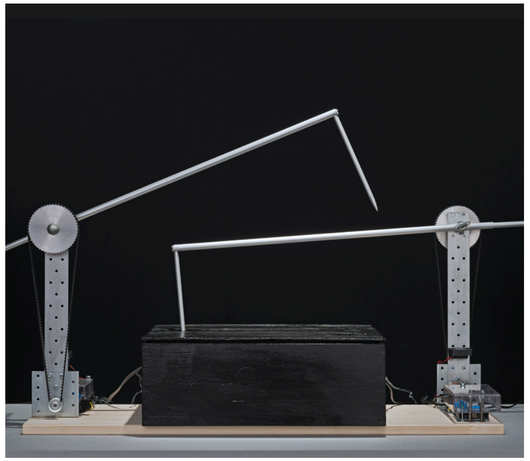
Tactile Agency

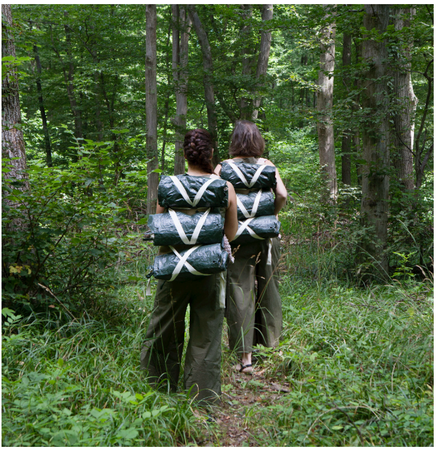
Rethinking Consent through Embodied AI Prototyping

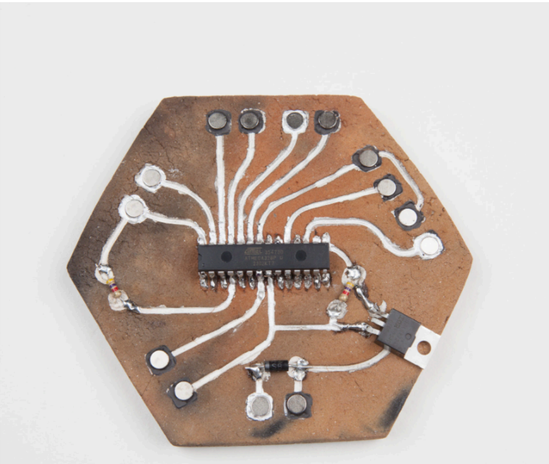
Patrícia J. Reis
7.6.25 | 13:00-18:00

This hands-on workshop explores the ethical, emotional and sensory dimensions of AI-driven human-machine interaction. Participants will engage with simple AI models and DIY touch-responsive interfaces using Arduino and open-source machine learning tools to examine how consent, agency and embodiment can be both problematic and reimagined within technological systems. By building interactive prototypes and experimenting with real-time feedback, the workshop invites critical reflection on intimacy, power and the politics of touch in today's technosphere. Beginners are also welcome; no prior experience is required.

about me







Patrícia J. Reis, ***Odalisques***, audiovisual interactive installation, 2006-2013. Exhibition at Edith-Russ-Haus für Medienkunst, Oldenburg, Germany. Photo © EDITH-RUSS-HAUS FÜR MEDIENKUNST

Patrícia J. Reis, ***Odalisques***, audiovisual interactive installation, 2006-2013. Exhibition at Edith-Russ-Haus für Medienkunst, Oldenburg, Germany. Photo © EDITH-RUSS-HAUS FÜR MEDIENKUNST

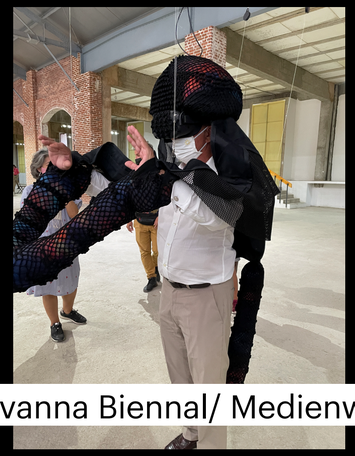
Patrícia J. Reis, ***Endosensorial Mask***, Virtual reality haptic interactive installation, 2022. Exhibition at Medienwerkstatt (Fotogalerie Wien) Vienna, Austria.
Photo © Fotogralerie Wien










Patrícia J. Reis, ***Endosensorial Mask***, Virtual reality haptic interactive installation, 2022. Exhibition at Havanna Biennal/ Medienwerkstatt (Fotogalerie Wien) Vienna, Austria.

Patrícia J. Reis, ***Blow***, Interactive haptic installation, 2019. Exhibition at Art Fair Parallel, Vienna, Austria. Photo © Sophie Thun

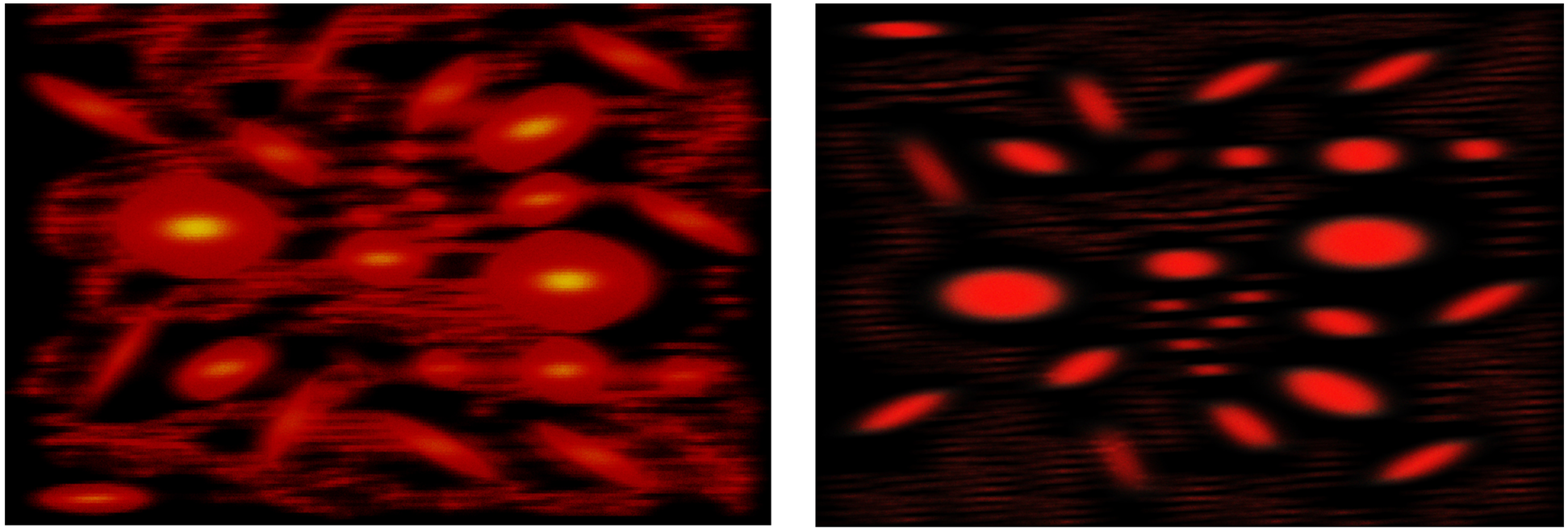
Patrícia J. Reis, ***Blow***, Interactive haptic installation, 2019. Exhibition at Art Fair Parallel, Vienna, Austria. Photo © Sophie Thun

Patrícia J. Reis, ***Underneath the skin another skin***, interactive haptic installation, 2015. Photo © Manfred Pichlbauer

Patrícia J. Reis, ***Underneath the skin another skin***, interactive haptic installation, 2015. Photo © Manfred Pichlbauer

Patrícia J. Reis, ***Underneath the skin another skin***, interactive haptic installation, 2015. Photo © Manfred Pichlbauer

Patrícia J. Reis, ***Underneath the skin another skin***, interactive haptic installation, 2015. Photo © Manfred Pichlbauer



Digital simulation of the perceived image while the frequencies are lower, 2014 © Patrícia Reis

Patrícia J. Reis, ***Massage chair series: "What's love"***, Audiovisual tactile interactive installation, 2023. Exhibition at Austrian Cultural Forum, Warsaw, Poland.
Photo © Flavio Palasciano

Patrícia J. Reis, ***Massage chair series: "What's love"***, Audiovisual tactile interactive installation, 2023. Exhibition at Austrian Cultural Forum, Warsaw, Poland.
Photo © Flavio Palasciano

Patrícia J. Reis, ***Massage chair series: "What's love"***, Audiovisual tactile interactive installation, 2023. Exhibition at Austrian Cultural Forum, Warsaw, Poland.
Photo © Flavio Palasciano

Patrícia J. Reis/ Ruth Schnell, ***Traces of Control***, Interactive mixed reality with HoloLens 2 and mixed media, 2025. Exhibition at AIL, Vienna, Austria.

Patrícia J. Reis/ Ruth Schnell, ***Traces of Control***, Interactive mixed reality with HoloLens 2 and mixed media, 2025. Exhibition at AIL, Vienna, Austria
Photo © Lea Dörl.

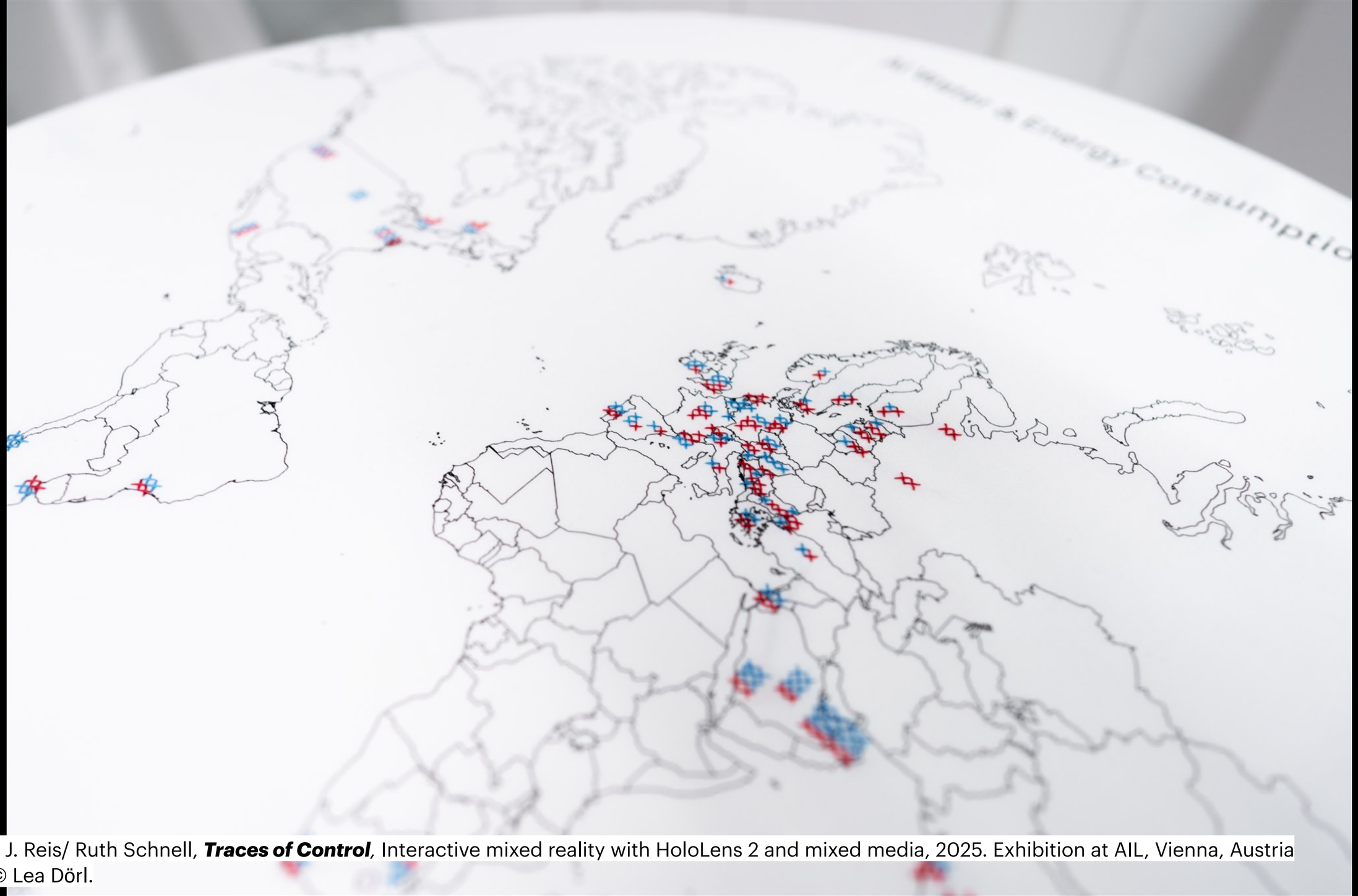
Patrícia J. Reis/ Ruth Schnell, ***Traces of Control***, Interactive mixed reality with HoloLens 2 and mixed media, 2025. Exhibition at AIL, Vienna, Austria
Photo © Lea Dörl.

Patrícia J. Reis/ Ruth Schnell, ***Traces of Control***, Interactive mixed reality with HoloLens 2 and mixed media, 2025. Exhibition at AIL, Vienna, Austria
Photo © Lea Dörl.

Patrícia J. Reis/ Ruth Schnell, ***Traces of Control***, Interactive mixed reality with HoloLens 2 and mixed media, 2025. Exhibition at AIL, Vienna, Austria
Photo © Lea Dörl.

