PATRICK EWING

SANTIAGO ECHEVERRY

ART 430/SENIOR PROJECT

12/13/11

Final Senior Project

My final project for ART 430 Physical Computing this semester is also doubling as my senior project. As an EMAT/New Media Production major, I have spent my time here at UT learning how to use and create many different forms of media. As I have began taking more difficult, higher level classes however, the shift has gone away from how to use and create media, and moved more towards finding new ways to use already existing media, as well as interacting with new media forms that are just emerging. I have enjoyed my time here, and decided to take the core idea of the major to heart. My project, put simply, uses an old form of media in a new way. I decided to use a musical instrument, known as a Theremin, as an input device to allow users to interact with a flash animation.

The Theremin is a musical instrument that gained popularity because it is one of the few instruments in existence that can be played without any physical contact by the user. Russian physicist and inventor, Leon Theremin, created the instrument in Russia October of 1920. Theremin created the instrument while doing research on proximity sensors for the Russian government. Theremin came to the United States on December 30, 1927, after going on a tour of Europe with his device, Once in the States, Theremin patented his device and gave commercial production rights to the Radio Corporation of America, or RCA as they are better known. The Theremin has had many different uses over the years in many forms of media

including concert and popular music, various film scores, videogame sound tracks, as well as on television in an episode of *The Simpsons*. The Theremin has never been a commercial success in the US largely because it was released just after the infamous black Tuesday stock market crash of 1929, however it has interested many including the famous electronic musician Robert Moog, whose music company still makes Theremin kits to this day. I myself was fascinated when I first learned about the Theremin in one of my electronic music classes. My fascination was what led me to decide to use the instrument in my project.

Being a musical instrument the Theremin is largely used to create sound effects and music. My project not only uses the Theremin for creating sound, but in an entirely new way as well. A Theremin generates sound using the heterodyne principal. Inside each Theremin are two frequency oscillators. One oscillator generates a constant frequency; the other is connected to an antenna and when electronic current flows through the oscillator, an electromagnetic field is generated around the antenna. The frequency of the second oscillator is controlled by distortions in this electromagnetic field, which are generated by the position of the user's hand relative to the antenna; these distortions are registered by the frequency oscillator through its antenna, and its frequency changes accordingly. Hence, no physical contact is needed to play the instrument. The frequencies two oscillators are then combined, and the resulting third frequency is played back

through a speaker. Nicer Theremin kits have two sets of these oscillators, one to control the frequency, or pitch of the instrument, and the other to control the amplitude, or volume of the instrument. The Theremin I used for my project however, only had the set for controlling the frequency.

I disconnected the speaker in my Theremin, and soldered extensions onto the wires. I then connected these wires to one of the analog ports on a computer chip we used in Physical Computing known as an Arduino Board. The Arduino Board uses a programming language known as processing to process input data from devices, such as a potentiometer, to control other devices or media forms. In my project, I connected my Theremin to my Arduino Board and sent its frequency into my laptop, where I programmed the Arduino interface to play the frequency back through a speaker (which was connected to one of the digital inputs on the Arduino Board). I also programmed the Arduino interface to send the frequency through another application called Serproxy (which allowed my laptop itself to register the input data) and into Adobe Flash, where I used it to control various parameters of a flash animation.

In its simplest form, my project allows the user to create sound and control a flash animation without having to touch the input device. I took an old form of media, a musical instrument, and found a new, and very fun way to use it, which is what the EMAT/New Media Production major is all about. My project however, is

just the tip of the iceberg. The beauty of the Arduino interface is that while I chose to use it to control flash, it could actually be used to control any form of media, so the uses for my Theremin are endless. I simply chose to demonstrate one possible use of it through Adobe Flash.

