuitPython!

Below is a complete example that shows changing the DAC voltage to a triangle wave that goes up and down repeatedly. Save this as **main.py** on your board and connect a multimeter to measure the Vout pin voltage to see it oscillate up and down from 0 to 3.3V and back. Remember to change to use **bitbangio** if necessary for your board!

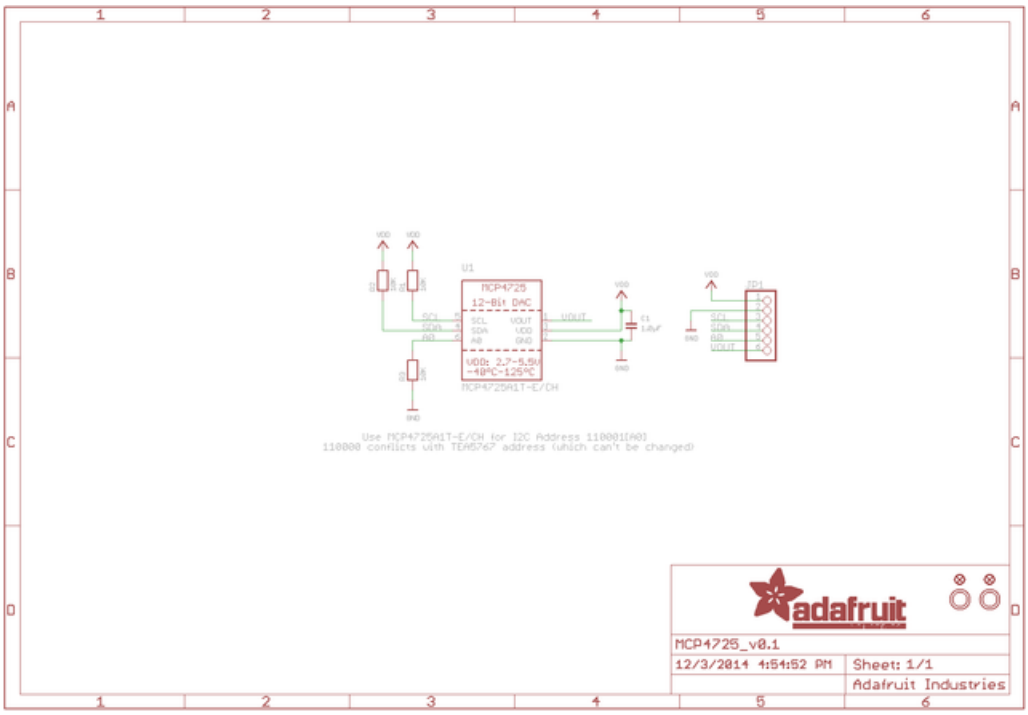
```
# Simple demo of setting the DAC value up and down through its entire range
# of values.
# Author: Tony DiCola
import board
import busio

import adafruit_mcp4725


# Initialize I2C bus.
i2c = busio.I2C(board.SCL, board.SDA)

# Initialize MCP4725.
dac = adafruit_mcp4725.MCP4725(i2c)
# Optionally you can specify a different addres if you override the A0 pin.
#amp = adafruit_max9744.MAX9744(i2c, address=0x63)

# There are a three ways to set the DAC output, you can use any of these:
dac.value = 65535  # Use the value property with a 16-bit number just like
                   # the AnalogOut class.  Note the MCP4725 is only a 12-bit
                   # DAC so quantization errors will occur.  The range of
                   # values is 0 (minimum/ground) to 65535 (maximum/Vout).

dac.raw_value = 4095  # Use the raw_value property to directly read and write
                      # the 12-bit DAC value.  The range of values is
                      # 0 (minimum/ground) to 4095 (maximum/Vout).

dac.normalized_value = 1.0  # Use the normalized_value property to set the
                            # output with a floating point value in the range
                            # 0 to 1.0 where 0 is minimum/ground and 1.0 is
                            # maximum/Vout.

# Main loop will go up and down through the range of DAC values forever.
while True:
    # Go up the 12-bit raw range.
    print('Going up 0-3.3V...')
    for i in range(4095):
        dac.raw_value = i
    # Go back down the 12-bit raw range.
    print('Going down 3.3-0V...')
    for i in range(4095, -1, -1):
        dac.raw_value = i
```

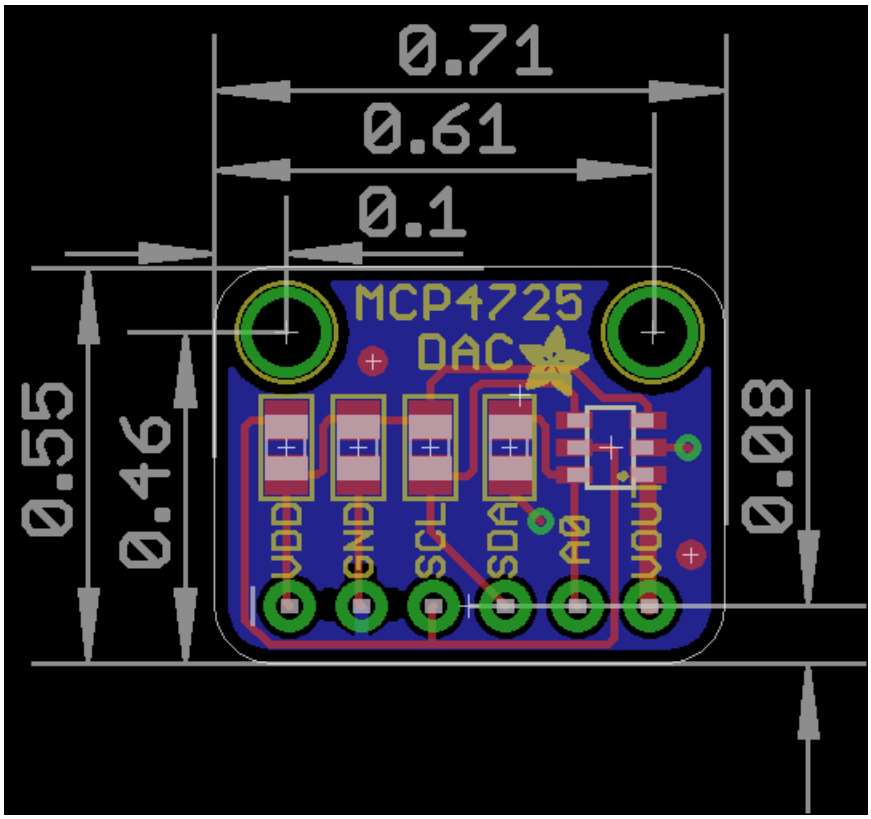
Download

Files

- For more details about the chip, please check out the MCP4725 datasheet
- MCP4725 Arduino Library is on GitHub
- Fritzing object in the Adafruit Fritzing library
- EagleCAD PCB files on GitHub

Schematic & Fabrication Print




0.71
0.61
0.1
0.55
0.46
0.08
MCP4725
DAC
VDD
GND
SCL
SDA
A0
VOUT