

<https://www.halvorsen.blog>

Arduino MCP4725 DAC Breakout Board

Hans-Petter Halvorsen



Introduction

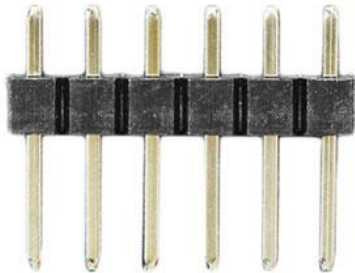
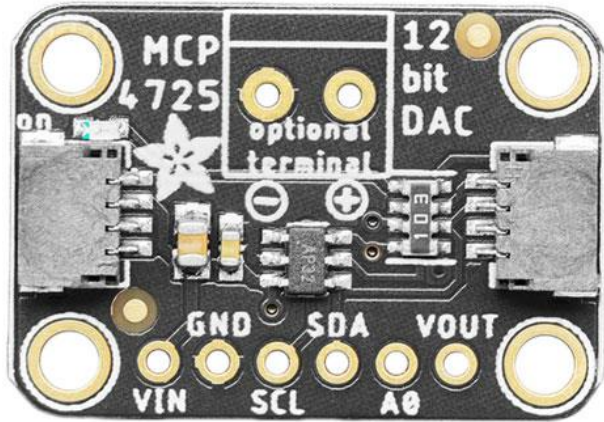
- Microcontrollers like e.g., **Arduino UNO** do not have Analog Outputs.
- The we can use an external Digital to Analog Converter (DAC) chip/IC.
- **MCP4725** is a Digital to Analog Converter (DAC).
- It comes in breadboard friendly version.
- We will use an Arduino UNO, a MCP4725 Breakout Board from Adafruit and a multimeter.

MCP4725

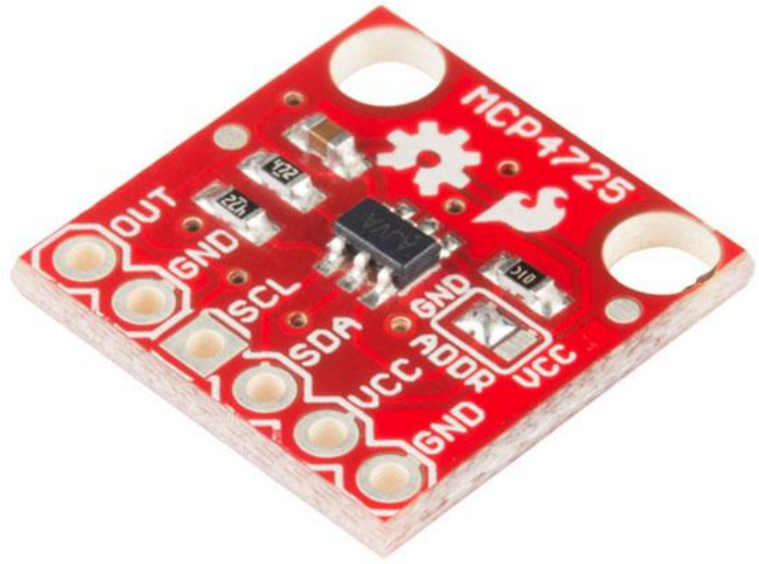
- MCP4725 is a Digital to Analog Converter (DAC).
- MCP4725 is a 12-bit DAC ($2^{12} = 4096$).
meaning you set values between minimum 0 (0V) a maximum 4095 (5V).
- MCP4725 has an I2C Interface.

MCP4725

Adafruit MCP4725 Breakout Board



Sparkfun MCP4725 Breakout Board



MCP4725 Resources

- MCP4725 at Adafruit:
<https://www.adafruit.com/product/935>
- Tutorial: <https://learn.adafruit.com/mcp4725-12-bit-dac-tutorial/>
- Hookup Guide:
<https://learn.sparkfun.com/tutorials/mcp4725-digital-to-analog-converter-hookup-guide>
- I2C Arduino:
<https://www.electronicwings.com/arduino/arduino-i2c>
- Wire Library:
<https://docs.arduino.cc/learn/communication/wire/>