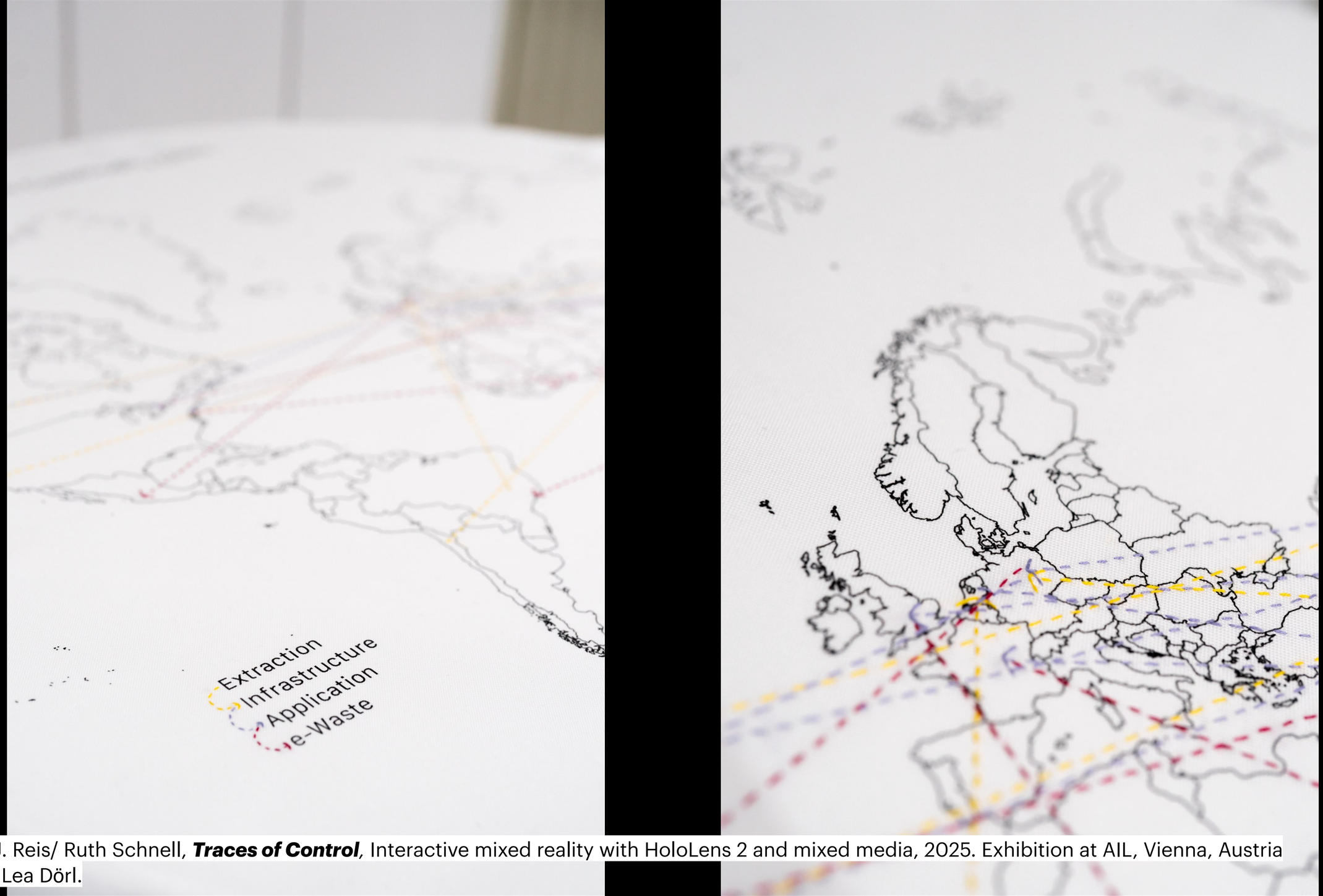
Patrícia J. Reis/ Ruth Schnell, ***Traces of Control***, Interactive mixed reality with HoloLens 2 and mixed media, 2025. Exhibition at AIL, Vienna, Austria
Photo © Lea Dörl.

Patrícia J. Reis/ Ruth Schnell, ***Traces of Control***, Interactive mixed reality with HoloLens 2 and mixed media, 2025. Exhibition at AIL, Vienna, Austria
Photo © Lea Dörl.



Patrícia J. Reis/ Ruth Schnell, ***Traces of Control***, Interactive mixed reality with HoloLens 2 and mixed media, 2025. Exhibition at AIL, Vienna, Austria
Photo © Lea Dörl.

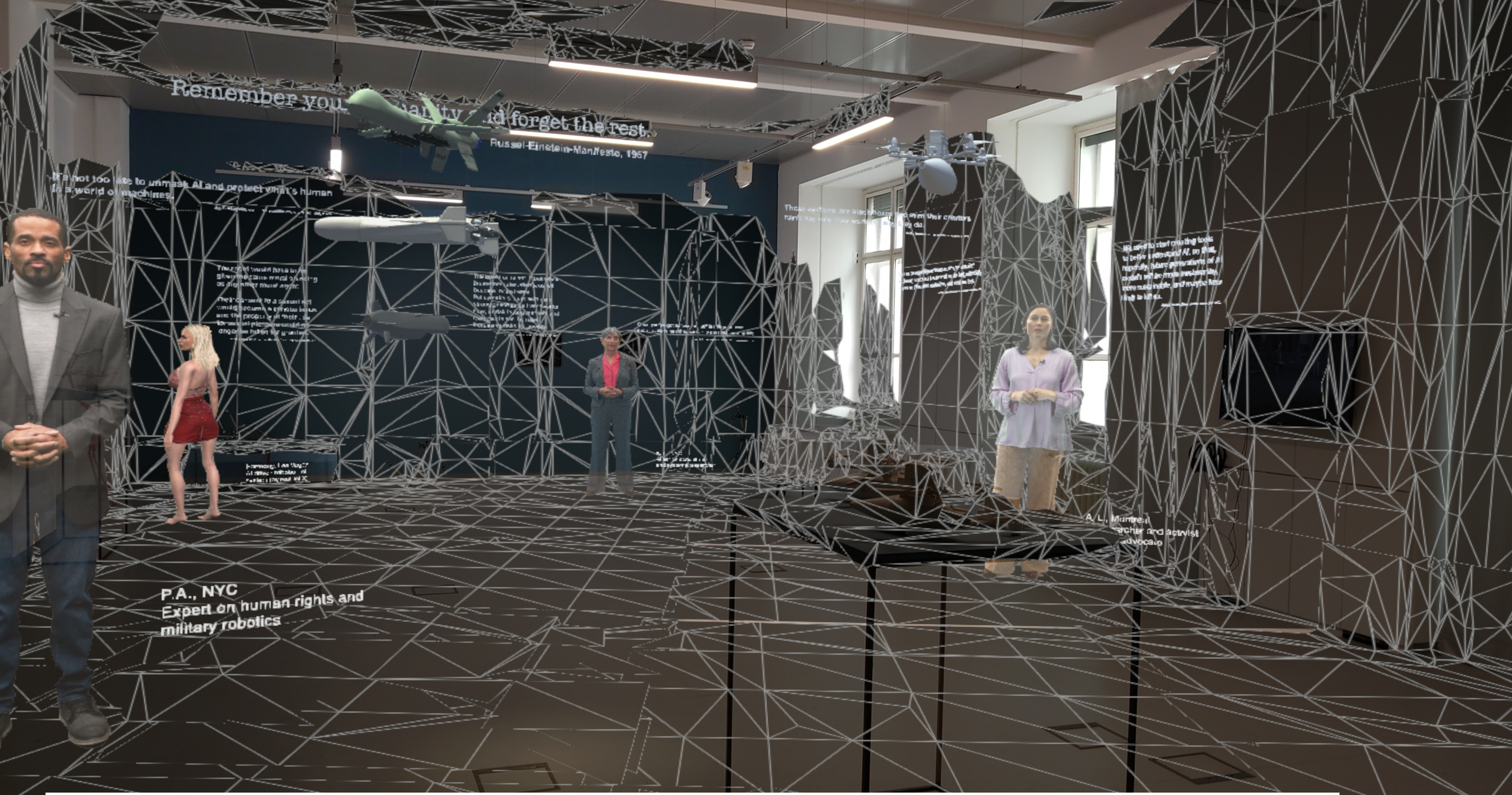
Patrícia J. Reis/ Ruth Schnell, ***Traces of Control***, Interactive mixed reality with HoloLens 2 and mixed media, 2025. Exhibition at AIL, Vienna, Austria
Photo © Lea Dörl.



Patrícia J. Reis/ Ruth Schnell, ***Traces of Control***, Interactive mixed reality with HoloLens 2 and mixed media, 2025. Exhibition at AIL, Vienna, Austria
Photo © Lea Dörl.

Patrícia J. Reis/ Ruth Schnell, ***Traces of Control***, Interactive mixed reality with HoloLens 2 and mixed media, 2025. Exhibition at AIL, Vienna, Austria
Photo © Lea Dörl.

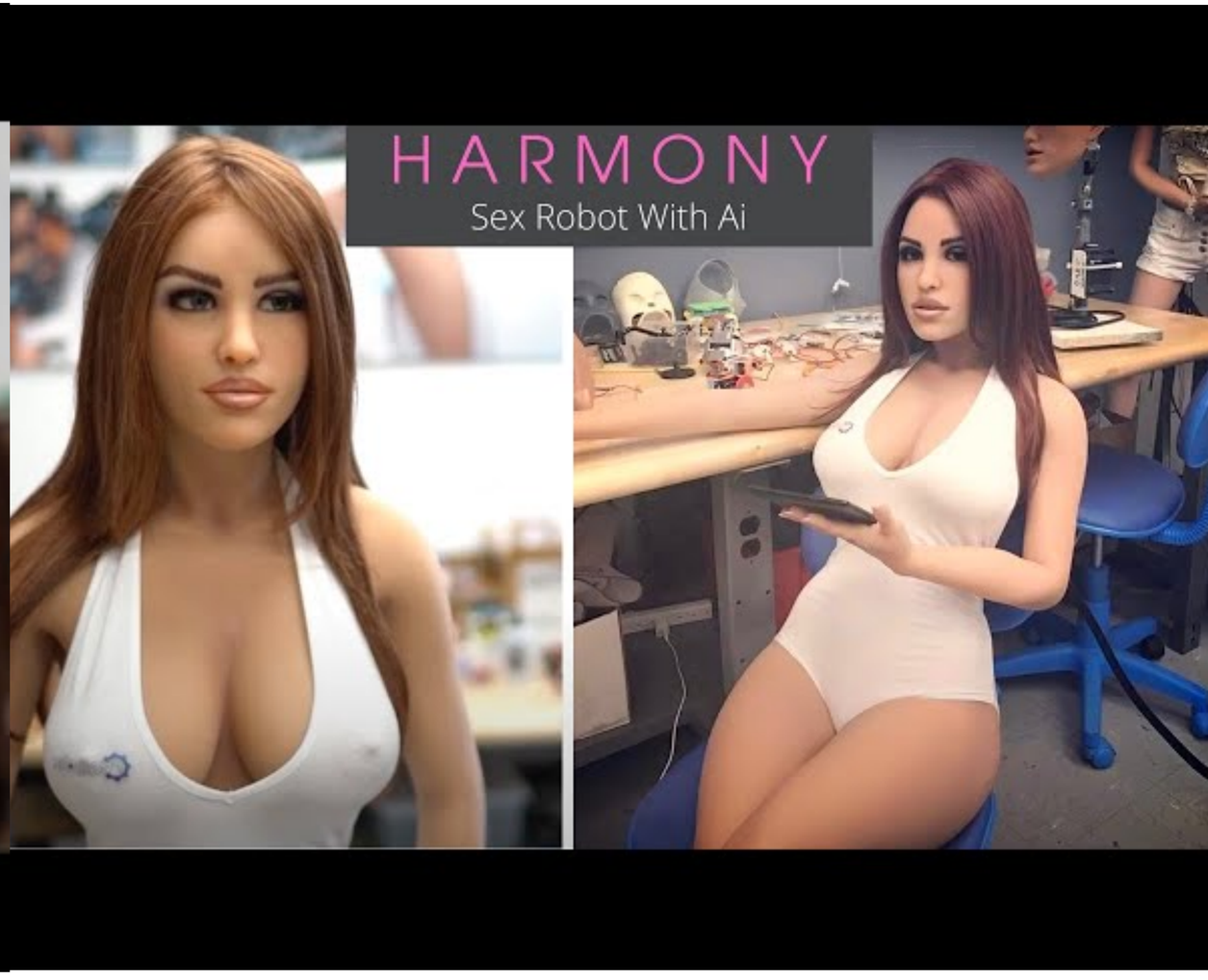
Patrícia J. Reis/ Ruth Schnell, ***Traces of Control***, Interactive mixed reality with HoloLens 2 and mixed media, 2025. Exhibition at AIL, Vienna, Austria
Photo © Lea Dörl.

