AD23300 Electronic Media Studio
Prof. Fabian Winkler
Spring 2011

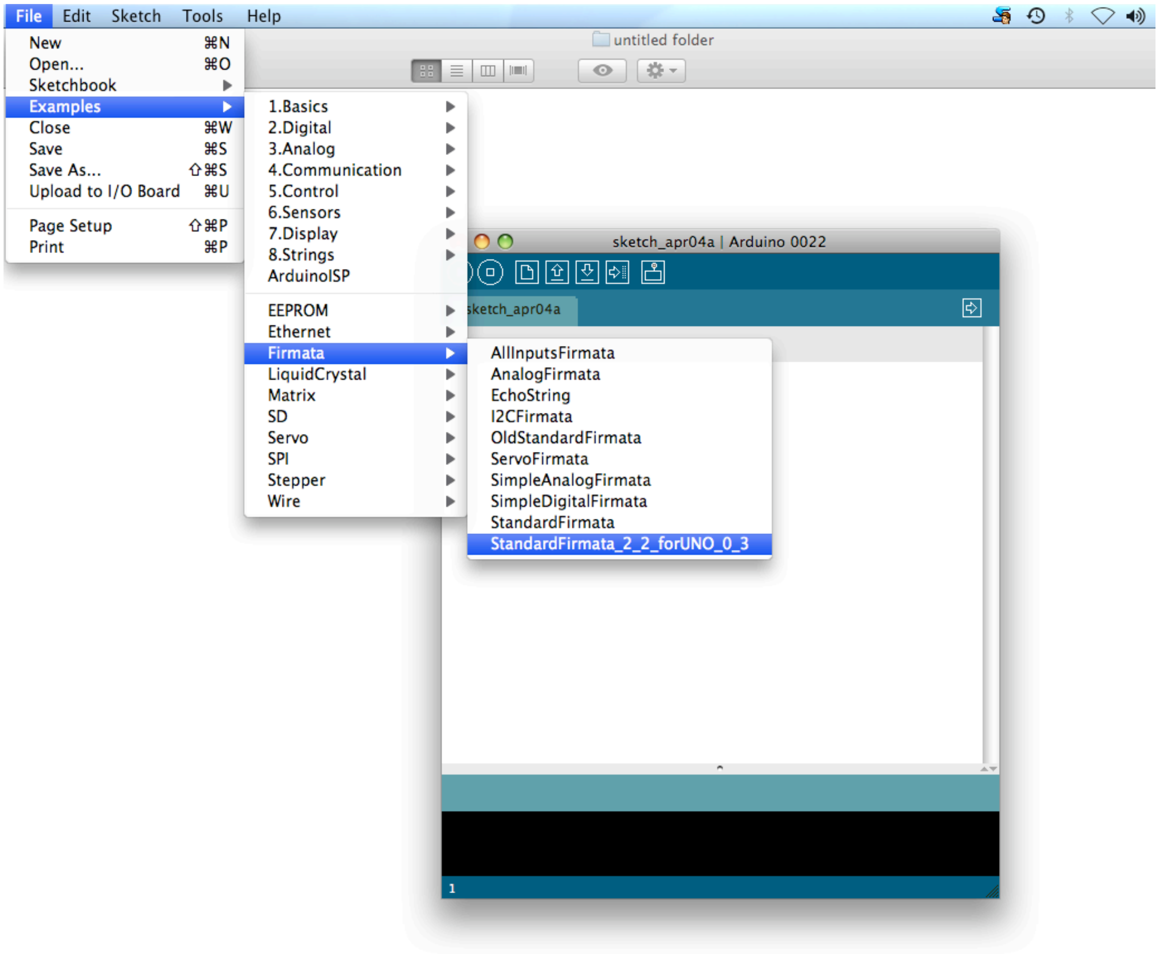
Arduino to Processing Communication with Firmata (rev. 5)

The following examples have been tested with the following software/hardware versions:

