Fabian Winkler

Controlling motors with Arduino and Processing

Today's workshop illustrates how to control two different types of motors with the Arduino board: DC motors and servo motors. Since we have started to work with firmata and a Processing to Arduino link, this workshop will use simple user interfaces in Processing to control motors connected to an Arduino board. We will use the Processing controlP5 library to create a simple GUI (graphical user interface).

(1) Install the Processing controlP5 library

Download the library at http://www.sojamo.de/libraries/controlP5/ and follow the installation instructions: http://www.sojamo.de/libraries/controlP5/#installation

Start writing your a simple processing sketch that uses 3 sliders to generate colors in the RGB color system:

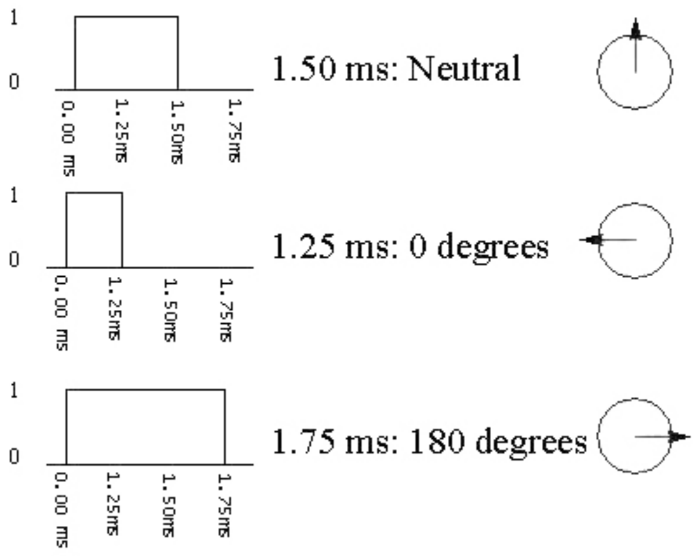
```
import controlP5.*;
ControlP5 controlP5;

int slider_RED = 200;
int slider_GREEN = 200;
int slider_BLUE = 200;

void setup() {

  size(400,400);

  controlP5 = new ControlP5(this);

  controlP5.addSlider("slider_RED",0,255,slider_RED,20,10,255,20);
  controlP5.addSlider("slider_GREEN",0,255,slider_GREEN,20,40,255,20);
  controlP5.addSlider("slider_BLUE",0,255,slider_BLUE,20,70,255,20);

  /*
  Check controlP5's javadocs to learn more about the arguments of the
"addSlider" method: addSlider(java.lang.String theName, float theMin,
float theMax, float theDefaultValue, int theX, int theY, int theW, int
theH)
 See: http://www.sojamo.de/libraries/controlP5/reference/index.html
  You can get the slider's value directly by referencing its name.
  */
}

void draw() {

  background(slider_RED, slider_GREEN, slider_BLUE);
}
```

You'll find many more examples on how to use this extremely powerful GUI library in the examples folder of the controlP5 library folder. For today we'll just need a simple slider that controls our motors.

(2) Connecting a servo motor to the Arduino

A servo motor is somewhat special in that it does not allow a continuous rotation of its output shaft (for example like regular DC motors) – rather it allows its output shaft to move very quickly to very precise output angles (usually somewhere between 0 and 180 degrees). It needs a precisely pulsed input signal to do so. The Seattle Robotics Society has a good and concise discussion of servo motors here:
http://www.seattlerobotics.org/guide/servos.html

This is their graphic illustrating the relationship between input pulse and output shaft rotation angle:



Servos are usually very powerful motors because of a gearbox that translates power and speed from the motor to the output shaft. With only minor modifications, a servo can be hacked and used as an inexpensive continuous rotation gearbox motor. See the Seattle Robotics Society's tutorial on how to do so:
http://www.seattlerobotics.org/guide/servohack.html

A good link for buying inexpensive DC gearbox motors (cheaper than modifying a servo) is this: http://www.solarbotics.com/motors_accessories/gear_motors/

Pages 121-123 in Sullivan and Igoe's book "Physical Computing" also has a more in depth discussion of servo motors.