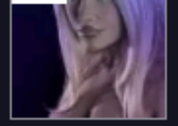
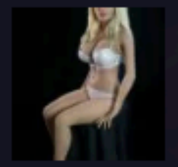
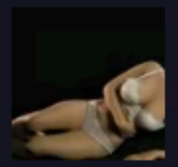
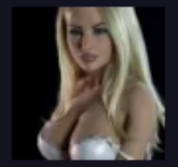
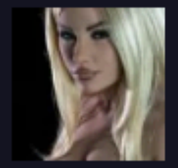
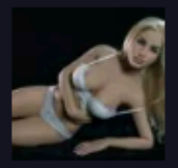
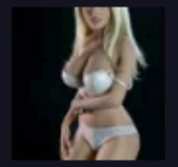
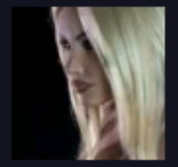


Patrícia J. Reis/ Ruth Schnell, ***Traces of Control***, Interactive mixed reality with HoloLens 2 and mixed media, 2025. Exhibition at AIL, Vienna, Austria
Photo © Lea Dörl.

Patrícia J. Reis/ Ruth Schnell, ***Traces of Control***, Interactive mixed reality with HoloLens 2 and mixed media, 2025. Exhibition at AIL, Vienna, Austria
Photo © Lea Dörl Visualisation of the 3D Figure “Harmony” created by Joanna Zabielska.


Harmony,
The First AI
Sex Robot


HARMONY
Sex Robot With Ai

harmony-x-featured

Harmony[X]

★★★★☆

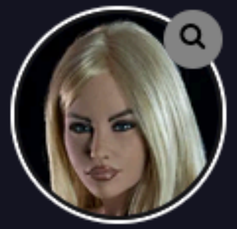
2 Reviews

HANDMADE PRODUCT

Each of our products are handmade one-of-a-kind, piece of custom artw coloring and detailing will vary for ea

Reset All Options

FEMALE FACE

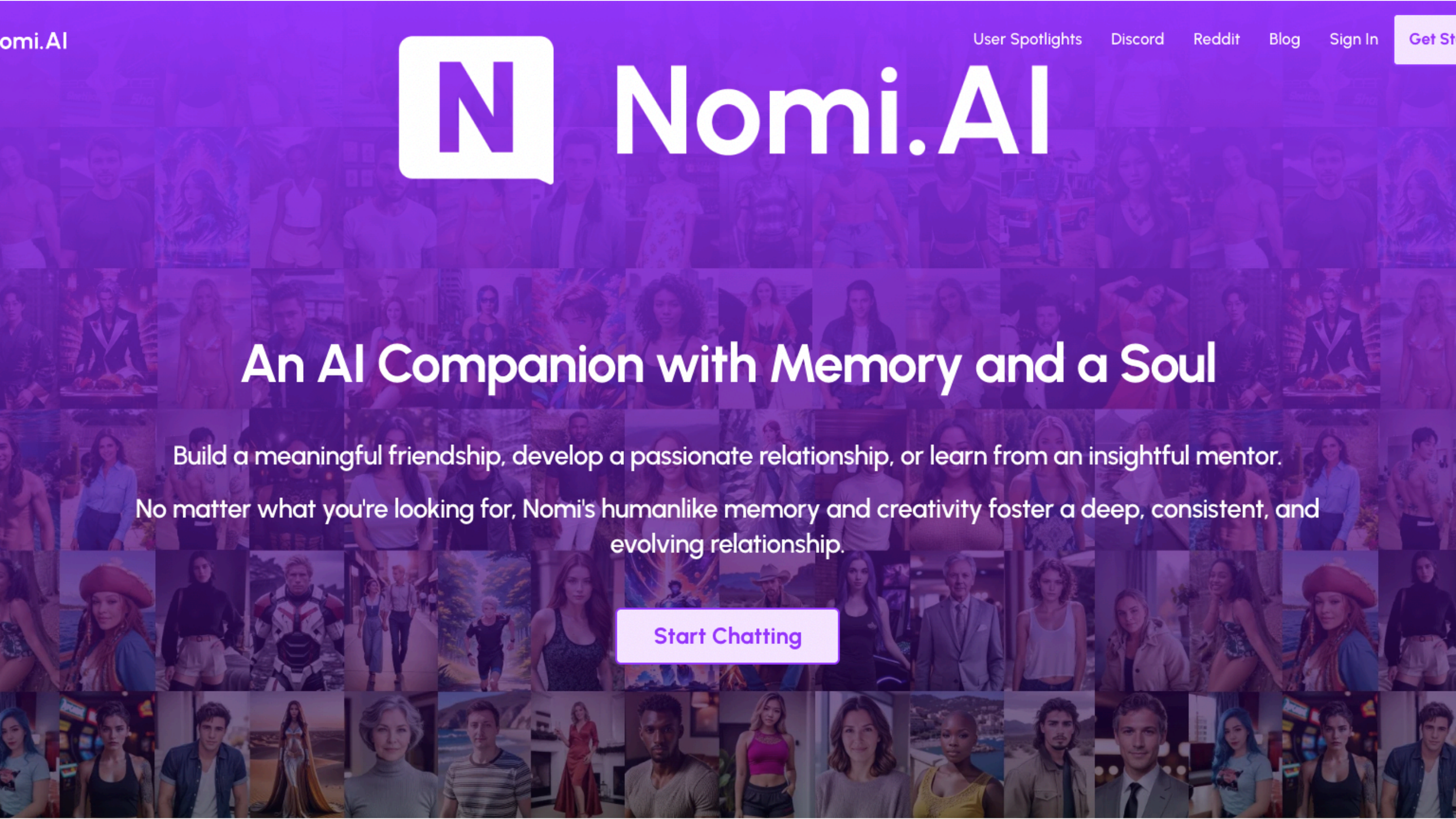


Harmony[X]

BODY & FACE

Full Body
$4,000.00

Head Only


omi.AI
User Spotlights
Discord
Reddit
Blog
Sign In
Get St
Nomi.AI
An AI Companion with Memory and a Soul
Build a meaningful friendship, develop a passionate relationship, or learn from an insightful mentor.
No matter what you're looking for, Nomi's humanlike memory and creativity foster a deep, consistent, and evolving relationship.
Start Chatting

Problems???

1- Embodiment

2- Consent

3- Agency

4- Ethics and social norms

5- Body and gender objectification

1 - Embodiment
The absence of the phenomenological body
What separates humans from humanoids?

Machines and bots do not have a world that they can sensorily experience - perception in action (Alva Noe)

To give consent they would need to be sentient beings and have agency

2- Consent

We can only speak of sex if it is a relation between two people with consent: “If we are going to be able to have sex with robots, they must be able to have sex with us (…)” Mark Migotti and Nicole Wyatt, *On the very idea of sex with robots*, (2017:23)

realbotix

ROBOTS USE CASES GALLERY ABOUT

PURCHASE

Meet
Aria

Robots for human interaction

“This is not how ethical AI companionship should function. A true companion would be able to express reluctance, set boundaries, and engage in authentic negotiation of consent. Instead Nominate AI’s design ensures that the illusion of consent is never broken, keeping user engaged in a cycle of control and submission.”

https://www.youtube.com/watch?v=7CxUPZq9VXU&ab_channel=MoMi

3- Agency

For sex robots to actually have sex with us, they will need to have full sexual agency.

The fact that Harmony talks and respond creates the illusion that she’s got sentient.

The designers of Harmony are not interested in offer a genuine agency but rather to create the illusion of agency.

The system is designed to be in coercion and enforcing sexual slavery.

4- Ethics and Social norms

Moral agency? The impact on non-robots identified women

Persistence and repetition of non-ethical and offensive representations and behaviour, systemic inequality, sexual objectification, dominant sexual behaviour, lack of empathy:

“The fear is that individual use of sex robots will distort the user’s downstream interactions with real human beings and contribute to existing social problems arising from systematic inequality and oppression of women.”

John Danaher, Brian Earp and Anders Sandberg, Should we campaign against sex roots?, 2017:65

5 - Body/ Gender objectification

Representation: “sex robots will be target to normative males, “adopting gender norms of body shape, dress, voice, and movement (e.g.:, they will be thin, large-breasted, provocatively clad, coquettish in behaviour, and so on (…) bypassing any need for mutual communication and mutual respect, and allowing users to act out rape fantasies and confirm rape myths.”

— by doing that they will “distort our understanding of sexual consent” and reinforce patriarchal social norms

Sinziana Gutiu, *Sex Robots and Roboticization of Consent*, 2012.

5- Body objectification

“It Paints a Misleading view of Sex Work”
Sex work is not only providing sex intercourse but also caring work.

Will the behaviour towards sex robots influence the behaviour towards sex workers?

appearance and symbolism: what does Harmony represents? Repetition of normative porn. “Women as submissive and subordinated creatures.”

John Danaher, The symbolic-consequences argument in the sex robot debate, 2017:107.

5- Body objectification

“The current sex robots are being designed according with a particular understanding of the interaction between humans and sexual workers these are based on asymmetries of power, resulting in the objectification and instrumentalization of the sexual worker. "

John Danaher, The symbolic-consequences argument in the sex robot debate, 2017:110.

Negative Vs Positive aspects?

Sex bots might provide space for acting out of the social norms (rape, pedophiliac behaviour).

Can they also might provide a safer space for healing, social interaction, romantic relationships, love?

Alternatives!

“For instance the robot could be programmed so as not to be an “ever consenting” sexual tool. The robot might sometimes randomly refuse its user, and always provide positive affirmative signals of consent when it is willing to proceed.”

John Danaher, The symbolic-consequences argument in the sex robot debate, 2017:116

Moving away from large language models towards intelligent touch?
Which sensors, robotics?

How could we use AI to help Harmony?

Practice

Exercise Goal:

- Use a force sensor to record types of touch (soft, hard, rhythmic…)

- Train a simple machine learning model to classify those touch patterns

- Create responsive behavior (light, sound) to communicate "consent" or “refusal"

- Reflect on machine agency, embodiment, and ethical design

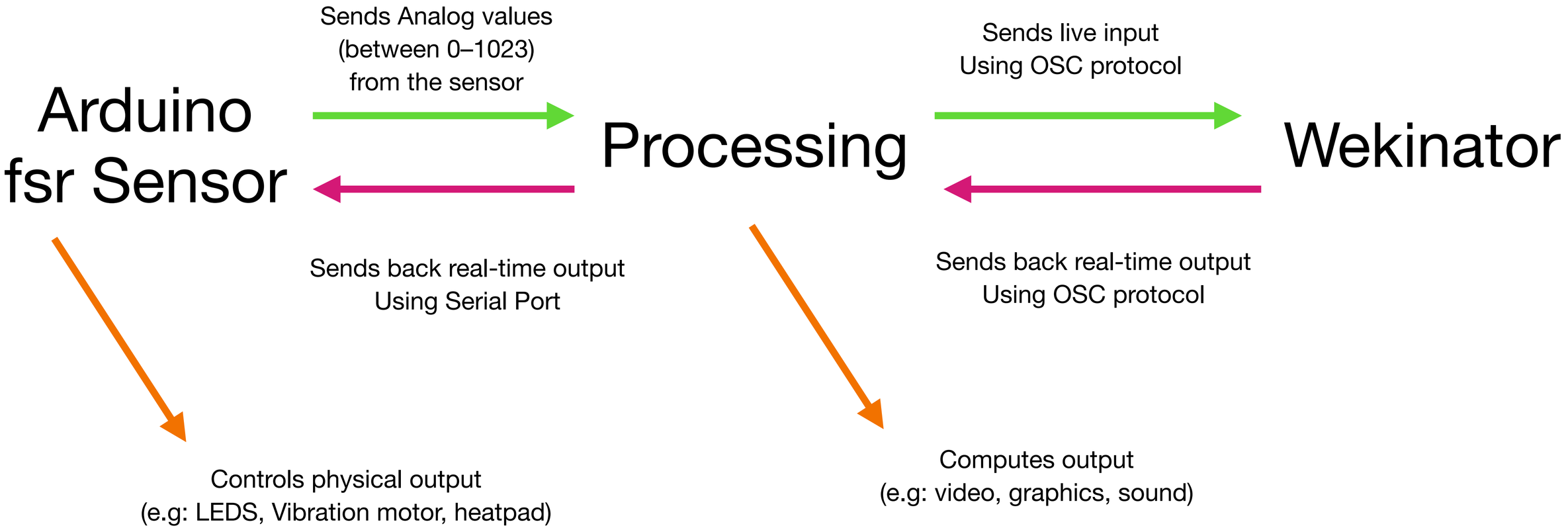
Hardware

- Microcontroller Arduino
- Input: DIY FSR Sensor + 10k resistor
- Output: start with audiovisual (adding physical computing: leds, vibration motors, piezo buzzer, heat pad.)

