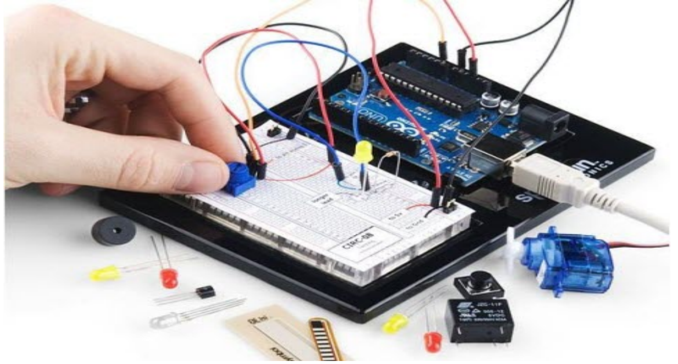
https://www.halvorsen.blog

Arduino PWM and Analog Output

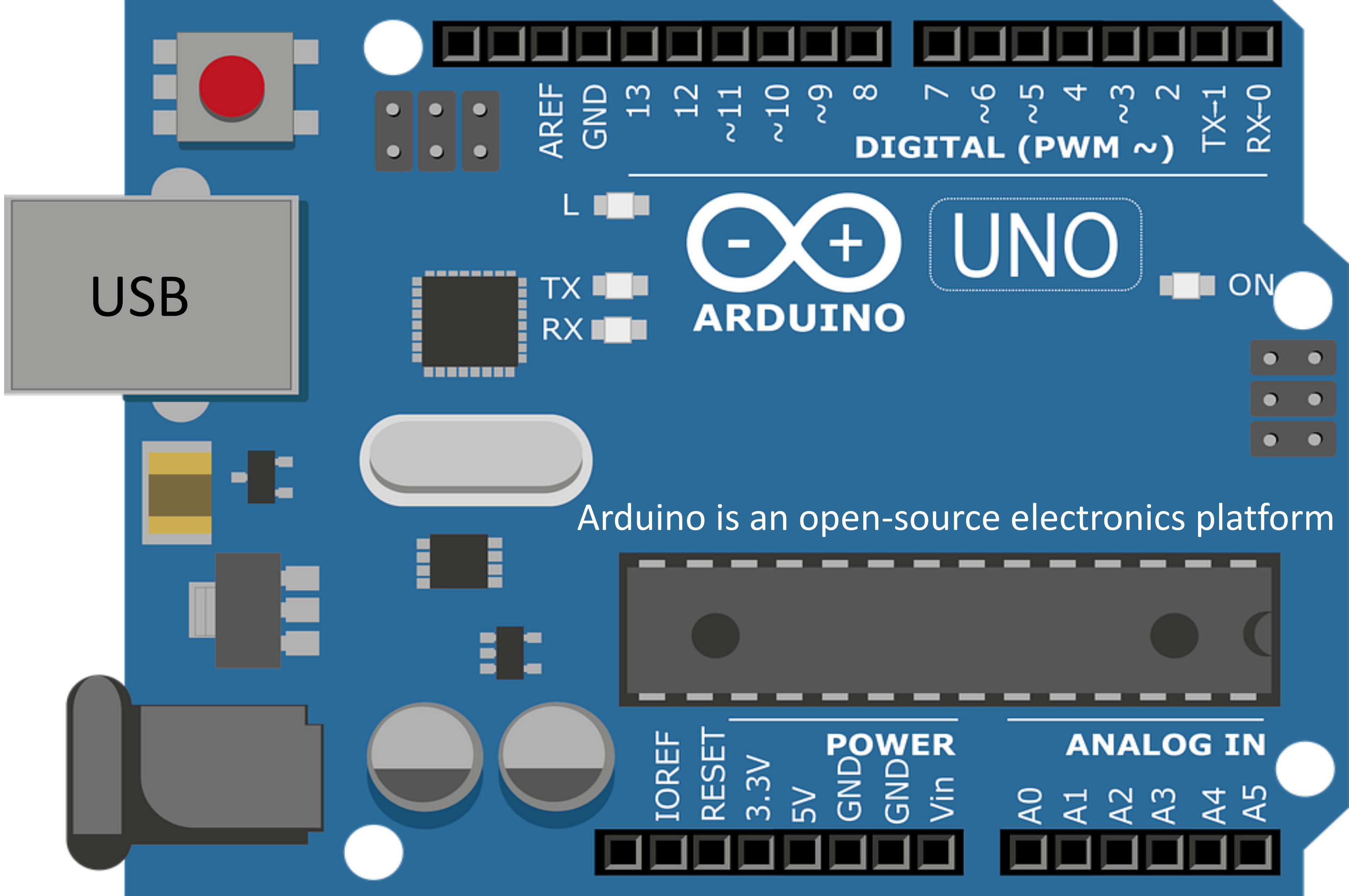
Hans-Petter Halvorsen



- Arduino is an open-source physical computing platform designed to make experimenting with electronics and programming more fun and intuitive.
- Arduino has its own unique, simplified programming language and a lots of premade examles and tutorials exists.