- Arduino 0022
- Processing 1.03
- Arduino UNO board

Upload firmata onto Arduino board:

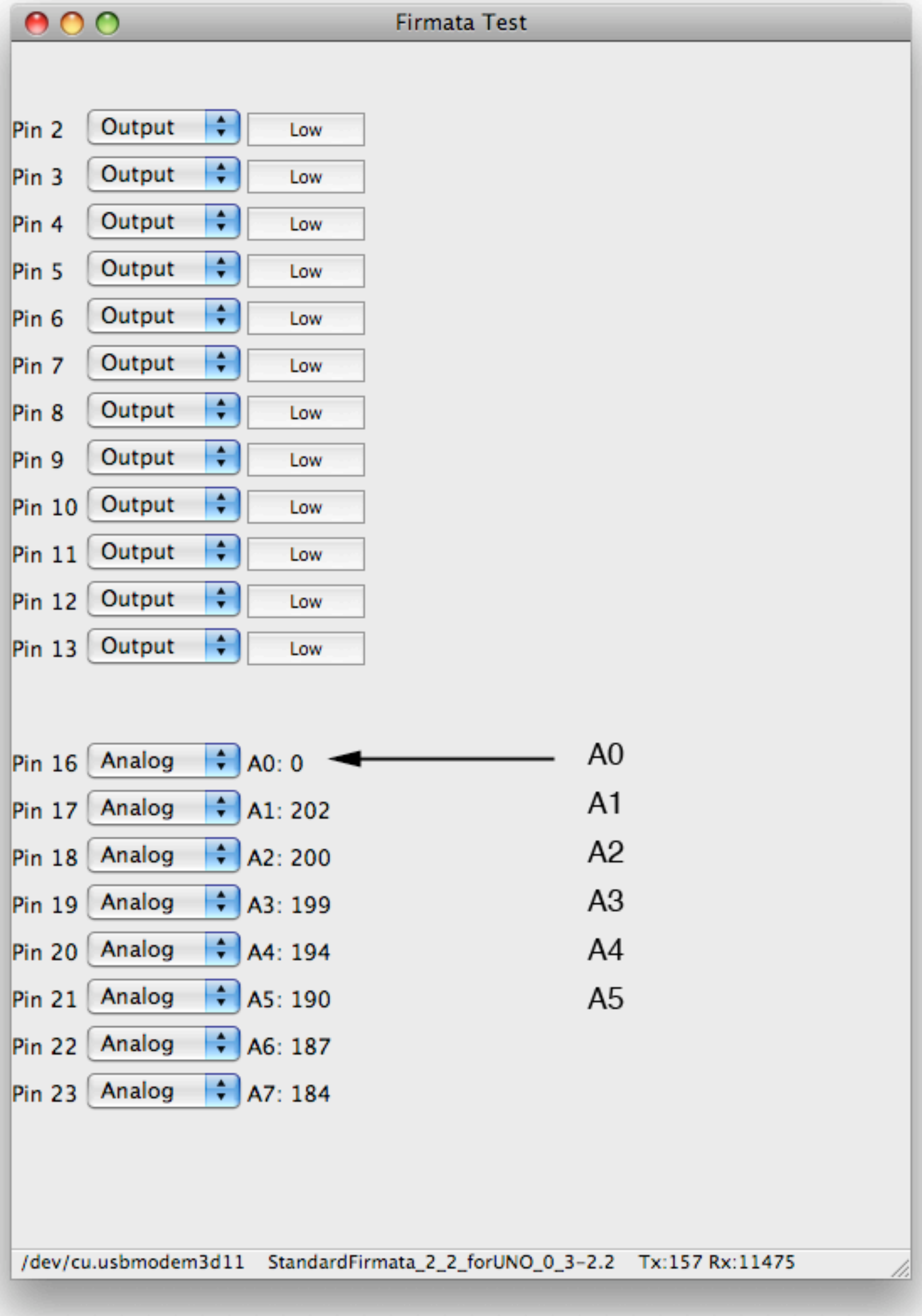
Open the Arduino software



Test firmata:

http://firmata.org/wiki/Main_Page

On the Macintosh, you should see a window like the one on the following page (make sure to set the correct serial port first: `Port > /dev/cu.USBxxx`