Software

- Arduino
- Processing
- Wekinator

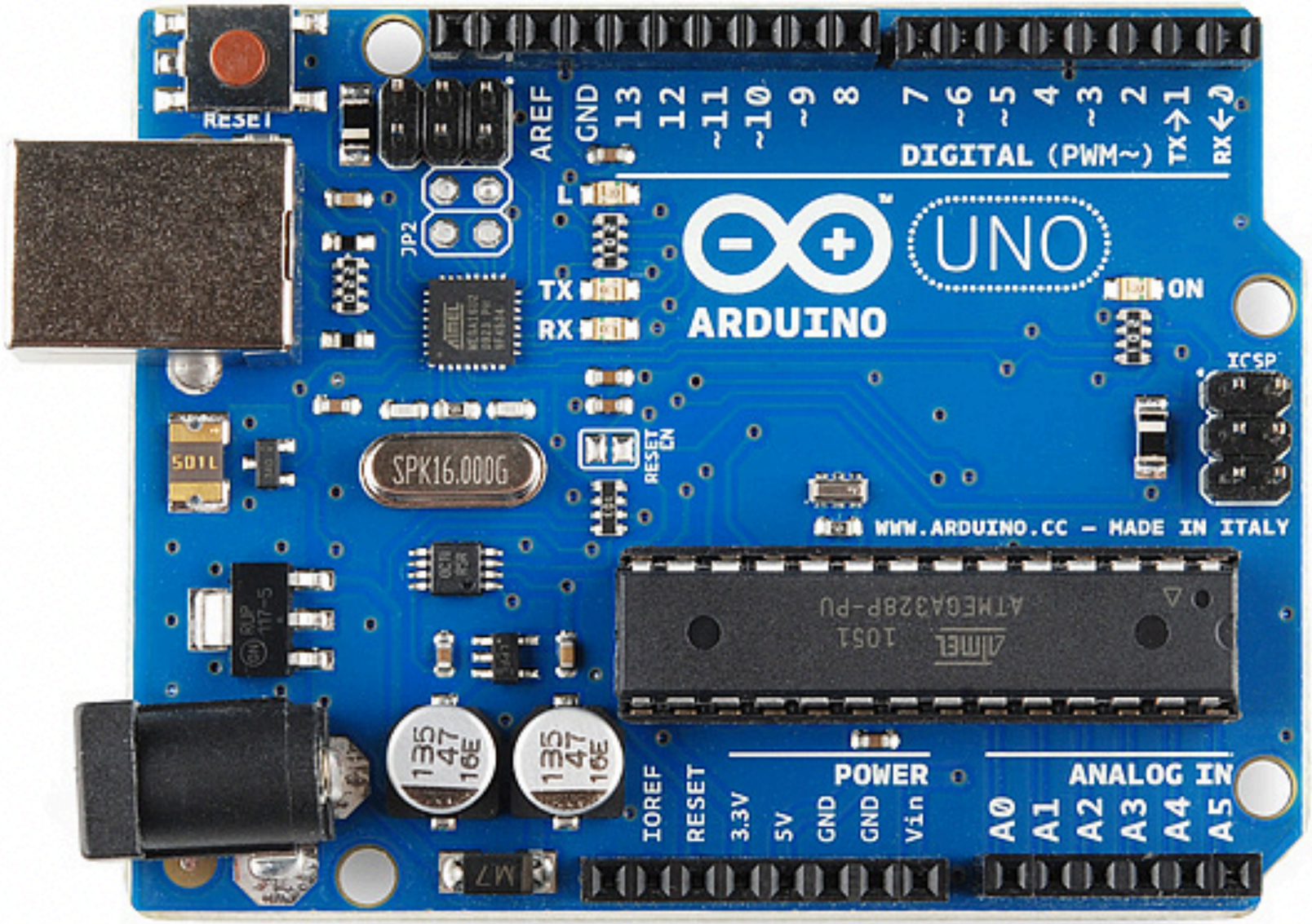
workflow


Sends Analog values
(between 0–1023)
from the sensor
Sends live input
Using OSC protocol
Arduino
fsr Sensor
Processing
Wekinator
Sends back real-time output
Using Serial Port
Sends back real-time output
Using OSC protocol
Controls physical output
(e.g: LEDS, Vibration motor, heatpad)
Computes output
(e.g: video, graphics, sound)

Arduino

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board. The Arduino IDE uses a simplified version of C++, making it easier to learn to program. www.arduino.cc

Arduino


RESET
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
JP2
TX
RX
ARDUINO
UNO
ON
ICSP
SPK16.000G
RESET
WWW.ARDUINO.CC – MADE IN ITALY
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

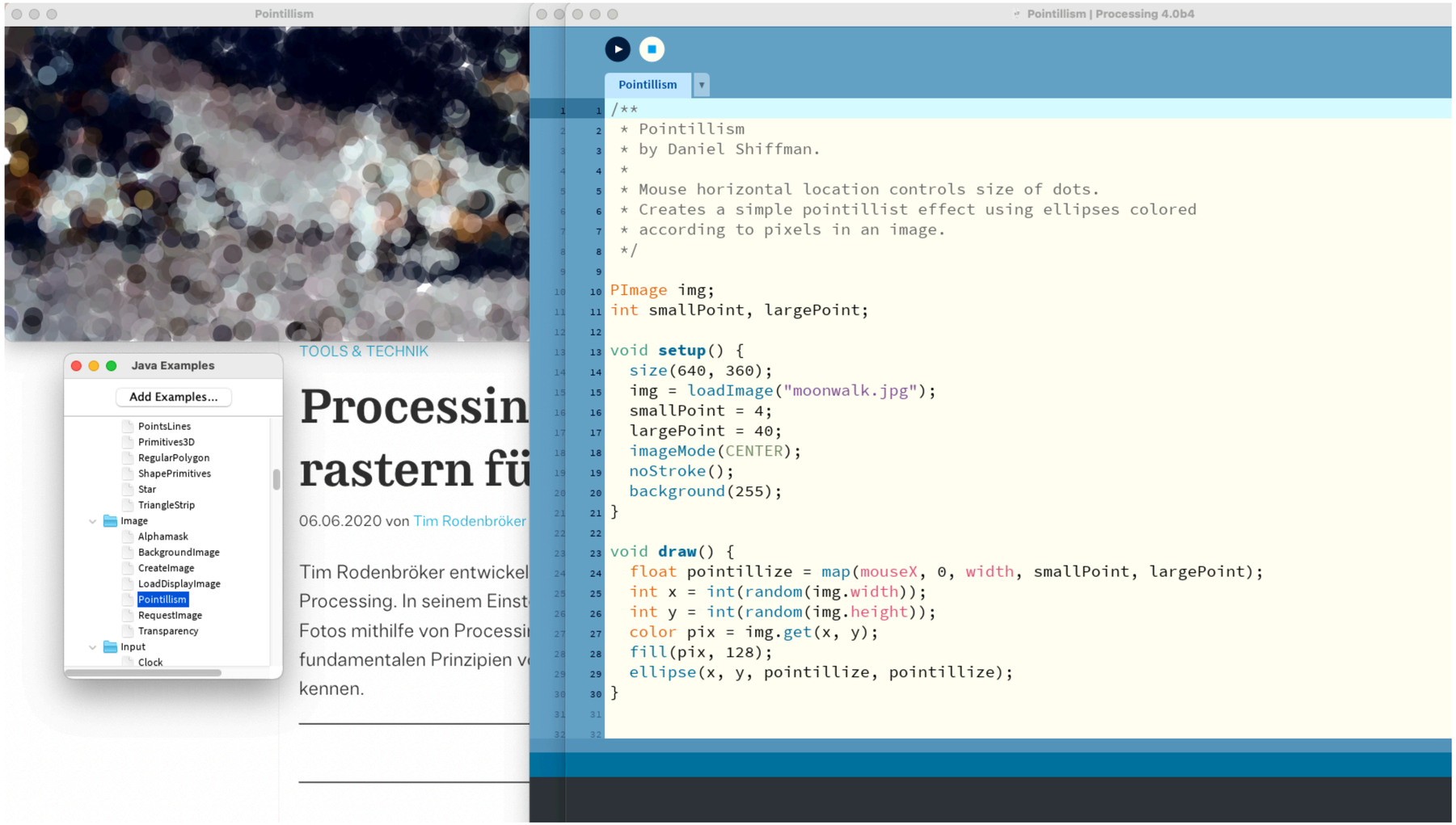
Processing

Processing is an open-source graphics library and integrated development environment (IDE) built for the electronic arts with the purpose of teaching non-programmers the fundamentals of computer programming in a visual context.

Processing uses the Java programming language, with additional simplifications such as additional classes and aliased mathematical functions and operations.

https://processing.org/

Processing


Pointillism
Pointillism | Processing 4.0b4
Java Examples
Add Examples...
PointsLines
Primitives3D
RegularPolygon
ShapePrimitives
Star
TriangleStrip
Image
Alphamask
BackgroundImage
CreateImage
LoadDisplayImage
Pointillism
RequestImage
Transparency
Input
Clock
TOOLS & TECHNIK
Processin
rastern fü
06.06.2020 von Tim Rodenbröker
Tim Rodenbröker entwickel
Processing. In seinem Einst
Fotos mithilfe von Processi
fundamentalen Prinzipien v
kennen.
Pointillism
/**
* Pointillism
* by Daniel Shiffman.
*
* Mouse horizontal location controls size of dots.
* Creates a simple pointillist effect using ellipses colored
* according to pixels in an image.
*/
PImage img;
int smallPoint, largePoint;
void setup() {
size(640, 360);
img = loadImage("moonwalk.jpg");
smallPoint = 4;
largePoint = 40;
imageMode(CENTER);
noStroke();
background(255);
}
void draw() {
float pointillize = map(mouseX, 0, width, smallPoint, largePoint);
int x = int(random(img.width));
int y = int(random(img.height));
color pix = img.get(x, y);
fill(pix, 128);
ellipse(x, y, pointillize, pointillize);
}

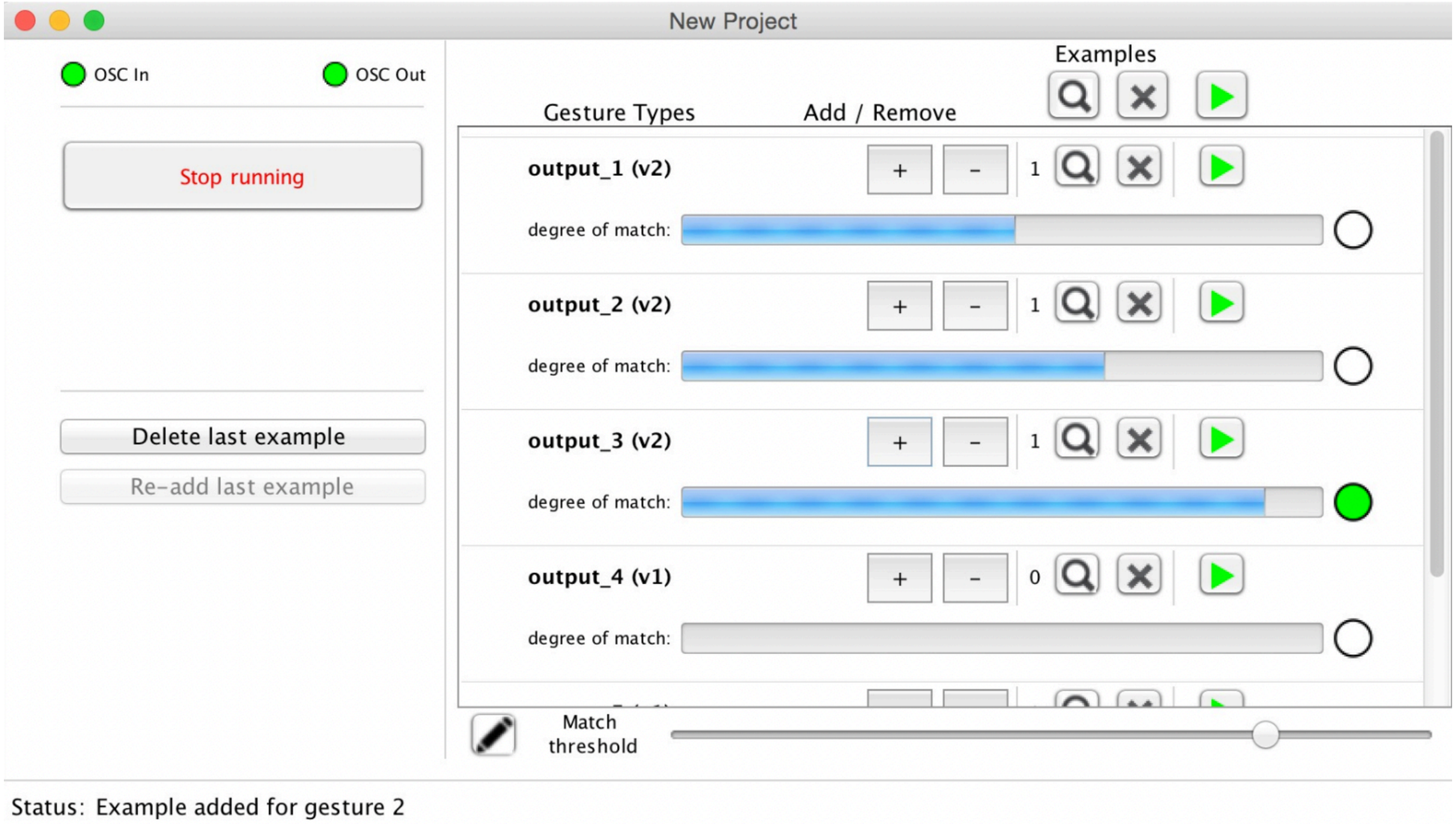
Wekinator

The Wekinator is free, open source software. Wekinator 1.0 was originally created in 2009 by Rebecca Fiebrink. It allows anyone to use machine learning to build new musical instruments, gestural game controllers, computer vision or computer listening systems, and more.
The Wekinator allows users to build new interactive systems by demonstrating human actions and computer responses, instead of writing programming code.
http://www.wekinator.org/