- With Arduino you can easily explore lots of small-scale sensors and actuators like motors, temperature sensors, etc.
- The possibilities with Arduino are endeless.

http://www.arduino.cc

Arduino Microcontroller



Arduino is an open-source electronics platform

Contents

- Introduction to Pulse Width Modulation (PWM)
- Arduino analogWrite() function
- Arduino UNO has no true Analog Outputs. What can we do?

https://www.halvorsen.blog

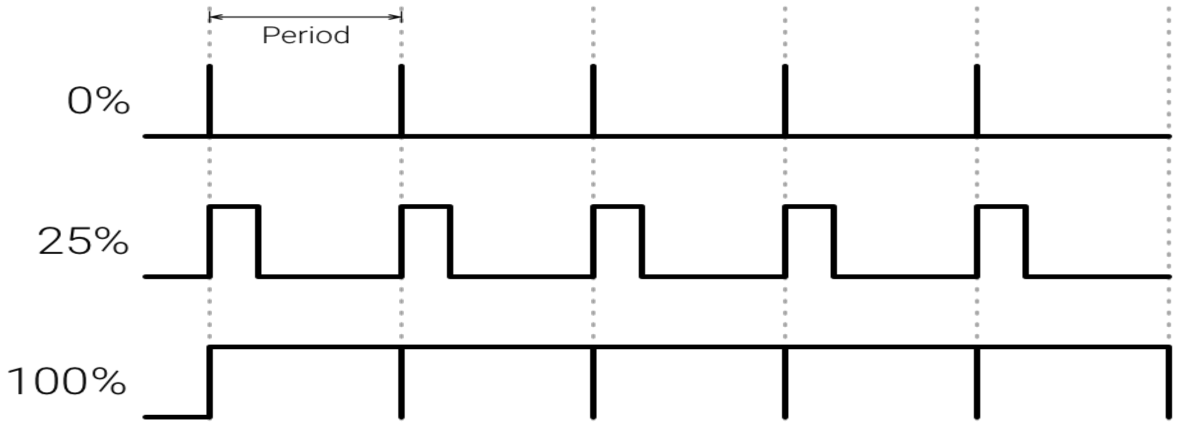
Pulse Width Modulation (PWM)