PATRICK EWING

SANTIAGO ECHEVERRY

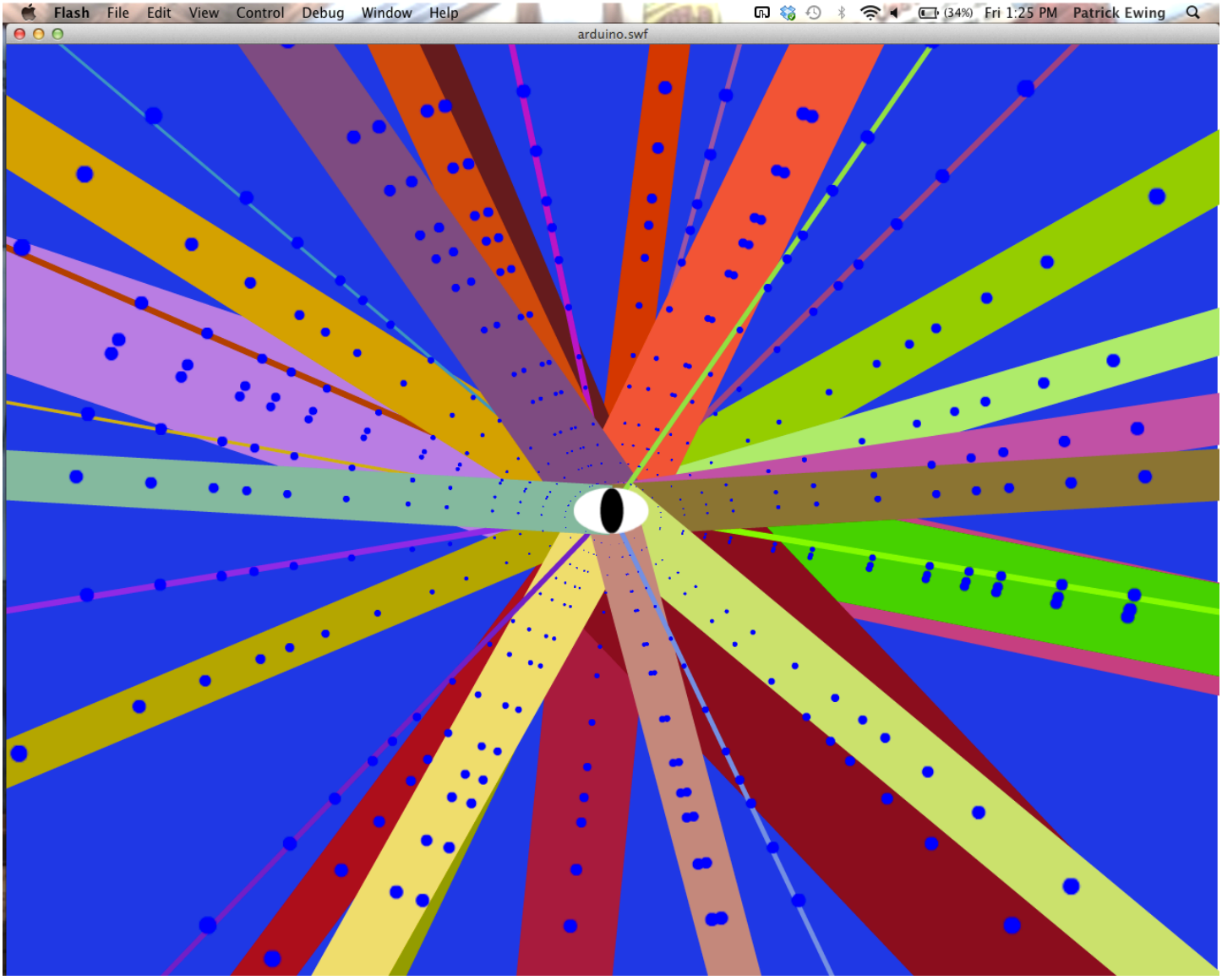
ART 430/SENIOR PROJECT

12/13/11

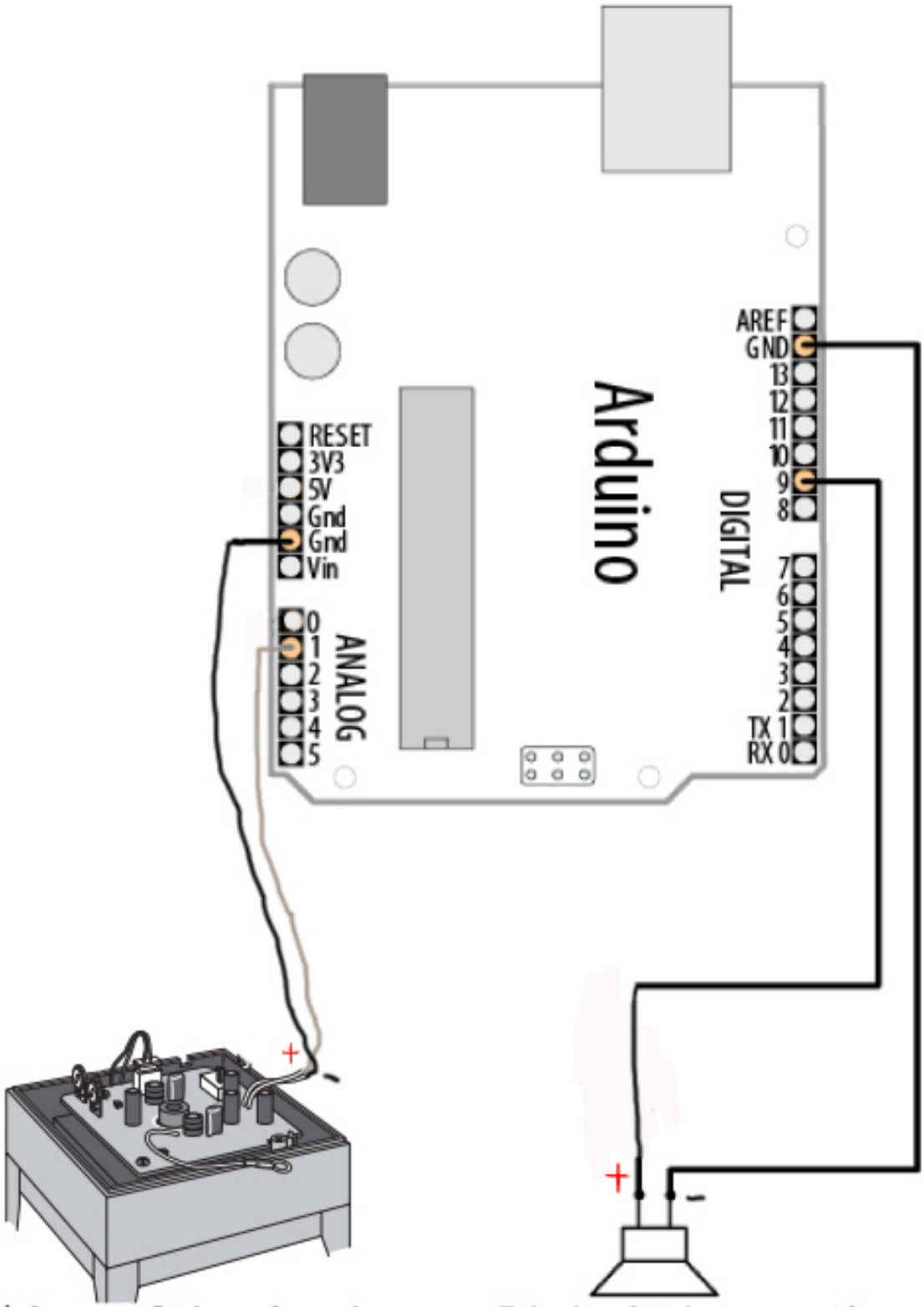
Works Cited

"Léon Theremin." *Wikipedia, the Free Encyclopedia*. Wikimedia Foundation. Web. 10 Dec. 2011. <http://en.wikipedia.org/wiki/Wikipedia>.