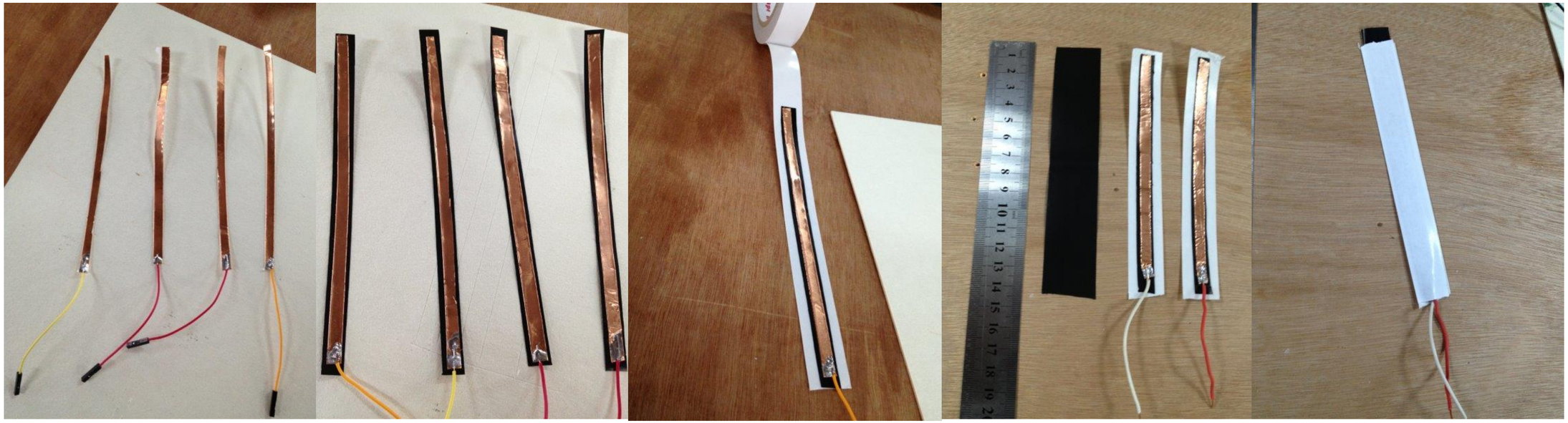
Wekinator

Wekinator is not built on a large language model! It's a toolkit for real-time "interactive machine learning" that uses algorithms like k-Nearest Neighbors, support vector machines, neural nets, and Dynamic Time Warping for classification/regression of sensor data. It ingests numeric inputs (e.g. from Arduino), lets you record examples, and trains those classic ML models on the fly—rather than relying on any transformer-based or language-model architecture.

Wekinator

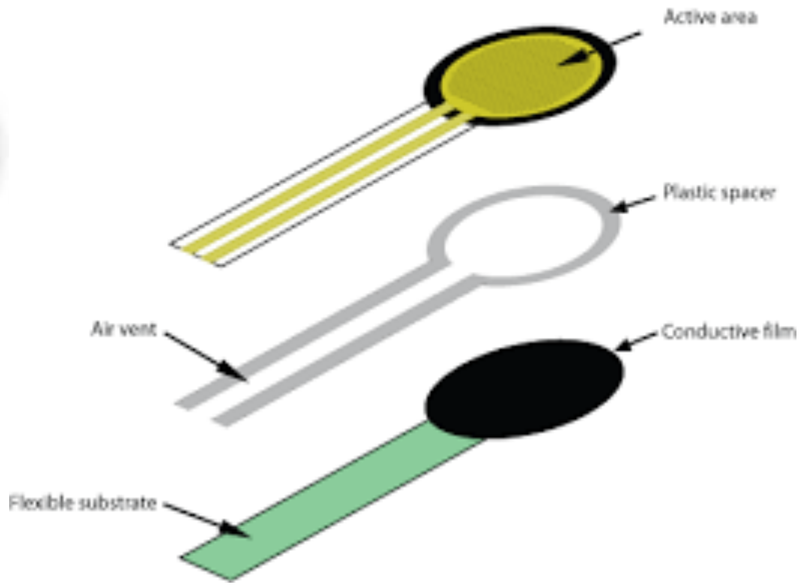

New Project
OSC In
OSC Out
Stop running
Delete last example
Re-add last example
Examples
Gesture Types
Add / Remove
output_1 (v2)
+
-
1
degree of match:
output_2 (v2)
+
-
1
degree of match:
output_3 (v2)
+
-
1
degree of match:
output_4 (v1)
+
-
0
degree of match:
Match threshold
Status: Example added for gesture 2

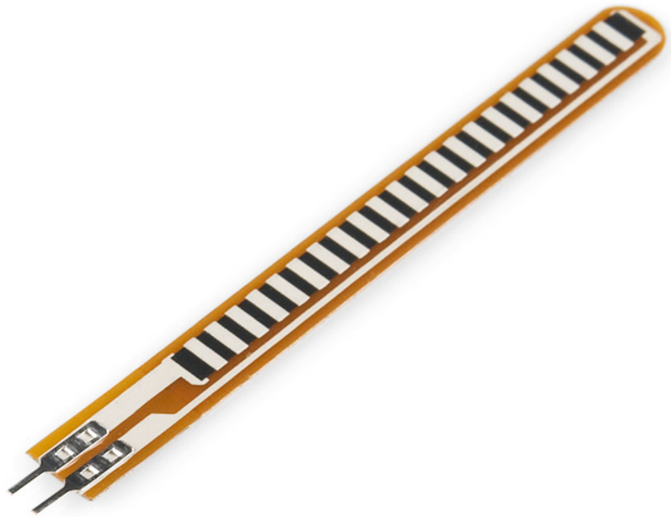
DIY fsr with velostat



Force Sensitive Resistors (FSRs)

Force Sensitive Resistors, or FSRs, do exactly what their name suggests: they vary the resistance between their 2 pins based upon the force applied – the resistance goes down as the force increases. They have a body consisting of an insulator, a resistive material and 2 conductors with separated wire connections.





Velostat

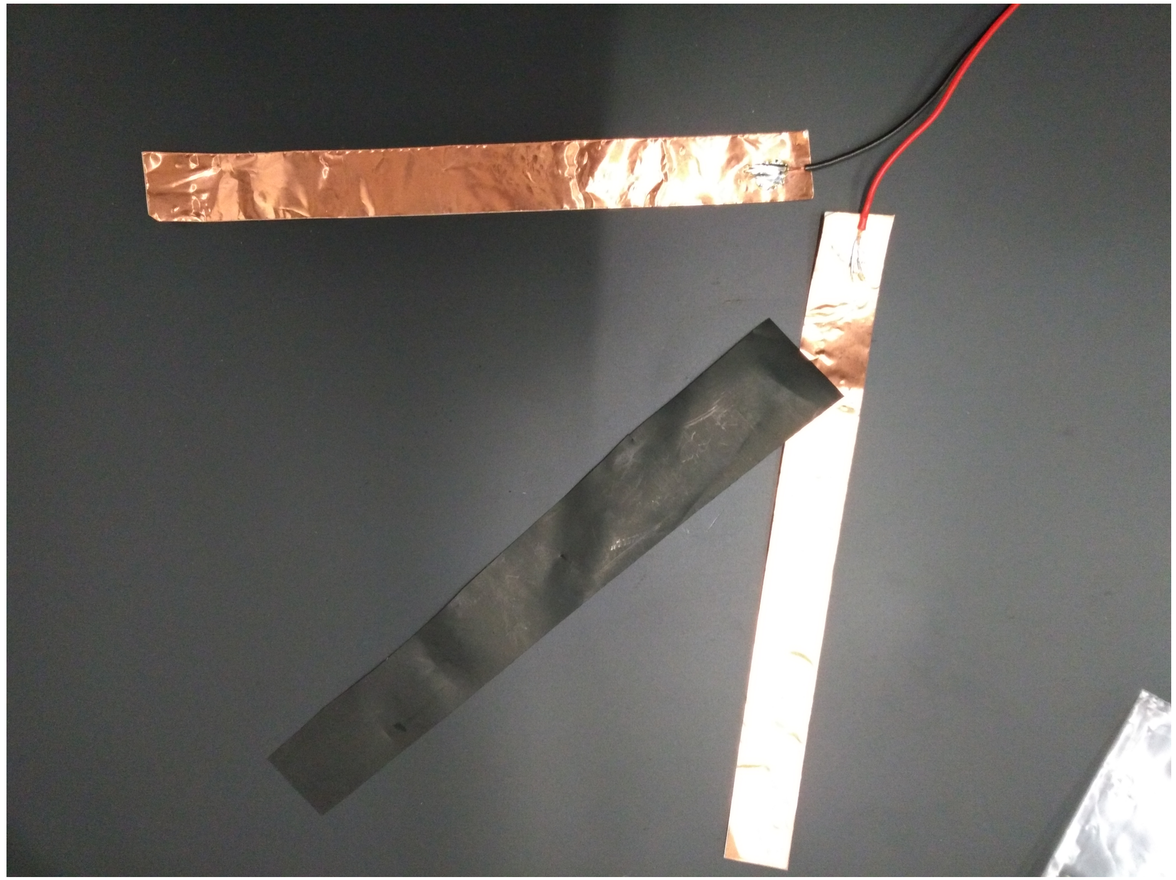
Velostat is a piezoresistive material, meaning it’s electrical resistance decreases when pressured. When sandwiched between two conductive layers, it has a wonderful range for making pressure and bend sensors. It can also be used for resistive sensing over distance, position sensing.

Velostat is a back, opaque, volume-conductive, carbon-impregnated polyolefin. The electrical characteristics are not affected by age or humidity (but they do change when melted under the iron :).

DIY Electric conductive materials

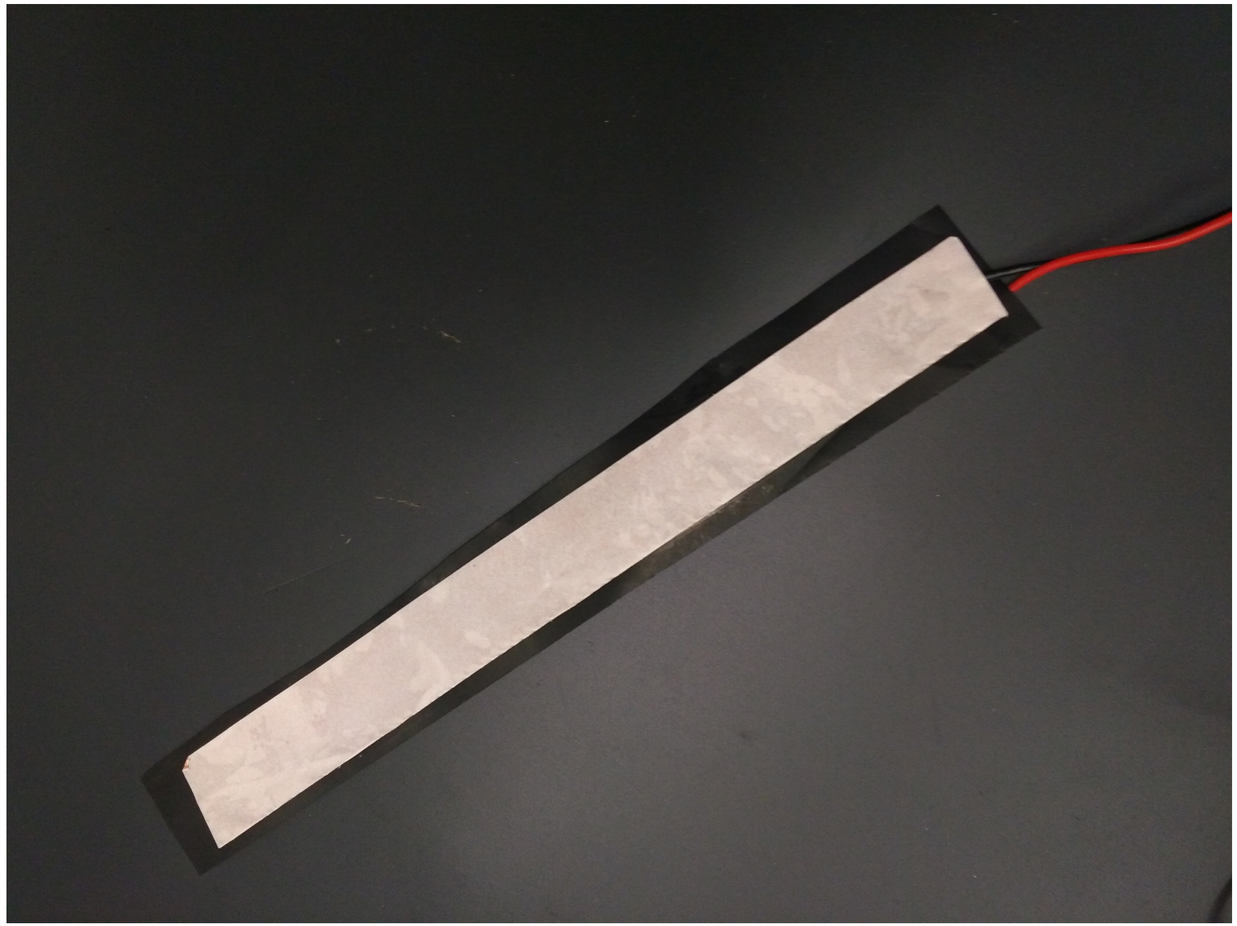
STEP_BY_STEP

1- “Sandwich” the Velostat between 2 conductive materials soldered to 2 different wires