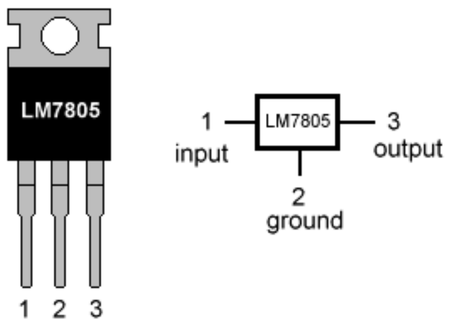
2a) Test your servo motor

Setup the servo and upload the code from the following example:
http://arduino.cc/en/Tutorial/Sweep

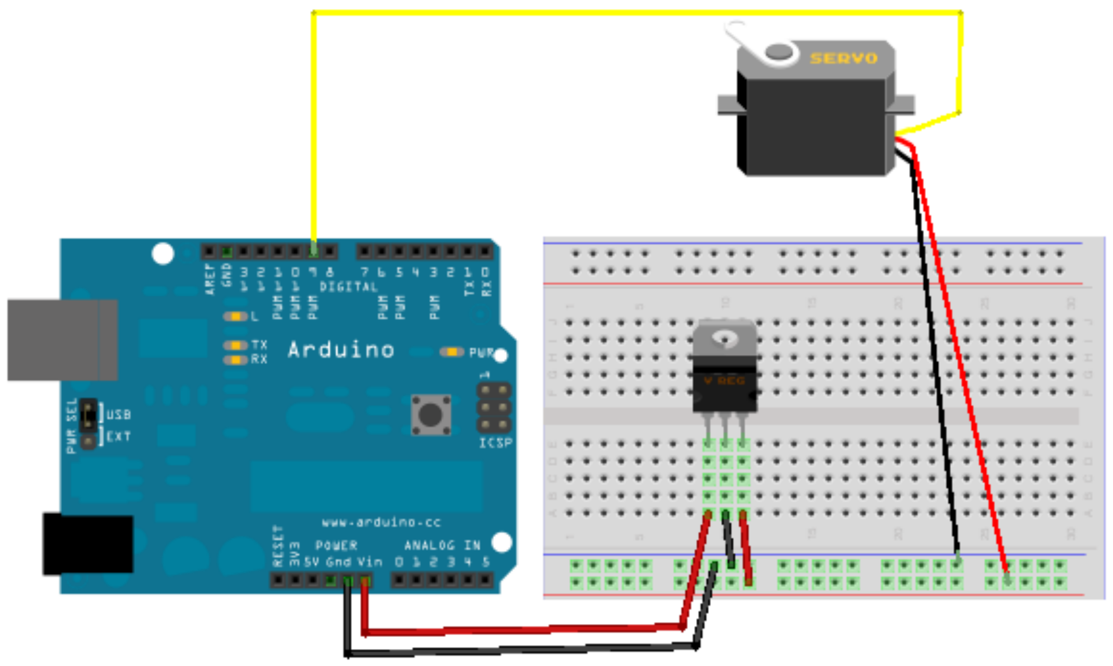
Servo motor power requirements

In contrast to DC motors which have varying power requirements, a servo motor almost always requires a voltage between 4.5 and 6VDC. If you power the servo from a power supply of less than +5V, that you will need to put a 1kOhm resistor between the I/O line and the servo's control line. If you plan to use the same power supply for the Arduino and the servo, make sure that it can supply at 1000mA of current. Especially if you plan to control more than one servo you will need an external power supply – servos can be quite power-demanding! If you are using two different power supplies, one for the Arduino and one for the servo make sure to connect both grounds (GND) together.

Since the power supply most of you have for this class provides 9V @ 1000mA, we need to find a way to convert the 9V into 5V to be able to power the servo, the 7805 voltage regulator IC does exactly this.