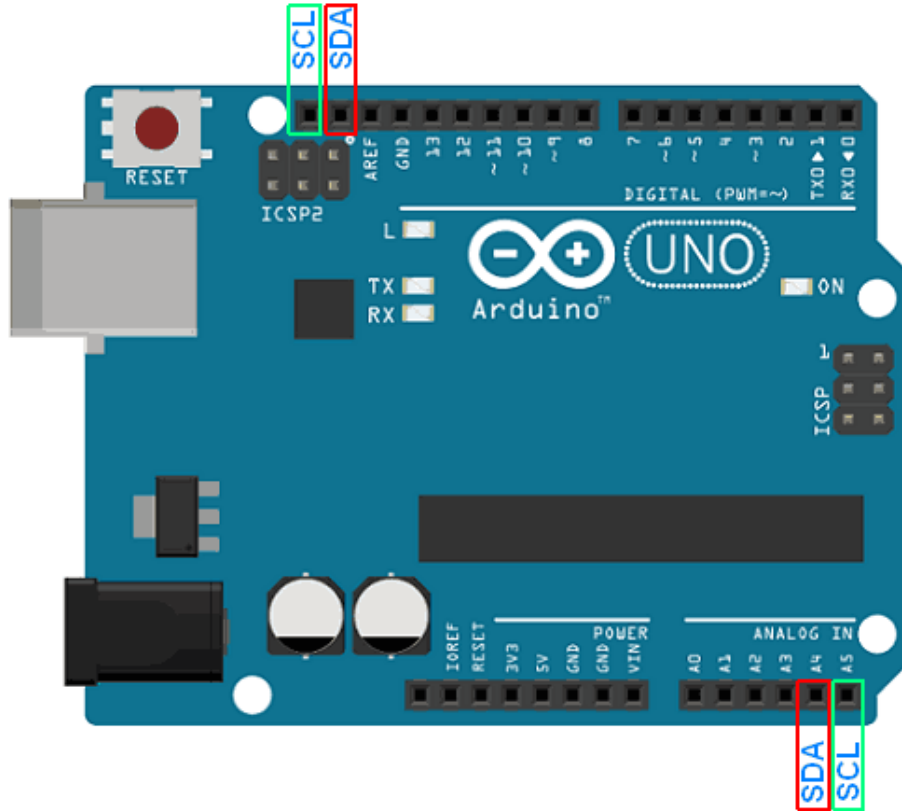
I2C Protocol

- I2C (Inter-Integrated Circuit) is a serial bus interface connection protocol.
- It is also called a TWI (two wire interface) since it uses only two wires for communication.
- Those two wires are SDA (serial data) and SCL (serial clock).
- In addition, you need to connect to Power and Ground.
- I2C is a Master–Slave Protocol.
- All microcontrollers like Arduino and Single Board Computers (SBC) like Raspberry Pi supports the I2C protocol.

I2C Protocol on Arduino

- You typically use I2C on Arduino to communicate with external components like small sensors, or chips/ICs like DAC and DAC chips/ICs.
- In our tutorial we will use the MCP4725 DAC.
- The Arduino is typically the Master and external components like MCP4725 is Slave.
- In Arduino you can use the built-in **Wire** Library to deal with I2C communication, this means you don't need to know too much details regarding the I2C protocol
- Many of the component vendors also create their own tailormade libraries, like the "Adafruit MCP4725" Arduino Library that can be used to communicate with the Arduino MCP4725 DAC Breakout Board.
- In this tutorial we will use both the "Adafruit MCP4725" Arduino Library and the "Wire" library.