Pin 2 Output Low
Pin 3 Output Low
Pin 4 Output Low
Pin 5 Output Low
Pin 6 Output Low
Pin 7 Output Low
Pin 8 Output Low
Pin 9 Output Low
Pin 10 Output Low
Pin 11 Output Low
Pin 12 Output Low
Pin 13 Output Low

Pin 16 Analog A0: 0
Pin 17 Analog A1: 202
Pin 18 Analog A2: 200
Pin 19 Analog A3: 199
Pin 20 Analog A4: 194
Pin 21 Analog A5: 190
Pin 22 Analog A6: 187
Pin 23 Analog A7: 184

```
/dev/cu.usbmodem3d11    StandardFirmata_2_2_forUNO_0_3-2.2    Tx:157 Rx:11475
```

Get values from the Arduino board in Processing:

see: http://www.arduino.cc/playground/Interfacing/Processing

In order to have the Arduino board with the firmata firmware work correctly with Processing you have to install the Arduino library for Processing:

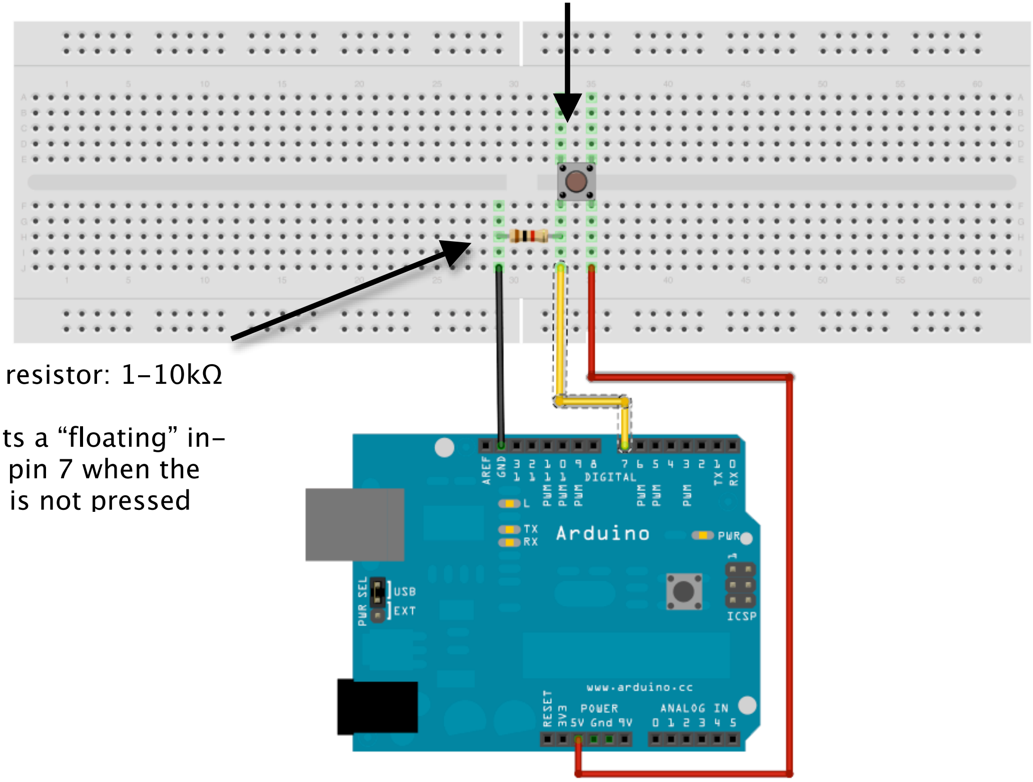
1. Download
Processing Library: http://arduino.cc/playground/uploads/Interfacing/processing-arduino-0017.zip (Updated 22 Sept. 2009)

