

# PingPong++

## *Instruction Manual*



## Contents

<b>0. Introduction</b>	<b>3</b>
<i>0.1 Required Materials</i>	<i>3</i>
<i>0.2 Software Requirements</i>	<i>3</i>
<i>0.3 Technical Overview</i>	<i>4</i>
<b>1. Hit Detection Circuitry</b>	<b>5</b>
<i>1.1 Theory</i>	<i>5</i>
<i>1.2 Parts</i>	<i>5</i>
<i>1.3 Piezo Preparation</i>	<i>6</i>
<i>1.4 Board Fabrication</i>	<i>7</i>
<b>Installation</b>	<b>7</b>
<i>2.1 Projector</i>	<i>7</i>
<i>2.2 Table</i>	<i>7</i>

<b><i>2.3 Connections</i></b>	<b><i>8</i></b>
<b><i>2.4 Arduino</i></b>	<b><i>8</i></b>
<b>3. Calibration</b>	<b>10</b>
<b>4. Running</b>	<b>12</b>
<b><i>4.1 Opening Koi Pond</i></b>	<b><i>13</i></b>
<b><i>4.2 Present Mode</i></b>	<b><i>13</i></b>
<b><i>4.3 Run Mode</i></b>	<b><i>14</i></b>

## 0. Introduction

PingPong++ is a ping pong table with projected visualizations that respond to locations of ball hits. Based on PingPongPlus, a Tangible Media Group project from 1998, PingPong++ was redesigned by Xiao Xiao and Michael Bernstein in spring 2010.

This document provides instructions for augmenting any standard ping pong table into PingPong++.

### 0.1 Required Materials

The following materials are necessary for PingPong++. The hit detection board and its parts are detailed in Section 1. Specific parts that we used in our installation are given in parentheses

- PingPong table
- Our custom circuit board (see Section 1)
- Arduino board (Decimila, Duemilanove or Uno.
- From Sparkfun: <http://www.sparkfun.com/products/11021>)
- Computer (Mac mini running OS X 10.6 Snow Leopard)
- USB cord to connect Arduino to computer
- Projector (Hitachi short throw CP-A100 placed on a pedestal)
- Gaffer tape to attach sensors to table



*Image: Arduino (left); Gaffer tape (middle); Our projector (right)*

### 0.2 Software Requirements

To run PingPong++, you will need to install the following:

- Arduino
- Processing
- Python 2.6
- serial, re, math, random, numpy, and scipy libraries for Python

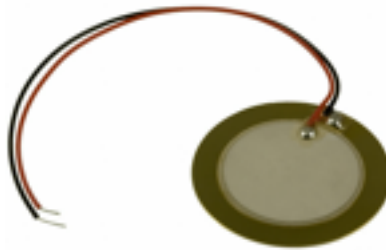
The PongPong++ software package will include the following

- pingpongplusplus.ino (the Arduino file in the pingpongplusplus folder)
- koi pond (Processing visualization)
- calibration (Processing visualization to help with calibration)
- training.py (Python program to for calibration calculations)

### 0.3 Technical Overview

PingPong++ detects locations of hits on the table using 4 piezoelectric buzzers taped to the underside of each side of the table. Timing differences between pairs of piezo elements are used to calculate where the ball hit.

Our circuit conditions the signal from the piezos, which allows the Arduino board connected to the computer to calculate timing differences. These timing differences are then passed to a Processing visualization running on the computer via the serial port. The Processing visualization is projected on the table.