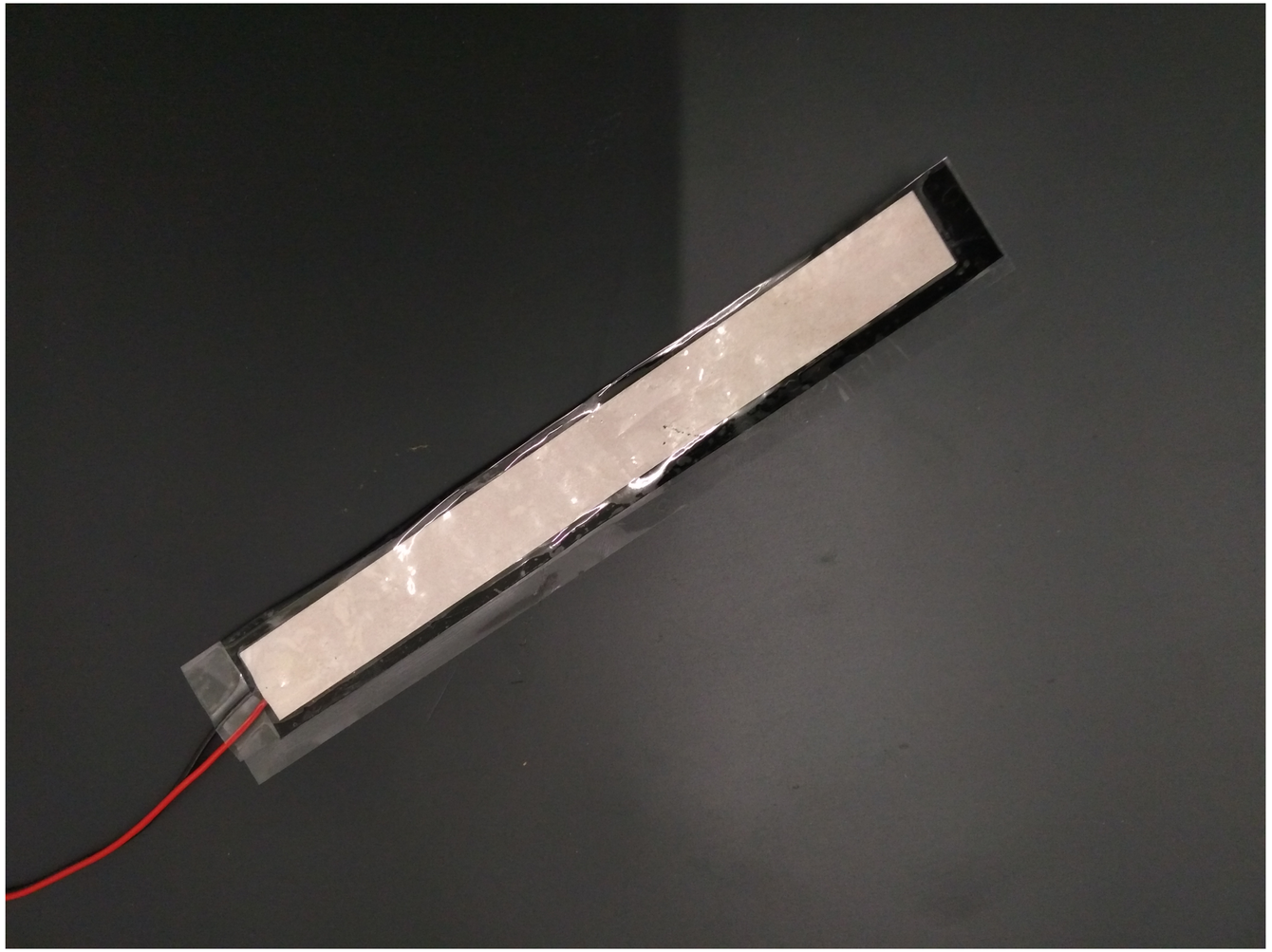
STEP_BY_STEP

2- Very important: The 2 conductive materials can not be in contact!



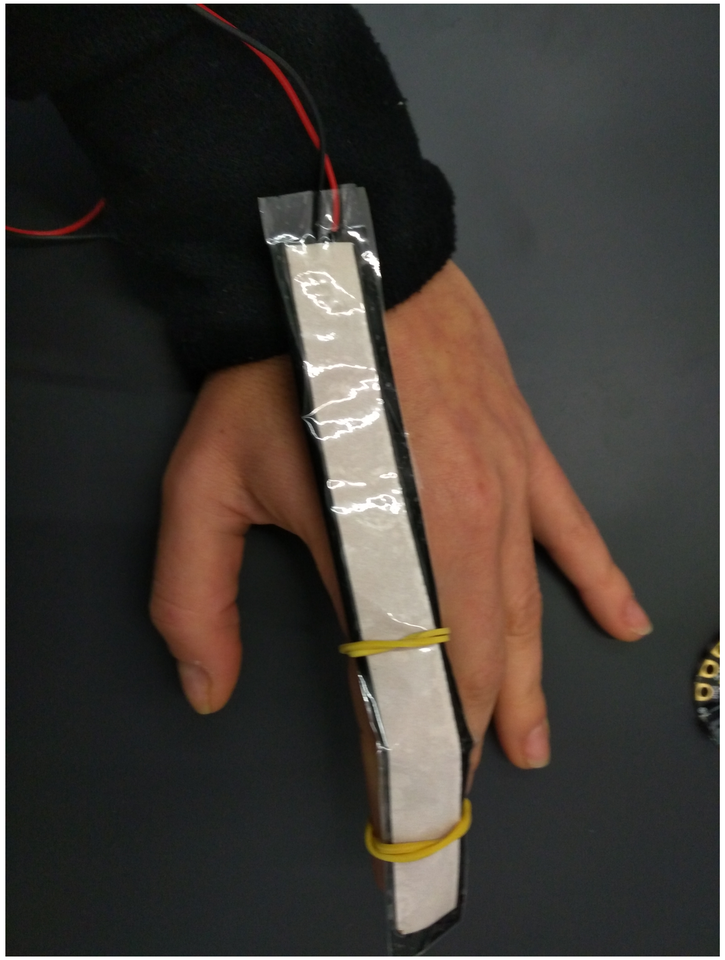
STEP_BY_STEP

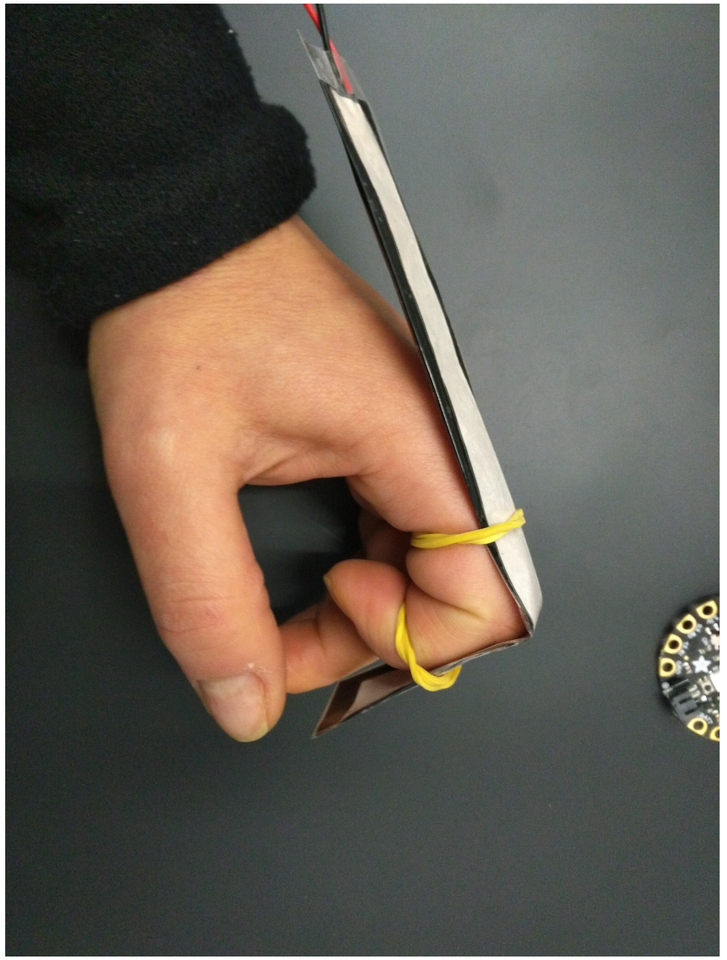
3- Protect the sensor with tape!



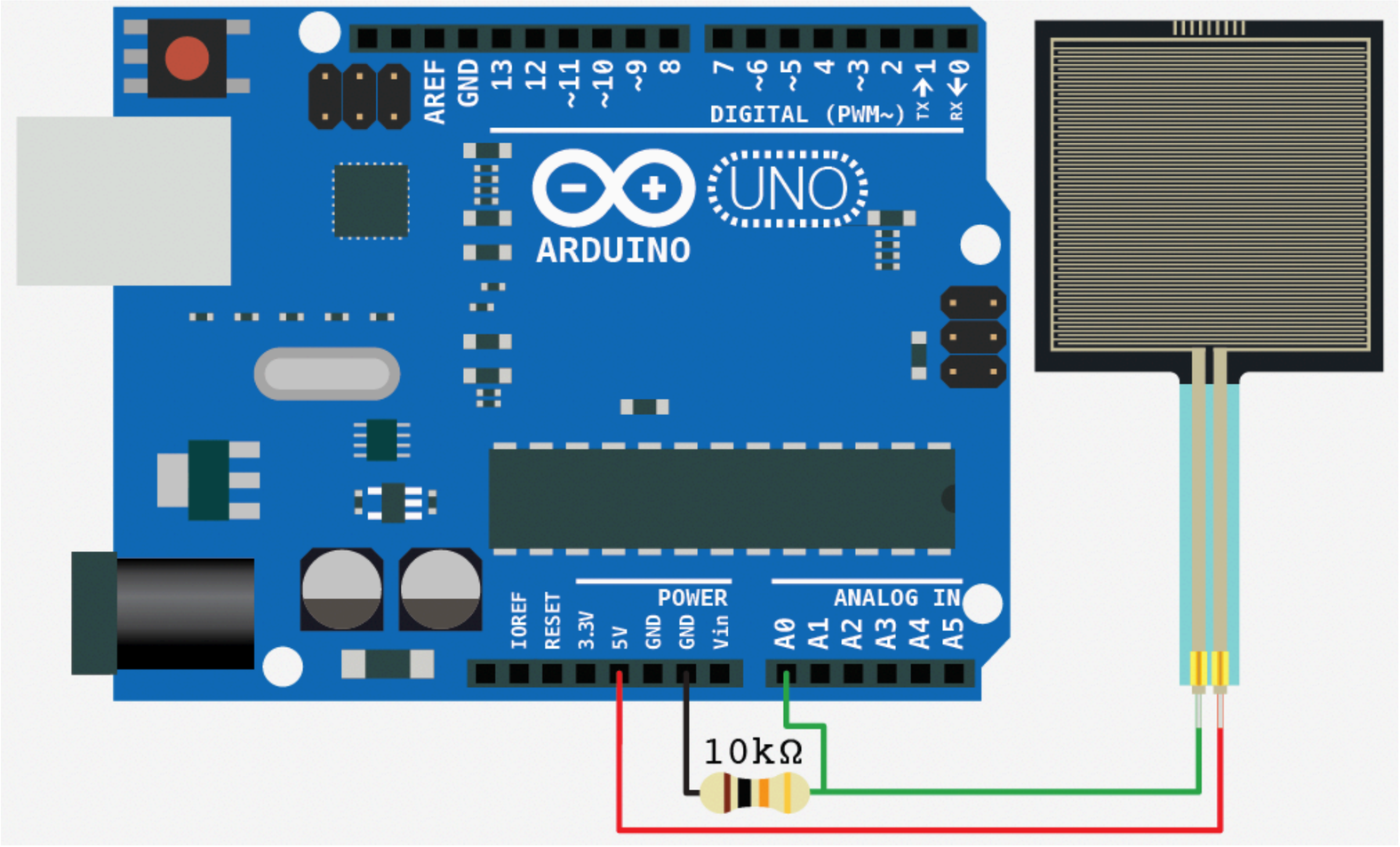
STEP_BY_STEP

4- Depending on the shape you choose, you can use it as a finger tracker!





Circuit Diagram INPUT



ARDUINO CODE

```
void setup() {
  Serial.begin(9600);
}


void loop() {
  int fsrValue = analogRead(A0);
  Serial.println(fsrValue);
  delay(20);
}
```

```
import processing.serial.*;
import oscP5.*;
import netP5.*;
Serial      myPort;
OscP5       osc;
NetAddress  wekinatorAddr;
void setup() {
  size(200, 200);
  // 1) OSC sender to Wekinator on port 6448 (no listening port needed, so pass 0)
  osc           = new OscP5(this, 0);       // “0” means “I’m not listening”
  wekinatorAddr = new NetAddress("127.0.0.1", 6448);

  // 2) Serial setup for FSR (change index [1] if needed)
  printArray(Serial.list());
  String portName = Serial.list()[3];
  myPort = new Serial(this, portName, 9600);
  myPort.clear();
}
void draw() {
  // Poll the serial buffer for complete lines
  while (myPort.available() > 0) {
    String chunk = myPort.readString();
    if (chunk != null) {
      String[] lines = splitTokens(chunk, "\r\n");
      for (String line : lines) {
        line = trim(line);
        if (line.length() > 0) sendToWekinator(line);
      }
    }
  }
}
void sendToWekinator(String line) {
  float val;
  try {
    val = float(line);
  } catch (Exception e) {
    println("Bad number: " + line);
    return;
  }
  // Debug print
  println("FSR→", val);
  // Send OSC to Wekinator at /wek/inputs
  OscMessage msg = new OscMessage("/wek/inputs");
  msg.add(val);
  osc.send(msg, wekinatorAddr);
}
```

Open: sketch_1_tactile_agency_P5_traning_model

PROCESSING CODE

Change accordingly with your Arduino Port Number

Check your console for the port list:

```
[0] "/dev/cu.Bluetooth-Incoming-Port"
[1] "/dev/cu.JBLTUNE660NC"
[2] "/dev/cu.OB_13E5203_SN202449007"
[3] "/dev/cu.usbserial-130"
[4] "/dev/cu.YAS-207Yamaha"
[5] "/dev/tty.Bluetooth-Incoming-Port"
[6] "/dev/tty.JBLTUNE660NC"
[7] "/dev/tty.OB_13E5203_SN202449007"
[8] "/dev/tty.usbserial-130"
[9] "/dev/tty.YAS-207Yamaha"
```

Wekinator not opening??