2. Install

1. Unzip the library and copy the "arduino" folder into the "libraries" sub-folder of your Processing Sketchbook. (You can find the location of your Sketchbook by opening the Processing Preferences. If you haven't made a "libraries" sub-folder, create one.)
2. Run Arduino, open the Examples > Firmata > StandardFirmata sketch, and upload it to the Arduino board.
3. Configure Processing for serial: http://processing.org/reference/libraries/serial/
4. In Processing, open one of the examples that comes with with the Arduino library.
5. Edit the example code to select the correct serial port.
6. Run the example code printed below (example 1).

Example 1:
This is the most basic setup that allows Processing to respond to events in the external world. Depending on the state of the button a square changes its color.

Circuit - using Fritzing (http://fritzing.org) to visualize the connections between the Arduino and the components on the breadboard:



Pushbutton switch

pullup resistor: 1-10kΩ

prevents a “floating” input on pin 7 when the button is not pressed

Processing code for example 1:

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

Arduino arduino;
int val;
int inputPin = 7;


void setup() {
  size(400, 400);
  noStroke();
  println(Arduino.list());
  arduino = new Arduino(this, Arduino.list()[0], 57600);
  arduino.pinMode(inputPin, Arduino.INPUT);
}

void draw() {
  val = arduino.digitalRead(inputPin);

  background(204);
  if (val == 0) {
    fill(255, 127, 0); // fill orange
  } else {
    fill(0, 170, 0); // fill green
  }
  rect(50, 50, 300, 300);
}
```

Example 2:
Animating a sequence of images in Processing (without the Arduino board) – in order for this example to work you need to "add" all of the frames of your animation to the sketch first. Go to: `Sketch > Add file…` and then choose the frames you want to display. These will be copied into a folder called "data" which resides in your sketch folder.

```
int numFrames = 15;  // The number of frames in the animation
int frame = 0;
PImage[] images = new PImage[numFrames];

void setup()
{
  size(400, 300);
  frameRate(10);  // you can put different numbers here to play with
                  // the playback speed

  images[0]  = loadImage("money_0000.jpg");
  images[1]  = loadImage("money_0001.jpg");
  images[2]  = loadImage("money_0002.jpg");
  images[3]  = loadImage("money_0003.jpg");
  images[4]  = loadImage("money_0004.jpg");
  images[5]  = loadImage("money_0005.jpg");
  images[6]  = loadImage("money_0006.jpg");
  images[7]  = loadImage("money_0007.jpg");
  images[8]  = loadImage("money_0008.jpg");
  images[9]  = loadImage("money_0009.jpg");
  images[10] = loadImage("money_0010.jpg");
  images[11] = loadImage("money_0011.jpg");
  images[12] = loadImage("money_0012.jpg");
  images[13] = loadImage("money_0013.jpg");
  images[14] = loadImage("money_0014.jpg");

  // If you don't want to load each image separately
  // and you know how many frames you have, you
  // can create the filenames as the program runs.
  // The nf() command does number formatting, which will
  // ensure that the number is (in this case) 4 digits.
  //  for(int i=0; i<numFrames; i++) {
  //  String imageName = "PT_anim" + nf(i, 4) + ".gif";
  //  images[i] = loadImage(imageName);
  //}
}