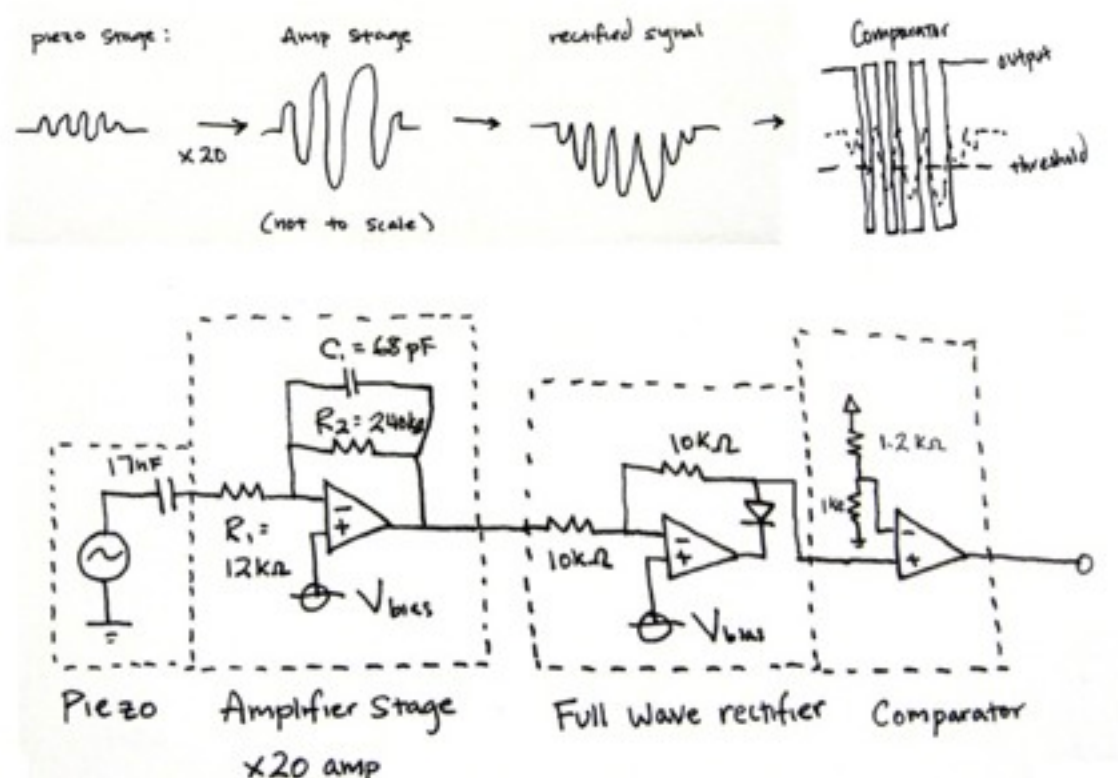
*Image: A piezo buzzer*

# 1. Hit Detection Circuitry

You will need to build a copy of our custom circuit for hit detection. Though we recommend using our PCB to simplify the process, We have used a breadboarded circuit and will provide instructions for those who prefer to build one.

## 1.1 Theory

The image below shows diagram of the circuit with capacitor and resistor values. The board includes 8 of this circuit, one for each piezo. The circuit works by first amplifying the piezo signal time 20. It then rectifies the signal so that the first “peaking” of the signal sets of the comparator to trigger a hit.



## 1.2 Parts

Through-hole components are given for breadboarding. Surface mount components are given for the packaging used in our PCB design.

- 8 piezo buzzers
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=102-1127-ND>
- 8 audio jacks (PCB only)
  - <http://www.sparkfun.com/products/8032>
- 8 audio cords (PCB only)
  - <http://www.sparkfun.com/products/8566>

- **8** 68uF ceramic capacitors
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=478-1314-2-ND> (surface mount)
- **6** TLV2374 Op-amps
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=296-12221-5-ND> (through hole)
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=296-11967-1-ND> (surface mount)
- **8** diodes 1N4148
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=1N4148FS-ND> (through hole)
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=1N4148WTCT-ND> (surface mount)
- **18** 10 K-Ohm, 1% tolerance resistors (**1% tolerance important**)
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=PPC10.0KYCT-ND> (through hole)
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=P10.0KCCT-ND> (surface mount)
- **8** 12 K-Ohm resistors
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=P12KBACT-ND> (through hole)
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=P12KADCT-ND> (surface mount)
- **8** 240 K-Ohm resistors
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=PPCQJ240KCT-ND> (through hole)
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=P240KACT-ND> (surface mount)
- **1** 1.2 K-Ohm resistor
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=P1.2KBACT-ND> (through hole)
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=P1.2KACT-ND> (surface mount)
- **1** 1 K-Ohm resistor
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=PPC1.0KW-1CT-ND> (through hole)
  - <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=541-1.0KACT-ND> (surface mount)
- Male headers

### 1.3 Piezo Preparation

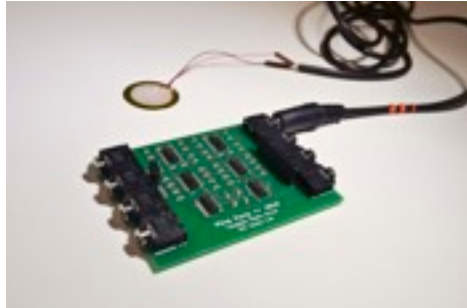
The piezos must be prepared so that they can connect to the board. For the breadboard, the two ends of the piezos can simply be soldered to two pieces of wires between 3-6 feet long.

For the PCB, each piezo must connect to one end of an audio cable. To do so, cut off the audio jack from one end and strip the cable to expose the wires within. Solder the red wire of the piezo to the red and white wires in the cable and the black wire of the piezo to the yellow wire in the cable.



## 1.4 Board Fabrication

Here is an image of the board with one piezo plugged in.



## Installation

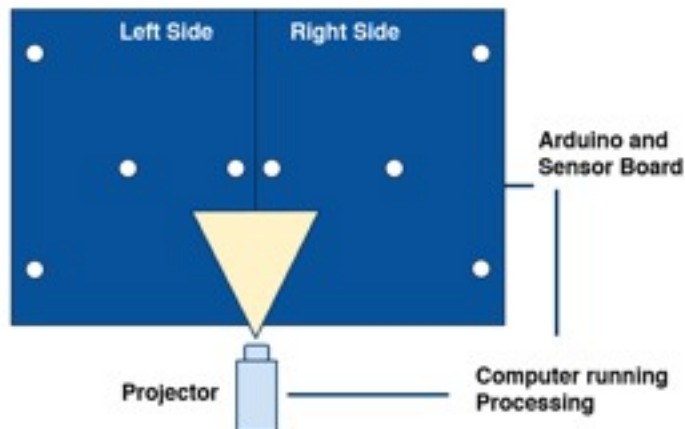
### 2.1 Projector

Our Hitachi projector is placed on a stand 5-feet (1.52m) tall so that it projects down onto the ping pong table. The distance between the stand and the table is about 3-feet but will need to be adjusted for specific situations.

Connect the projector to the computer and run the Calibration program in Processing. Position the projector such that the gray area of the program covers the entire ping pong table. The screen resolution of the computer may have to be adjusted for this.

### 2.2 Table

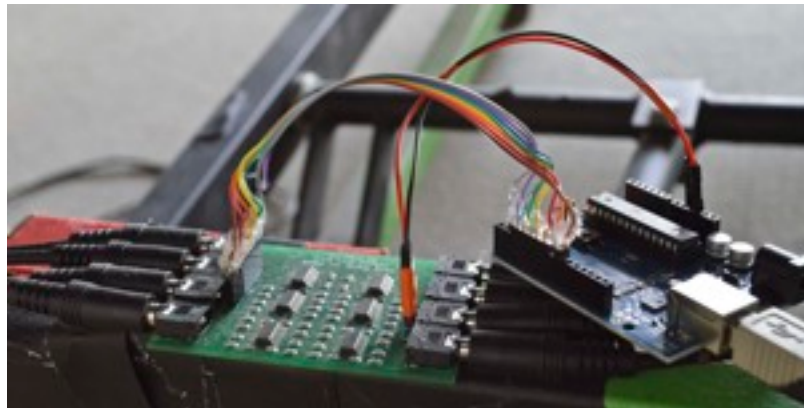
We attached piezos to the underside of the table using gaffer tape. 4 piezos are placed on each side in the configuration described in the figure. The piezo cables can be taped to the bottom of the table without interfering with the sensing and should be to avoid clutter. We ran the cables to one side of the table and attached them to legs of the table as in the image.





## 2.3 Connections

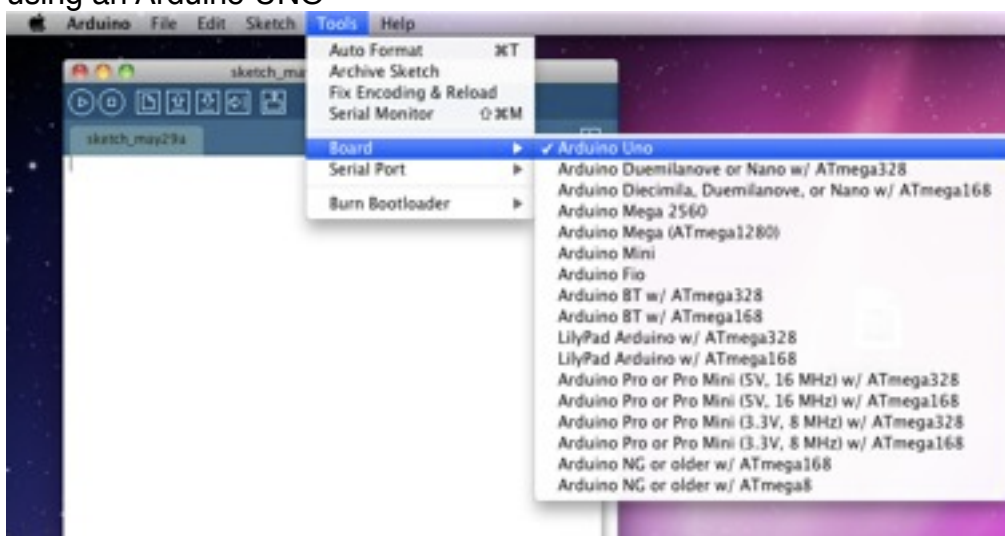
- Plug the sensor board into the Arduino according to the image
- Plug the 4 audio cables on the two sides of the table to the audio jacks on the two sides of the board. Make sure the left side and right side of the piezos and of the board match. Left and right are defined by the system configuration image.
- Connect the Arduino board and the computer with the USB cable



## 2.4 Arduino

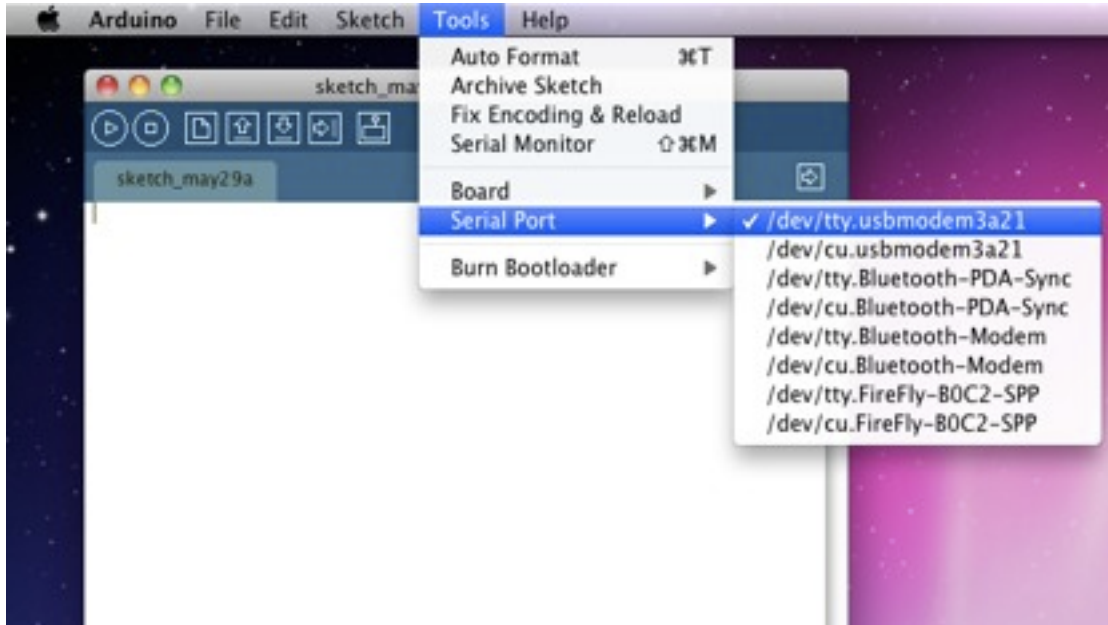
We need to verify that Arduino is receiving data from serial with the following steps. Your Arduino should be connected to your computer via USB when you do this.

1. Open the Arduino program
2. Go to Tools > Board and make sure the correct Arduino board is selected. Here we're using an Arduino UNO

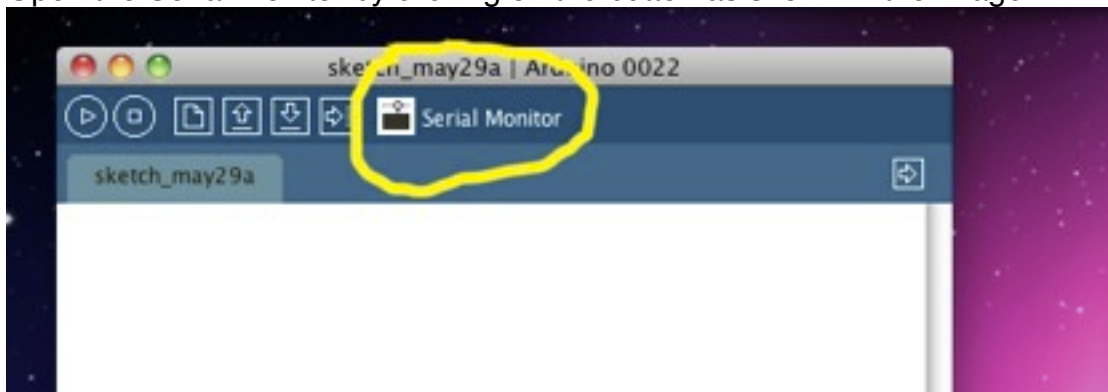




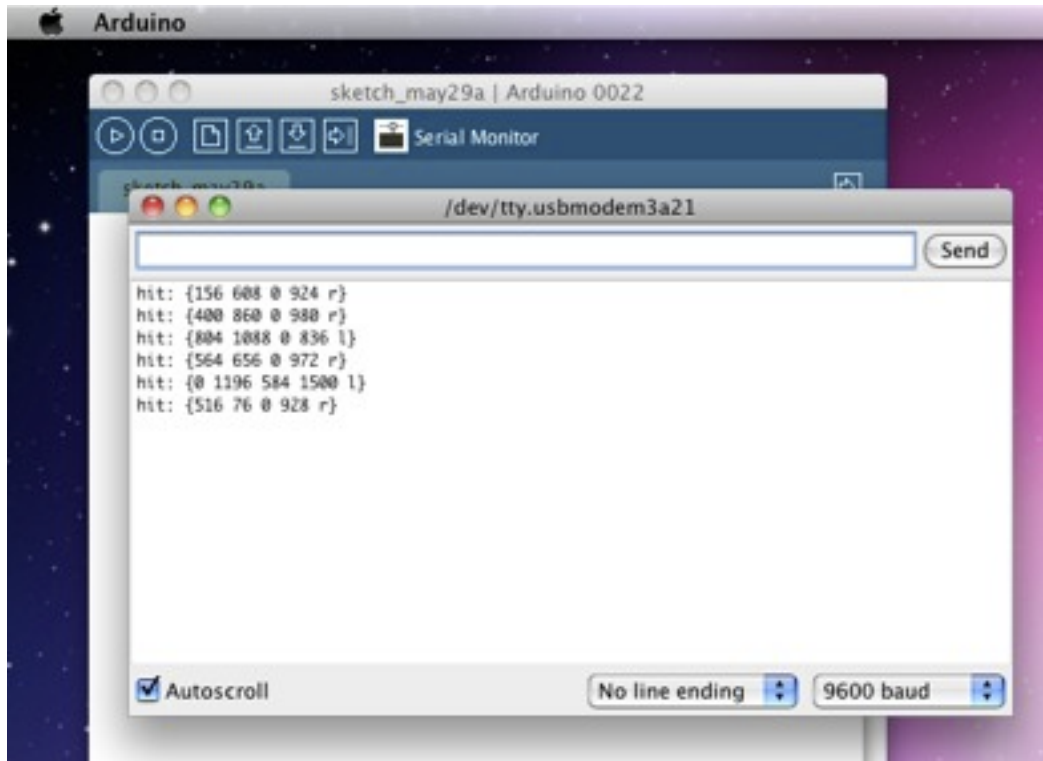
1. Go to Tools > Serial and select the first item. Write it down somewhere. You will need it later for calibration.



2. Open the Serial Monitor by clicking on the button as shown in the image



3. Hit the table a few times with the serial monitor open. Lines showing “hit” with some numbers should pop up. The “l” and “r” at the end indicate “left” and “right” side of the table.



If everything works according to described, your computer is now receiving hit data from the Arduino! If nothing show up on the serial monitor when you hit the table, something is probably not connected properly.

### 3. Calibration

The calibration works by asking you to drop the ball at 20 points on each side of the table. The Calibration visualization shows you where to drop the balls.

Open Processing and run the Calibration visualization. See section 4 for instructions on how to run a program in Processing.

On another screen, you will run the Python program training.py to calibrate. Before running it, open it in a text editor program like TextEdit and change the port to what you copied from the Serial setting in Arduino. In our case, we are going to replace the highlighted line in the image with the following:

```
PORT = '/dev/tty.usbmodem3a21' # FOR MAC
```

```

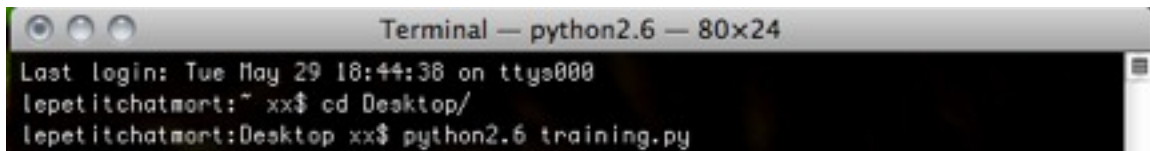
#
# (0,0)
# on right
# side
#
# x| 4 LL      2 UR| 6      8 |
#
# | 3 LR      | 5 |
#
# |-----|
#
# (0,0) on left side
# (left as your back is to the
# projector -- e.g., screen's left)

DEBUG = False
DEBUG_INPUT = 'hitdata'
FILE_OUTPUT = 'coefficients'

BAUD = 9600
#PORT = 6 # PORT = 5 means COM6. FOR WINDOWS
PORT = '/dev/tty.usbserial-A5002vax' # FOR MAC
SERIAL_TIMEOUT = .1 # in seconds
|

```

Now, open a terminal window and cd to where training.py is (we have it on the desktop). Type python2.6 training.py to run it.

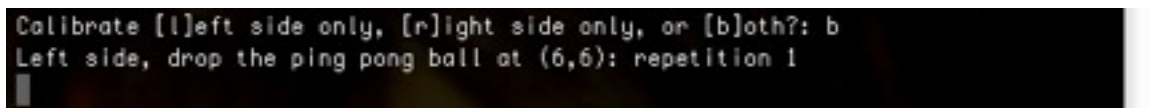


```

Terminal — python2.6 — 80x24
Last login: Tue May 29 18:44:38 on ttys000
lepetitchatmont:~ xx$ cd Desktop/
lepetitchatmont:Desktop xx$ python2.6 training.py

```

It will ask you whether you want to calibrate the left, right or both sides. Press the key for your selection and press enter. Then follow the directions the script gives you.



```

Calibrate [l]eft side only, [r]ight side only, or [b]oth?: b
Left side, drop the ping pong ball at (6,6): repetition 1
|

```

For each point, it will ask you to drop the ball 5 times. For best results, vary how hard you drop the ball for each of the 5 drops. Start with light drops and gradually increase the strength it hits the table.

At the end of 5 repetitions, it will ask whether you want to do that point again or move on. If your ball accidentally bounced in a wrong place, you can press “r” to repeat that point. Otherwise, hit enter to continue.

```
Terminal — Python — 80x24
pppp:~ tag-admin$ cd Desktop/
pppp:Desktop tag-admin$ python training.py
Calibrate [l]eft side only, [r]ight side only, or [b]oth?: b
Left side, drop the ping pong ball at (6,6): repetition 1
hit: {0 12 24 36 1}

{'four': 36, 'three': 24, 'two': 12, 'one': 0}
Left side, drop the ping pong ball at (6,6): repetition 2
hit: {0 16 24 40 1}

{'four': 40, 'three': 24, 'two': 16, 'one': 0}
Left side, drop the ping pong ball at (6,6): repetition 3
hit: {0 12 24 40 1}

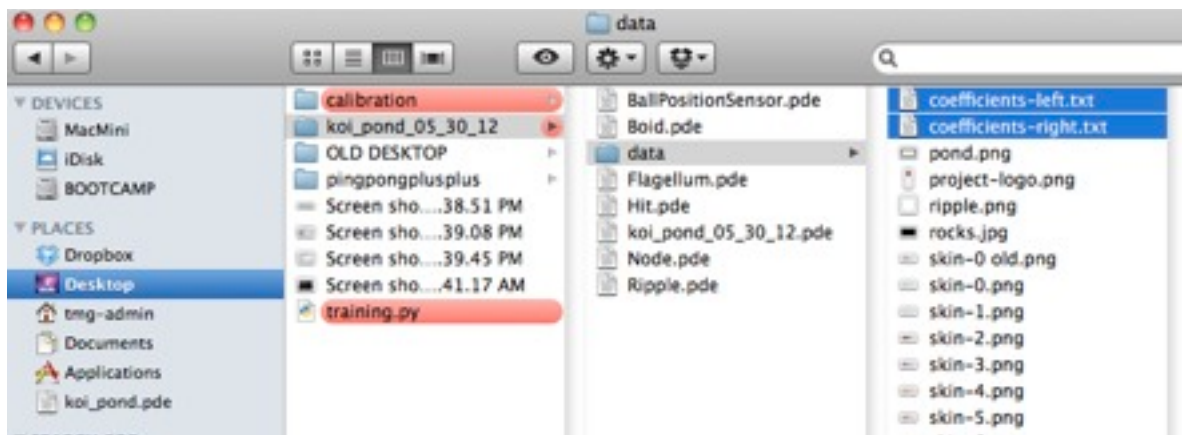
{'four': 40, 'three': 24, 'two': 12, 'one': 0}
Left side, drop the ping pong ball at (6,6): repetition 4
hit: {0 12 24 36 1}

{'four': 36, 'three': 24, 'two': 12, 'one': 0}
Left side, drop the ping pong ball at (6,6): repetition 5
hit: {0 16 24 40 1}

{'four': 40, 'three': 24, 'two': 16, 'one': 0}
Were all the tests OK? Press enter if OK, or enter 'r' to redo: 
```

The calibration script will generate two files: coefficients-left.txt and coefficients-right.txt in the same folder as training.py.

You will need to place these into koi pond > data, replacing the existing coefficients-left and coefficients-right files.

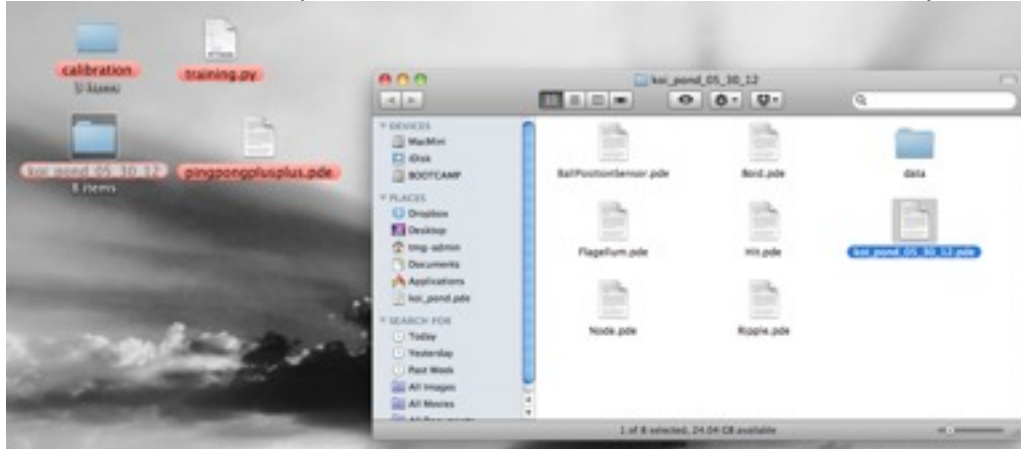


## 4. Running

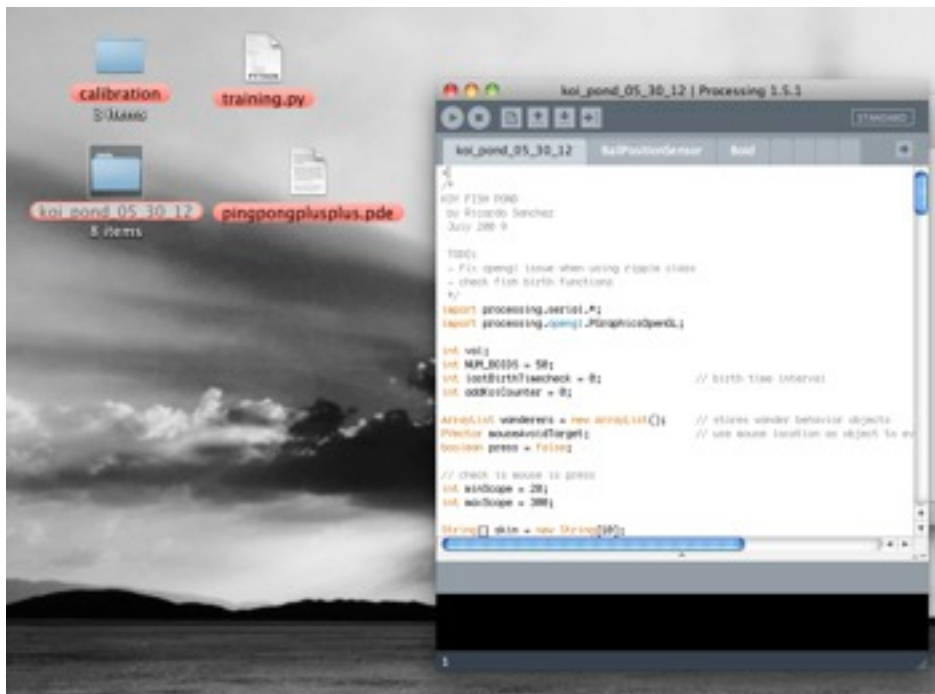
- \* Before you run any visualization that uses hit detection:
  - \* Make sure the Arduino with the custom circuit is connected to the computer
  - \* Make sure the serial monitor of Arduino is **CLOSED**

## 4.1 Opening Koi Pond

Double click the koi\_pond\_05\_30\_12 folder and double click koi\_pond\_05\_30\_12.pde



A Processing window will show up containing the code. You do not need to modify the code for



There are 2 ways of running a Processing application. **Run** mode and **Present** mode. **Run** mode displays the visualization in a window while **Present** mode is full screen. We recommend running visualizations in **Present** mode.

## 4.2 Present Mode

To run an application in present mode, go to *Sketch > Present*



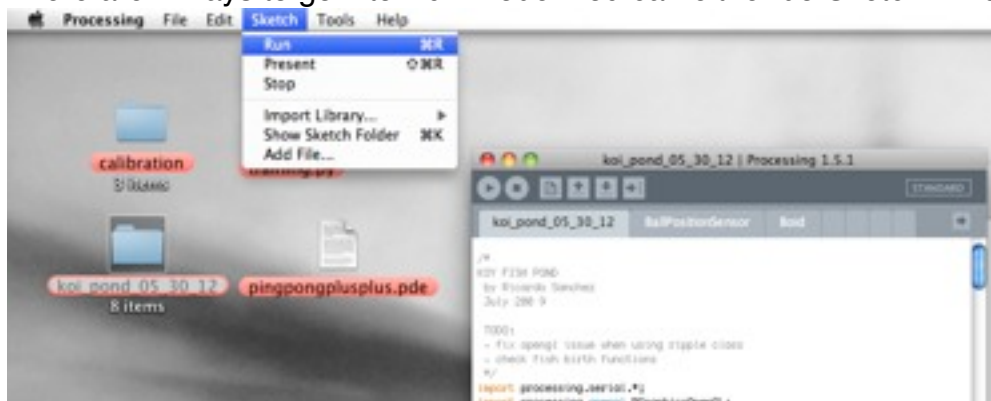
The visualization will be displayed in full screen mode. You can use the mouse to click on the screen to simulate ball hits to see the fish reacting.

To exit, either press *Esc* or click on the “stop” in the bottom left corner with the mouse.



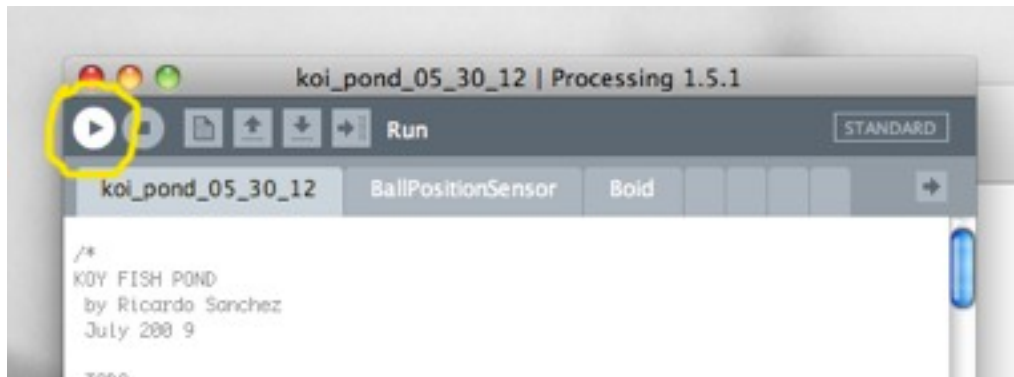
### 4.3 Run Mode

There are 2 ways to go into Run mode. You can either do Sketch > Run:



Or click on the circled button:





Either way, the application will run in a separate window. You can close the window to close the application.