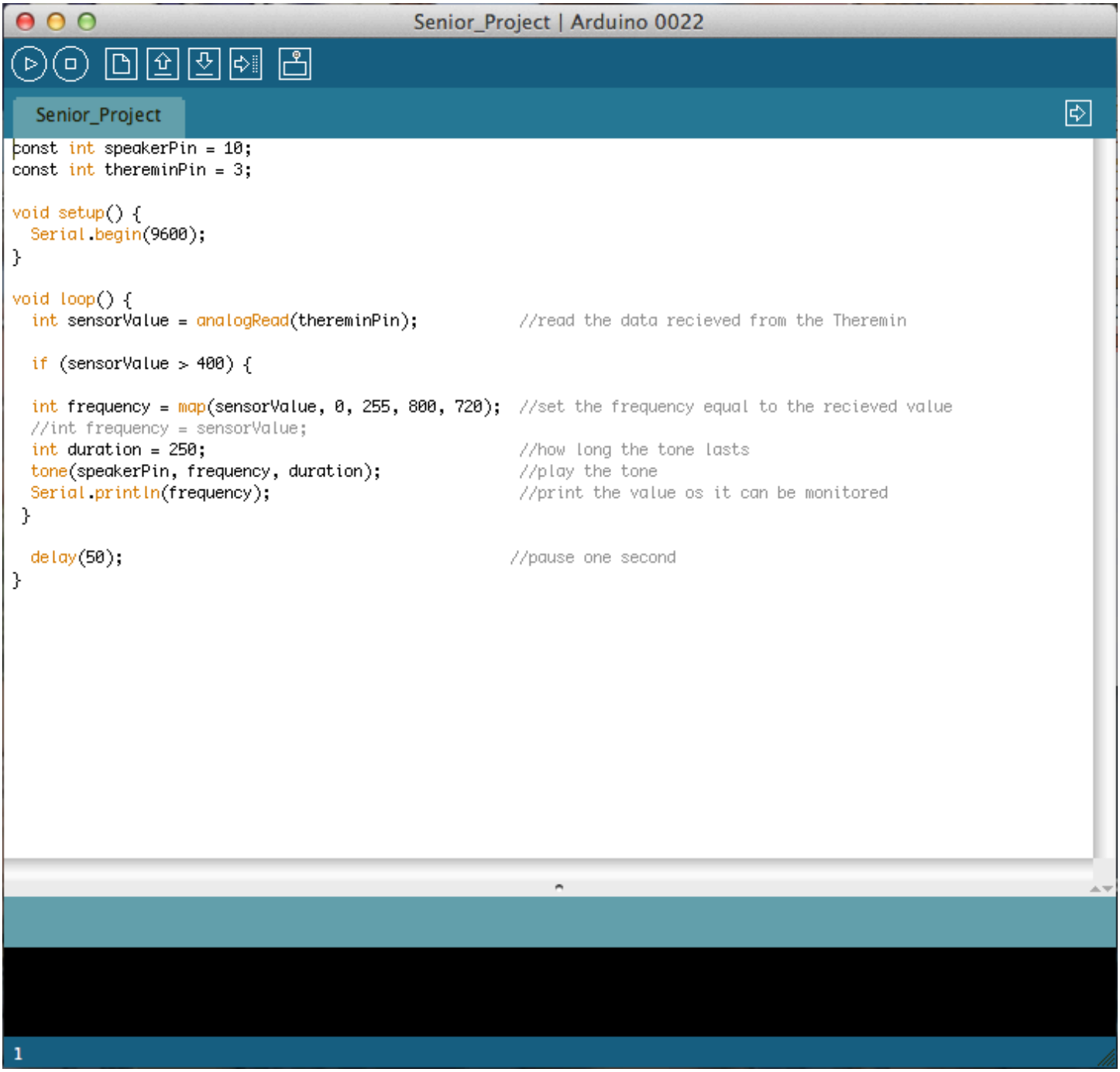
"Theremin." *Wikipedia, the Free Encyclopedia*. Wikimedia Foundation. Web. 10 Dec. 2011. <http://en.wikipedia.org/wiki/Wikipedia>.

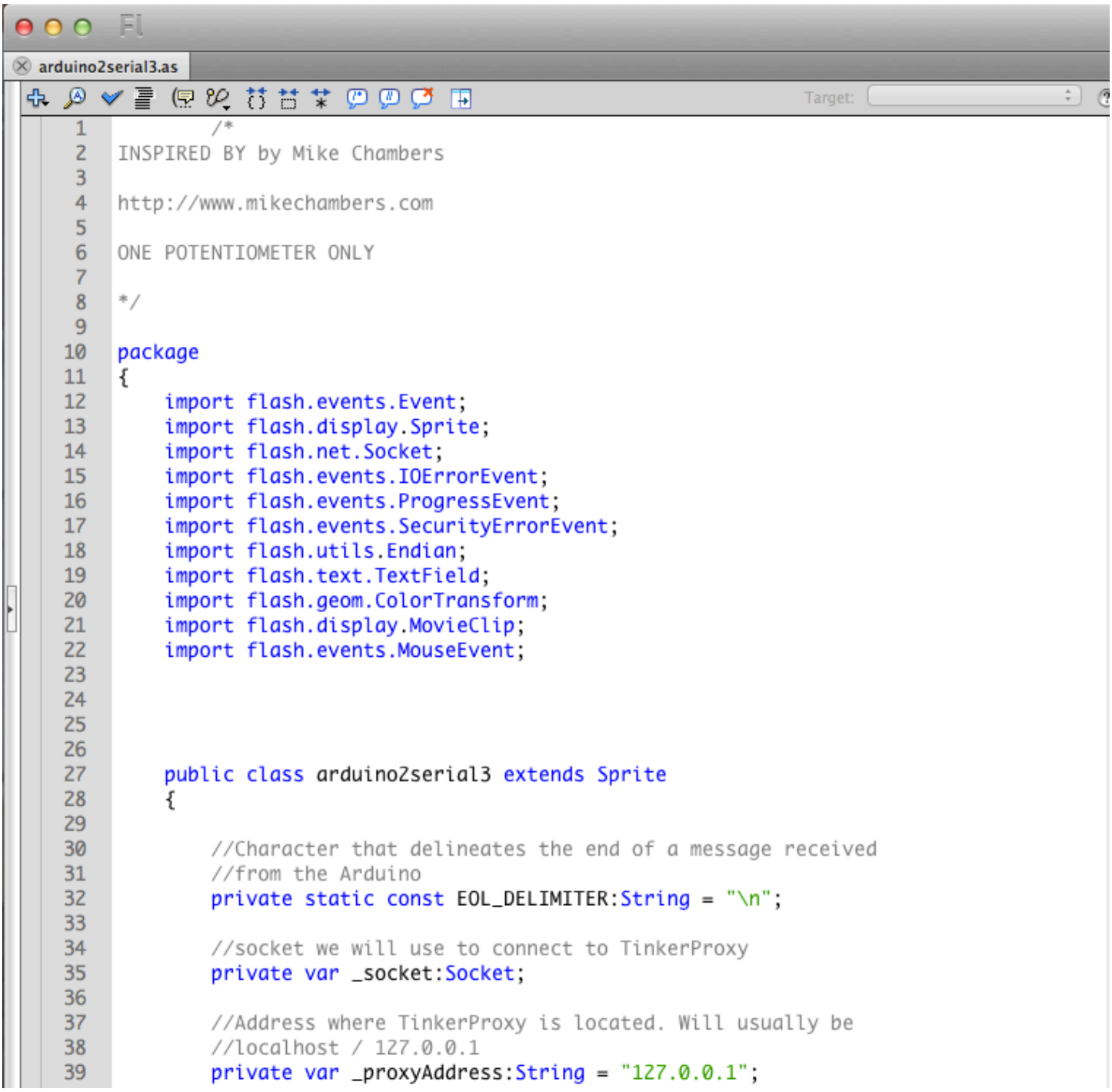

Flash File Edit View Control Debug Window Help
(34%) Fri 1:25 PM Patrick Ewing
arduino.swf