void draw()
{

  if (frame<numFrames-1) {
    frame = (frame+1);
  } else {
    frame = 0;
  }

  image(images[frame], 0, 0);
}
```

Example 3:
Animating a sequence of images based on pushing a button
The circuit setup is the same as in example 1 (see page 3)

Processing code:

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

Arduino arduino;
int val;
int inputPin = 7;


int numFrames = 15;  // The number of frames in the animation
int frame = 0;
PImage[] images = new PImage[numFrames];

void setup()
{

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

  size(400, 300);
  images[0]  = loadImage("money_0000.jpg");
  images[1]  = loadImage("money_0001.jpg");
  images[2]  = loadImage("money_0002.jpg");
  images[3]  = loadImage("money_0003.jpg");
  images[4]  = loadImage("money_0004.jpg");
  images[5]  = loadImage("money_0005.jpg");
  images[6]  = loadImage("money_0006.jpg");
  images[7]  = loadImage("money_0007.jpg");
  images[8]  = loadImage("money_0008.jpg");
  images[9]  = loadImage("money_0009.jpg");
  images[10] = loadImage("money_0010.jpg");
  images[11] = loadImage("money_0011.jpg");
  images[12] = loadImage("money_0012.jpg");
  images[13] = loadImage("money_0013.jpg");
  images[14] = loadImage("money_0014.jpg");

  // If you don't want to load each image separately
  // and you know how many frames you have, you
  // can create the filenames as the program runs.
  // The nf() command does number formatting, which will
  // ensure that the number is (in this case) 4 digits.
  //  for(int i=0; i<numFrames; i++) {
  //  String imageName = "PT_anim" + nf(i, 4) + ".gif";
  //  images[i] = loadImage(imageName);
  //}
}

void draw()
{
  val = arduino.digitalRead(inputPin);

  if (val == 1) {
   if (frame<numFrames-1) {
     frame = (frame+1);
   } else {
     frame = 0;
  }
  }

  image(images[frame], 0, 0);
}
```

Example 4:
Randomly triggering images based on pushing a button.
This example code is very similar to the previous one, only now the images are not played back sequentially but based on the value of a random number.

Processing code:

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

Arduino arduino;
int val;
int rand_frame = 0;

int numFrames = 15;  // The number of frames in the animation
int frame = 0;
PImage[] images = new PImage[numFrames];

void setup()
{

  println(Arduino.list());
  arduino = new Arduino(this, Arduino.list()[0], 57600);
  arduino.pinMode(7, Arduino.INPUT);

  size(400, 300);
  frameRate(15);

  images[0]  = loadImage("money_0000.jpg");
  images[1]  = loadImage("money_0001.jpg");
  images[2]  = loadImage("money_0002.jpg");
  images[3]  = loadImage("money_0003.jpg");
  images[4]  = loadImage("money_0004.jpg");
  images[5]  = loadImage("money_0005.jpg");
  images[6]  = loadImage("money_0006.jpg");
  images[7]  = loadImage("money_0007.jpg");
  images[8]  = loadImage("money_0008.jpg");
  images[9]  = loadImage("money_0009.jpg");
  images[10] = loadImage("money_0010.jpg");
  images[11] = loadImage("money_0011.jpg");
  images[12] = loadImage("money_0012.jpg");
  images[13] = loadImage("money_0013.jpg");
  images[14] = loadImage("money_0014.jpg");

  // If you don't want to load each image separately
  // and you know how many frames you have, you
  // can create the filenames as the program runs.
  // The nf() command does number formatting, which will
  // ensure that the number is (in this case) 4 digits.
  //  for(int i=0; i<numFrames; i++) {
  //  String imageName = "PT_anim" + nf(i, 4) + ".gif";
  //  images[i] = loadImage(imageName);
  //}
}

void draw()
{
  val = arduino.digitalRead(7);