Hans-Petter Halvorsen

PWM

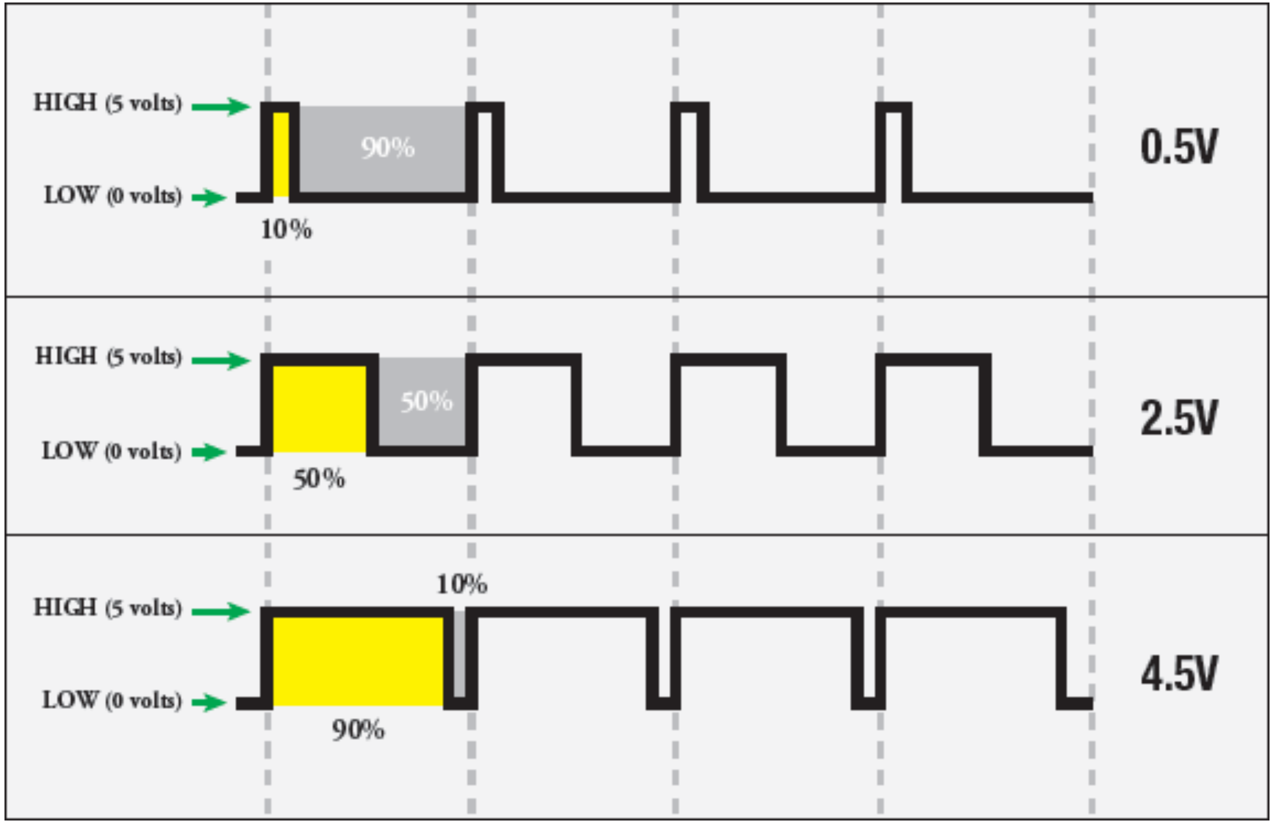
PWM is a digital (i.e. square wave) signal that oscillates according to a given *frequency* and *duty cycle*.
The frequency (expressed in Hz) describes how often the output pulse repeats.
The period is the time each cycle takes and is the inverse of frequency.
The duty cycle (expressed as a percentage) describes the width of the pulse within that frequency window.

You can adjust the duty cycle to increase or decrease the average "on" time of the signal. The following diagram shows pulse trains at 0%, 25%, and 100% duty:



https://developer.android.com/things/sdk/pio/pwm.html

Pulse-Width Modulation (PWM)



Pulse-Width Modulation (PWM)

The “shocking truth” behind analogWrite():

- We know that the Arduino can read analog voltages (voltages between 0 and 5 volts) using the analogRead() function.
- Is there a way for the Arduino to output analog voltages as well? The answer is no... and yes. Arduino does not have a true analog voltage output. But, because Arduino is so fast, it can fake it using something called PWM ("Pulse-Width Modulation"). The pins on the Arduino with “~” next to them are PWM/Analog out compatible.
- The Arduino is so fast that it can blink a pin on and of almost 1000 times per second. PWM goes one step further by varying the amount of time that the blinking pin spends HIGH vs. the time it spends LOW. If it spends most of its time HIGH, a LED connected to that pin will appear bright. If it spends most of its time LOW, the LED will look dim. Because the pin is blinking much faster than your eye can detect, the Arduino creates the illusion of a "true" analog output.
- To smooth the signal even more, we will create and use a RC circuit (Lowpass Filter)

Pulse-Width Modulation (PWM)

- The Arduino's programming language makes PWM easy to use; simply call analogWrite(pin, dutyCycle), where dutyCycle is a value from 0 to 255, and pin is one of the PWM pins (3, 5, 6, 9, 10, or 11).
- The analogWrite function provides a simple interface to the hardware PWM, but doesn't provide any control over frequency. (Note that despite the function name, the output is a digital signal, often referred to as a square wave.)

$$0-5V \rightarrow 0-255 \ \rightarrow y(x)=51x$$
$$u=0V \ \rightarrow analogWrite(0)$$
$$u=5V \ \rightarrow analogWrite(255)$$
$$u=xV \ \rightarrow analogWrite(51*x)$$

analogWrite():
https://www.arduino.cc/en/Reference/AnalogWrite

Secrets of Arduino PWM:
https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM

https://www.halvorsen.blog

Using analogWrite()