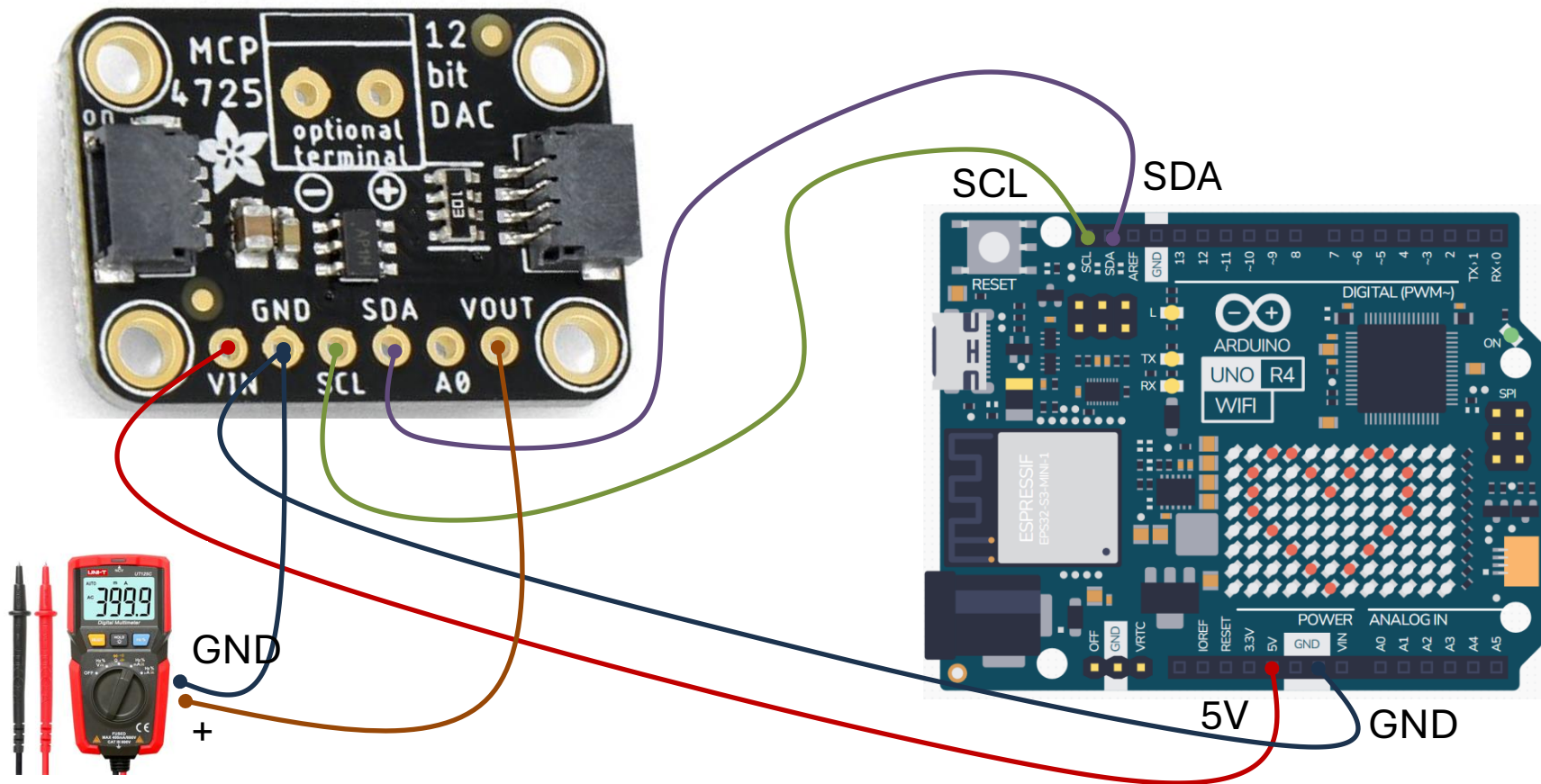
I2C Arduino Pins



- The Arduino Uno board has only one I2C module, but it provides these SDA and SCL line at two different locations, as seen in the image to the left.
- **SDA** (serial data) wire is used for data exchange in between the master and slave device.
- **SCL** (serial clock) is used for the synchronous clock in between master and slave device.