  if (val == 1) {
   rand_frame = int(random(numFrames-1));
  }

  image(images[rand_frame], 0, 0);
}
```

If you only like to trigger one random image while the button is pressed use the following code instead. It uses a variable “oldval” to compare the incoming sensor value against, a new random image is only triggered when your sensor values is 1 and oldval is not equal to val (i.e. the previous sensor reading had to be a 0):

Processing code:

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

Arduino arduino;
int val;
int rand_frame = 0;
int oldval = 0;

int numFrames = 15;  // The number of frames in the animation
int frame = 0;
PImage[] images = new PImage[numFrames];

void setup()
{

  println(Arduino.list());
  arduino = new Arduino(this, Arduino.list()[0], 57600);
  arduino.pinMode(7, Arduino.INPUT);

  size(400, 300);
  frameRate(15);

  images[0]  = loadImage("money_0000.jpg");
  images[1]  = loadImage("money_0001.jpg");
  images[2]  = loadImage("money_0002.jpg");
  images[3]  = loadImage("money_0003.jpg");
  images[4]  = loadImage("money_0004.jpg");
  images[5]  = loadImage("money_0005.jpg");
  images[6]  = loadImage("money_0006.jpg");
  images[7]  = loadImage("money_0007.jpg");
  images[8]  = loadImage("money_0008.jpg");
  images[9]  = loadImage("money_0009.jpg");
  images[10] = loadImage("money_0010.jpg");
  images[11] = loadImage("money_0011.jpg");
  images[12] = loadImage("money_0012.jpg");
  images[13] = loadImage("money_0013.jpg");
  images[14] = loadImage("money_0014.jpg");

  // If you don't want to load each image separately
  // and you know how many frames you have, you
  // can create the filenames as the program runs.
  // The nf() command does number formatting, which will
  // ensure that the number is (in this case) 4 digits.
  //  for(int i=0; i<numFrames; i++) {
  //  String imageName = "PT_anim" + nf(i, 4) + ".gif";
  //  images[i] = loadImage(imageName);
  //}
}

void draw()
{
  val = arduino.digitalRead(7);

  if (val != oldval && val == 1) {
  // the line above makes sure we advance only one frame with
  // each pressing of the button
```

```
   rand_frame = int(random(numFrames-1));
  }

  image(images[rand_frame], 0, 0);
  oldval = val;

}
```

Example 5:
Playing back sounds by pressing a button.
This example plays back a sound (and loops it) when the button is pressed. When the button is released the sound stops and it continues to play where it was paused the next time the button is pressed. This example uses the “minim” library (see: http://code.compartmental.net/tools/minim/). You might have to check the Javadocs at: http://code.compartmental.net/minim/javadoc/ to find out all about the methods you can use with certain classes, such as the AudioPlayer class. Make sure you have the file “expressway.aiff” from the “media_for_Processing_examples.zip” folder in your sketch’s data folder:

```
import cc.arduino.*;
import processing.serial.*;
import ddf.minim.*;

Minim minim;
AudioPlayer mySound;
Arduino arduino;

int val;
int oldval = 0;
int playback_pos = 0;

int numFrames = 15;  // The number of frames in the animation
int frame = 0;
PImage[] images = new PImage[numFrames];

void setup()
{

  println(Arduino.list());
  arduino = new Arduino(this, Arduino.list()[0], 57600);
  arduino.pinMode(7, Arduino.INPUT);

  size(400, 400);
  noStroke();

  minim = new Minim(this);
  mySound = minim.loadFile("expressway.aiff");
  // make sure you the file “expressway.aiff” in your “data” folder
  fill(255, 127, 0);
}

void draw()
{
  val = arduino.digitalRead(7);
  background(204);

  if (val != oldval && val == 1) {
  // the line above makes sure we only trigger the sound once
  // when the pushbutton is pressed & held down
     fill(0, 170, 0); // fill green
     mySound.play(playback_pos);
     // print("played back at: ");
     // println(playback_pos);
     // continues to playback where the sound was paused by
     // remembering the playback head's position
     mySound.loop();
   }

  if (val != oldval && val == 0) {
  // the line above makes sure we only pause the sound once
  // when the pushbutton is released
     fill(255, 127, 0); // fill orange
     playback_pos = mySound.position();
     // print("paused at: ");
     // println(playback_pos);
     // saves the position of the playback head when paused
     mySound.pause();
   }

  rect(50, 50, 300, 300);
  oldval = val;
  println(playback_pos);
}


void stop()
{
  // always close Minim audio classes when you are done with them
 mySound.close();
 minim.stop();
 super.stop();
}
```

Example 6:
This examples randomly triggers one of four sounds, in contrast to the example above, it uses Minim's AudioSnippet class which is useful to play back short sounds (not more than 5-15 seconds). This is the fastest way to play back sounds immediately after a switch has been triggered (AudioPlayer takes a couple of milliseconds to load the sounds into RAM).