Hans-Petter Halvorsen

analogWrite

This functions write an "analog value" (PWM signal) to the specified pin. You can e.g., use it to make a LED light with different intensity.
Syntax:
analogWrite(pin, value)
where value is a value between 0 and 255

```
int ledPin = 9;
int value = 0;
void setup()
{
	pinMode(ledPin, OUTPUT);
}

void loop()
{
	value = random(256);
	analogWrite(ledPin, value);
	delay(1000);
}s
```

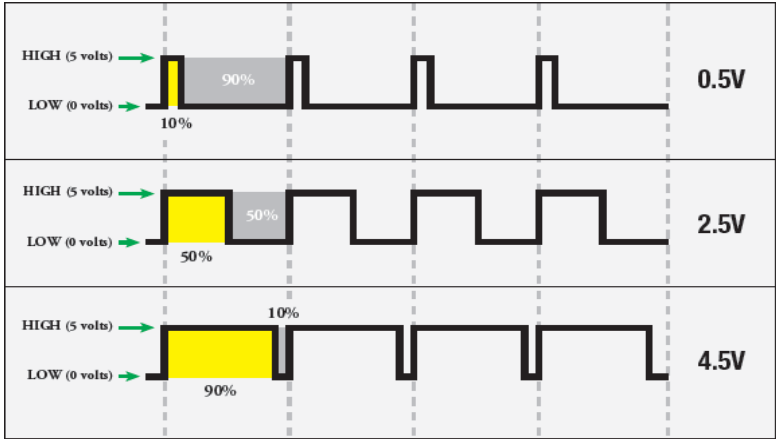
Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightness's or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady rectangular wave of the specified duty cycle until the next call to analogWrite()

Note! You need to use one of the pins marked with ~

https://www.arduino.cc/en/Reference/AnalogWrite

analogWrite()

Arduino can give a signal between 0 and $5V$



$$0.5V\colon \frac{0.5}{5}100\% \rightarrow 10\%$$

0.5V (10% of 255) -> analogWrite(pin, 25)

$$2.5V\colon \frac{2.5}{5}100\% \rightarrow 50\%$$

2.5V (50% of 255) -> analogWrite(pin, 127)

$$4.5V\colon \frac{4.5}{5}100\% \rightarrow 90\%$$

4.5V (90% of 255) -> analogWrite(pin, 229)

Arduino syntax: **analogWrite(pin, value)**
value: the duty cycle: between 0 (always off) and 255 (always on).
0-5V -> 0-255

https://www.halvorsen.blog

True Analog Out

Hans-Petter Halvorsen

Arduino Analog Out

- Arduino UNO has no true built-in Analog Output Channels (only PWM)
- What if we need a real Analog Out Signal (0-5V)?
- We will use a 2 different options:
  - Create a RC Lowpass Filter that converts PWM to Voltage
  - Use a DAC chip/IC (Digital to Analog Converter)
    - Such a chip uses either the SPI bus or the I2C bus

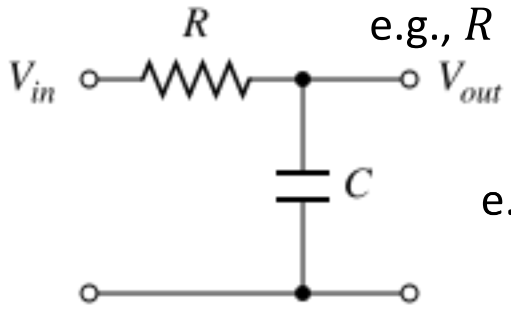
https://www.halvorsen.blog

Lowpass Filter using RC Circuit

Hans-Petter Halvorsen

Option 1: Convert PWM to Voltage

RC Lowpass Filter:



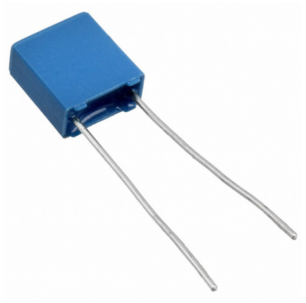
e.g., $R = 3.9k\Omega$

e.g., $C = 10\mu F$

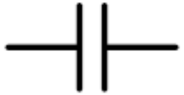
- http://www.instructables.com/id/Analog-Output-Convert-PWM-to-Voltage/
- http://provideyourown.com/2011/analogwrite-convert-pwm-to-voltage/