Here is a Fritzing sketch that illustrates how you connect the servo to an Arduino board:



You could also use the Arduino's on-board voltage regulator and get the 5VDC power for the servo from the Arduion's 5V output. However, larger servos draw more current and may overheat and damage the smaller on board voltage regulator.

2b) Processing to Servo

Now after setting up the Arduino circuit, let's control the servo's rotation angle with a slider in Processing. We will use firmata for the communication between Processing and Arduino and have the servo library handle the servo control from the Arduino board. Just upload the Servo firmata onto your Arduino board, in Arduino go to
`File > Examples > Firmata > Servo Firmata`

And upload the code to your board.

In Processing type in the following:

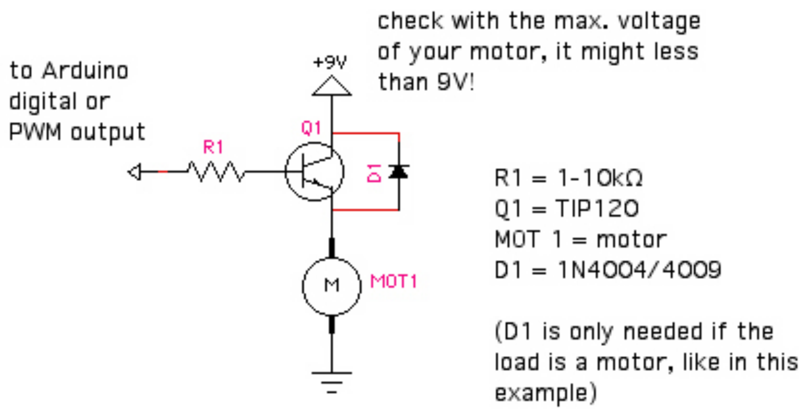
```
import processing.serial.*;
import cc.arduino.*;
import controlP5.*;

ControlP5 controlP5;
Arduino arduino;

int servoAngle = 90;

void setup() {

  size(400,400);

  println(Arduino.list());
  arduino = new Arduino(this, Arduino.list()[0], 57600);

  for (int i = 0; i <= 13; i++)
    arduino.pinMode(i, Arduino.OUTPUT);

  controlP5 = new ControlP5(this);
  controlP5.addSlider("servoAngle",0,180,servoAngle,20,10,180,20);

}

void draw() {

  arduino.analogWrite(9, servoAngle);
  //delay(15);
}
```

If you are not interested in using Processing to control the movements of your servo but would like to control the servo using only the Arduino and electronic circuitry, this example might be interesting, using the Arduino servo library: http://arduino.cc/en/Tutorial/Knob and http://arduino.cc/en/Reference/Servo