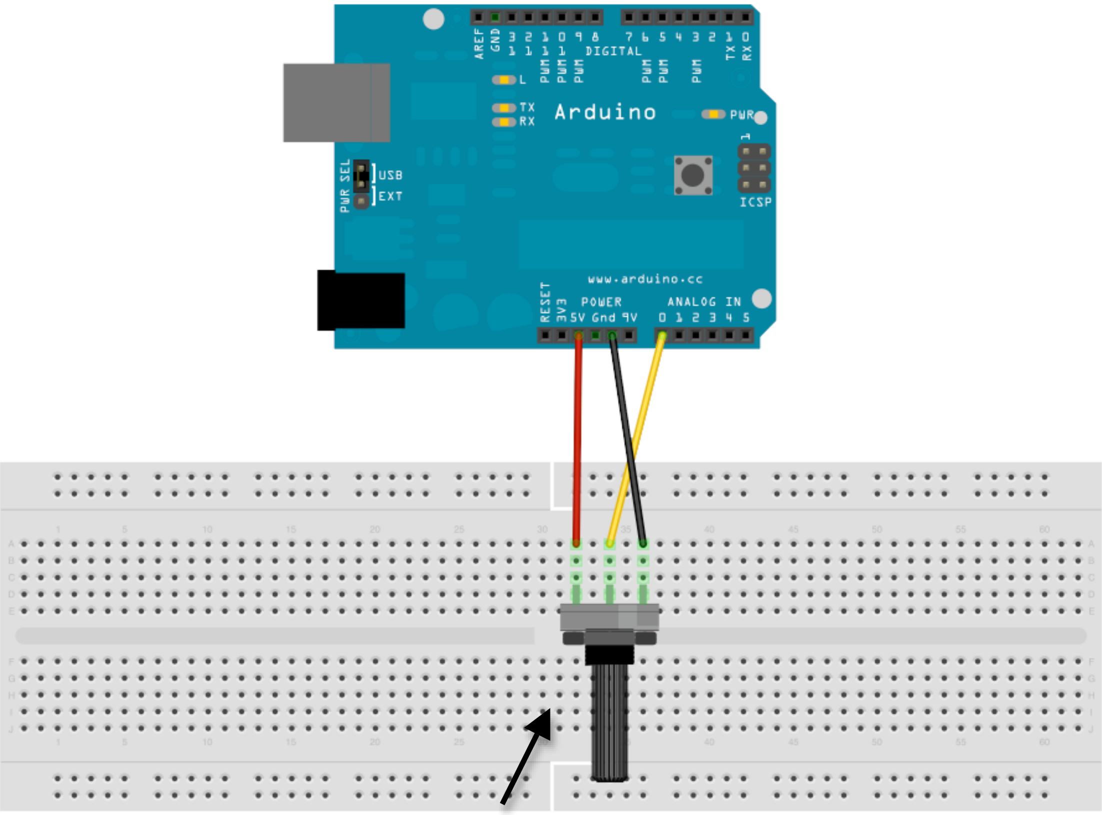
See the code example in the attached folder with the prefix: "example6".