Electrical Components

Capacitor



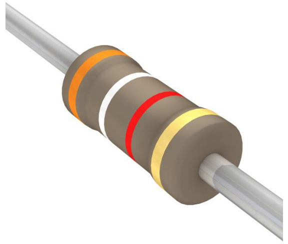
e.g., $C = 10\mu F$



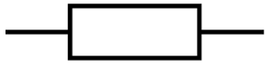
A capacitor stores and releases electrical energy in a circuit. When the circuits voltage is higher than what is stored in the capacitor, it allows current to flow in, giving the capacitor a charge. When the circuits voltage is lower, the stored charge is released. Often used to smooth fluctuations in voltage

https://en.wikipedia.org/wiki/Capacitor

Resistor



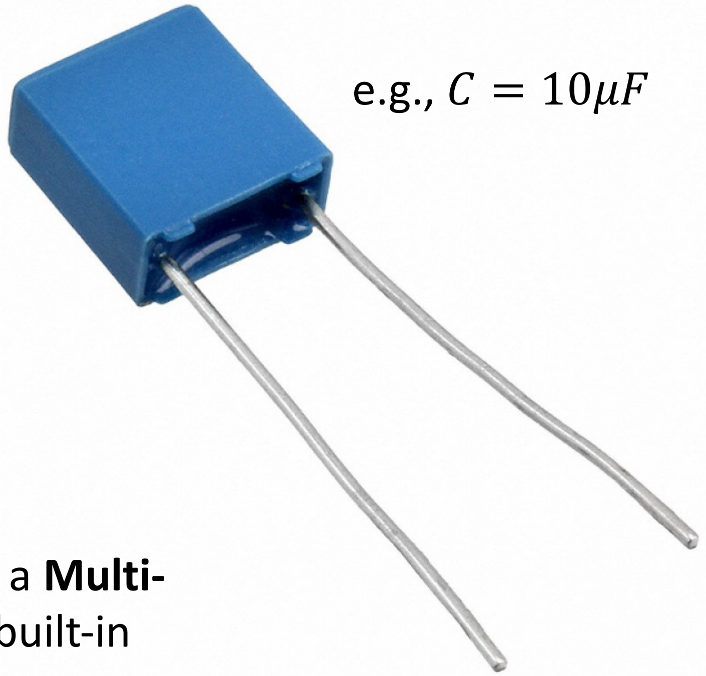
$R = 3.9k\Omega$



A resistor resists the flow of electrical energy in a circuit, changing the voltage and current as a result (according to Ohms law, $U = RI$). Resistor values are measured in ohms (Ω). The color stripes on the sides of the resistor indicate their values. You can also use a Multi-meter in order to find the value of a given resistor.

These electronics components are typically included in a “Starter Kit”, or they can be bought “everywhere” for a few bucks.

Capacitor

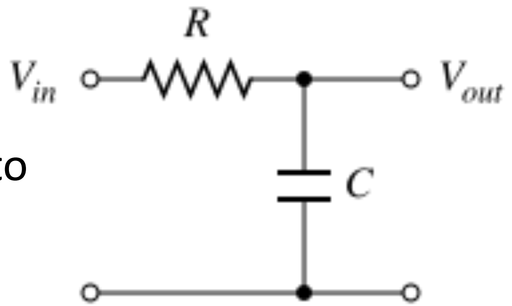


e.g., $C = 10\mu F$

The Capacitor is typically included in the Arduino Starter Kit (or similar Kits).

If you don't have such a Kit you may buy capacitors from Elfa, Kjell & Company, etc.

Note! You can also easily measure the capacitance using a **Multi-meter**. A Multi-meter that cost from 400-500+ NOK has built-in support for measuring capacitors (same for resistors and resistance).