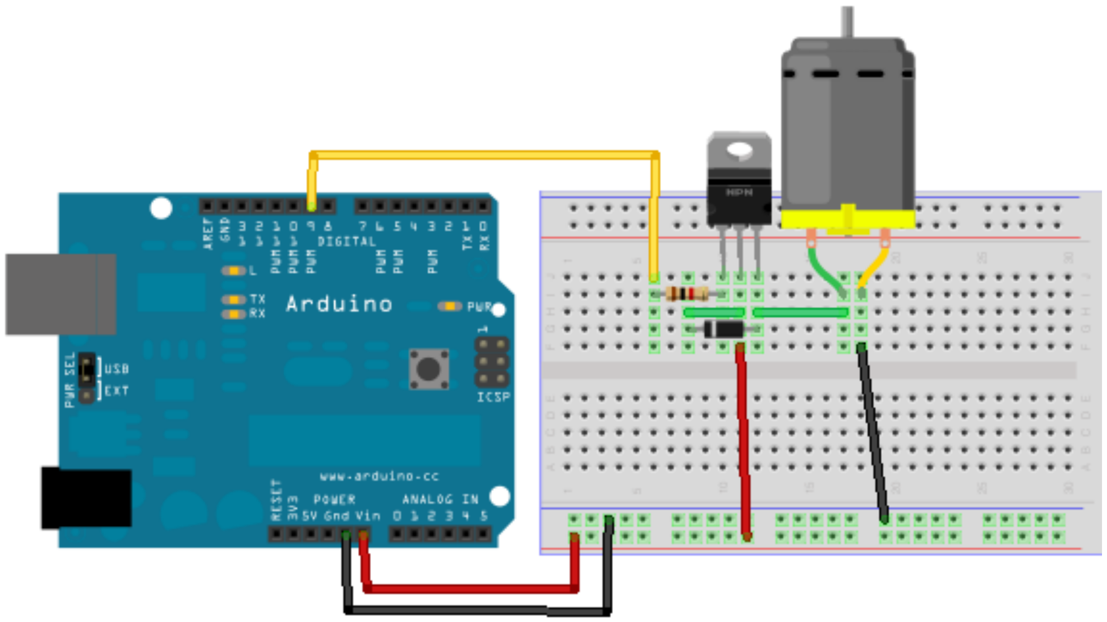
(3) DC motor control – simple

Regular DC motors are controlled differently than servo motors, for very simple DC motor control use a power transistor, such as the TIP120 controlled by one of the Arduino's PWM pins. Since DC motors draw a considerable amount of power they can't be powered directly by the Arduino PWM pin. In many cases it is also a good idea to use an external power supply with the Arduino that provides the necessary voltage and current for the motor. This example shows a simple way of controlling a DC motor which only allows you to change the motor speed not the direction in which the motor is turning (forward/backward).

This is what the circuit setup looks like:


check with the max. voltage of your motor, it might less than 9V!
+9V
to Arduino digital or PWM output
R1
Q1
D1
M
MOT1
R1 = 1-10kΩ
Q1 = TIP120
MOT 1 = motor
D1 = 1N4004/4009
(D1 is only needed if the load is a motor, like in this example)

On the breadboard this would translate into something like this:



With this setup make sure your Arduino board is powered with the external power supply!

Arduino code for this project: upload the SimpleAnalogFirmata example – File > Examples > Firmata > SimpleAnalogFirmata

Processing code
(allows you to control the speed of the motor with a GUI slider):

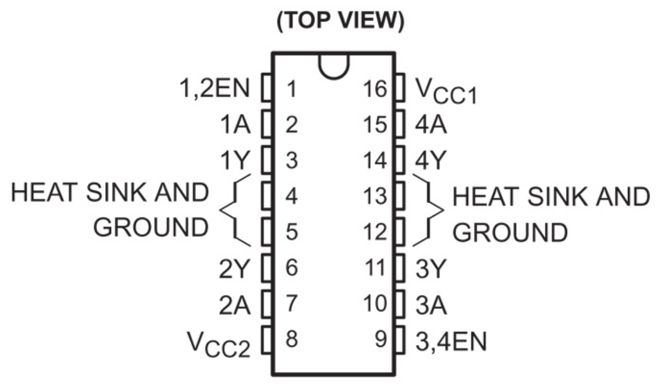
```
import processing.serial.*;
import cc.arduino.*;
import controlP5.*;

ControlP5 controlP5;
Arduino arduino;
```

```
int DC_speed = 150; // 0-255

void setup() {

  size(400,400);

  println(Arduino.list());
  arduino = new Arduino(this, Arduino.list()[0], 57600);

  for (int i = 0; i <= 13; i++)
    arduino.pinMode(i, Arduino.OUTPUT);

  controlP5 = new ControlP5(this);
  controlP5.addSlider("DC_speed",0,255,DC_speed,20,10,255,20);

}

void draw() {

  arduino.analogWrite(9, DC_speed);

}
```

(4) DC motor control – SN754410

This is a slightly more complicated way to control a motor, however it allows you to change the direction of the motor as well as the motor speed. The SN754410 is a handy IC that allows you to control the speed and direction of a DC motor with only one PWM output and two digital outputs from your Arduino board