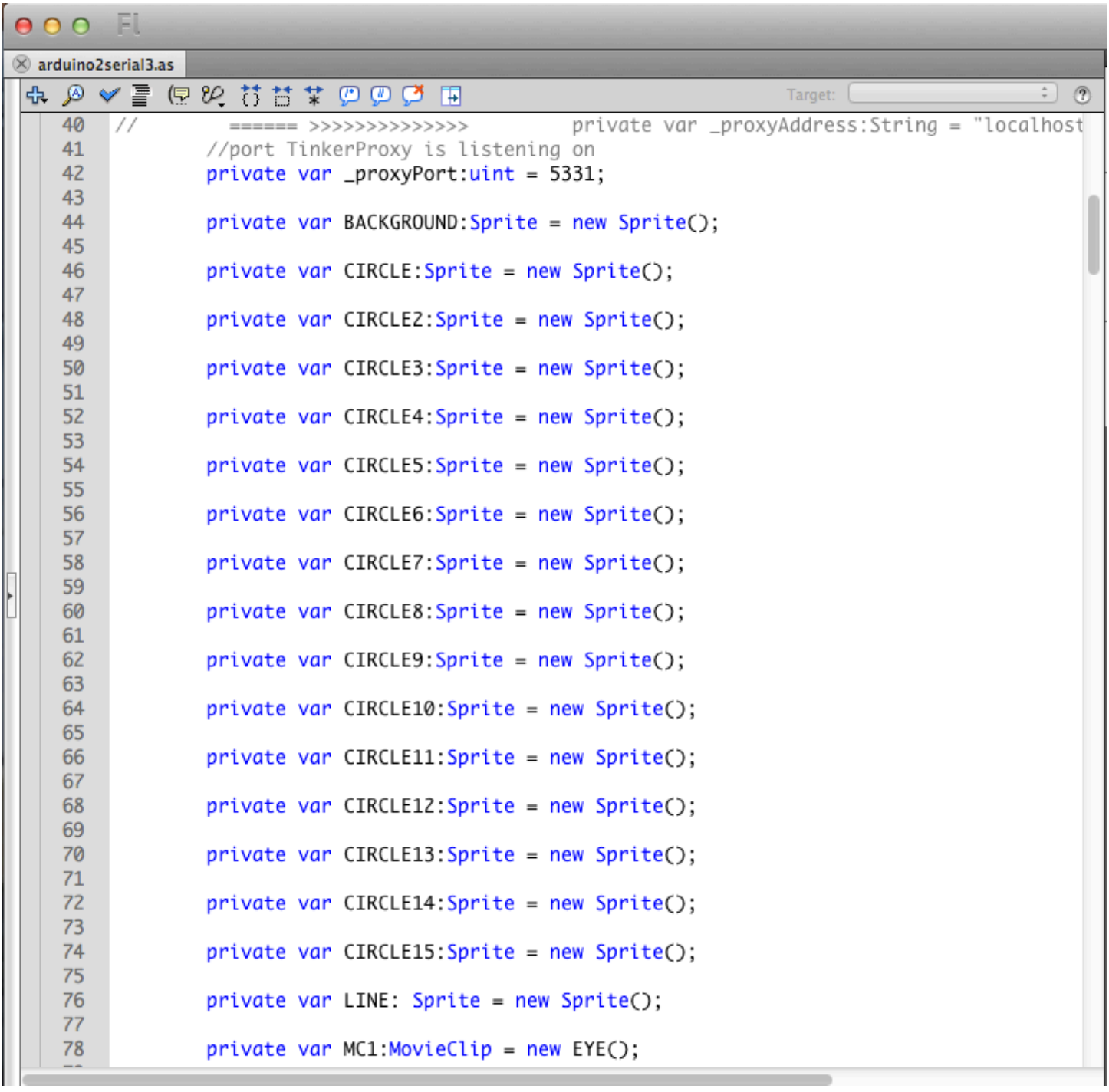
Wiring Diagram

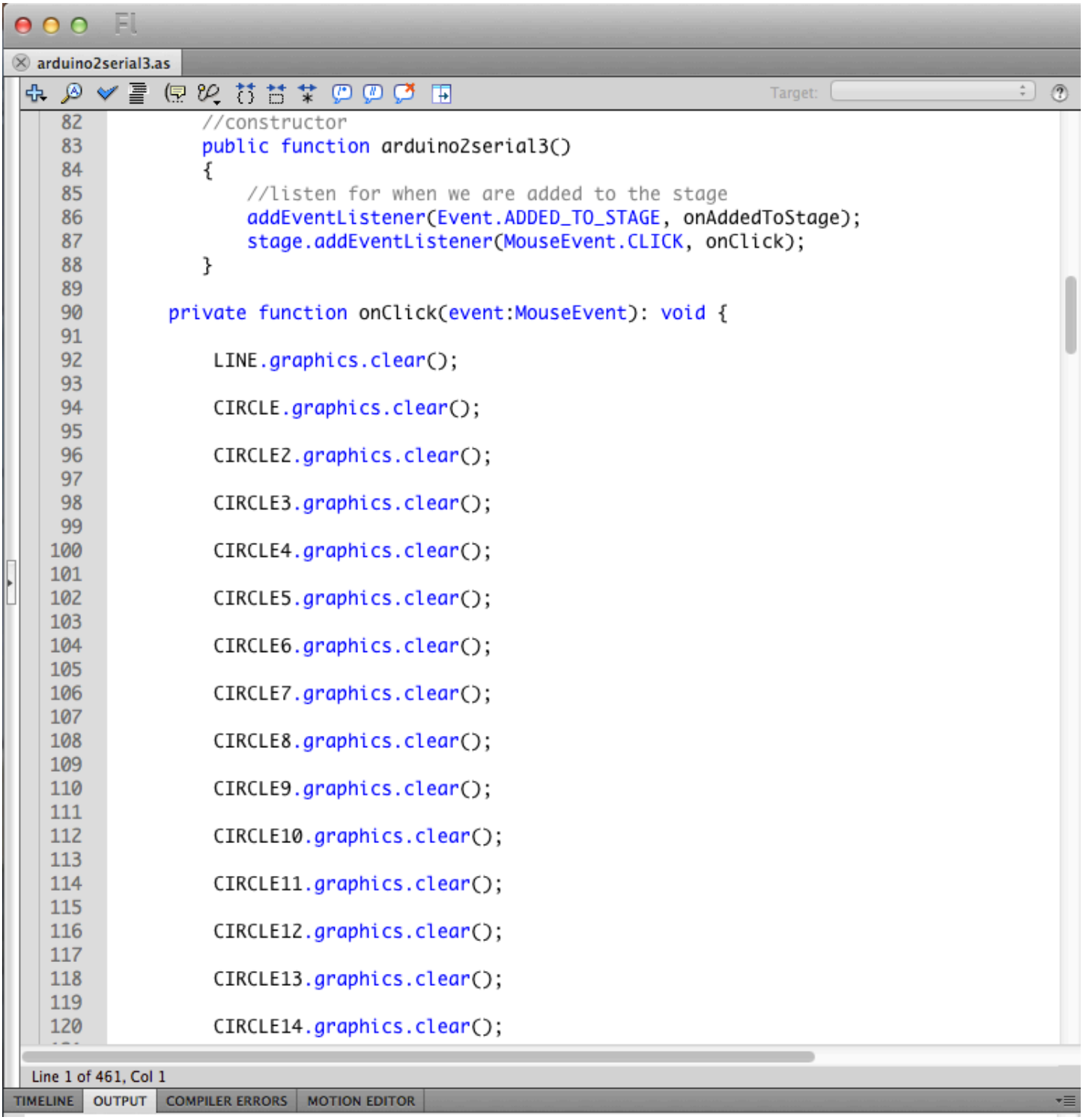