You normally don't need to use the A0

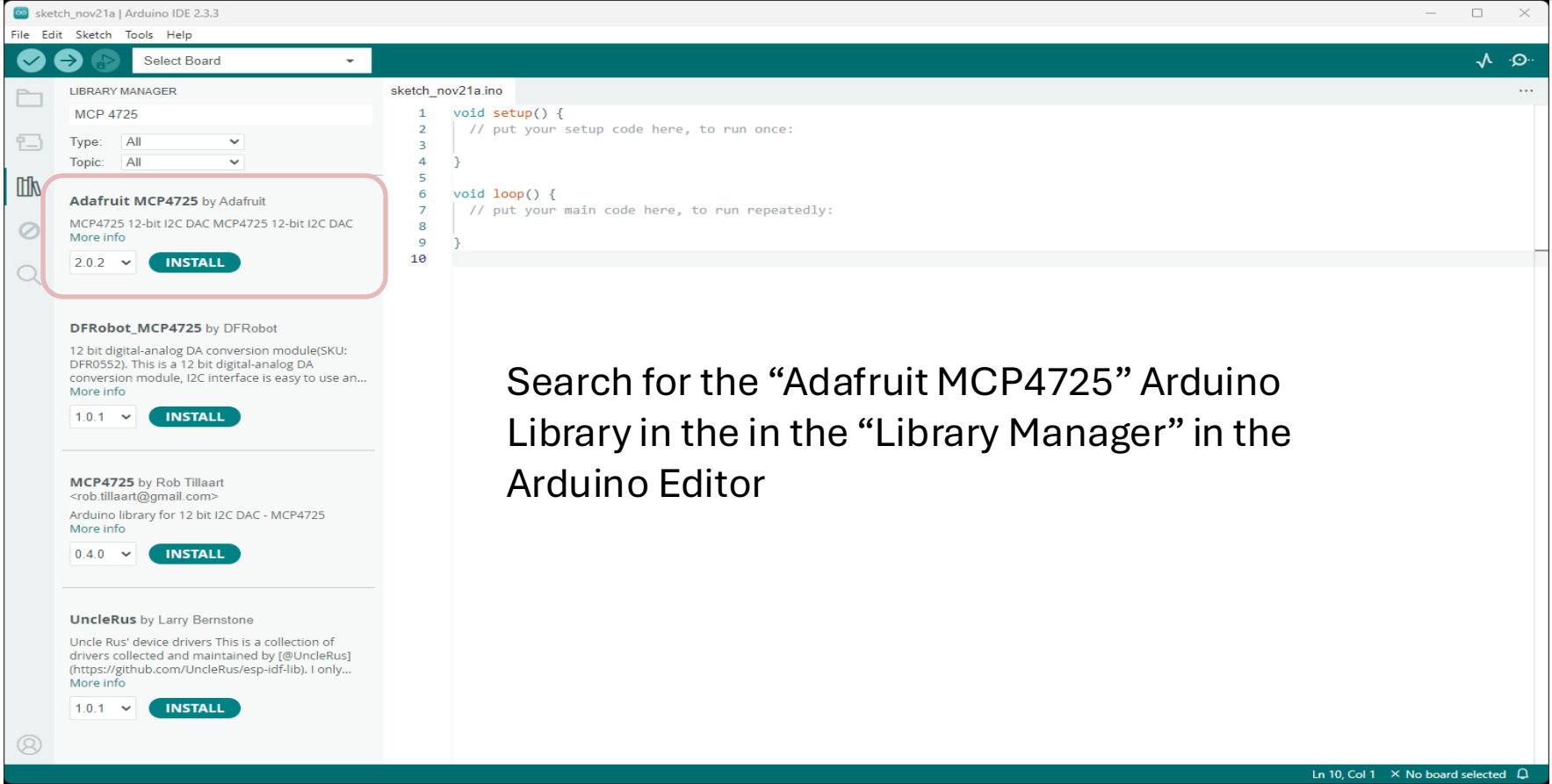
Wiring Example



A0 Pin

- 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.

Adafruit MCP4725 Arduino Library



The screenshot shows the Arduino IDE interface with the Library Manager open. The search results for 'MCP 4725' are displayed, with the 'Adafruit MCP4725' library highlighted by a red box. The library details for 'Adafruit MCP4725' are as follows:

- Library Name:** Adafruit MCP4725 by Adafruit
- Description:** MCP4725 12-bit I2C DAC MCP4725 12-bit I2C DAC
- Version:** 2.0.2
- Action:** INSTALL

The other search results are:

- DFRobot_MCP4725** by DFRobot: 12 bit digital-analog DA conversion module(SKU: DFR0552). This is a 12 bit digital-analog DA conversion module, I2C interface is easy to use an... (Version: 1.0.1)
- MCP4725** by Rob Tillaart: Arduino library for 12 bit I2C DAC - MCP4725 (Version: 0.4.0)
- UncleRus** by Larry Bernstone: Uncle Rus' device drivers This is a collection of drivers collected and maintained by [UncleRus] (https://github.com/UncleRus/esp-idf-lib). I only... (Version: 1.0.1)