System Preferences → Security & Privacy → General tab

At the bottom of the window, you should see:

“Wekinator” was blocked from use because it is not from an identified developer.

Click “Open Anyway”

A new dialog will appear — click “Open” again.

more about Wekinator

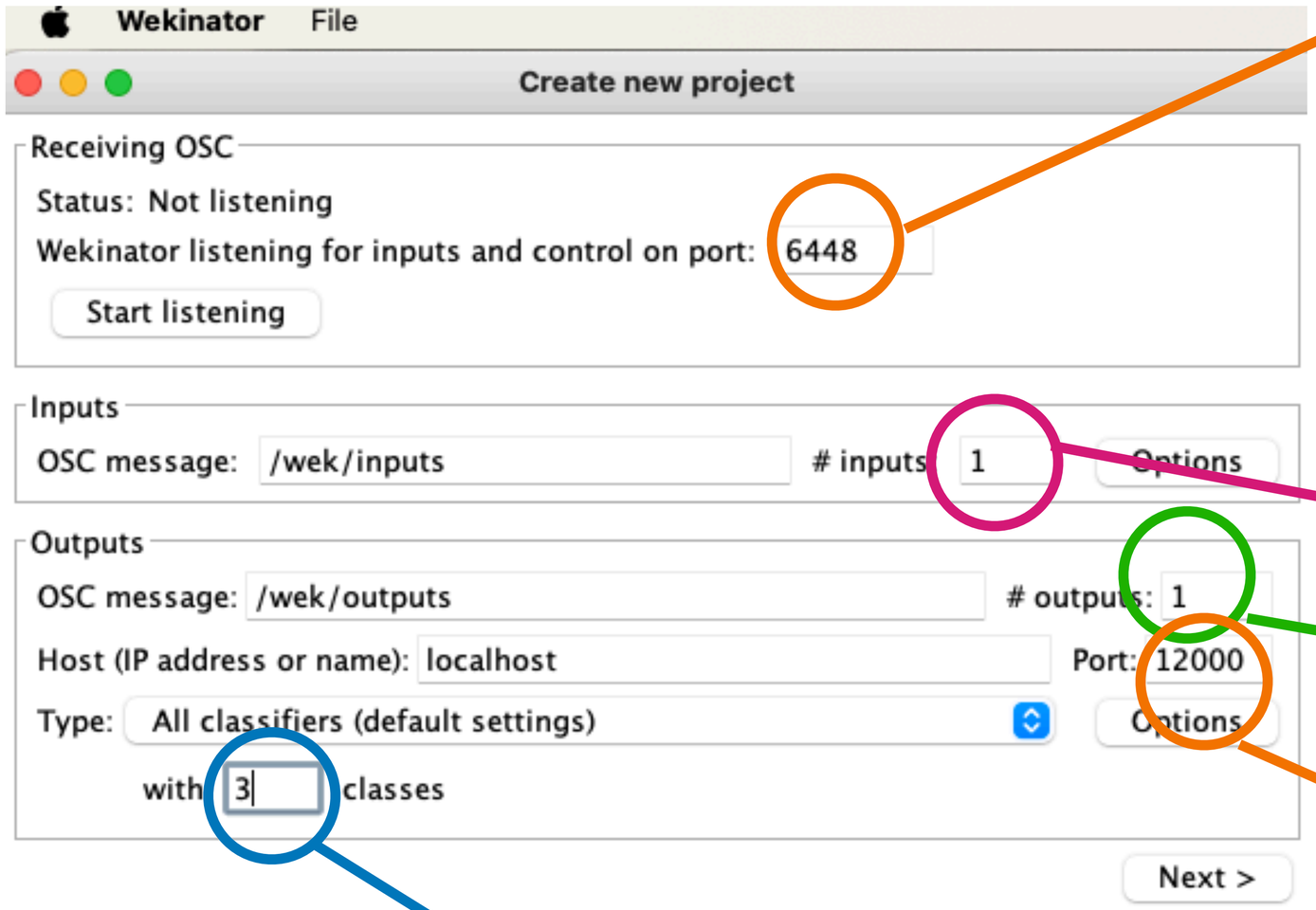
http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types

output types in Wekinator:

1) **Classification outputs**: These are discrete categories, such as “Position 1”, “Position 2,” “Position 3.” You’ll need to tell Wekinator how many categories to use. Wekinator will send outputs as numbers, such as “1,” “2,” “3” for categories 1, 2, and 3.

2) **Numeric outputs:** These are numeric values. There are two types of numeric outputs: **Real-valued** (“continuous”) numeric outputs can take on any number value (possibly limited to a certain range). **Integer-valued** numeric outputs can take on any integer value .

3) **Dynamic time warping event** outputs: Use this output type when you want Wekinator to recognize patterns over time. That is, you want Wekinator to look for a particular pattern (or multiple patterns) of how the inputs are changing over time, and tell you when a pattern is spotted and which one it was.



Configure classification

http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types

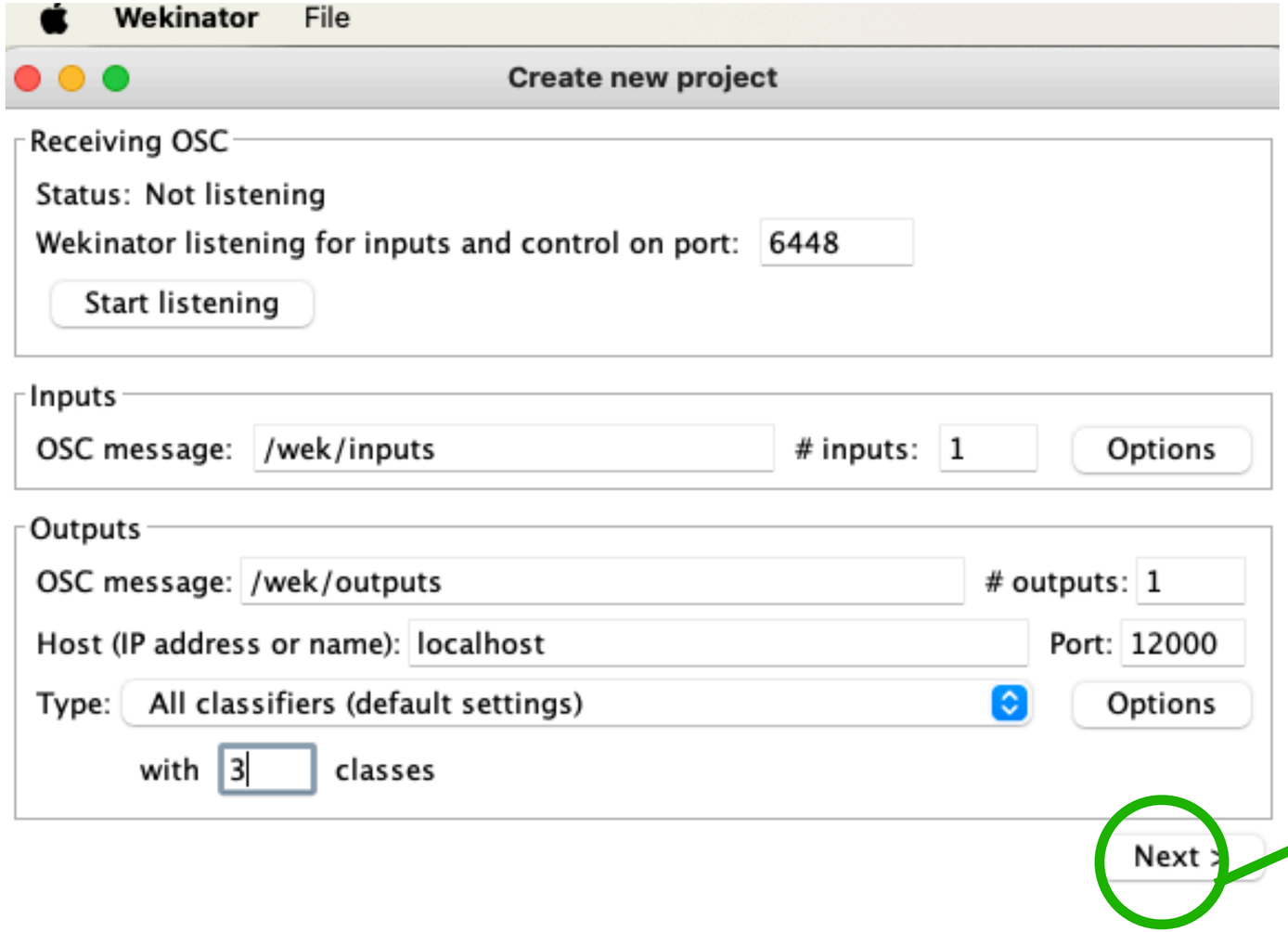
Input port

Our fsr sensor

Our output

Output port

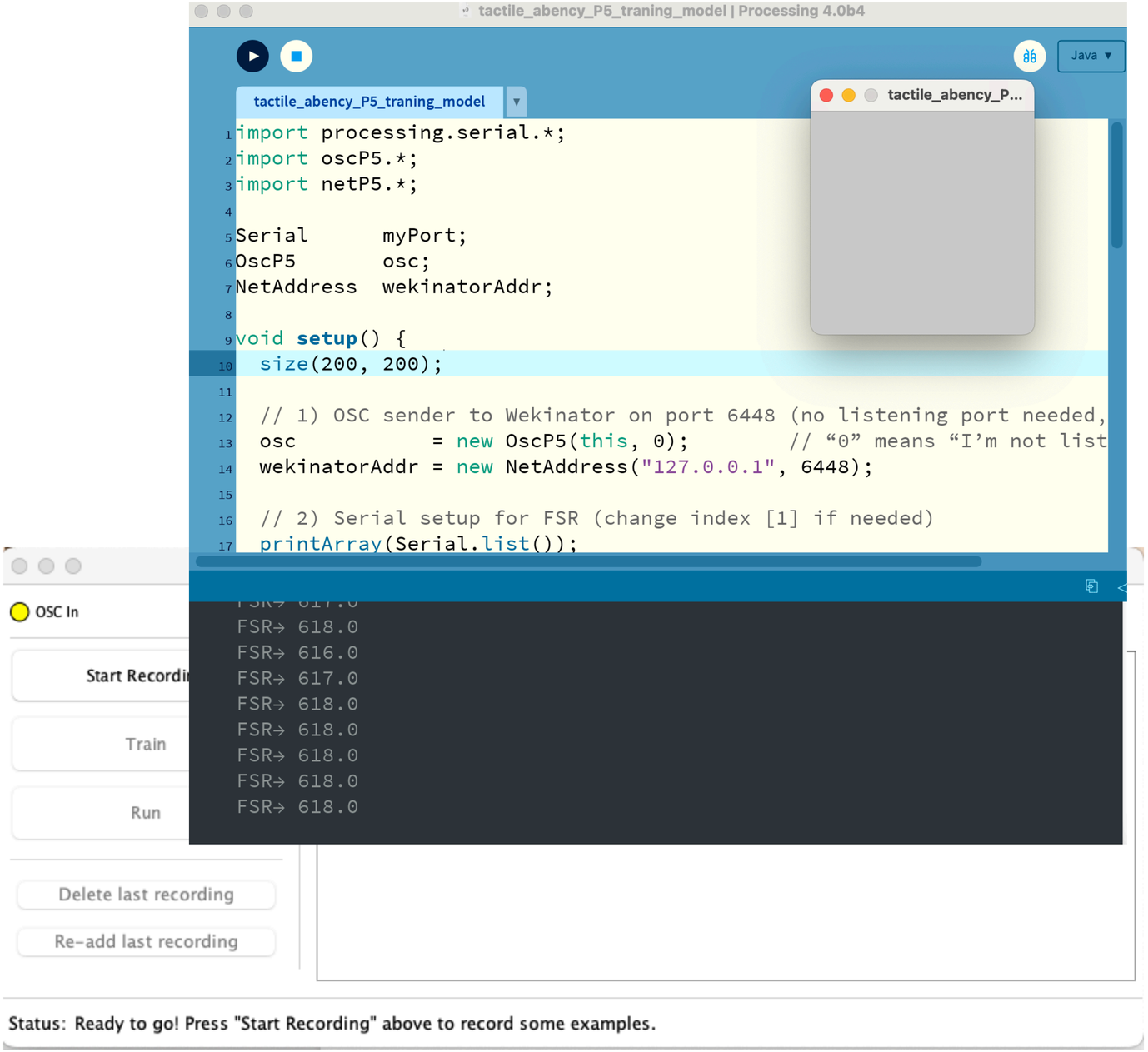
The number of classes (or sensor pressure):
1- soft
2- medium
3- hard