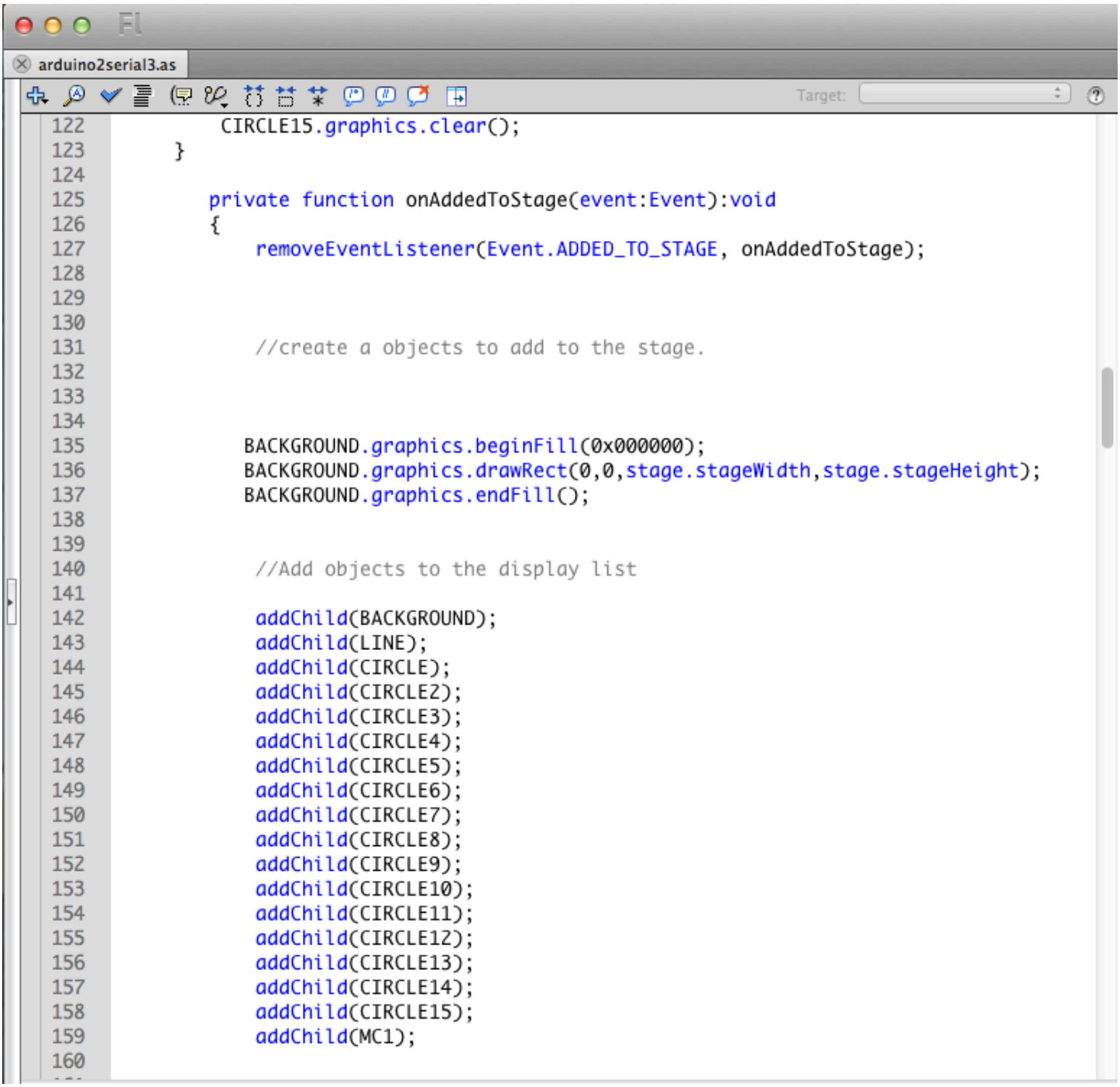
*Any of the Analog or Digital pins on the Arduino board could be used