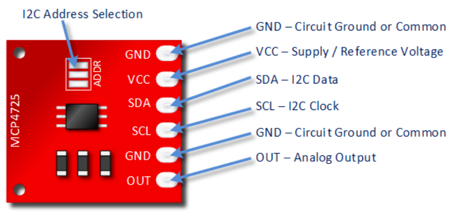
We will use the capacitor to create a RC Lowpass Filter in order to smooth the PWM signal from the Arduino to make a “true” Analog Out Signal

https://www.halvorsen.blog

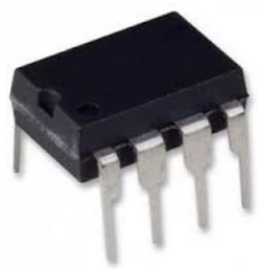
Digital to Analog Converters (DAC)

Hans-Petter Halvorsen

Option 2Using a DAC chip



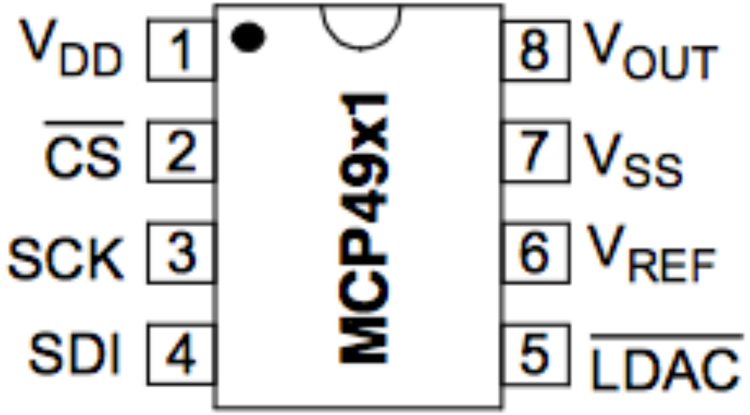
- DAC – Digital to Analog Converter
- Use, e.g., Microchip MCP4911, MCP4725 or similar
- **SPI** Arduino Library: https://www.arduino.cc/en/Reference/SPI
- MCP49XX Arduino Library: https://github.com/exscape/electronics/tree/master/Arduino/Libraries



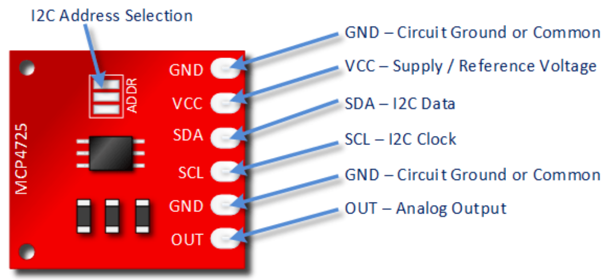
DAC

Arduino UNO has no Analog Output Pins, so we need a DAC such as, e.g., Microchip MCP4911, MCP4725 or similar

MCP4911: 10-bit single DAC, SPI Interface



MCP4725



12-bit resolution
I2C Interface

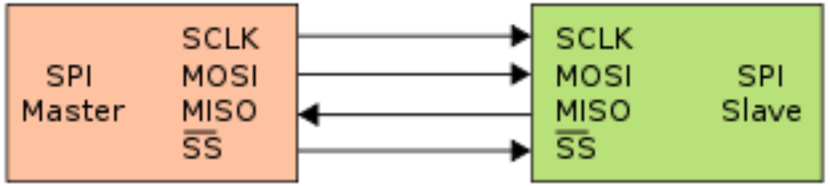
The MCP4725 is a little more expensive, but simpler to use

Microchip MCP4911 can be bought “everywhere” (10 NOK).

SPI Bus

- Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances.
- With an SPI connection there is always one master device (usually a microcontroller) which controls the peripheral devices.
- SPI devices communicate in full duplex mode using a master-slave architecture with a single master.
- The interface was developed by Motorola and has become a de facto standard.
- Typical applications include sensors, Secure Digital cards, and liquid crystal displays (LCD).

SCLK : Serial Clock (output from master)
MOSI : Master Output, Slave Input (output from master)
MISO : Master Input, Slave Output (output from slave)
SS (or SC) : Slave Select (active low, output from master)

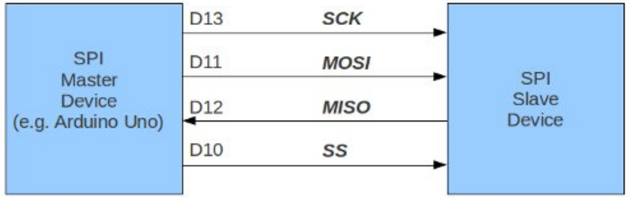


http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi

Arduino SPI

- https://www.arduino.cc/en/Reference/SPI
- http://tronixstuff.com/2011/05/13/tutorial-arduino-and-the-spi-bus/
- http://arduino.stackexchange.com/questions/16348/how-do-you-use-spi-on-an-arduino
- https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi



I2C Bus

- I^{2}C (Inter-Integrated Circuit), is a multi-master, multi-slave, single-ended, serial computer bus
- It is typically used for attaching lower-speed peripheral ICs to processors and microcontrollers.
- I^{2}C is typically spelled I2C (pronounced I-two-C)
- The I^{2}C bus was developed in 1982 by Philips Semiconductor.
- The I^{2}C protocol requires only 2 wires for connecting all the peripheral to a microcontroller.

https://learn.sparkfun.com/tutorials/i2c

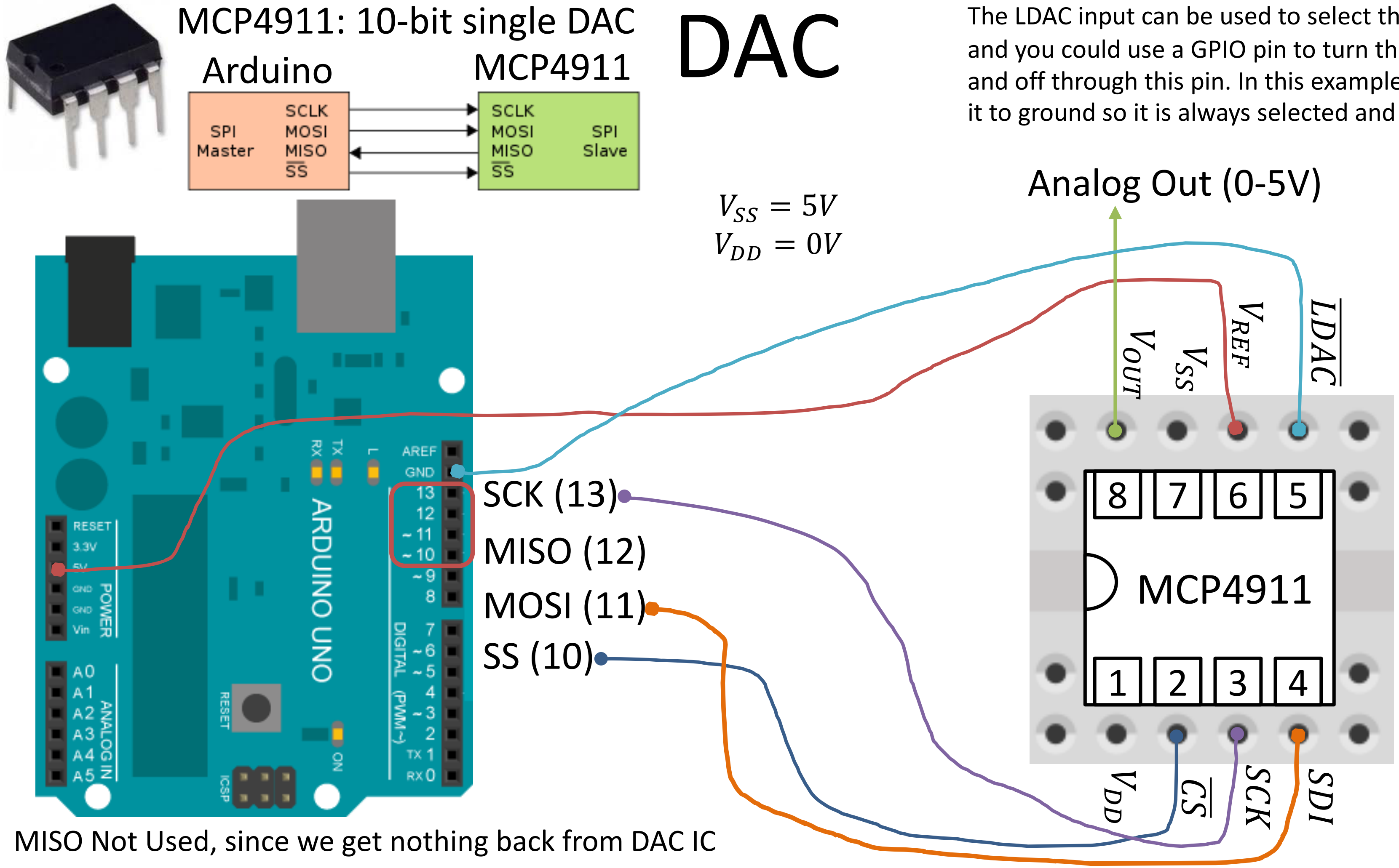
http://en.wikipedia.org/wiki/I2C

DAC

MCP4911: 10-bit single DAC

The LDAC input can be used to select the device, and you could use a GPIO pin to turn the device on and off through this pin. In this example, we just tie it to ground so it is always selected and powered.