Example 7:
Switching between different videos by pushing a button. This one is a little more complicated since we need to use the jmcvideo library for Processing: http://www.mat.ucsb.edu/~a.forbes/PROCESSING/jmcvideo/jmcvideo.htmlthe installation instructions for this library are quite complicated, I tried to make things easier (at least for the Mac users:) by including the folder "jmcvideo" that you downloaded with this workshop document. Simply place this folder into your Processing libraries folder, restart Processing and you should be ready to go. PC users please follow

the installation instructions at
http://www.mat.ucsb.edu/~a.forbes/PROCESSING/jmcvideo/jmcvideo.html

```
/**
  * This processing sketch randomly plays back video excerpts from D.W.
  * Griffith’s film “Abraham Lincoln” –
  * http://www.archive.org/details/abraham_lincoln. It creates non-
  * linear and possibly new interpretations of Lincoln’s life through
  * recombinant strategies.
  *
  * How it works:
  * Pressing a pushbutton connected to the Arduino board randomly plays
  * one of six videos. The video playback stops at the end of the video
  * and the Processing sketch waits for the next button push to play
  * another randomly chosen video.
  *
  * You will need to install the jmcvideo library for Quicktime video
  * playback. More information on this installation and the download of
  * the library can be found here:
  * http://www.mat.ucsb.edu/~a.forbes/PROCESSING/jmcvideo/jmcvideo.html
  *
  * This example is based on the VideoSwitchGL example by Angus Forbes
  * copyleft, 2011: Fabian Winkler
  *
 */

import cc.arduino.*;
import processing.serial.*;
import jmcvideo.*;
import processing.opengl.*;
import javax.media.opengl.*;

Arduino arduino;
JMCMovieGL myMovie;
int pvw, pvh;
int val = 0;
int oldval = 0;

// change the following code line to put in your own videos,
// don't forget to put them into the sketches' "data" folder
String[] vids = new String[]{"lincoln_01.mov", "lincoln_02.mov",
"lincoln_03.mov", "lincoln_04.mov", "lincoln_05.mov",
"lincoln_06.mov"};
int vidNum = 0;

void setup() {
  size(320, 240, OPENGL);
  frame.setResizable(true);
  background(0);

  println(Arduino.list());
  arduino = new Arduino(this, Arduino.list()[0], 57600);
  arduino.pinMode(7, Arduino.INPUT);

  // always start out with video number 1:
  myMovie = movieFromDataPath(vids[vidNum]);
  myMovie.play();
}

void draw() {
val = arduino.digitalRead(7);

  // make sure we only trigger the video switch once
  // when the pushbutton is pressed & held down
  if (val != oldval && val == 1) {

   //create random numbers from 0 to (the amount of videos imported) -1
     vidNum = (int)random(vids.length);
     // switch video to the number randomly generated
     myMovie.switchVideo(vids[vidNum]);
  }

  // start playing the video when the push button is released
  if (val != oldval && val == 0) {
    myMovie.play();
  }


  // leave everything below this line alone...
  // ----------------------------------------
  PGraphicsOpenGL pgl = (PGraphicsOpenGL) g;

  GL gl = pgl.beginGL();
  {
    if (pvw != width || pvh != height)
    {
      background(0);
      gl.glViewport(0, 0, width, height);
      pvw = width;
      pvh = height;
    }
    myMovie.centerImage(gl);
  }
  pgl.endGL();

  oldval = val;
}



JMCMovieGL movieFromDataPath(String filename)
{
  return new JMCMovieGL(this, filename, RGB);
}
```

Example 8:
Connecting an analog sensor to the Arduino board and read the sensor's values in Processing. This is the most basic example, it uses the square from example 1 but this time, by turning the potentiometer the color changes gradually. On the following page is the circuit diagram for this example.



potentiometer, you have to solder some wires to the solder lugs on the component to make it fit into the breadboard

Processing code:

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

Arduino arduino;
int color_val;

 void setup() {
  size(400, 400);
  noStroke();
  println(Arduino.list());
  arduino = new Arduino(this, Arduino.list()[0], 57600);
  //arduino.pinMode(7, Arduino.INPUT);
}

void draw() {
  color_val = arduino.analogRead(0)/4;
  // we only need values from 0 - 255

  background(204);
  //from green to orange:
  fill(color_val, 127, 0);

  rect(50, 50, 300, 300);
}
```