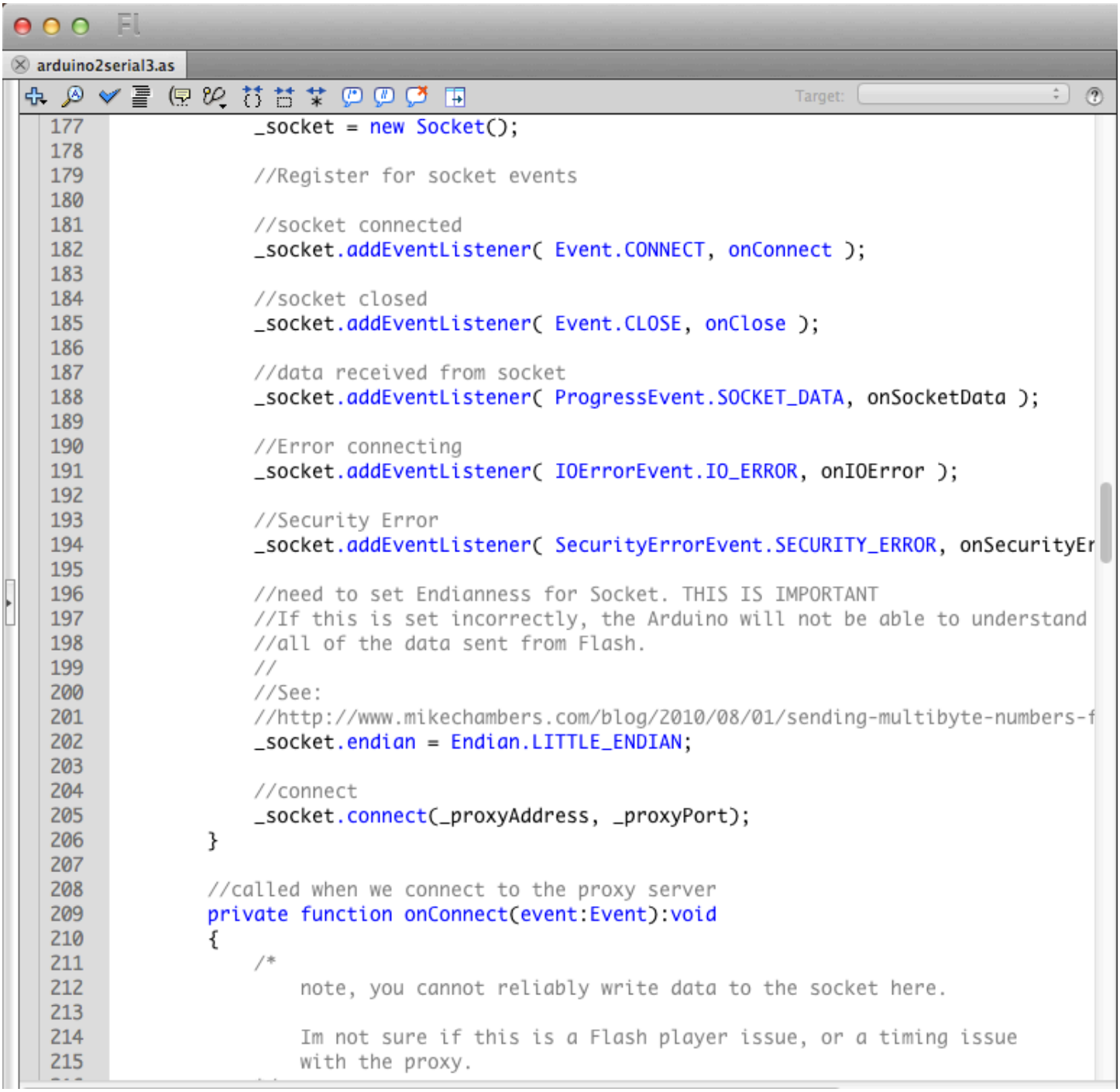

Senior_Project | Arduino 0022
Senior_Project
const int speakerPin = 10;
const int thereminPin = 3;
void setup() {
Serial.begin(9600);
}
void loop() {
int sensorValue = analogRead(thereminPin); //read the data recieved from the Theremin
if (sensorValue > 400) {
int frequency = map(sensorValue, 0, 255, 800, 720); //set the frequency equal to the recieved value
//int frequency = sensorValue;
int duration = 250; //how long the tone lasts
tone(speakerPin, frequency, duration); //play the tone
Serial.println(frequency); //print the value os it can be monitored
}
delay(50); //pause one second
}
1