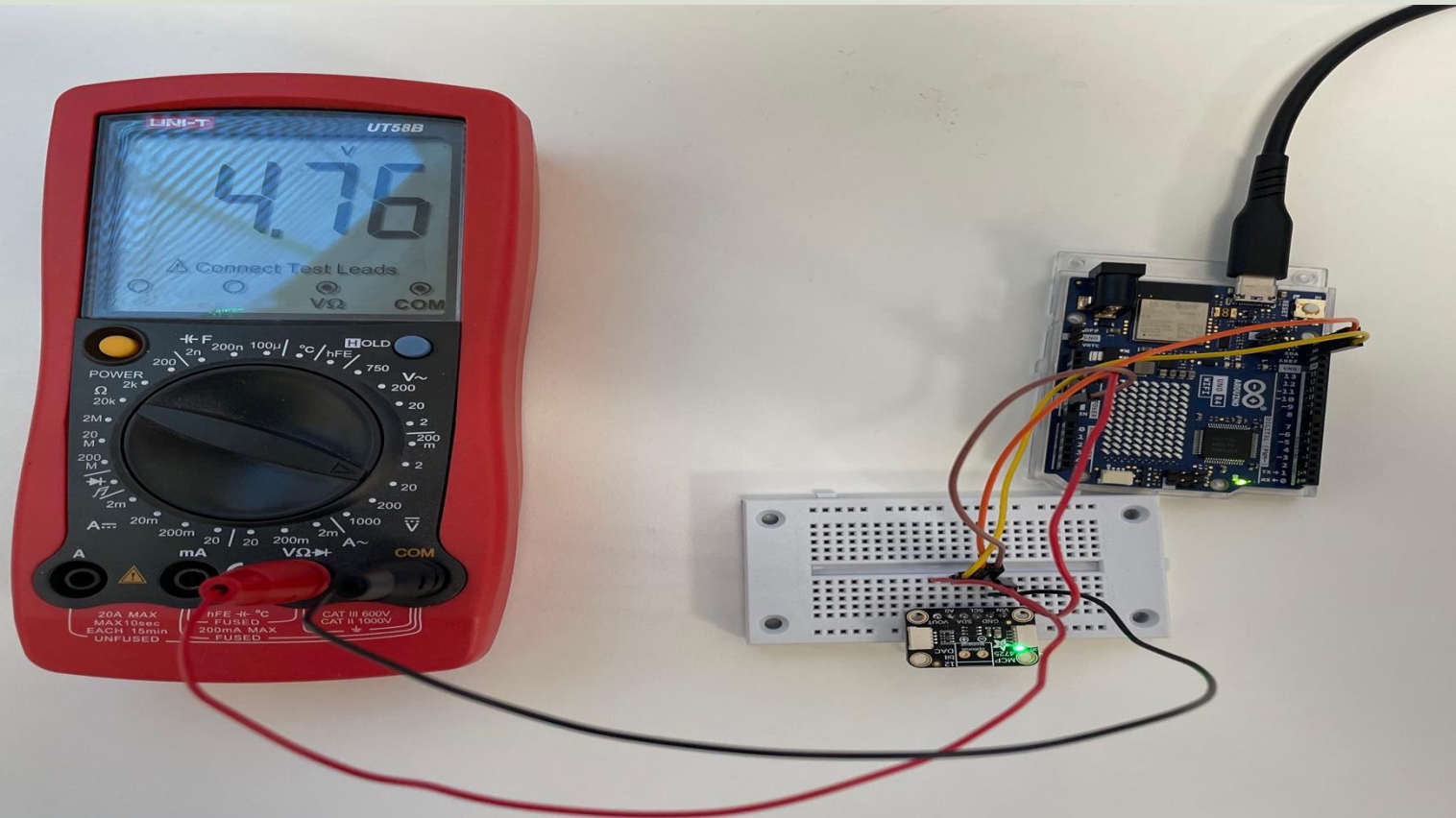
The code editor on the right shows the following code for sketch_nov21a.ino:

```
1 void setup() {  
2   // put your setup code here, to run once:  
3 }  
4  
5  
6 void loop() {  
7   // put your main code here, to run repeatedly:  
8 }  
9  
10
```

At the bottom right of the IDE, the status bar indicates 'Ln 10, Col 1' and 'No board selected'.