SN754410
QUADRUPLE HALF-H DRIVER

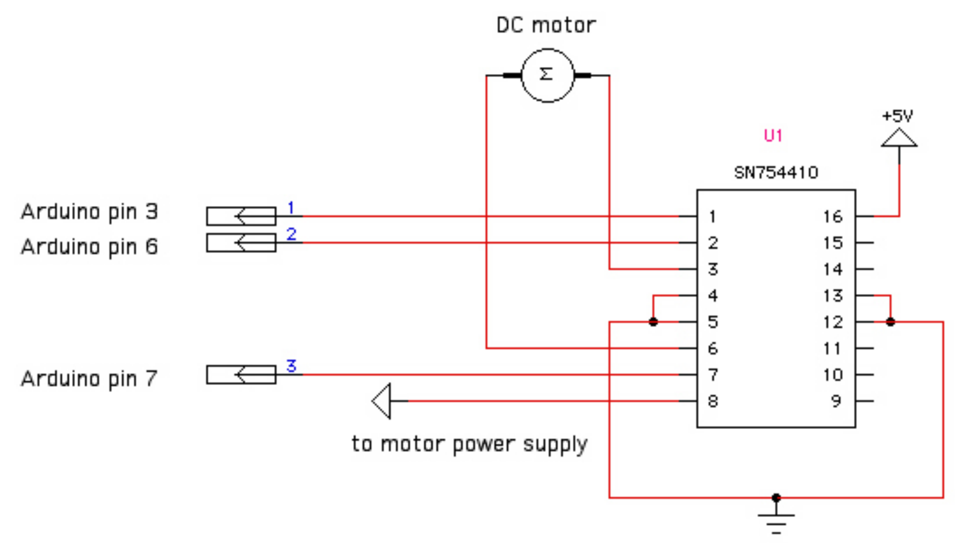


FUNCTION TABLE
(each driver)

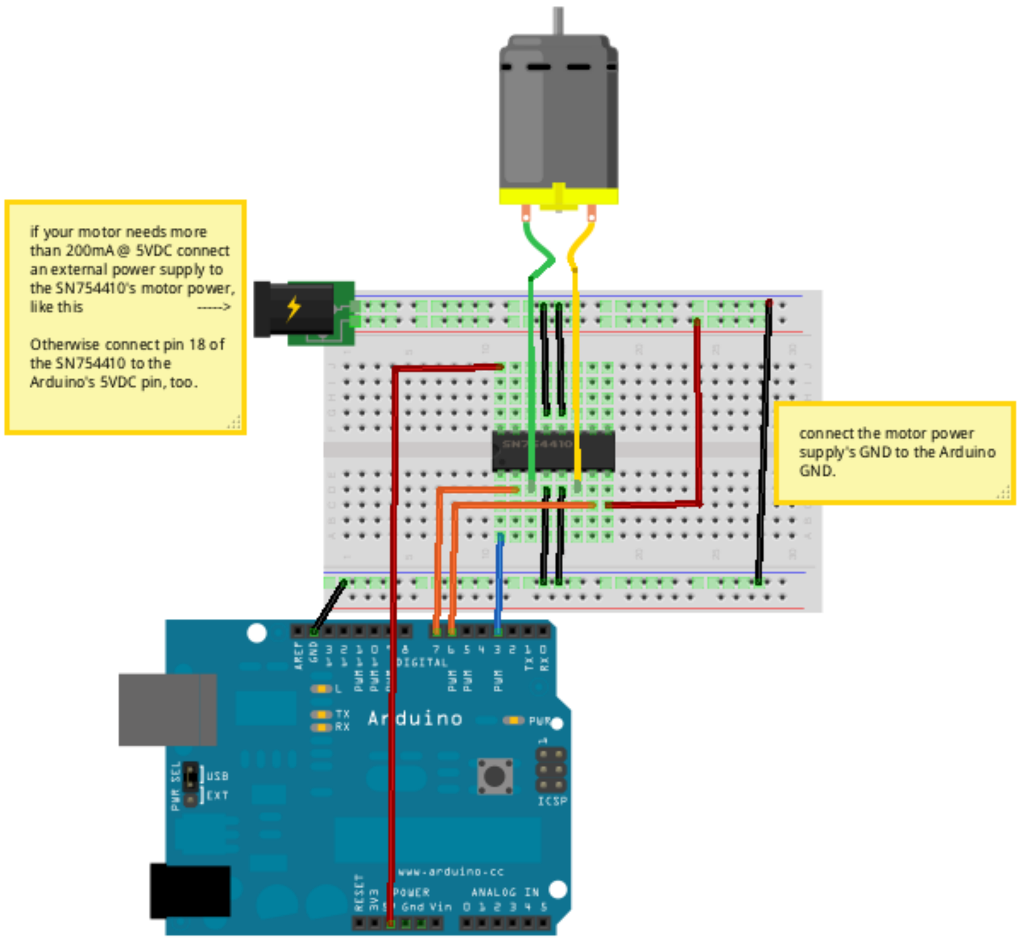
INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high-level, L = low-level
X = irrelevant
Z = high-impedance (off)
† In the thermal shutdown mode, the output is in a high-impedance state regardless of the input levels.