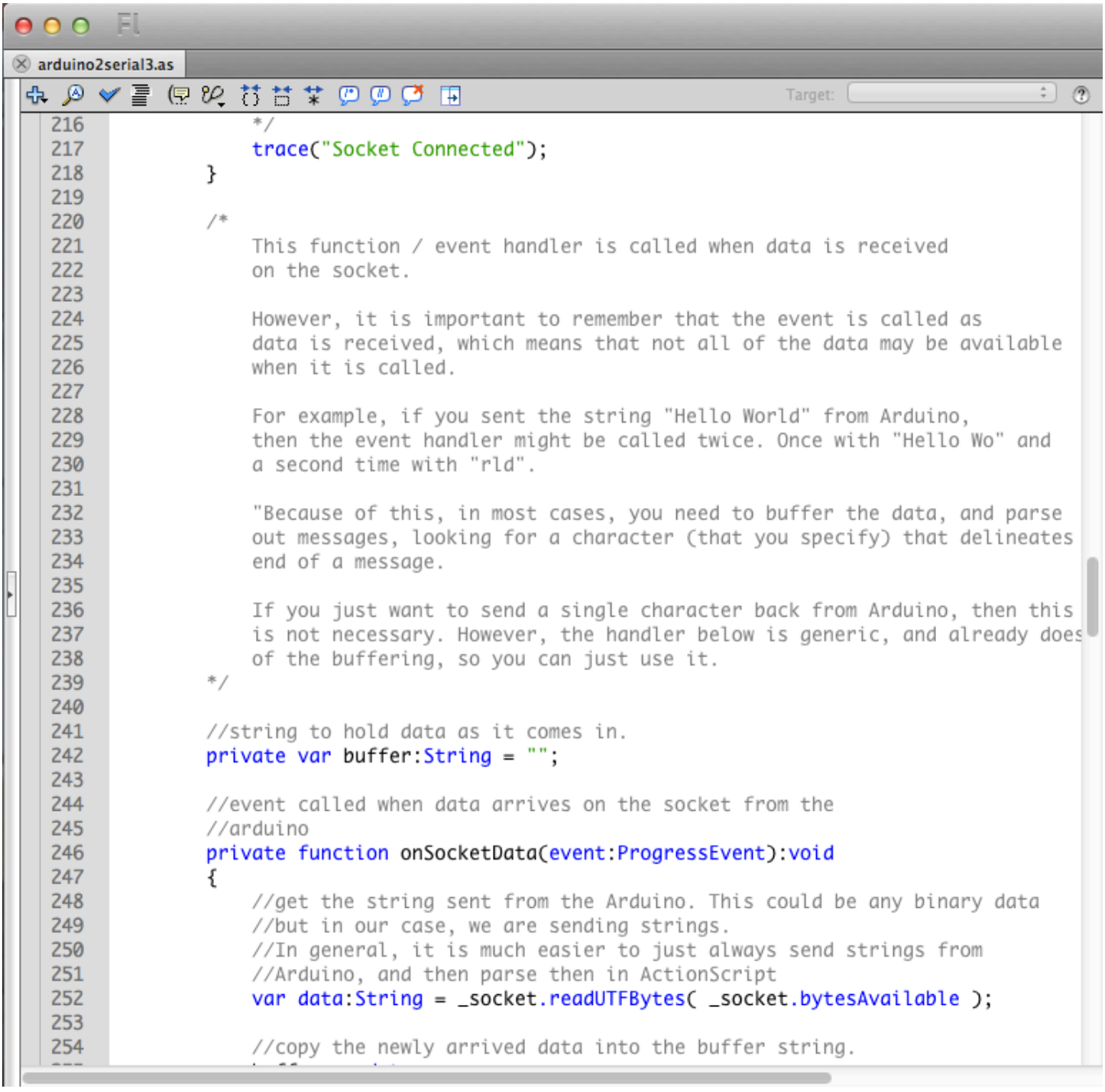

arduino2serial3.as
Target:
/*
INSPIRED BY by Mike Chambers
http://www.mikechambers.com
ONE POTENTIOMETER ONLY
*/
package
{
import flash.events.Event;
import flash.display.Sprite;
import flash.net.Socket;
import flash.events.IOErrorEvent;
import flash.events.ProgressEvent;
import flash.events.SecurityErrorEvent;
import flash.utils.Endian;
import flash.text.TextField;
import flash.geom.ColorTransform;
import flash.display.MovieClip;
import flash.events.MouseEvent;
public class arduino2serial3 extends Sprite
{
//Character that delineates the end of a message received
//from the Arduino
private static const EOL_DELIMITER:String = "\n";
//socket we will use to connect to TinkerProxy
private var _socket:Socket;
//Address where TinkerProxy is located. Will usually be
//localhost / 127.0.0.1
private var _proxyAddress:String = "127.0.0.1";


arduino2serial3.as
Target:
// ====== >>>>>>>>>>>>>> private var _proxyAddress:String = "localhost
//port TinkerProxy is listening on
private var _proxyPort:uint = 5331;
private var BACKGROUND:Sprite = new Sprite();
private var CIRCLE:Sprite = new Sprite();
private var CIRCLE2:Sprite = new Sprite();
private var CIRCLE3:Sprite = new Sprite();
private var CIRCLE4:Sprite = new Sprite();
private var CIRCLE5:Sprite = new Sprite();
private var CIRCLE6:Sprite = new Sprite();
private var CIRCLE7:Sprite = new Sprite();
private var CIRCLE8:Sprite = new Sprite();
private var CIRCLE9:Sprite = new Sprite();
private var CIRCLE10:Sprite = new Sprite();
private var CIRCLE11:Sprite = new Sprite();
private var CIRCLE12:Sprite = new Sprite();
private var CIRCLE13:Sprite = new Sprite();
private var CIRCLE14:Sprite = new Sprite();
private var CIRCLE15:Sprite = new Sprite();
private var LINE: Sprite = new Sprite();
private var MC1:MovieClip = new EYE();


arduino2serial3.as
Target:
//constructor
public function arduino2serial3()
{
//listen for when we are added to the stage
addEventListener(Event.ADDED_TO_STAGE, onAddedToStage);
stage.addEventListener(MouseEvent.CLICK, onClick);
}
private function onClick(event:MouseEvent): void {
LINE.graphics.clear();
CIRCLE.graphics.clear();
CIRCLE2.graphics.clear();
CIRCLE3.graphics.clear();
CIRCLE4.graphics.clear();
CIRCLE5.graphics.clear();
CIRCLE6.graphics.clear();
CIRCLE7.graphics.clear();
CIRCLE8.graphics.clear();
CIRCLE9.graphics.clear();
CIRCLE10.graphics.clear();
CIRCLE11.graphics.clear();
CIRCLE12.graphics.clear();
CIRCLE13.graphics.clear();
CIRCLE14.graphics.clear();
Line 1 of 461, Col 1
TIMELINE
OUTPUT
COMPILER ERRORS
MOTION EDITOR