Search for the “Adafruit MCP4725” Arduino Library in the in the “Library Manager” in the Arduino Editor

Hardware Setup



Code Structure

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>

Adafruit_MCP4725 dac;

int dacValue = 0;

void setup()
{
  dac.begin(0x62);
  dacValue = 2048; //2.5V
  dac.setVoltage(dacValue, false);
}

void loop()
{
}
```

- Initialize the DAC with **Adafruit_MCP4725 dac;**
- Then call **begin(addr)** where **addr** is the i2c address. Default address is 0x62.
- Set the DAC output value by calling **setVoltage(value, storeflag)**
- **value** should range from 0 to 4095.
- **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.

Code Ex.

When you have a multimeter connected, you should first see 0v for 2 seconds, then 2.5V for 2 seconds and then 5v for 2 seconds, then it goes back to 0v, etc.



0V ->2.5V ->5V

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>
```

```
Adafruit_MCP4725 dac;
int dacValue = 0; //Min 0(0V) and Max 4095(5V)
```

```
void setup()
{
  Serial.begin(9600);
  dac.begin(0x62); //MCP4725 Address is 0x62
}
```

```
void loop()
{
  Serial.println("0V");
  dacValue = 0; //0V
  dac.setVoltage(dacValue, false);
  delay(2000);
```

```
  Serial.println("2.5V");
  dacValue = 2048; //2.5V
  dac.setVoltage(dacValue, false);
  delay(2000);
```

```
  Serial.println("5V");
  dacValue = 4095; //5V
  dac.setVoltage(dacValue, false);
  delay(2000);
}
```

Updated Example

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>

Adafruit_MCP4725 dac;

void setup() {
  Serial.begin(9600);
  dac.begin(0x62);
}

void loop() {
  for (int voltageValue = 0; voltageValue <= 5; voltageValue++)
  {
    WriteDAC(voltageValue);
  }
}
```

0V ->1V ->2V ->3V ->4V ->5V



Updated Example cont.

```
void WriteDAC(int voltageValue)
{
    int dacValue;

    Serial.print(voltageValue);
    Serial.println("V");

    dacValue = Voltage2DAC(voltageValue);
    dac.setVoltage(dacValue, false);

    delay(2000);
}
```

We have made 2 new functions in order to make the code in the main() loop a bit neater.

```
int Voltage2DAC(double voltageValue) //0-5V=>0-4095
{
    int dacValue;

    dacValue = (4095/5)*voltageValue;
    return dacValue;
}
```


Arduino Wire Library

- The Wire library is what Arduino uses to communicate with I2C devices.
- It is included in all board packages, so you don't need to install it manually in order to use it.
- For more information and overview of functions:

<https://docs.arduino.cc/learn/communication/wire/>

Arduino Wire Library

Main code structure using the Wire Library:

```
#include <Wire.h>

const int MCP4725_ADDR = 0x62;

Wire.begin();

Wire.beginTransmission(MCP4725_ADDR);

Wire.write(dacValue >> 4); //8 most significant bits
Wire.write((dacValue & 15) << 4); //4 least significant bits

Wire.endTransmission();

Wire.end();
```

- MCP4725 uses 12-bit
- In the I2C protocol we can only send 8 bit of data at a time
- So, we need to split the 12-bit data into 2 operations

Alternative Code

Here we only use the Wire Library

```
#include <Wire.h>
int dacValue = 0;
const int MCP4725_ADDR = 0x62;
```

```
void setup() {
  Serial.begin(9600);
  Wire.begin();
}
```

```
void loop() {
  Serial.println("0V");
  dacValue = 0; //0V
  DACWrite(dacValue);
  delay(2000);
```

```
  Serial.println("5V");
  dacValue = 4095; //5V
  DACWrite(dacValue);
  delay(2000);
}
```

```
void DACWrite(int dacValue)
{
  Wire.beginTransaction(MCP4725_ADDR);
  Wire.write(64); //cmd to update the DAC
  Wire.write(dacValue >> 4); //8 most significant bits
  Wire.write((dacValue & 15) << 4); //4 least significant bits
  Wire.endTransmission();
}
```

The example is based on:

<https://learn.sparkfun.com/tutorials/mcp4725-digital-to-analog-converter-hookup-guide>

Summary

- I needed an Analog Output for my Arduino UNO project.
- This tutorial used the MCP4725 DAC Breakout Board made by Adafruit.
- In this tutorial we will use both the tailormade “Adafruit MCP4725” Arduino Library and the general “Wire” I2C library that is included with the Arduino software.
- We made some code examples and tested the code using a multimeter.

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: <https://www.halvorsen.blog>