Please read pp.255 –260 in O'Sullivan and Igoe's book "Physical Computing" for more details on how to use the SN754410 motor driver IC with a microcontroller. Here is a circuit diagram for how to interface the SN754410 with the Arduino board:



And here is a picture what this should look like on your breadboard:



if your motor needs more than 200mA @ 5VDC connect an external power supply to the SN754410's motor power, like this ----->

Otherwise connect pin 18 of the SN754410 to the Arduino's 5VDC pin, too.

connect the motor power supply's GND to the Arduino GND.

Processing code (make sure you have the standard firmata code uploaded onto your Arduino board):

```
import processing.serial.*;
import cc.arduino.*;
import controlP5.*;

ControlP5 controlP5;
Arduino arduino;

int DC_speed = 150; // 0-255
int direction = 1;  // 0: backward, 1: forward

void setup() {

  size(400,400);

  println(Arduino.list());
  arduino = new Arduino(this, Arduino.list()[0], 57600);

  for (int i = 0; i <= 13; i++)
    arduino.pinMode(i, Arduino.OUTPUT);
    // pin3: PWM, pin 6: 1A, pin 7: 2A (see SN754410 datasheet)

  controlP5 = new ControlP5(this);
  controlP5.addSlider("DC_speed",0,255,DC_speed,20,10,255,20);

  Radio r = controlP5.addRadio("radio",20,50);
  //r.deactivateAll();
  // use deactiveAll to NOT make the first radio button active.
  r.add("forward",0);
  r.add("backward",1);

}

void draw() {

  arduino.analogWrite(3, DC_speed);

  if (direction == 1) { // run in one direction, i.e. forward
    arduino.digitalWrite(6, 1);
    arduino.digitalWrite(7, 0);
  }
  else { // run in the opposite direction, i.e. backward
    arduino.digitalWrite(6, 0);
    arduino.digitalWrite(7, 1);
  }

}

void radio(int theID) {
  switch(theID) {
    case(0):
      direction = 1;  // forward
      break;
    case(1):
      direction = 0;  // backward
      break;
    }
}
```

If you are interested in the kinetic and mechanic aspects of motor output, please read pp.271-283 “Basic Mechanics” in O’Sullivan and Igoe’s book “Physical Computing.”

LEGO technic pieces are also a great way to experiment with kinetic systems and Arduino-controlled motors. Look at these nice tutorials for a start:
http://www.clear.rice.edu/elec201/Book/legos.html
http://sariel.pl/2009/09/gears-tutorial/
http://neuron.eng.wayne.edu/LEGO.../lego_building_tutorial.pdf

You can buy inexpensive used LEGO pieces here:
http://www.bricklink.com/

Look specifically for gears and LEGO technic bricks (the ones with the holes). Also good are the LEGO 9V motors (part#2838c01) – it’s safe to buy them right away if you find them in good working condition for around $5.00.