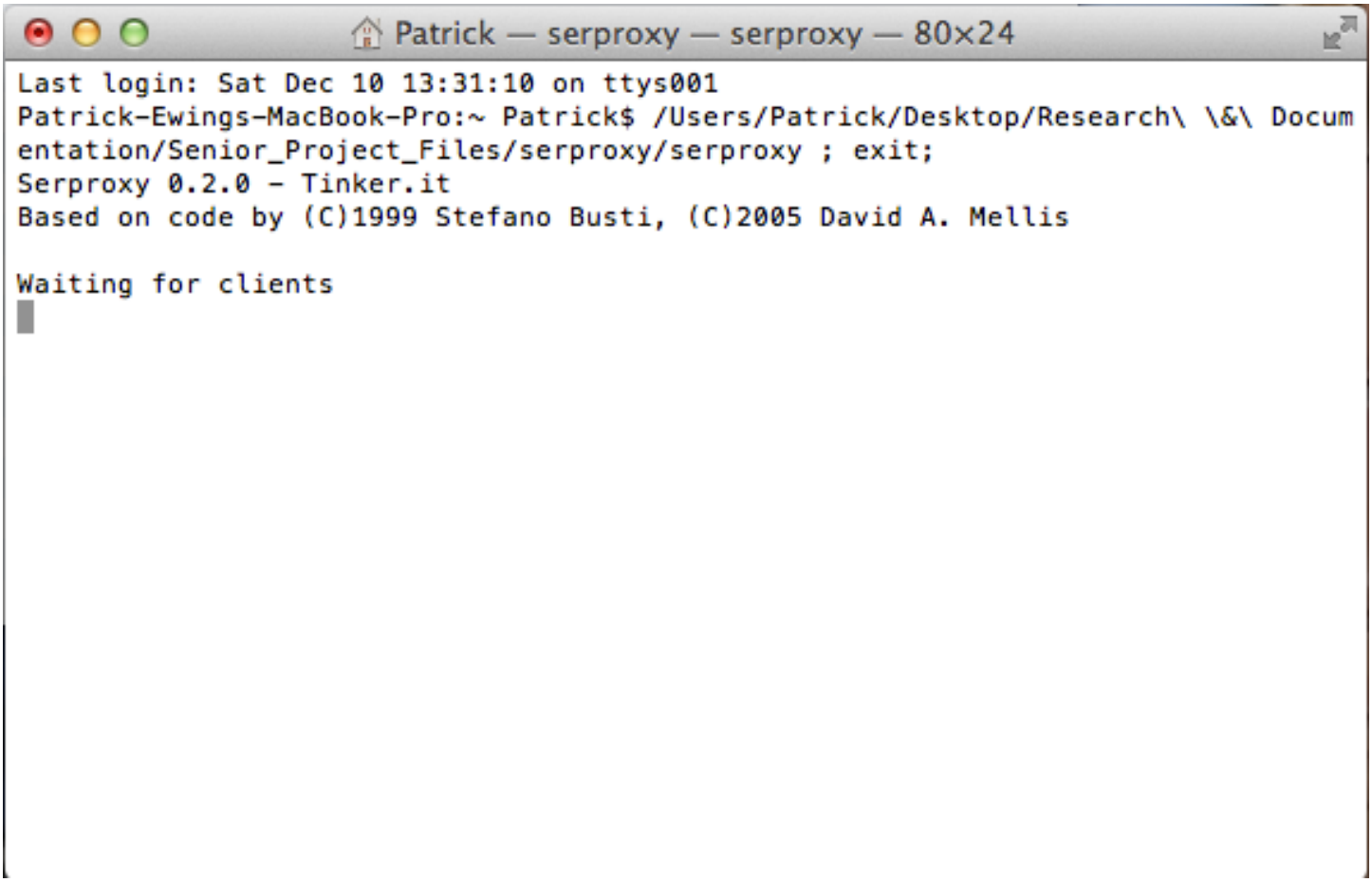

arduino2serial3.as
Target:
122 CIRCLE15.graphics.clear();
123 }
124
125 private function onAddedToStage(event:Event):void
126 {
127 removeEventListener(Event.ADDED_TO_STAGE, onAddedToStage);
128
129
130
131 //create a objects to add to the stage.
132
133
134
135 BACKGROUND.graphics.beginFill(0x000000);
136 BACKGROUND.graphics.drawRect(0,0,stage.stageWidth,stage.stageHeight);
137 BACKGROUND.graphics.endFill();
138
139
140 //Add objects to the display list
141
142 addChild(BACKGROUND);
143 addChild(LINE);
144 addChild(CIRCLE);
145 addChild(CIRCLE2);
146 addChild(CIRCLE3);
147 addChild(CIRCLE4);
148 addChild(CIRCLE5);
149 addChild(CIRCLE6);
150 addChild(CIRCLE7);
151 addChild(CIRCLE8);
152 addChild(CIRCLE9);
153 addChild(CIRCLE10);
154 addChild(CIRCLE11);
155 addChild(CIRCLE12);
156 addChild(CIRCLE13);
157 addChild(CIRCLE14);
158 addChild(CIRCLE15);
159 addChild(MC1);
160


arduino2serial3.as
Target:
177 _socket = new Socket();
178
179 //Register for socket events
180
181 //socket connected
182 _socket.addEventListener(Event.CONNECT, onConnect);
183
184 //socket closed
185 _socket.addEventListener(Event.CLOSE, onClose);
186
187 //data received from socket
188 _socket.addEventListener(ProgressEvent.SOCKET_DATA, onSocketData);
189
190 //Error connecting
191 _socket.addEventListener(IOErrorEvent.IO_ERROR, onIOError);
192
193 //Security Error
194 _socket.addEventListener(SecurityErrorEvent.SECURITY_ERROR, onSecurityEr
195
196 //need to set Endianness for Socket. THIS IS IMPORTANT
197 //If this is set incorrectly, the Arduino will not be able to understand
198 //all of the data sent from Flash.
199 //
200 //See:
201 //http://www.mikechambers.com/blog/2010/08/01/sending-multibyte-numbers-f
202 _socket.endian = Endian.LITTLE_ENDIAN;
203
204 //connect
205 _socket.connect(_proxyAddress, _proxyPort);
206 }
207
208 //called when we connect to the proxy server
209 private function onConnect(event:Event):void
210 {
211 /*
212 note, you cannot reliably write data to the socket here.
213
214 Im not sure if this is a Flash player issue, or a timing issue
215 with the proxy.


arduino2serial3.as
Target:
216 */
217 trace("Socket Connected");
218 }
219
220 /*
221 This function / event handler is called when data is received
222 on the socket.
223
224 However, it is important to remember that the event is called as
225 data is received, which means that not all of the data may be available
226 when it is called.
227
228 For example, if you sent the string "Hello World" from Arduino,
229 then the event handler might be called twice. Once with "Hello Wo" and
230 a second time with "rld".
231
232 "Because of this, in most cases, you need to buffer the data, and parse
233 out messages, looking for a character (that you specify) that delineates
234 end of a message.
235
236 If you just want to send a single character back from Arduino, then this
237 is not necessary. However, the handler below is generic, and already does
238 of the buffering, so you can just use it.
239 */
240
241 //string to hold data as it comes in.
242 private var buffer:String = "";
243
244 //event called when data arrives on the socket from the
245 //arduino
246 private function onSocketData(event:ProgressEvent):void
247 {
248 //get the string sent from the Arduino. This could be any binary data
249 //but in our case, we are sending strings.
250 //In general, it is much easier to just always send strings from
251 //Arduino, and then parse then in ActionScript
252 var data:String = _socket.readUTFBytes(_socket.bytesAvailable);
253
254 //copy the newly arrived data into the buffer string.

serproxy.cfg

```
# Config file for serproxy
# Original documentation not included you can download the original package at
# http://www.arduino.cc/en/Main/Software
#
# Change the settings below for your configuration

# Transform newlines coming from the serial port into nils
# if using Flash AS2 or AS1 : true
# if using Flash AS3 : false

newlines_to_nils=false

# Same number as in Serial.begin() in the Arduino code
# With StandardFirmata try 57600 or (older Arduino versions) 115200

comm_baud=9600
comm_databits=8
comm_stopbits=1
comm_parity=none

# Idle time out in seconds

timeout=300

# Different configuration for Mac and Windows
# Change the settings and make sure you comment everything
# of the OS you don't use.

# == WINDOWS =====================================================================
#
# Choose the COMport where Arduino is connected
# You can find this in the Arduino IDE Tool - Serialport
# Change the net_port number in the same number as the serialport number at the "comm_ports" line
# example : if comm_ports=3 -> net_port3=5331
# decomment below.............

#comm_ports=3
#net_port3=5331

# == OSX =========================================================================
#
# decomment ONE of the two lines below
# change the part marked with $ in the number of your device
# You can find the serial_device name in the Arduino IDE Tool - Serialport
# example Arduino Duemilanove and older:
#  serial_device1=/dev/cu.usbserial-A6008rZN
# example Arduino UNO:
#  serial_device1=/dev/cu.usbmodem641

serial_device1=/dev/cu.usbmodemfd131

# decomment below

comm_ports=1
net_port1=5331
```


Patrick — serproxy — serproxy — 80×24
Last login: Sat Dec 10 13:31:10 on ttys001
Patrick-Ewings-MacBook-Pro:~ Patrick$ /Users/Patrick/Desktop/Research\ \&\ Docum
entation/Senior_Project_Files/serproxy/serproxy ; exit;
Serproxy 0.2.0 - Tinker.it
Based on code by (C)1999 Stefano Busti, (C)2005 David A. Mellis
Waiting for clients