$$V_{SS} = 5V$$
$$V_{DD} = 0V$$



MISO Not Used, since we get nothing back from DAC IC

MCP49xx Arduino Library Example

Example

```
#include <SPI.h>             //Include the Arduino SPI Library
#include <DAC_MCP49xx.h>     //Include the MCP49xx Arduino Library

// The Arduino pin used for the slave select / chip select
#define SS_PIN 10

//Set up the DAC DAC MCP4911
DAC_MCP49xx dac(DAC_MCP49xx::MCP4911, SS_PIN);

void setup()
{
}

void loop()
{
  double u; //Control Signal
  // For MCP4911, use values below (but including) 1023 (10 bit)
  u = 255; //Simulating the Control Value
  dac.output(u);
  delay(5000);

  u = 512; //Simulating the Control Value
  dac.output(u);
  delay(5000);
}
```

The control signal (u) should come from the PI/PID controller function.
It need to be converted from 0-5V (or 0-100%) -> 0-1023 before we send it to the DAC

Connect the circuit (Arduino + MCP4911) on a breadboard. Use a multi-meter so see if you get the correct output signal

https://github.com/exscape/electronics/tree/master/Arduino/Libraries

MCP49xx Arduino Library Example

```
#include <SPI.h>            //Include the Arduino SPI Library
#include <DAC_MCP49xx.h>    //Include the MCP49xx Arduino Library

// The Arduino pin used for the slave select / chip select
#define SS_PIN 10

DAC_MCP49xx dac(DAC_MCP49xx::MCP4911, SS_PIN);

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

void loop()
{
  double u; //Control Signal
  int aiPin = 0;
  int aiValue;

  for (int i=0; i<1023; i++)
  {
    u = i;
    dac.output(u);

    aiValue = analogRead(aiPin);
    Serial.print("AIValue=");
    Serial.println(aiValue);

    delay(1000);
  }
}
```

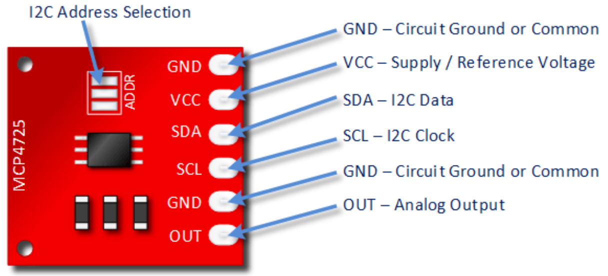
Connect the circuit (Arduino + MCP4911) on a breadboard. Use a multi-meter so see if you get the correct output signal.

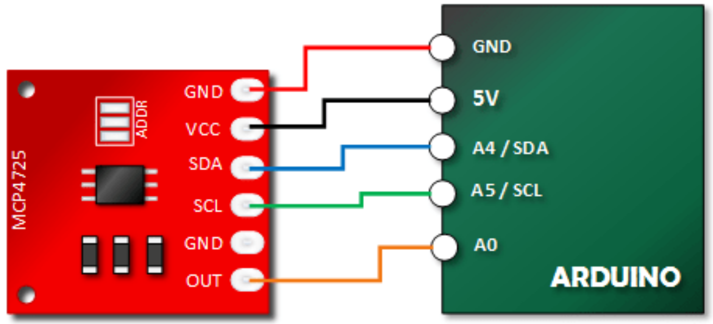
On the Multimeter you should see the output slowly increasing from ~0V to ~5V with intervals of 1000ms.

You can also connect the output from the DAC to an Analog Input Pin on the Arduino. Write the value to the Serial Monitor.

Alternative Solution

MCP4725





12-bit resolution
I2C Interface

The MCP4725 is a little more expensive (than MCP49xx), but simpler to use.

http://henrysbench.capnfatz.com/henrys-bench/arduino-output-devices/arduino-mcp4725-digital-to-analog-converter-tutorial/

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog