

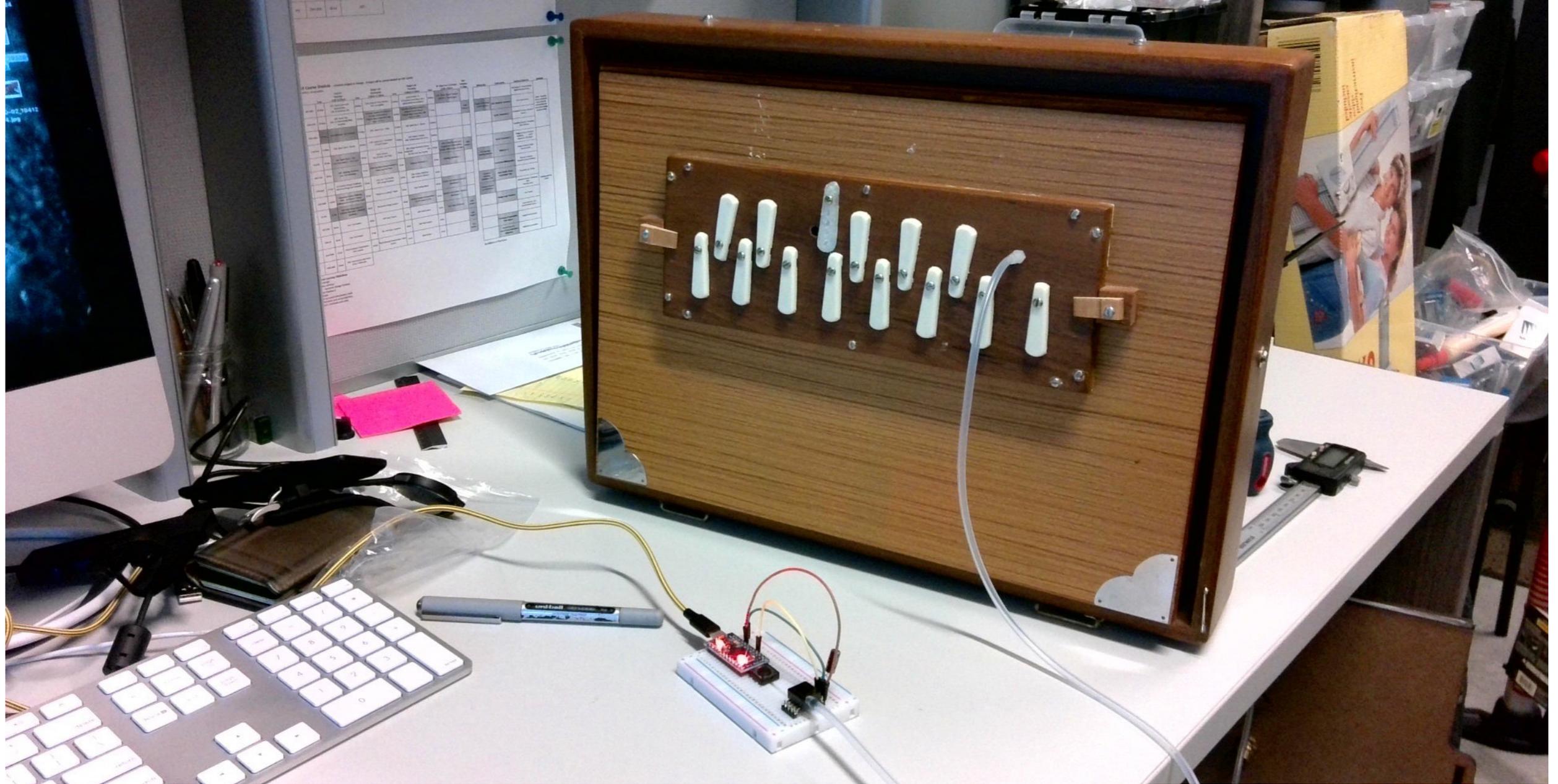


---

# GRAVITY

*UWTMC*

*Matt Borland - 2019*



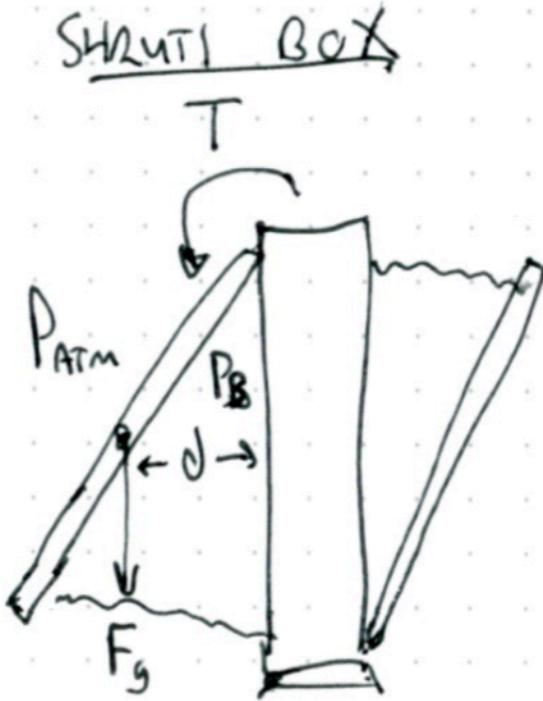
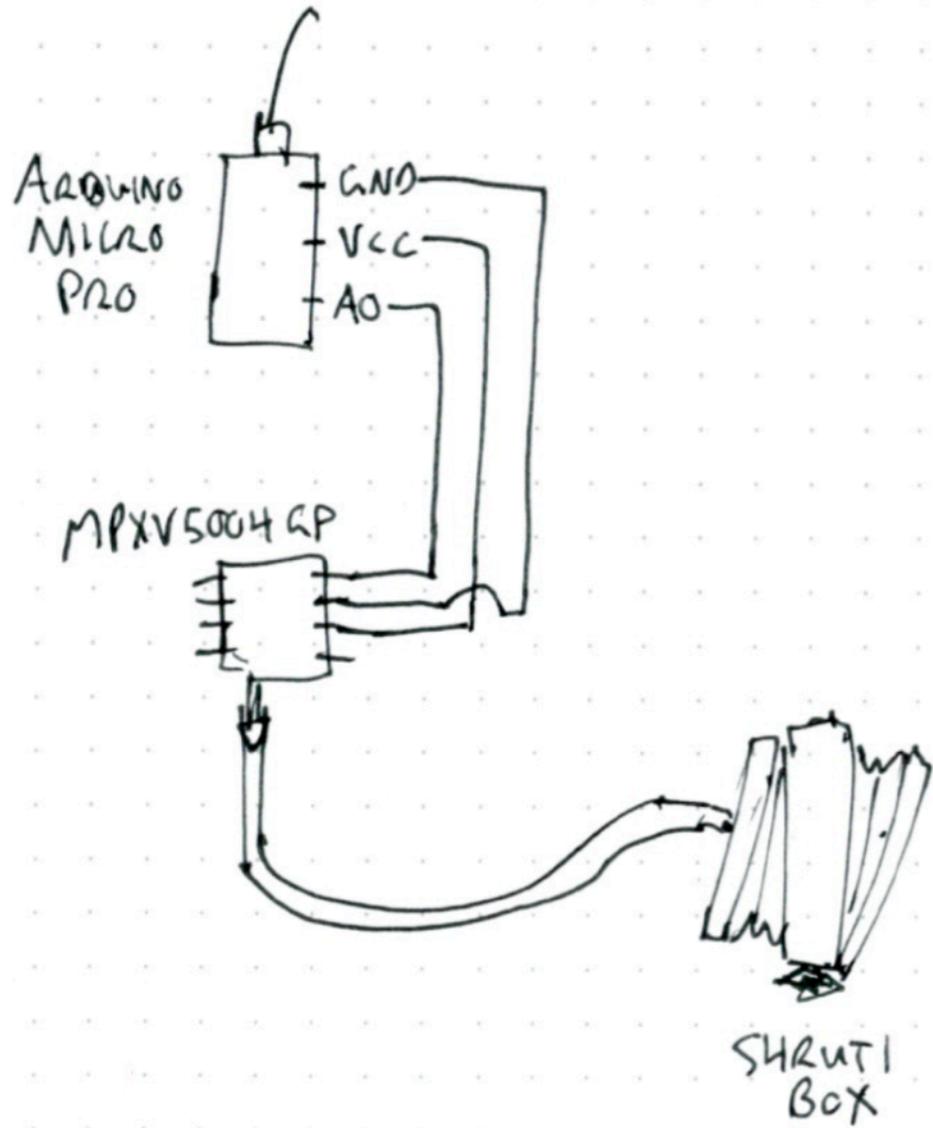
## WE'RE USING GRAVITY TO CREATE MUSIC

---

*We'll use sensors to measure change in pressure as a function of gravitational force, then convert those readings to MIDI messages which your computer will then turn into musical sounds!*

*We'll also simulate gravity and make music on our computers.*

# UWTML - GRAVITY

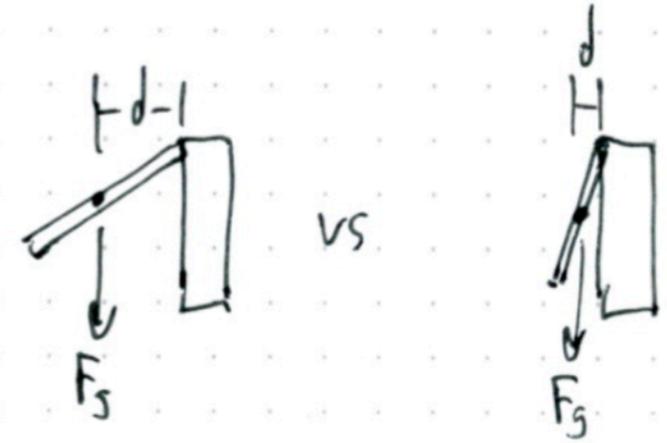


$$T = F_g d$$

→ THE APPLICATION OF TORQUE CREATES A PRESSURE DIFFERENCE BETWEEN THE AIR INSIDE AND OUTSIDE THE BELLOWS.

$$P_{SENSOR} = P_B - P_{ATM}$$

MB  
2019



→ THE AMOUNT OF APPLIED TORQUE IS A FUNCTION OF THE ANGLE OF THE BELLOWS PLATE.

THIS IS THEN MAPPED TO CONTROL THE VOLUME OF OUR SYNTHESIZER.

# DOWNLOADS

---

*You need three pieces of software. All are free and multi-platform!*

Download the Arduino IDE



The screenshot shows the Arduino IDE download page. On the left is the Arduino logo, a teal circle with a white infinity symbol containing a minus and a plus sign. To its right, the text reads: **ARDUINO 1.8.10**. Below this, it says: "The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions." On the right side of the page, there are download links for Windows (installer and ZIP), Mac OS X, and Linux (32-bit and 64-bit). At the bottom, there are links for Release Notes, Source Code, and Checksums (sha512).



*Arduino IDE - a software platform used to program your microcontroller.*

<https://www.arduino.cc/en/Main/Software>

*Helm - a software synthesizer to make musical sounds with your computer.*

<https://tytel.org/helm/>

# DOWNLOADS

---



3.5.3 (3 February 2019)

[Windows](#) 64-bit  
[Windows](#) 32-bit

[Linux](#) 64-bit  
[Linux](#) 32-bit  
[Linux](#) ARM  
([running on Pi?](#))

[Mac OS X](#)

*Processing IDE - a software platform used to create simple programs.*

<https://processing.org/download/>

# FILES

---

Workshops this term are at CML! All sessions run 3:30-5:30pm at Critical Media Lab, located in Communitech, 151 Charles St. W., Kitchener.

Sept. 18th: Deformation - MIDI Balloons: [DeformationFiles.zip](#)

Oct. 2nd: Gravity - Shruti Box: [GravityFiles.zip](#)



Oct. 23rd: Continuity - Seaboard and Bop Pad

Nov. 6th: Complexity - Modular Synthesis

Nov. 20th: Exploration - Co-play Patch Tables

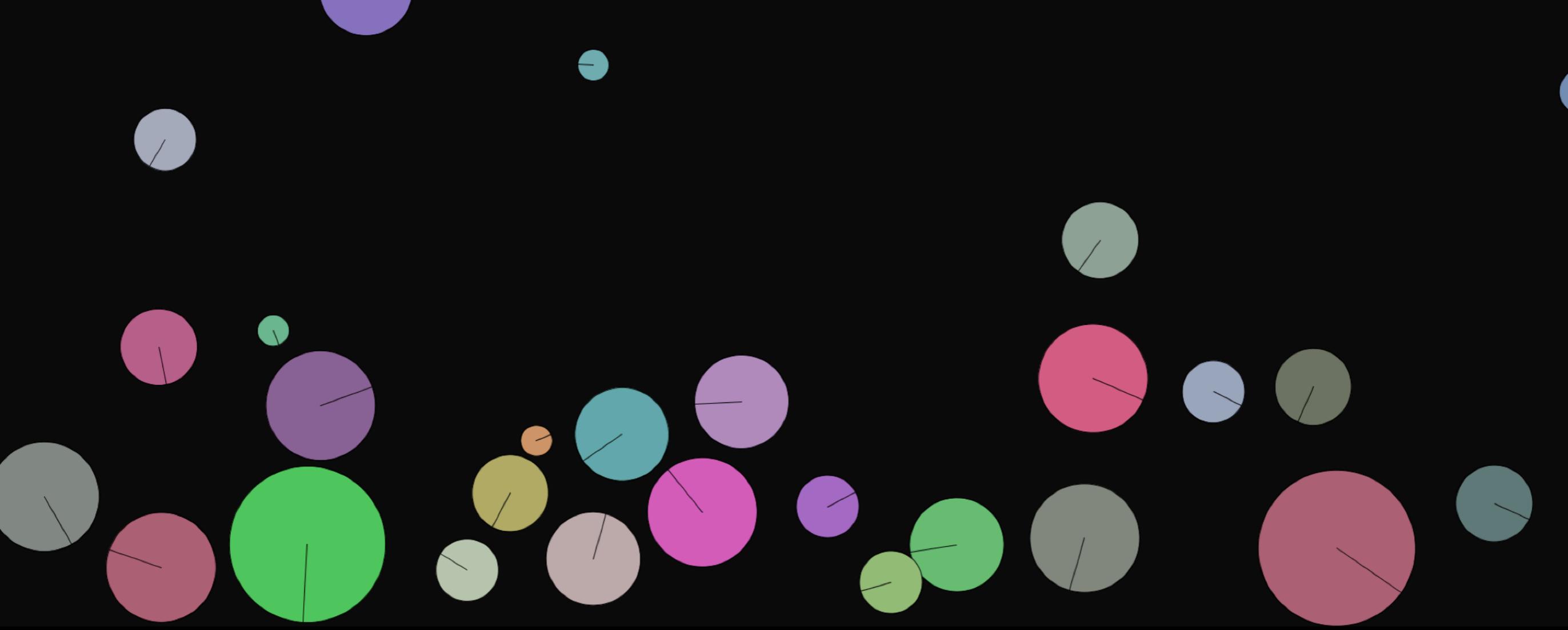


Files are available as a ZIP at <https://uwatmc.com>

LET'S DO IT!

---

*Matt will explain everything, so follow along with him on the big screen. Ask questions if you're unsure - other people are probably in the same boat.*



---

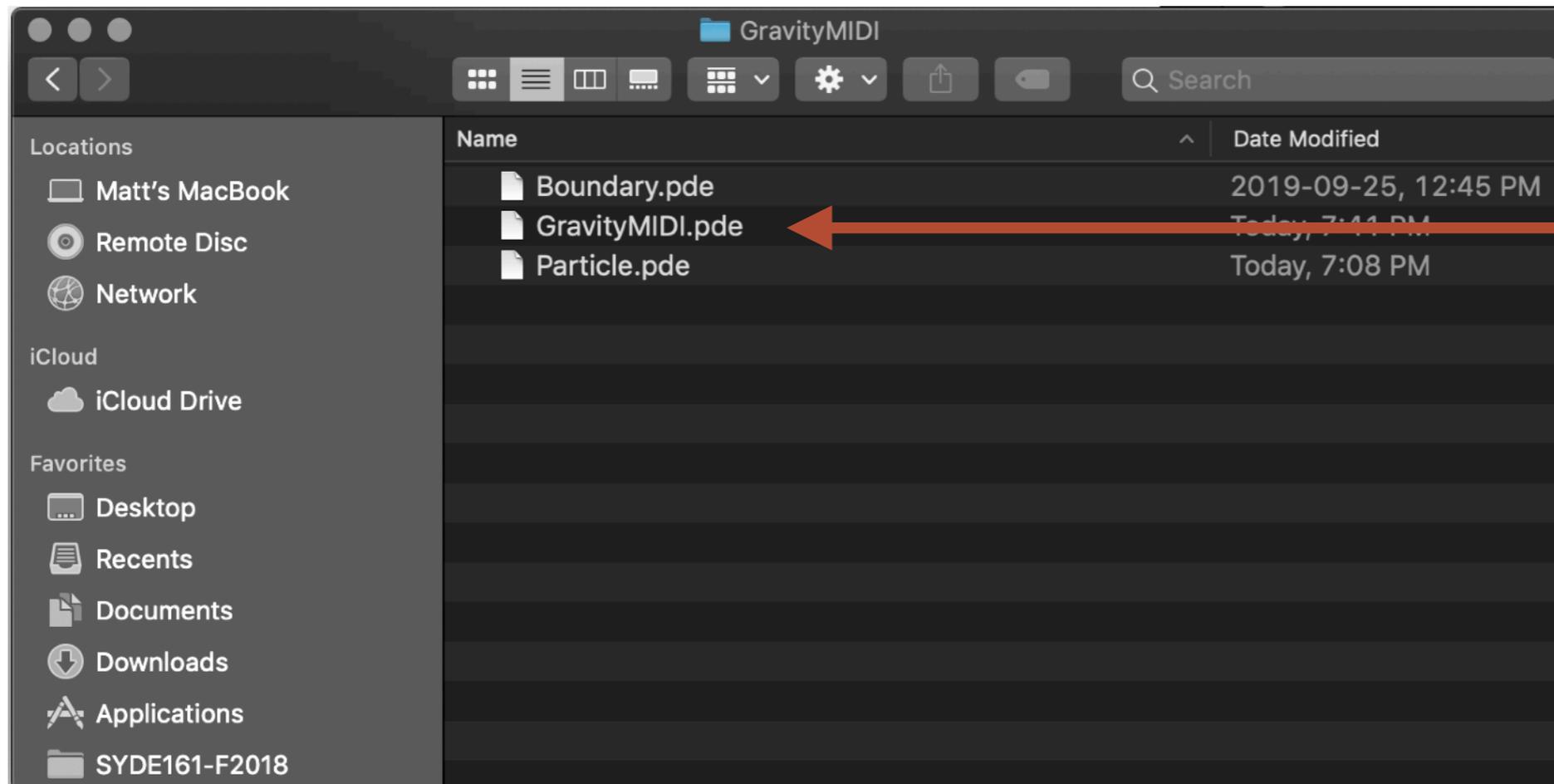
# SIMULATING GRAVITY

*UWTMC*

*Matt Borland - 2019*

# SIMULATING GRAVITY

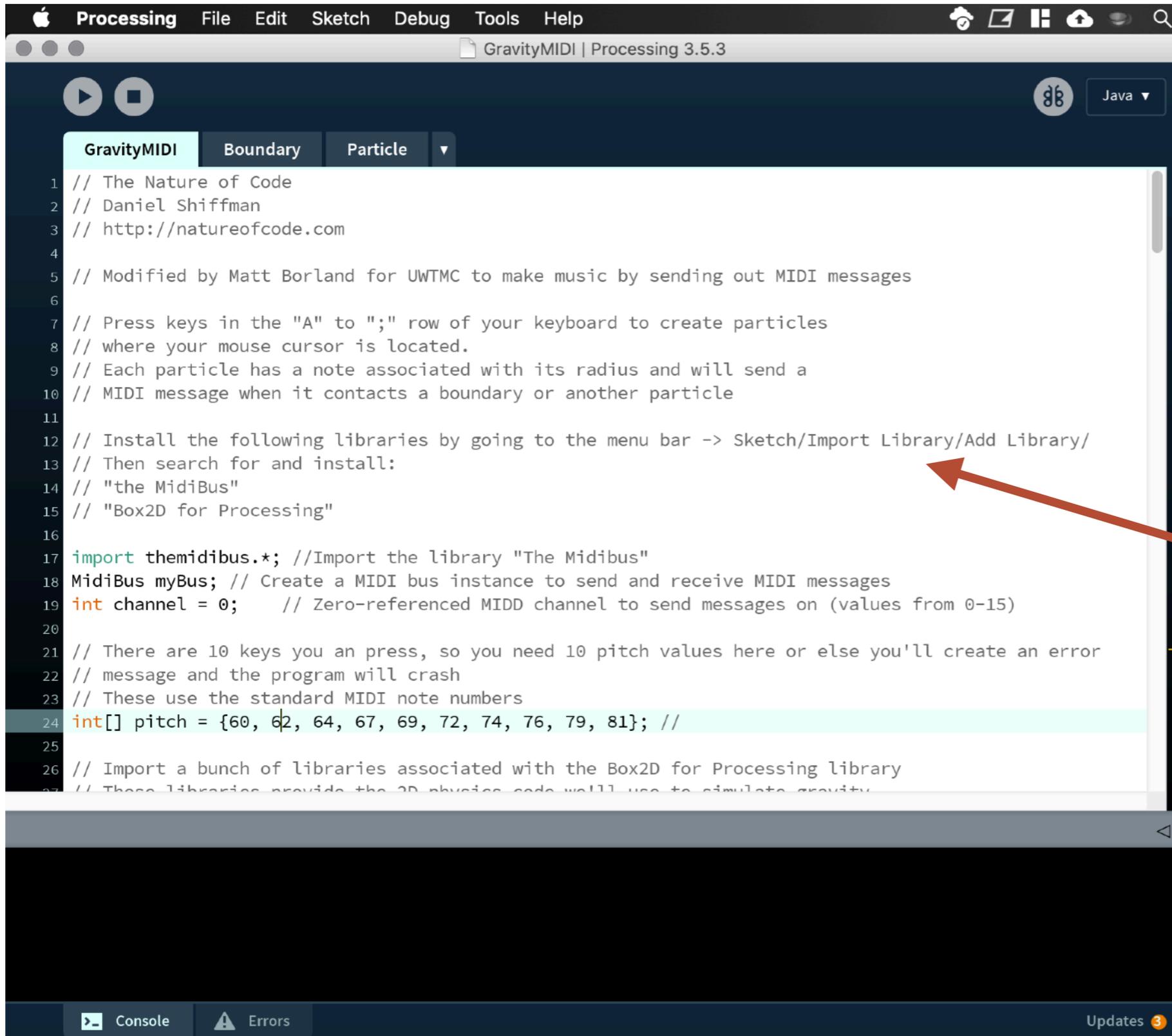
---



*Open “GravityMIDI.pde”, the other files will automatically load.*

# SIMULATING GRAVITY - INSTALL LIBRARIES

---



```
1 // The Nature of Code
2 // Daniel Shiffman
3 // http://natureofcode.com
4
5 // Modified by Matt Borland for UWTMC to make music by sending out MIDI messages
6
7 // Press keys in the "A" to ";" row of your keyboard to create particles
8 // where your mouse cursor is located.
9 // Each particle has a note associated with its radius and will send a
10 // MIDI message when it contacts a boundary or another particle
11
12 // Install the following libraries by going to the menu bar -> Sketch/Import Library/Add Library/
13 // Then search for and install:
14 // "the MidiBus"
15 // "Box2D for Processing"
16
17 import themidibus.*; //Import the library "The Midibus"
18 MidiBus myBus; // Create a MIDI bus instance to send and receive MIDI messages
19 int channel = 0; // Zero-referenced MIDD channel to send messages on (values from 0-15)
20
21 // There are 10 keys you an press, so you need 10 pitch values here or else you'll create an error
22 // message and the program will crash
23 // These use the standard MIDI note numbers
24 int[] pitch = {60, 62, 64, 67, 69, 72, 74, 76, 79, 81}; //
25
26 // Import a bunch of libraries associated with the Box2D for Processing library
27 // These libraries provide the 2D physics code we'll use to simulate gravity
```

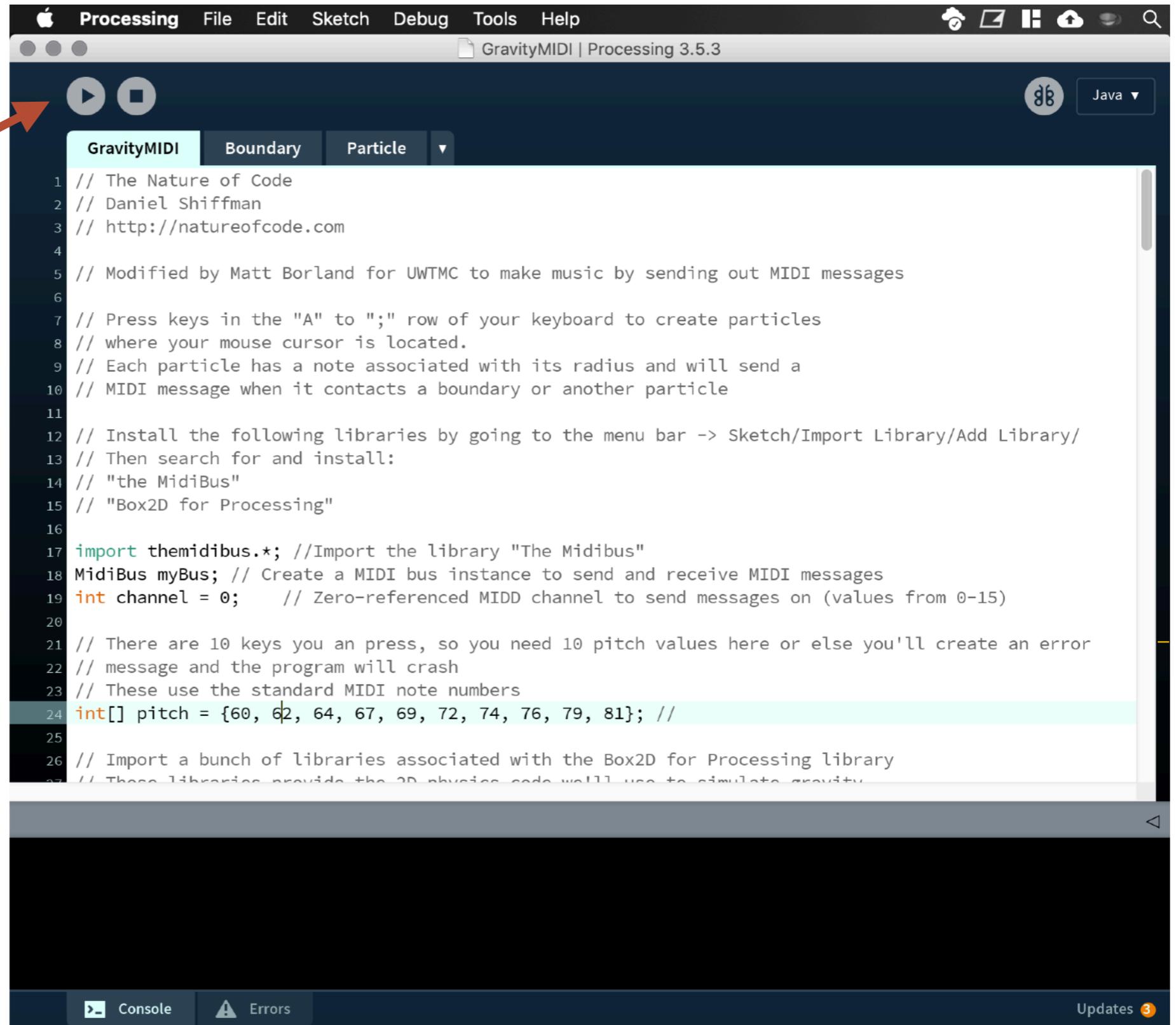
*Follow these instructions to install the required libraries.*

# SIMULATING GRAVITY – RUN THE CODE

---

*Press the run button to try to run the code.*

*It's not going to work because we need to set your midi port, but you need to run it to see what midi ports are available.*



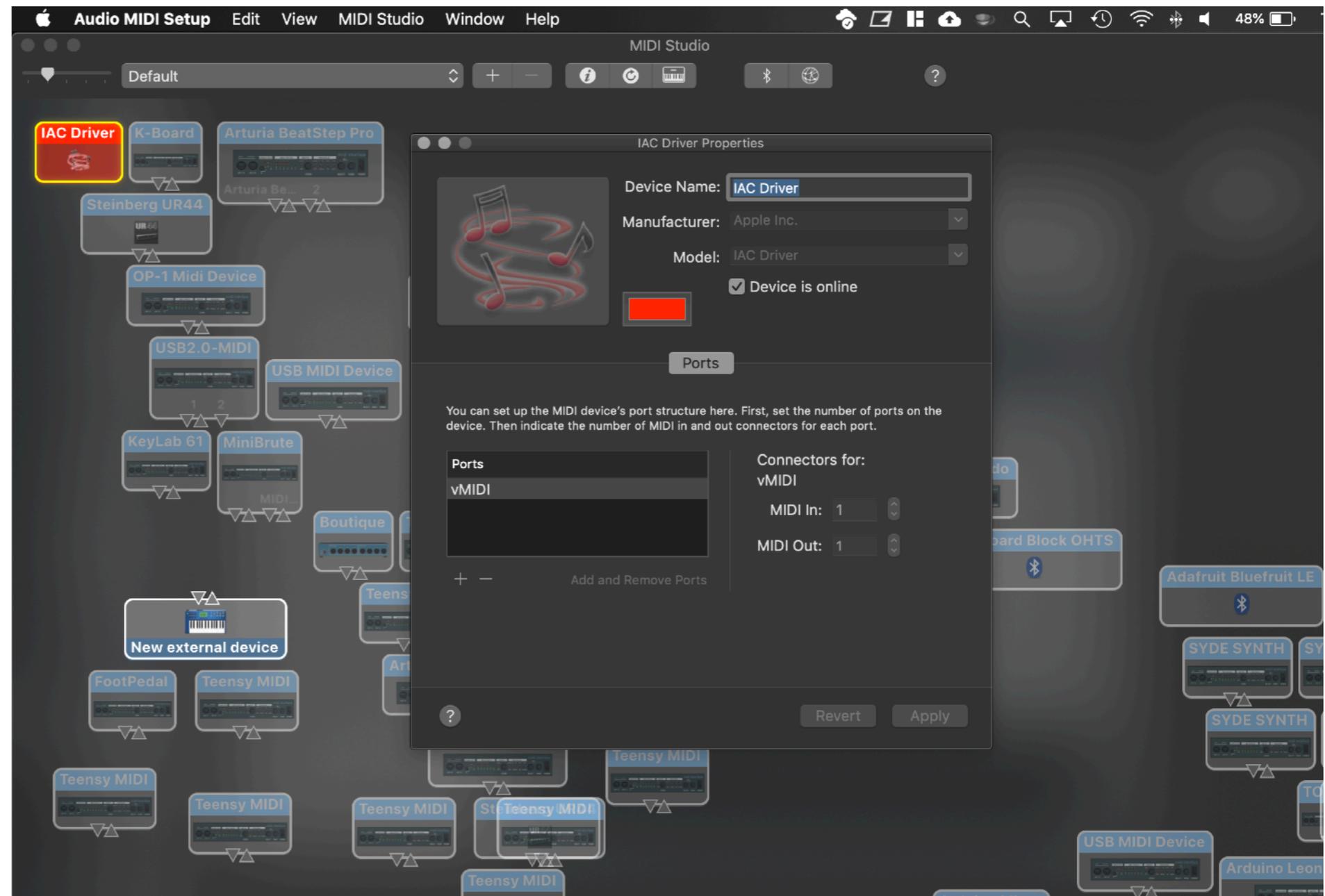
```
Processing File Edit Sketch Debug Tools Help
GravityMIDI | Processing 3.5.3

GravityMIDI Boundary Particle
1 // The Nature of Code
2 // Daniel Shiffman
3 // http://natureofcode.com
4
5 // Modified by Matt Borland for UWTMC to make music by sending out MIDI messages
6
7 // Press keys in the "A" to ";" row of your keyboard to create particles
8 // where your mouse cursor is located.
9 // Each particle has a note associated with its radius and will send a
10 // MIDI message when it contacts a boundary or another particle
11
12 // Install the following libraries by going to the menu bar -> Sketch/Import Library/Add Library/
13 // Then search for and install:
14 // "the MidiBus"
15 // "Box2D for Processing"
16
17 import themidibus.*; //Import the library "The Midibus"
18 MidiBus myBus; // Create a MIDI bus instance to send and receive MIDI messages
19 int channel = 0; // Zero-referenced MIDD channel to send messages on (values from 0-15)
20
21 // There are 10 keys you can press, so you need 10 pitch values here or else you'll create an error
22 // message and the program will crash
23 // These use the standard MIDI note numbers
24 int[] pitch = {60, 62, 64, 67, 69, 72, 74, 76, 79, 81}; //
25
26 // Import a bunch of libraries associated with the Box2D for Processing library
27 // These libraries provide the 2D physics code we'll use to simulate gravity
```

# USING GRAVITY – VIRTUAL MIDI PORT ON MAC

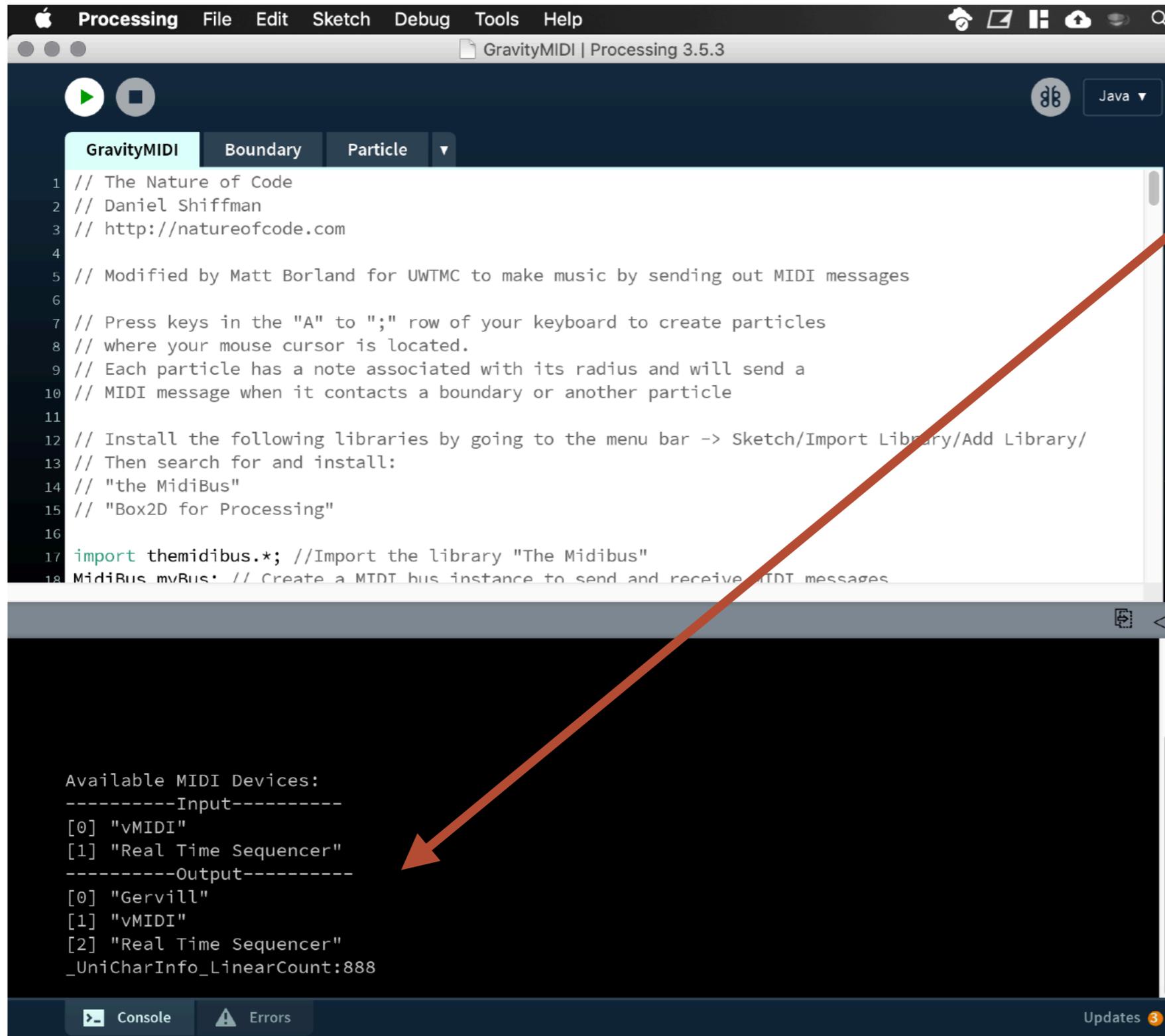
---

*On Mac OS you may have to turn on a virtual MIDI port. Open “Audio MIDI Setup” to get a window that lists all of your MIDI instruments. Double click “IAC Driver”, and make sure the “device is online” box is checked.*



*Make a note of the port name - this is used in the Processing code.*

# SIMULATING GRAVITY – SET MIDI PORT



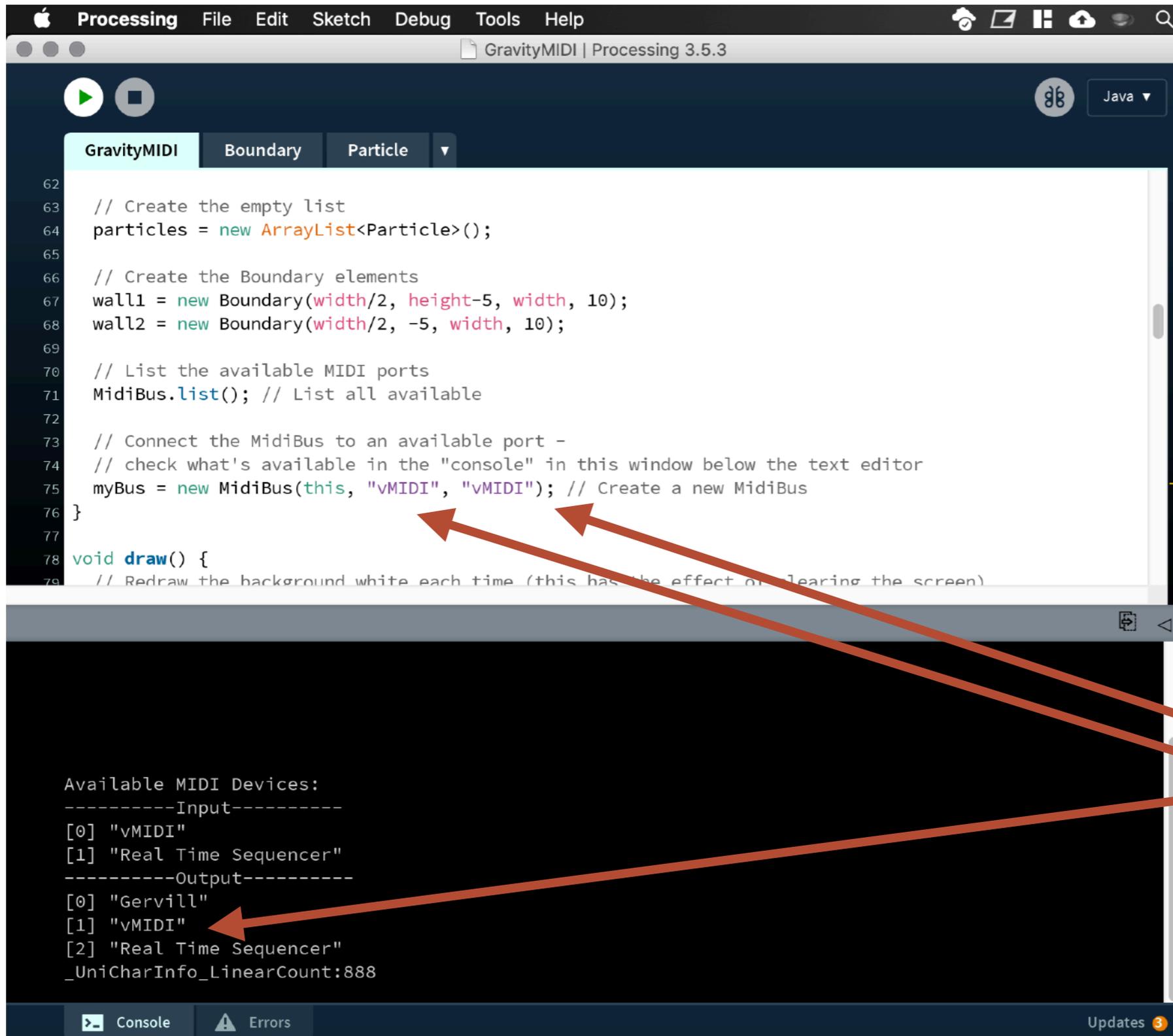
```
Processing File Edit Sketch Debug Tools Help
GravityMIDI | Processing 3.5.3
GravityMIDI Boundary Particle
1 // The Nature of Code
2 // Daniel Shiffman
3 // http://natureofcode.com
4
5 // Modified by Matt Borland for UWTMC to make music by sending out MIDI messages
6
7 // Press keys in the "A" to ";" row of your keyboard to create particles
8 // where your mouse cursor is located.
9 // Each particle has a note associated with its radius and will send a
10 // MIDI message when it contacts a boundary or another particle
11
12 // Install the following libraries by going to the menu bar -> Sketch/Import Library/Add Library/
13 // Then search for and install:
14 // "the MidiBus"
15 // "Box2D for Processing"
16
17 import themidibus.*; //Import the library "The Midibus"
18 MidiBus myBus; // Create a MIDI bus instance to send and receive MIDI messages

Available MIDI Devices:
-----Input-----
[0] "vMIDI"
[1] "Real Time Sequencer"
-----Output-----
[0] "Gervill"
[1] "vMIDI"
[2] "Real Time Sequencer"
_UniCharInfo_LinearCount:888
Console Errors Updates 3
```

*The MIDI device name here is used in the setup code for the program, so make note of what options you have.*

*Your options will be different than mine!*

# SIMULATING GRAVITY – SET MIDI PORT



The screenshot shows the Processing IDE interface. The top menu bar includes 'Processing', 'File', 'Edit', 'Sketch', 'Debug', 'Tools', and 'Help'. The window title is 'GravityMIDI | Processing 3.5.3'. The code editor shows the following code:

```
62
63 // Create the empty list
64 particles = new ArrayList<Particle>();
65
66 // Create the Boundary elements
67 wall1 = new Boundary(width/2, height-5, width, 10);
68 wall2 = new Boundary(width/2, -5, width, 10);
69
70 // List the available MIDI ports
71 MidiBus.list(); // List all available
72
73 // Connect the MidiBus to an available port -
74 // check what's available in the "console" in this window below the text editor
75 myBus = new MidiBus(this, "vMIDI", "vMIDI"); // Create a new MidiBus
76 }
77
78 void draw() {
79 // Redraw the background white each time (this has the effect of clearing the screen)
```

The console window at the bottom displays the following output:

```
Available MIDI Devices:
-----Input-----
[0] "vMIDI"
[1] "Real Time Sequencer"
-----Output-----
[0] "Gervill"
[1] "vMIDI"
[2] "Real Time Sequencer"
_UniCharInfo_LinearCount:888
```

Three red arrows point from the console output to the code: one from the first 'vMIDI' in the console to the 'vMIDI' parameter in the `list()` call, one from the second 'vMIDI' in the console to the second 'vMIDI' parameter in the `list()` call, and one from the 'vMIDI' in the console to the 'vMIDI' parameter in the `new MidiBus()` call.

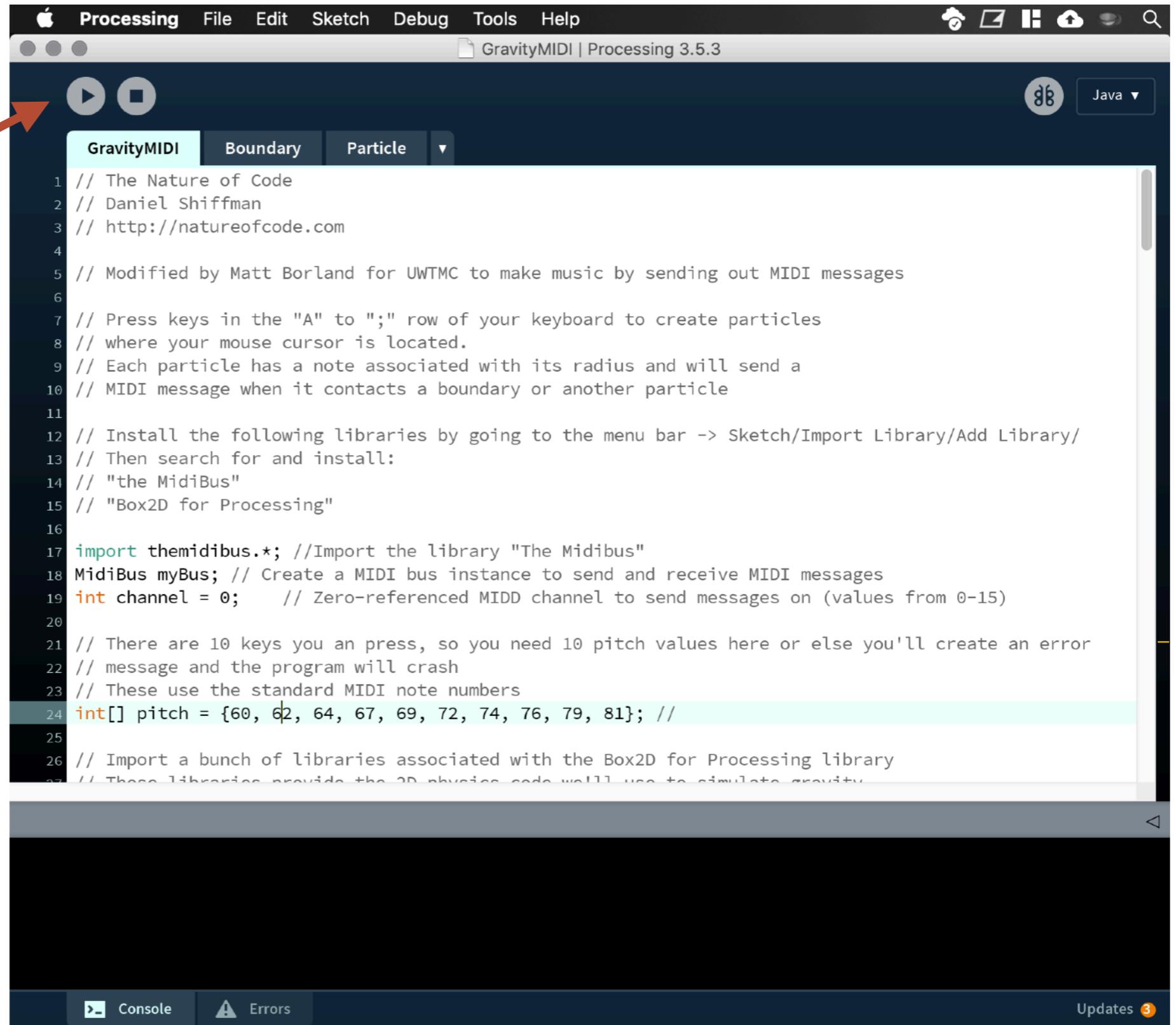
*Scroll down to the setup loop so you can set your MIDI ports.*

*On Mac this variable should match the name of the port found in “Audio MIDI Setup” two slides ago.*

# SIMULATING GRAVITY – RUN THE CODE AGAIN

---

*Press the run button to try to run the code. Hopefully it works this time!*



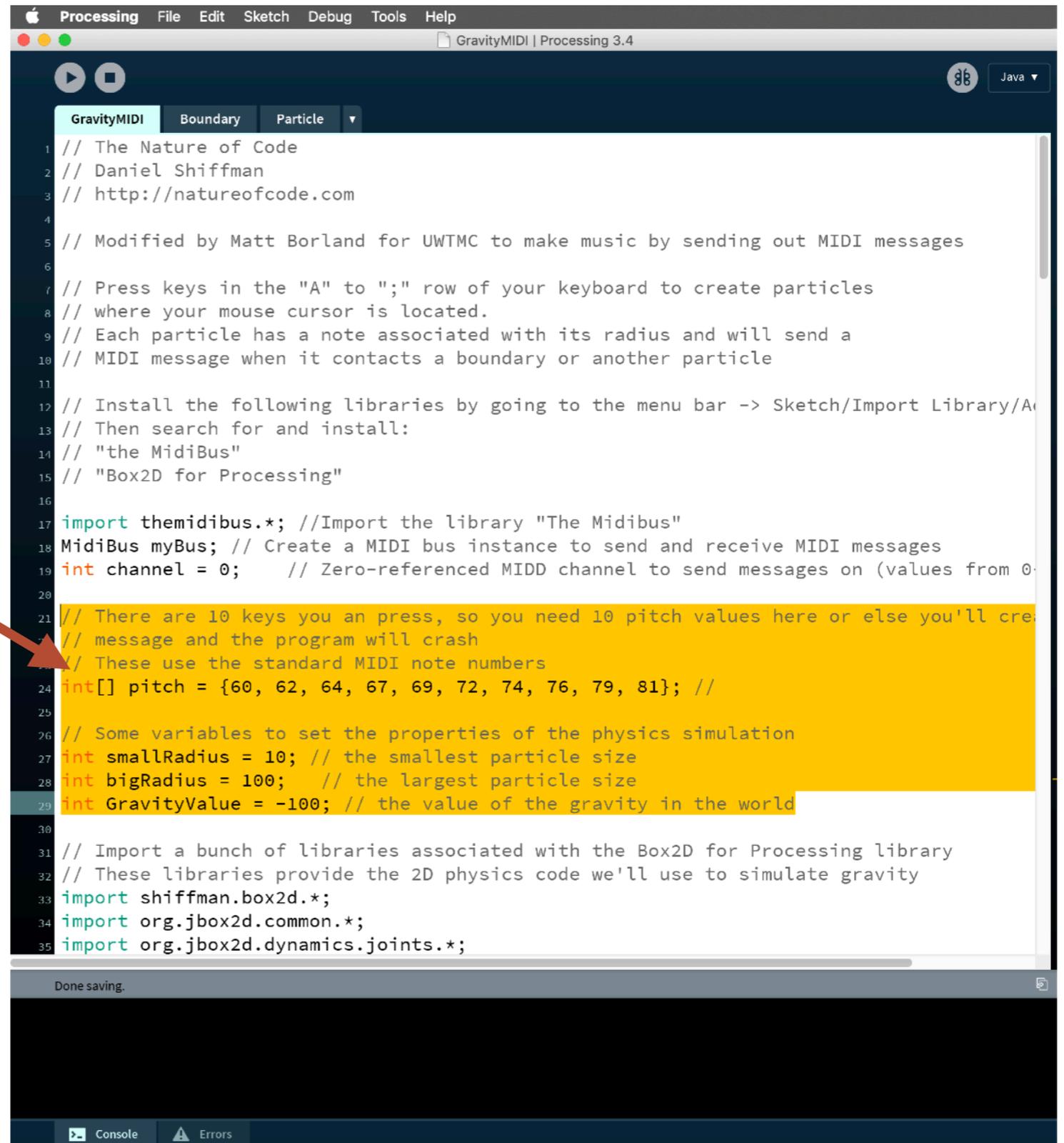
```
Processing File Edit Sketch Debug Tools Help
GravityMIDI | Processing 3.5.3

GravityMIDI Boundary Particle
1 // The Nature of Code
2 // Daniel Shiffman
3 // http://natureofcode.com
4
5 // Modified by Matt Borland for UWTMC to make music by sending out MIDI messages
6
7 // Press keys in the "A" to ";" row of your keyboard to create particles
8 // where your mouse cursor is located.
9 // Each particle has a note associated with its radius and will send a
10 // MIDI message when it contacts a boundary or another particle
11
12 // Install the following libraries by going to the menu bar -> Sketch/Import Library/Add Library/
13 // Then search for and install:
14 // "the MidiBus"
15 // "Box2D for Processing"
16
17 import themidibus.*; //Import the library "The Midibus"
18 MidiBus myBus; // Create a MIDI bus instance to send and receive MIDI messages
19 int channel = 0; // Zero-referenced MIDD channel to send messages on (values from 0-15)
20
21 // There are 10 keys you an press, so you need 10 pitch values here or else you'll create an error
22 // message and the program will crash
23 // These use the standard MIDI note numbers
24 int[] pitch = {60, 62, 64, 67, 69, 72, 74, 76, 79, 81}; //
25
26 // Import a bunch of libraries associated with the Box2D for Processing library
27 // These libraries provide the 2D physics code we'll use to simulate gravity
```

Console Errors Updates 3

# SIMULATING GRAVITY – CHANGE VARIABLES

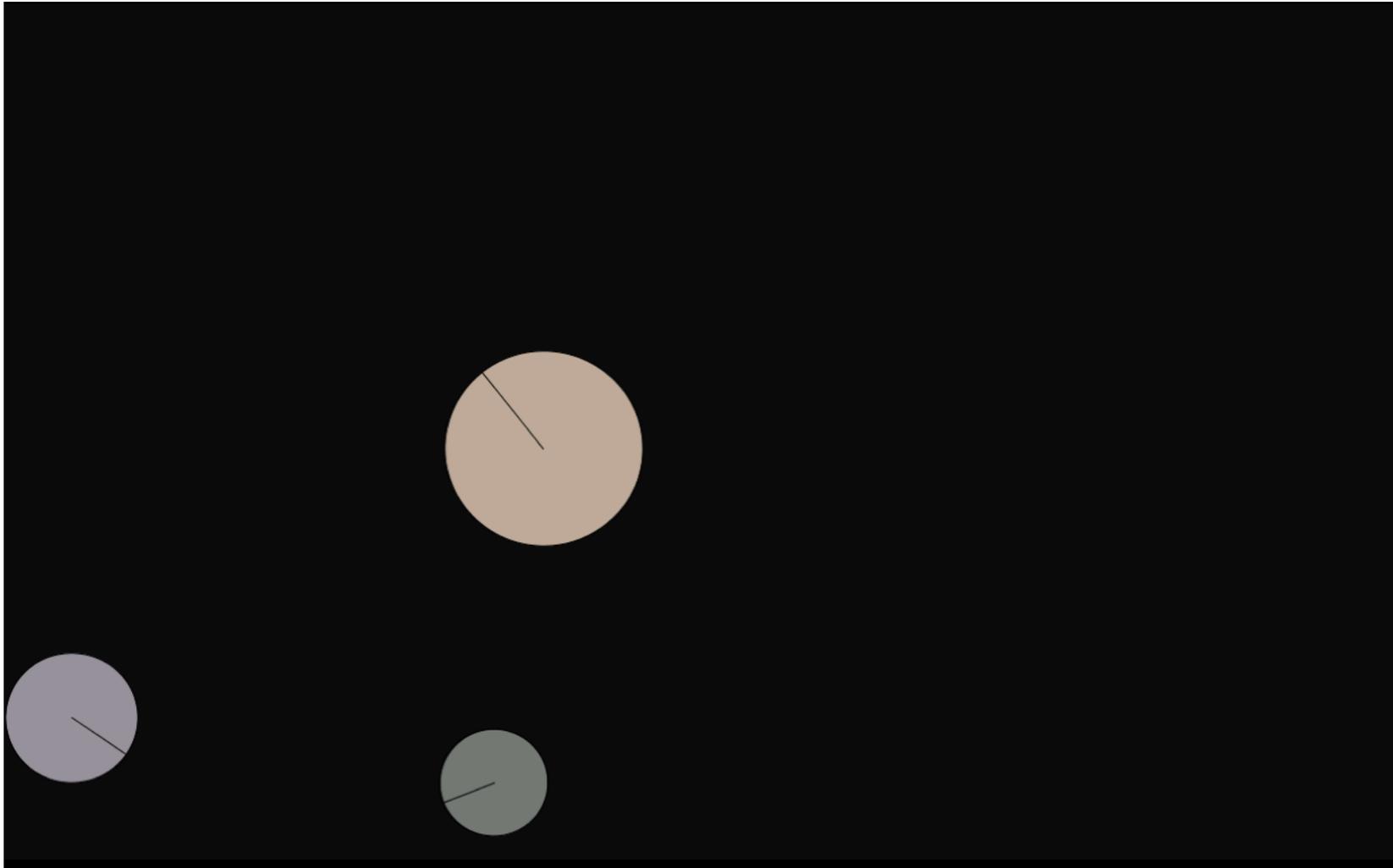
*These are the variables in the code you can change easily. Follow the instructions and think about what each one means. You can make the program crash if you choose bad numbers, so if your code stops working, come back tot these values.*



```
Processing File Edit Sketch Debug Tools Help
GravityMIDI | Processing 3.4
GravityMIDI Boundary Particle
1 // The Nature of Code
2 // Daniel Shiffman
3 // http://natureofcode.com
4
5 // Modified by Matt Borland for UWTC to make music by sending out MIDI messages
6
7 // Press keys in the "A" to ";" row of your keyboard to create particles
8 // where your mouse cursor is located.
9 // Each particle has a note associated with its radius and will send a
10 // MIDI message when it contacts a boundary or another particle
11
12 // Install the following libraries by going to the menu bar -> Sketch/Import Library/A
13 // Then search for and install:
14 // "the MidiBus"
15 // "Box2D for Processing"
16
17 import themidibus.*; //Import the library "The Midibus"
18 MidiBus myBus; // Create a MIDI bus instance to send and receive MIDI messages
19 int channel = 0; // Zero-referenced MIDD channel to send messages on (values from 0
20
21 // There are 10 keys you an press, so you need 10 pitch values here or else you'll cre
22 // message and the program will crash
23 // These use the standard MIDI note numbers
24 int[] pitch = {60, 62, 64, 67, 69, 72, 74, 76, 79, 81}; //
25
26 // Some variables to set the properties of the physics simulation
27 int smallRadius = 10; // the smallest particle size
28 int bigRadius = 100; // the largest particle size
29 int GravityValue = -100; // the value of the gravity in the world
30
31 // Import a bunch of libraries associated with the Box2D for Processing library
32 // These libraries provide the 2D physics code we'll use to simulate gravity
33 import shiffman.box2d.*;
34 import org.jbox2d.common.*;
35 import org.jbox2d.dynamics.joints.*;
```

# SIMULATING GRAVITY – RUN THE CODE

---



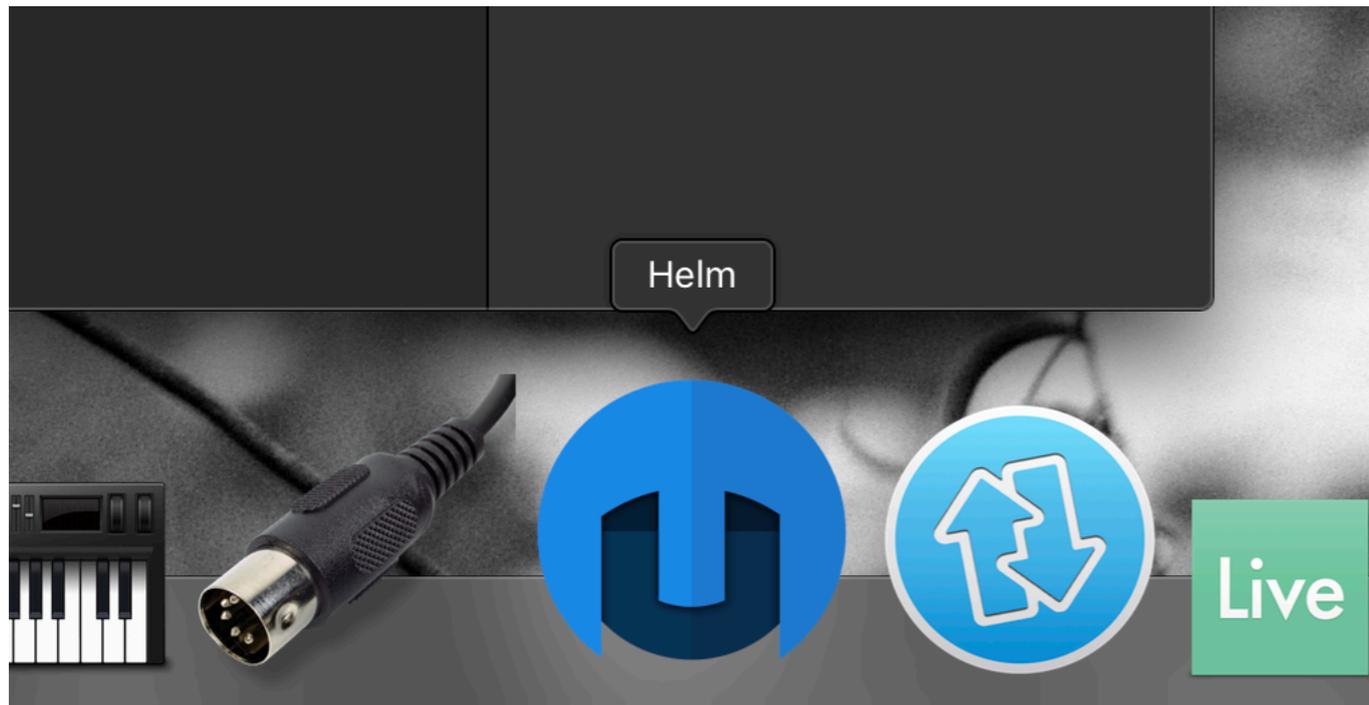
*Your screen should fill up with the program running the simulation.*

*Press keys in your “A” to “;” row to create particles which will follow the laws of gravity.*

*Each collision will send a MIDI message which we’ll interpret in a synthesizer and turn into sound!*

# OPEN HELM - SOFTWARE SYNTHESIZER

---



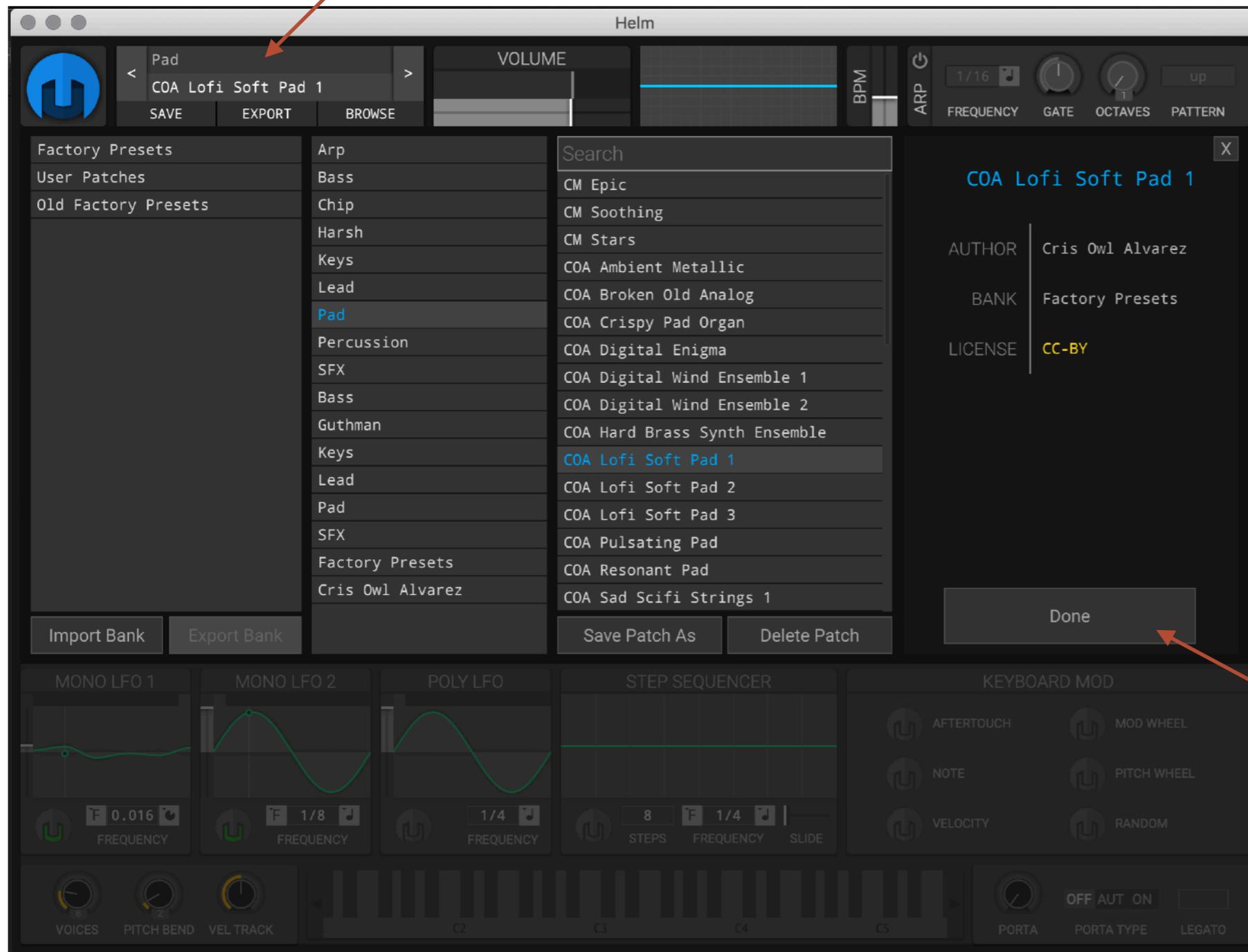
*Helm - a software synthesizer  
to make musical sounds with  
your computer.*

<https://tytel.org/helm/>

# SETUP HELM

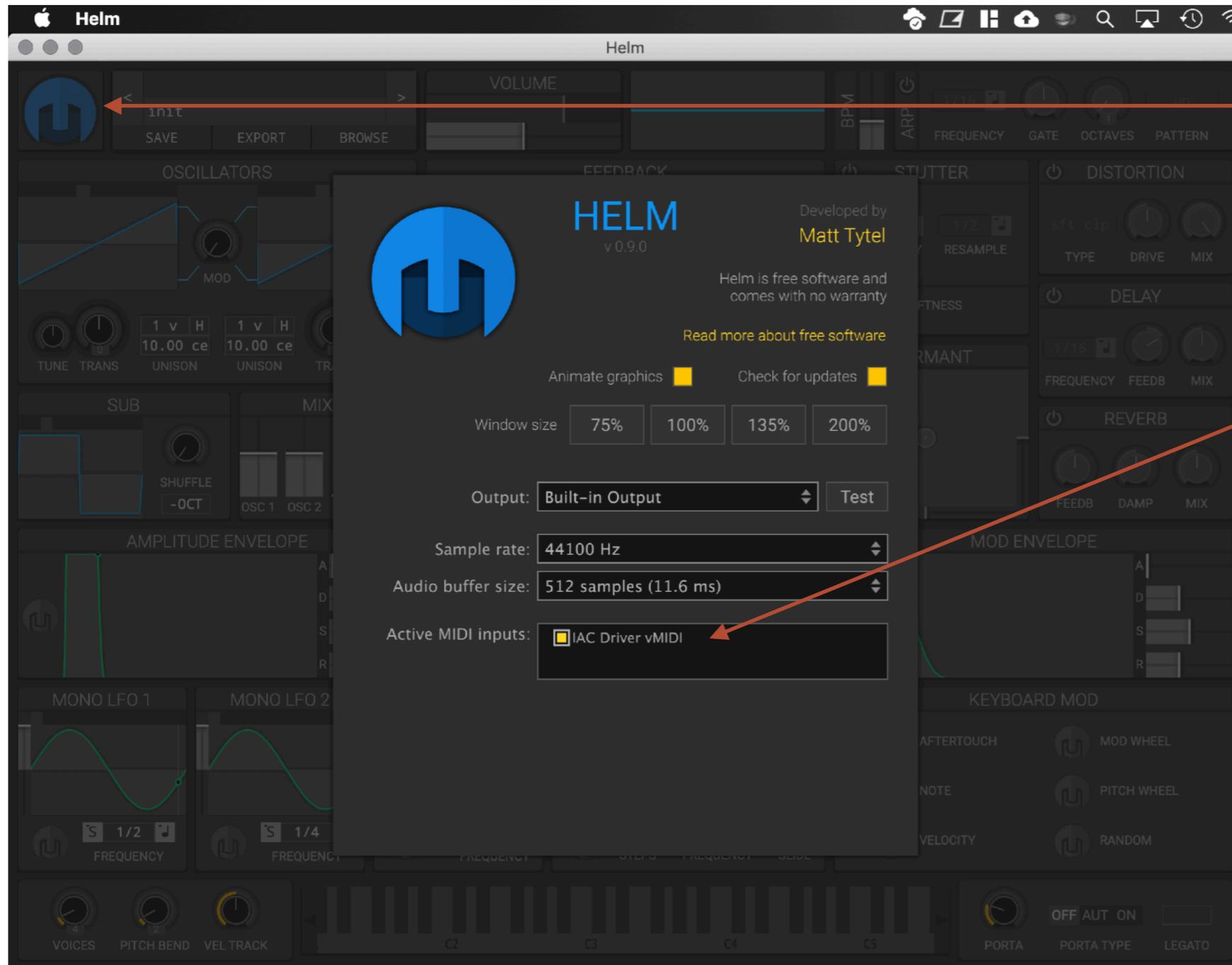
*Click here to open the menu to select a sound*

*Make sure your laptop's speakers are on and turned up!*



*Try different sounds, then click done.*

# SETUP MIDI

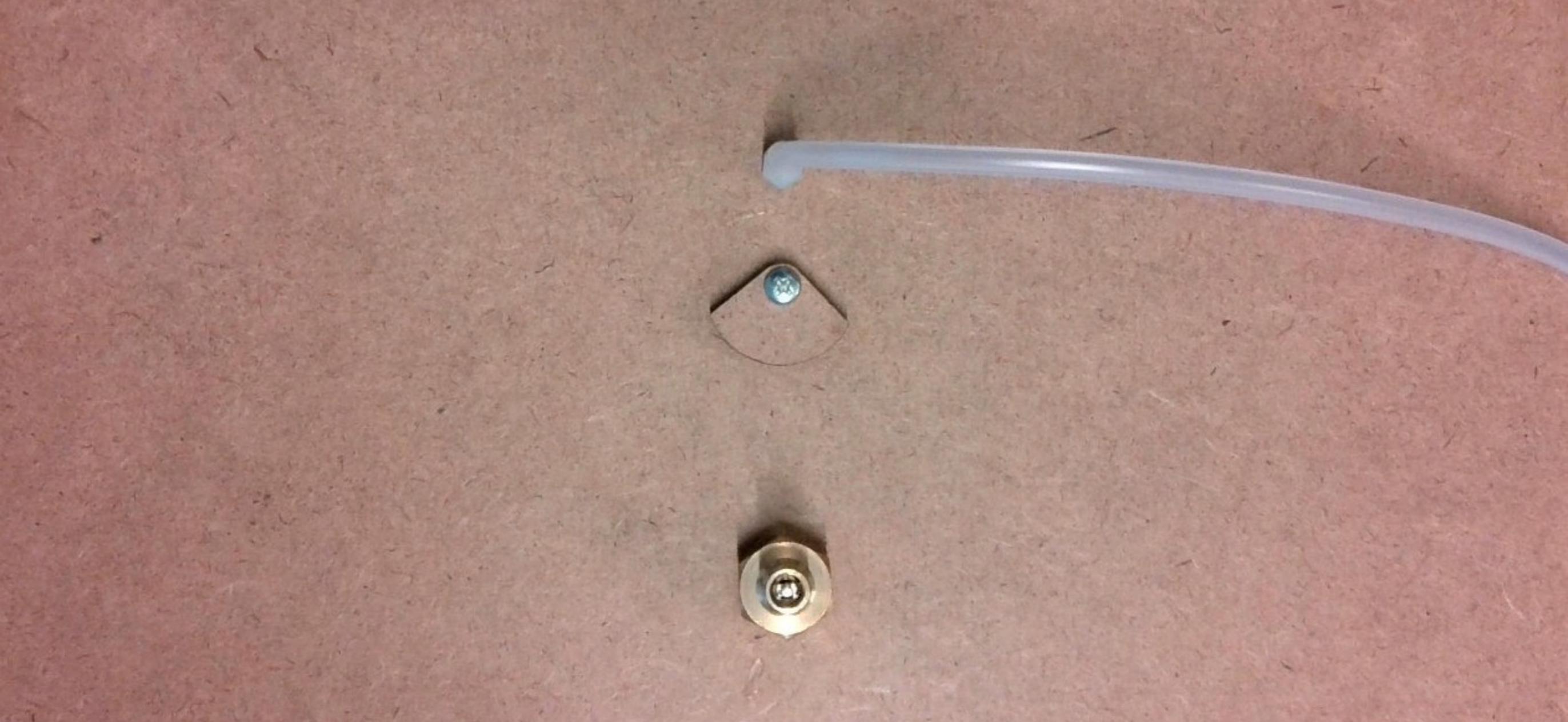


*Click the HELMet and then check that you see IAC driver selected under active MIDI inputs.*

# SETUP HELM



*You should hear sound and see things flashing and changing on screen if you have particles running in your simulation.*



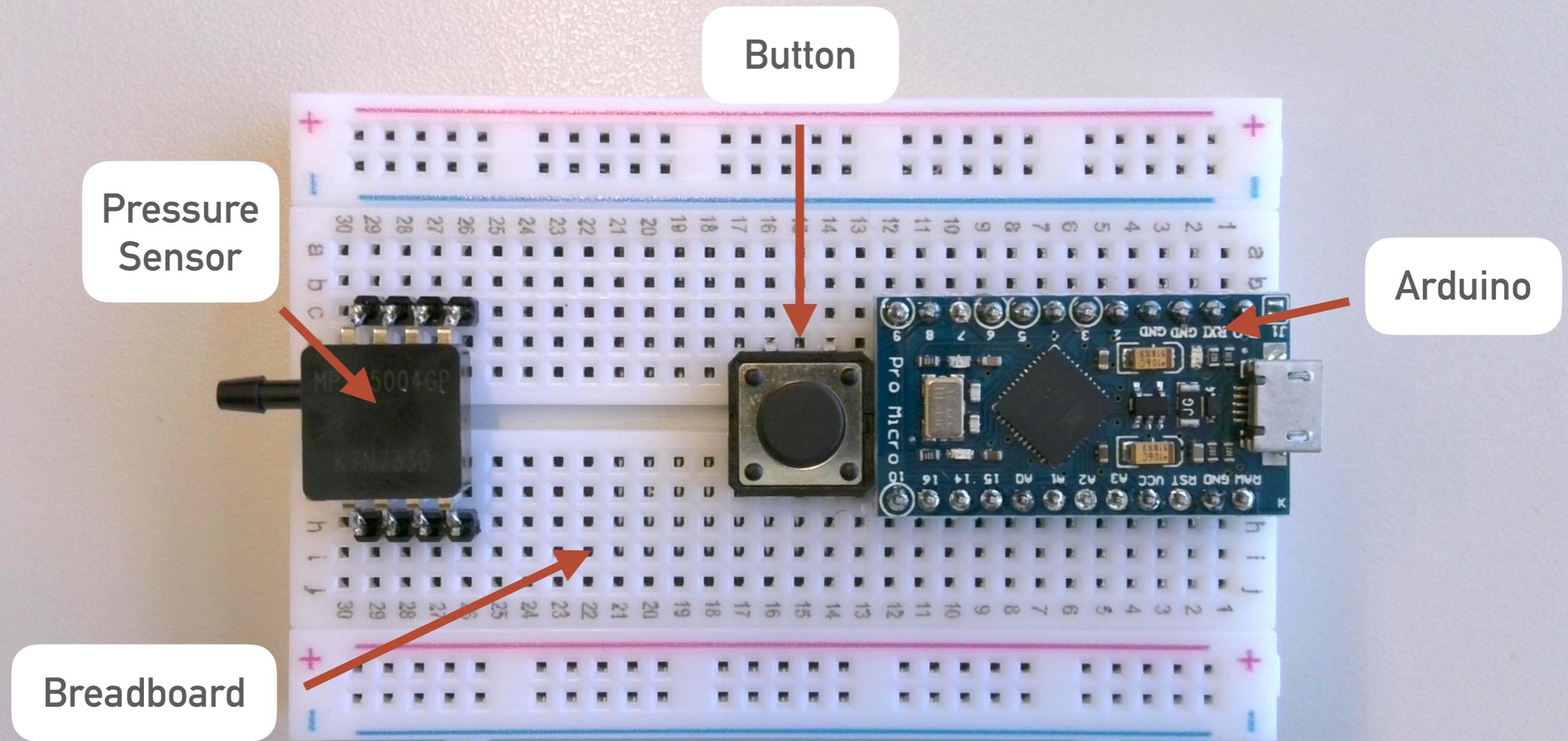
# USING GRAVITY

*UWTMC*

*Matt Borland - 2019*

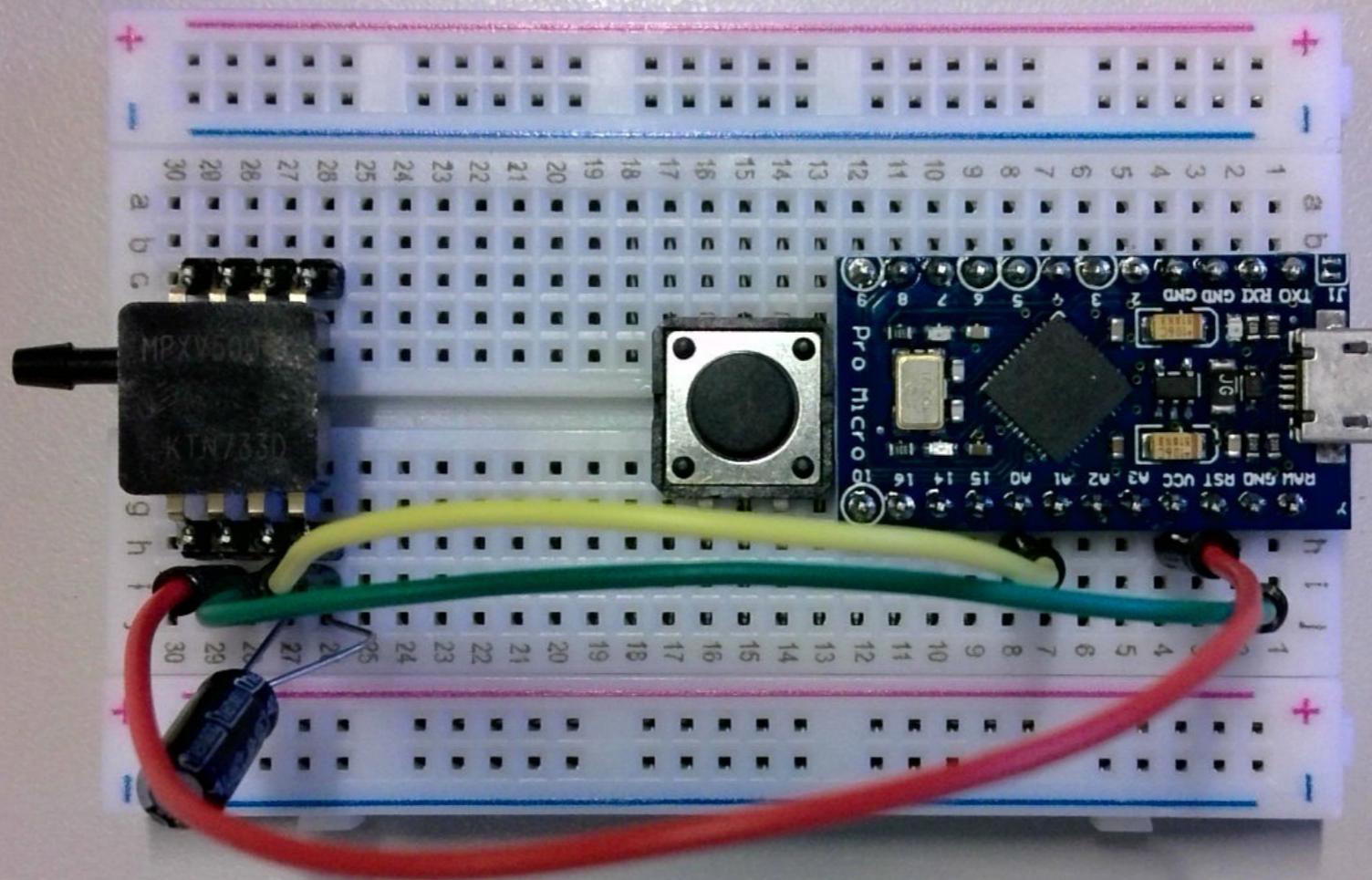
# HOOKING UP YOUR SENSORS! FOLLOW MB ON THE PROJECTOR

---



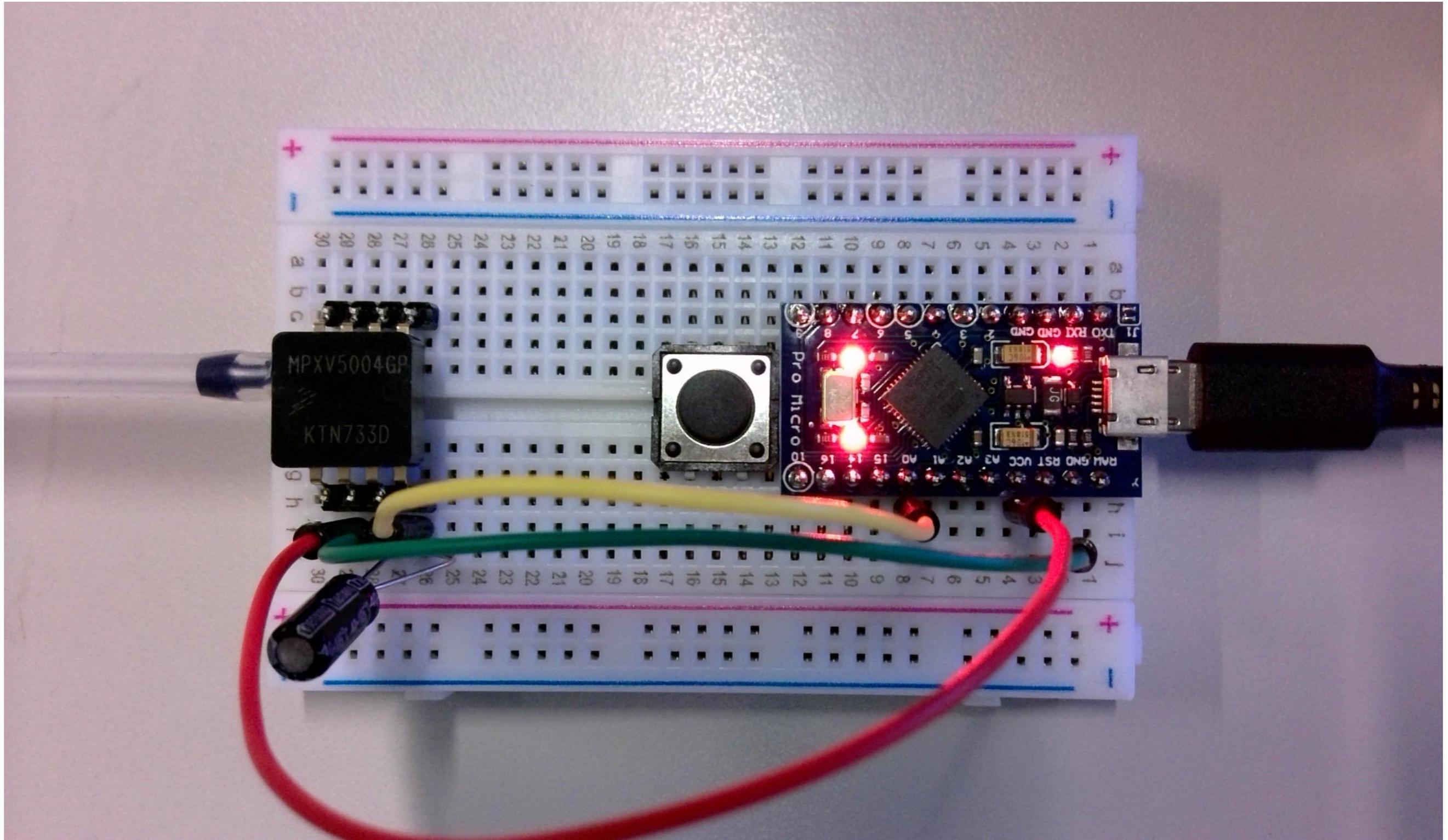
# HOOKING UP YOUR SENSORS! SEE THE SCHEMATIC

---



# PLUG IN USB AND THE PRESSURE TUBE

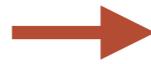
---



# OPEN ARDUINO IDE - UPLOAD GRAVITYDEBUG.INO

---

*Program your board with the debugging file to see if your sensor is working and to figure out the range of values you will see when playing the instrument.*

A screenshot of the Arduino IDE interface. The window title is "GravityDebug | Arduino 1.8.7". The menu bar includes "Arduino", "File", "Edit", "Sketch", "Tools", and "Help". The toolbar shows icons for "Upload Using Programmer". The code editor contains the following C++ code:

```
// Programmed using the knockoff Pro Micro boards - use Arduino/Genuino Micro as board type.
// Watch out - your board's name might change and you'll have to select it again in Tools/Port after
// you program it the first time.

// Anything on a line after "//" is commented out - the microcontroller will ignore it!

// This sketch is just to check your sensor is working
// This is good practice to make sure all your hardware is working the way you are expecting it to

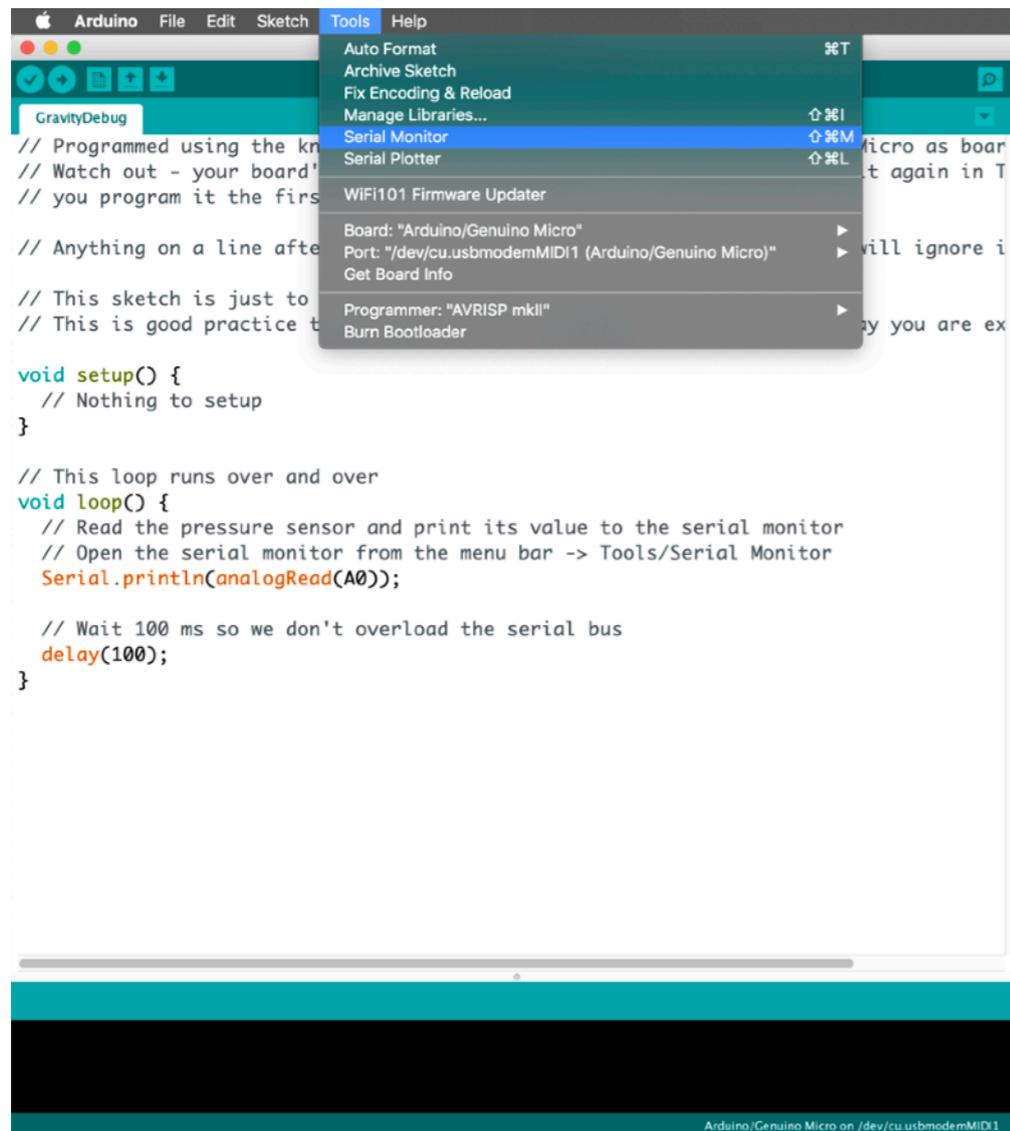
void setup() {
  // Nothing to setup
}

// This loop runs over and over
void loop() {
  // Read the pressure sensor and print its value to the serial monitor
  // Open the serial monitor from the menu bar -> Tools/Serial Monitor
  Serial.println(analogRead(A0));

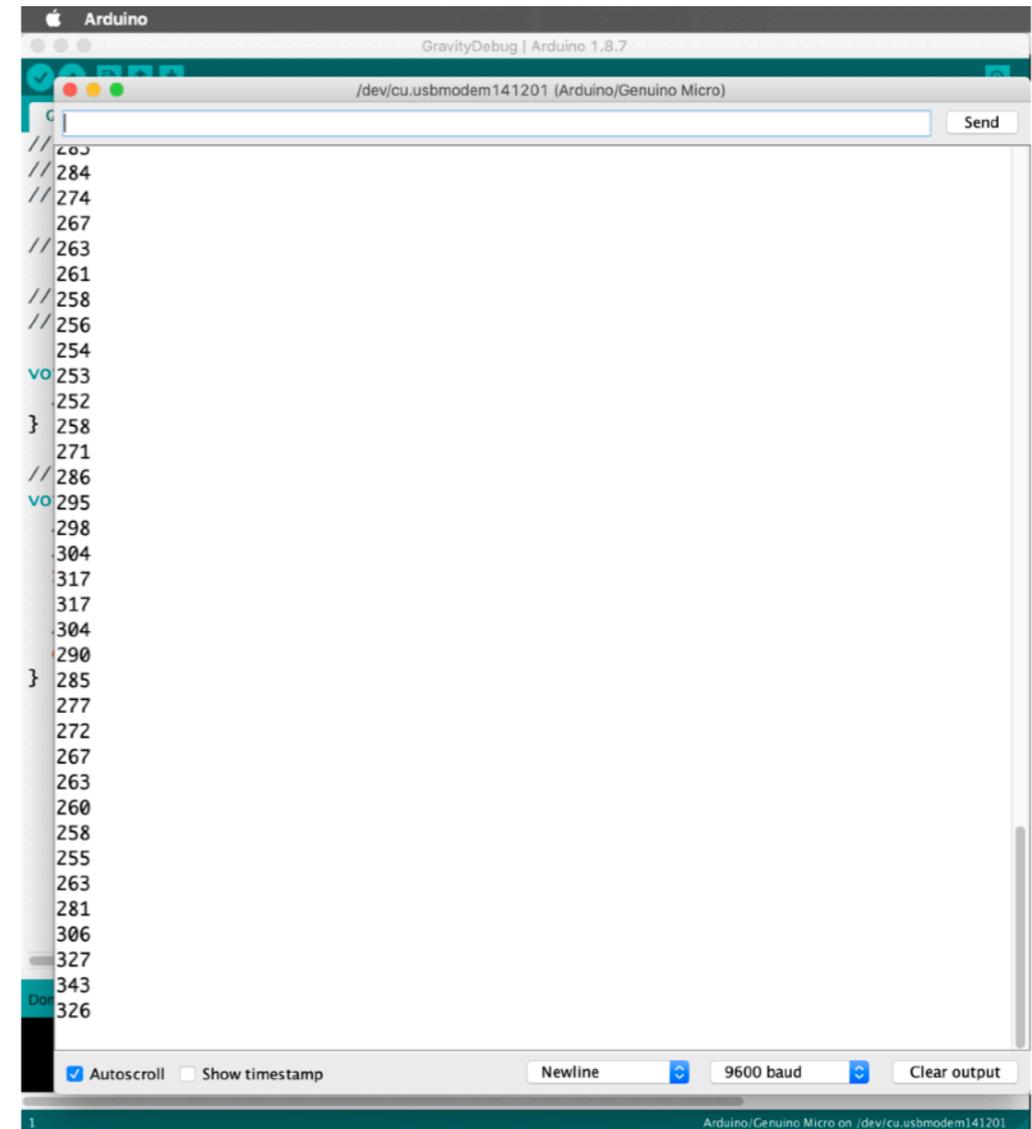
  // Wait 100 ms so we don't overload the serial bus
  delay(100);
}
```

The status bar at the bottom shows the file path: "/Users/mattborland/ownCloud/TEACHING/UWTMC/Public Sessions/F2019/Session 2 - Gravity/GravityFiles/Grav". The system tray at the bottom right shows "Arduino/Genuino Micro on /dev/cu.usbmodem141201".

# OPEN HELM – RUN GRAVITYDEBUG.INO AND CHECK OUTPUT



*Open the serial monitor from the menu bar.*



*Play the instrument and see what the biggest value your sensor typically outputs.*

# OPEN ARDUINO IDE - UPLOAD GRAVITYCC.INO

*Program your board with the MIDI file to send MIDI CC Messages that correspond to the pressure level inside the shruti box. You need to adjust the “highP” value with the number you found as the max pressure when using the debug file. You can change this and re-upload if the instrument doesn’t play the way you want it to.*



```
Arduino File Edit Sketch Tools Help
GravityCC | Arduino 1.8.7
GravityCC
// Programmed using the knockoff Pro Micro boards - use Arduino/Genuino Micro as board type.
// Watch out - your board's name might change and you'll have to select it again in Tools/Port after
// you program it the first time.

// Anything on a line after "//" is commented out - the microcontroller will ignore it!

// Libraries
// You need to install these libraries from the menu bar ->
// Sketch/Include Library/Manage Libraries, and then search for the word in front of the .h
#include <elapsedMillis.h>
#include <MIDIUSB.h>

//// MIDI Setup
// Send MIDI on channel 1
int channel = 1;

// Send Control Change messages on CC #7
int CC = 7;

// A variable for the lower pressure limit
int lowP = 0;

// A variable for the high pressure limit - you should set this appropriately after running
// Gravity debug - a value in the range 260-300 is probably right
int highP = 260;

// A timer variable that is used to limit the number of messages that are sent
elapsedMillis timeElapsed;

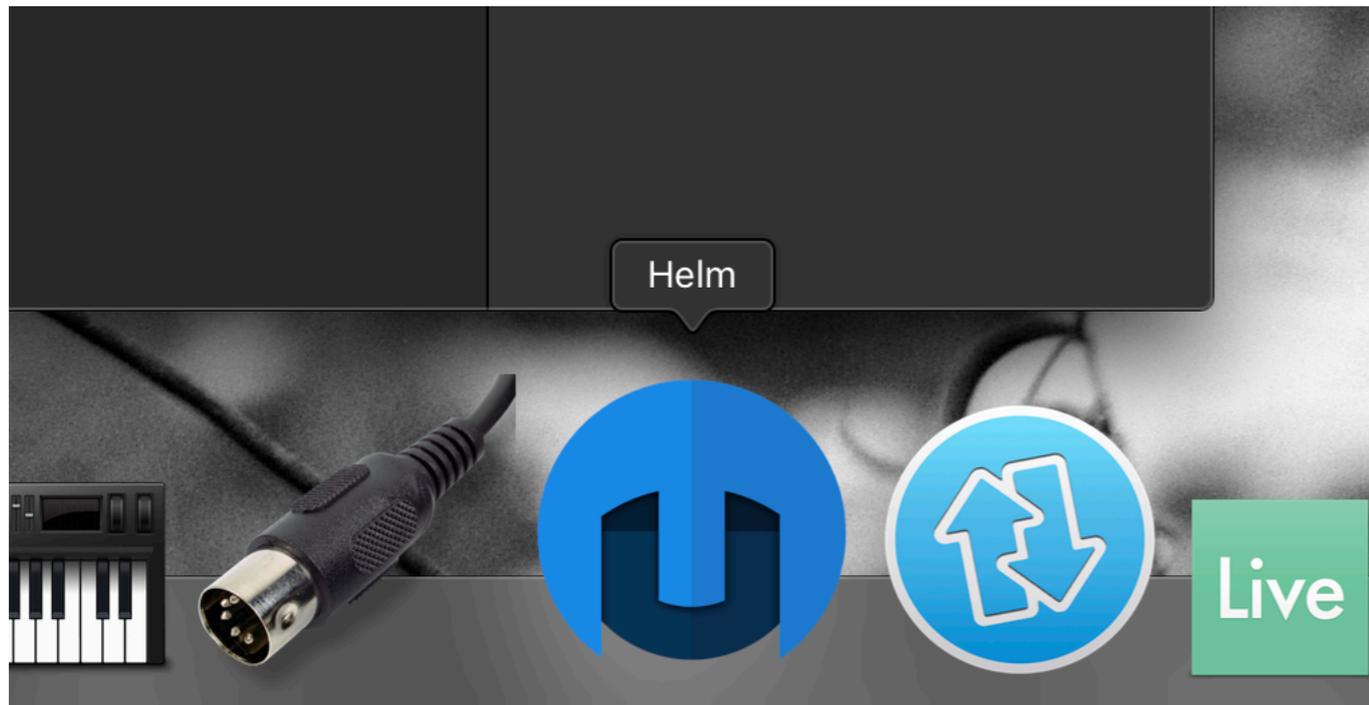
// The setup loop runs once:
void setup() {
  lowP = analogRead(A0);
}

// The main loop runs repeatedly forever
void loop() {
  // if 20 ms have elapsed since the last message was sent, send another one
  if(timeElapsed > 20){
    timeElapsed = 0;
    // read the pressure and map the current pressure reading to an appropriate MIDI CC
    // value in the range 0-127
  }
}

Done uploading.
Arduino/Genuino Micro on /dev/cu.usbmodem141201
```

# OPEN HELM – SOFTWARE SYNTHESIZER

---



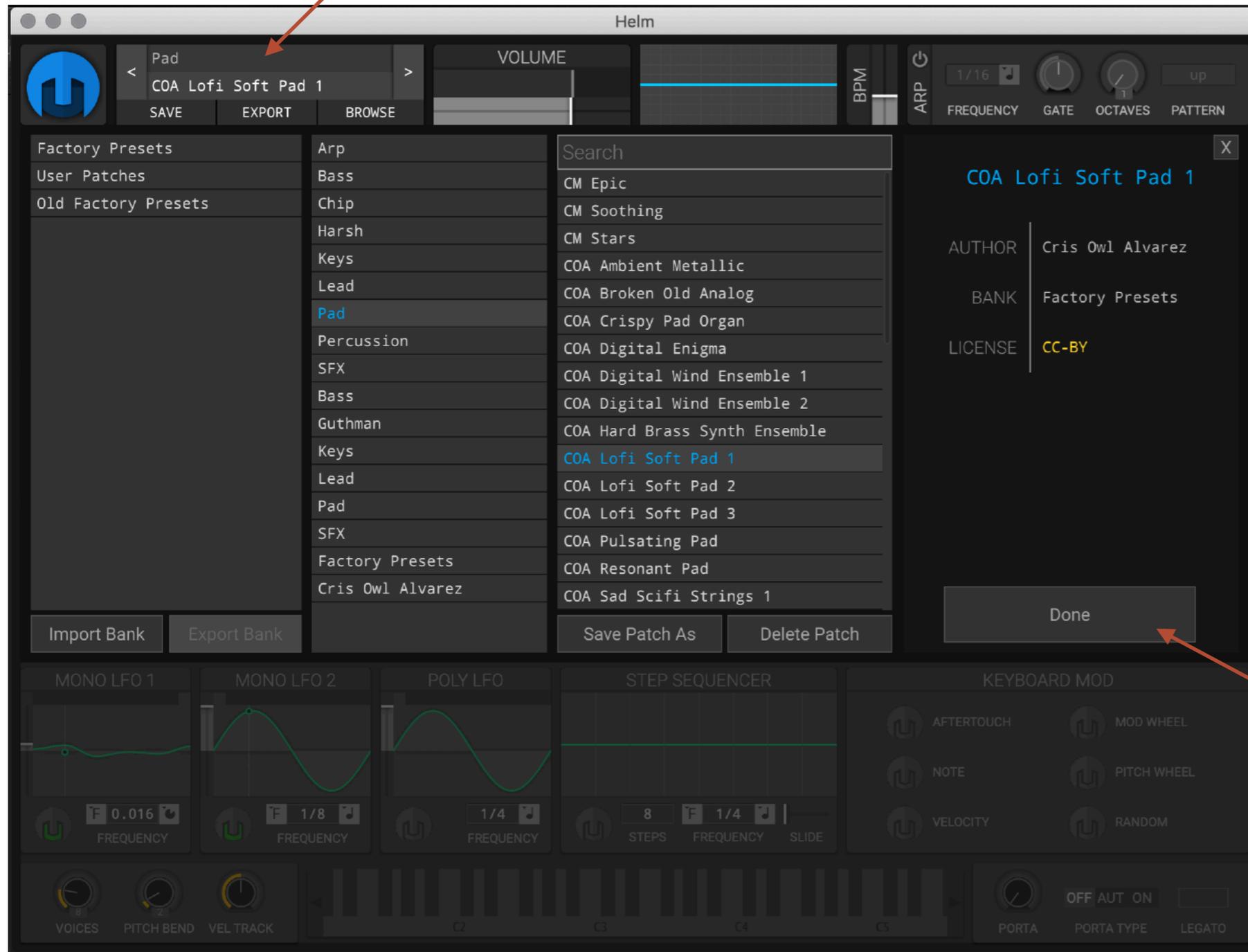
*Helm - a software synthesizer  
to make musical sounds with  
your computer.*

<https://tytel.org/helm/>

# SETUP HELM – SOUNDS

*Click here to open the menu to select a sound*

*Make sure your laptop's speakers are on and turned up!*



*Try different sounds, then click done.*

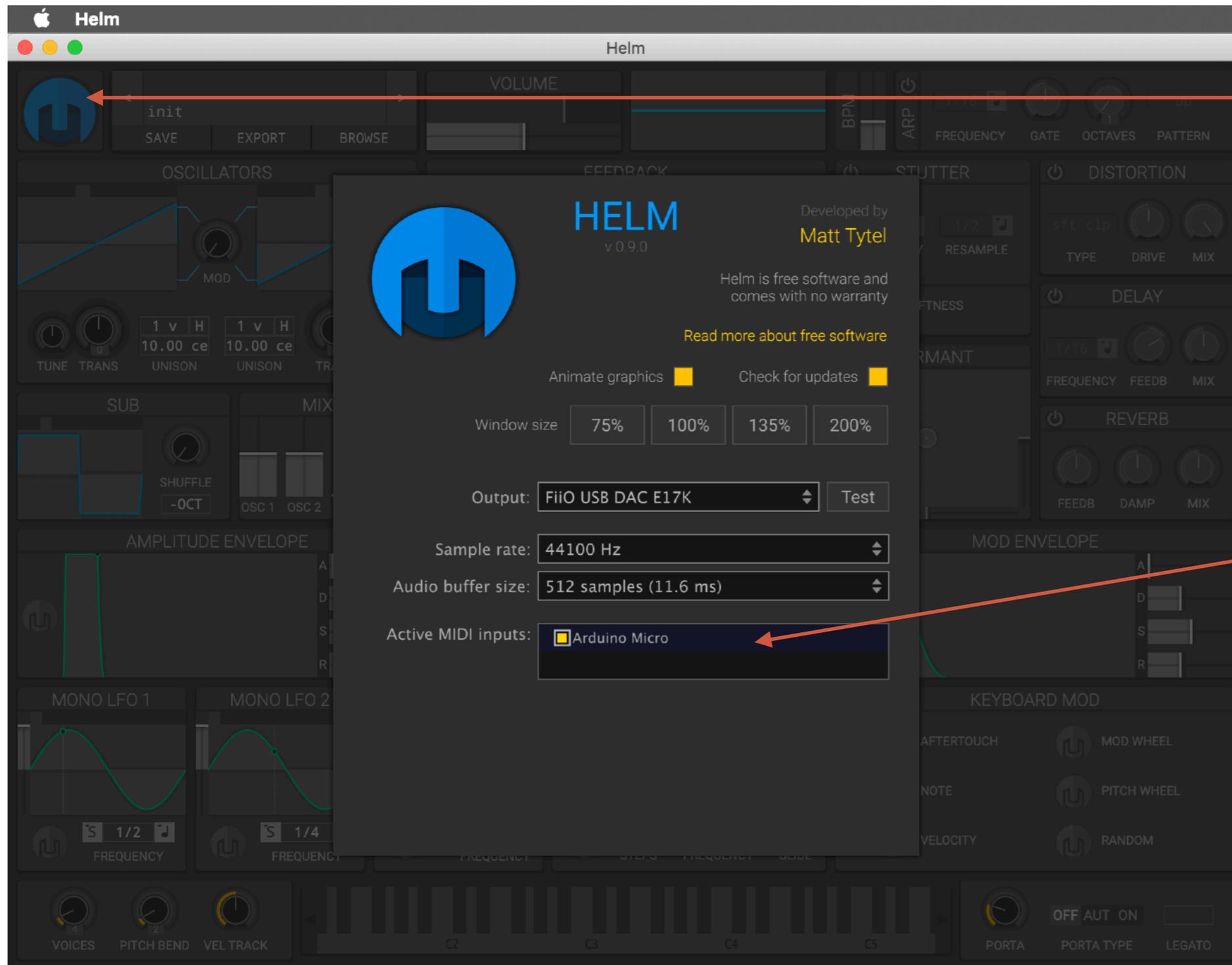
# SETUP HELM – MAP CC TO OVERALL VOLUME

*Right click on this bar below VOLUME, then click “LEARN MIDI ASSIGNMENT”. You should then see this bar slide back and forth as you squeeze the shruti box. If the bar is all the way down, you won’t hear anything because you’ve turned the volume all the way down!*



*The Arduino only sends CC messages, so no notes will be played. You need to play them by using your computer keyboard. The “A” to “L” row are the white keys of a piano. You’ll see them light up here as you play them.*

# NOT WORKING? CHECK THE MIDI PORT



*Click the HELMet to get this dialog to open.*

*You should see “Arduino Micro” as a MIDI input, with a yellow box showing it is selected.*