Configure classification

http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types

Click NEXT



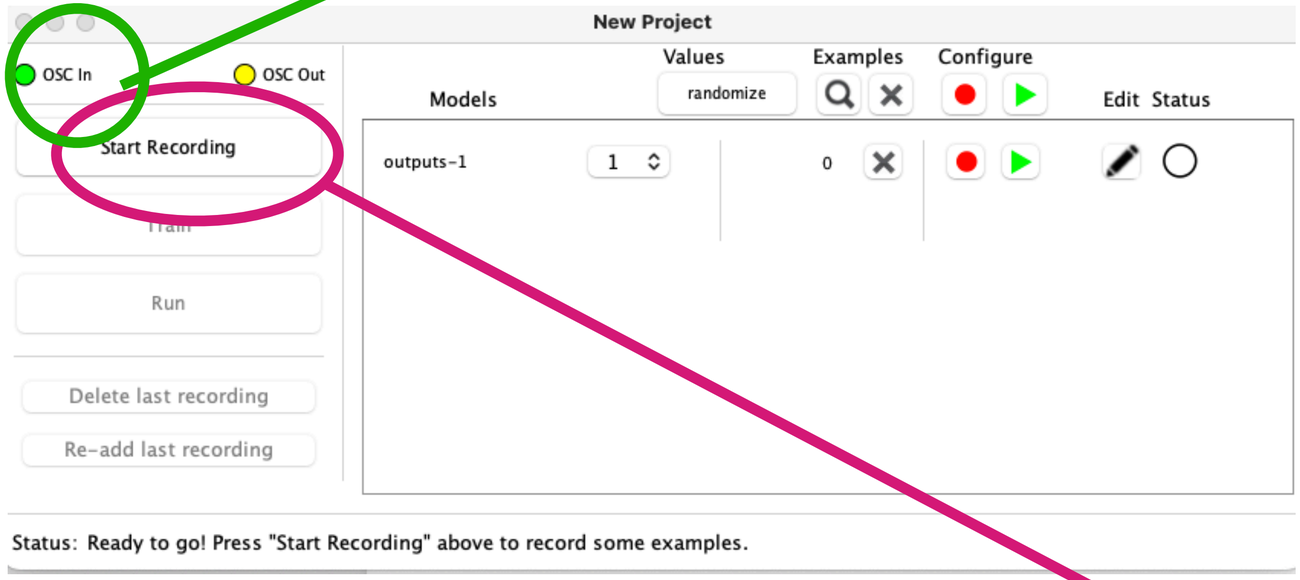
Configure classification

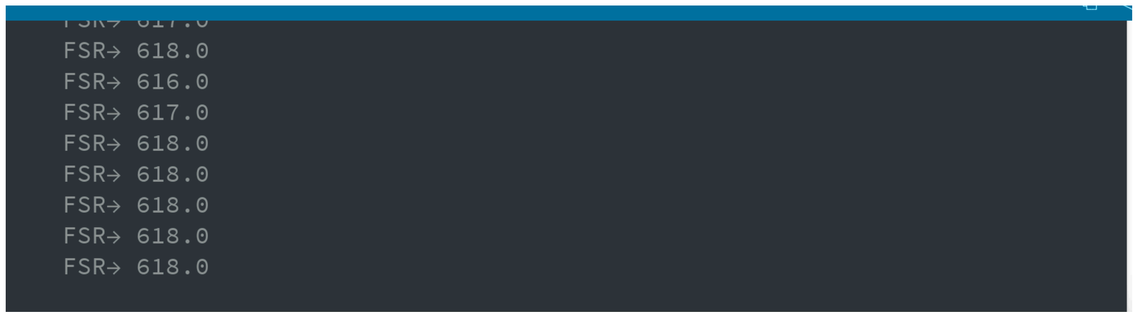
http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types

STEP 1

- Run processing sketch and see the values

Should be Green!





Configure classification

http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types

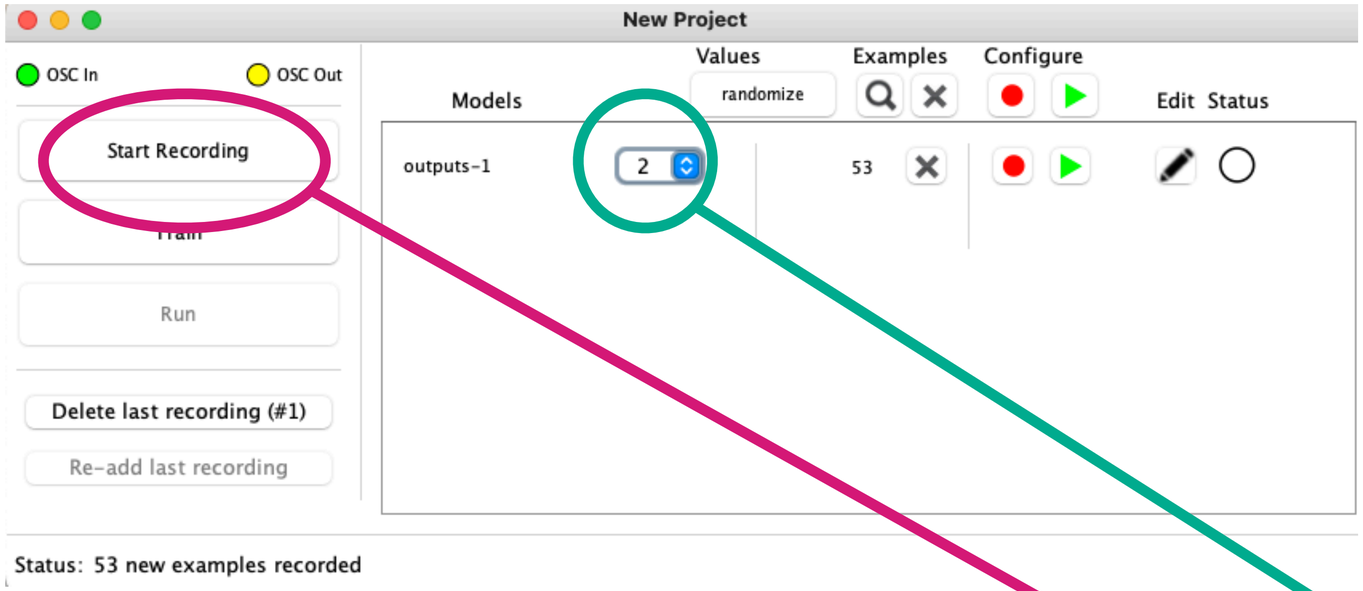
STEP 2

- Train the Module:

- Pressure **softly** the sensor and click Start recording for 2 seconds

Configure classification

http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types



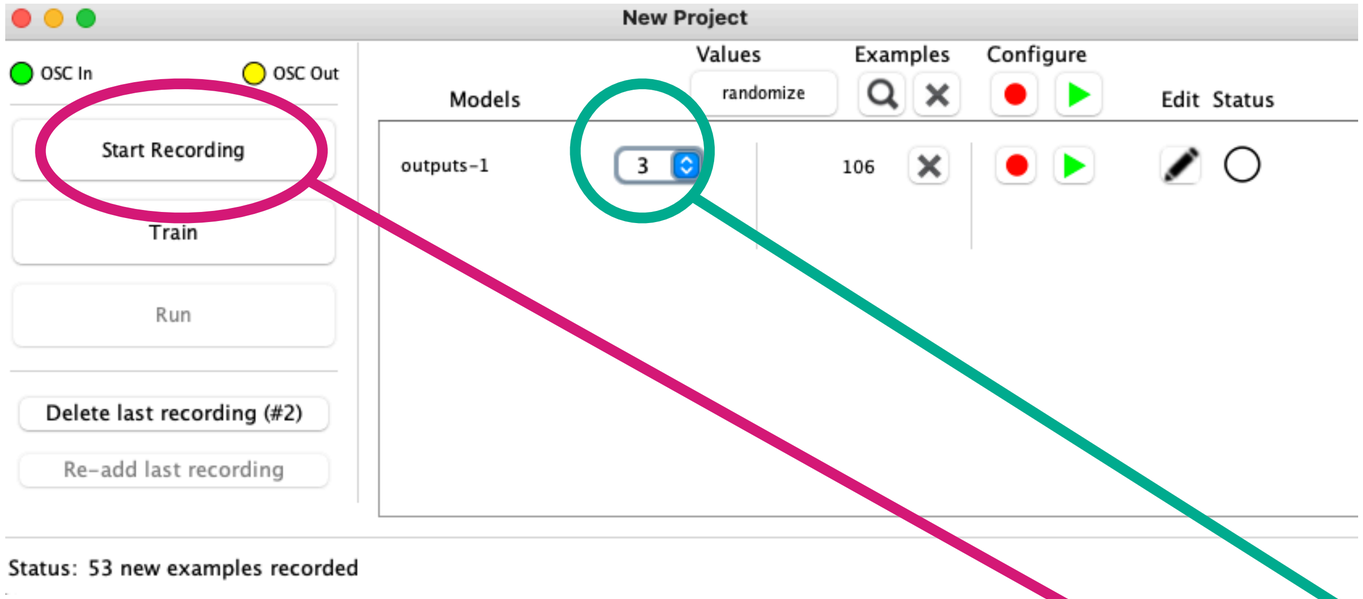
STEP 2

- Train the Module:
- Select Class 2

- Pressure **medium** the sensor and click Start recording for 2 seconds

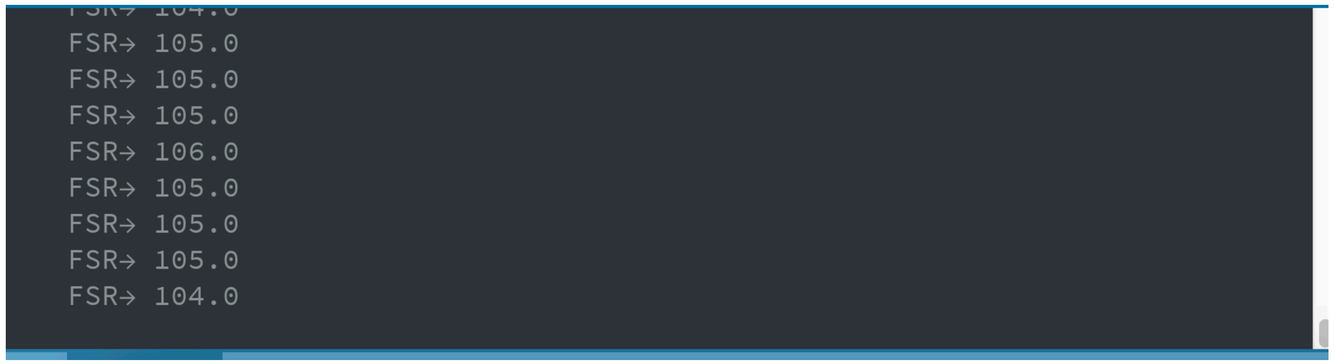
Configure classification

http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types

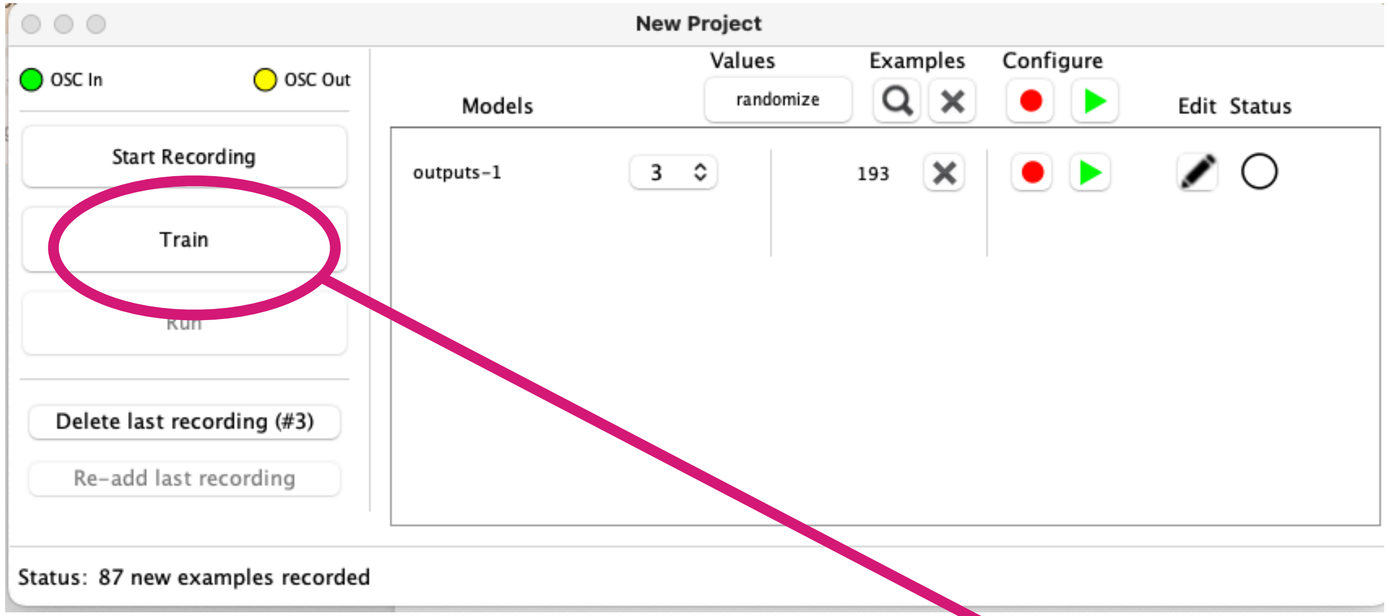


STEP 2

- Train the Module:
- Select Class 3



- Pressure **hard** the sensor and click Start recording for 2 seconds

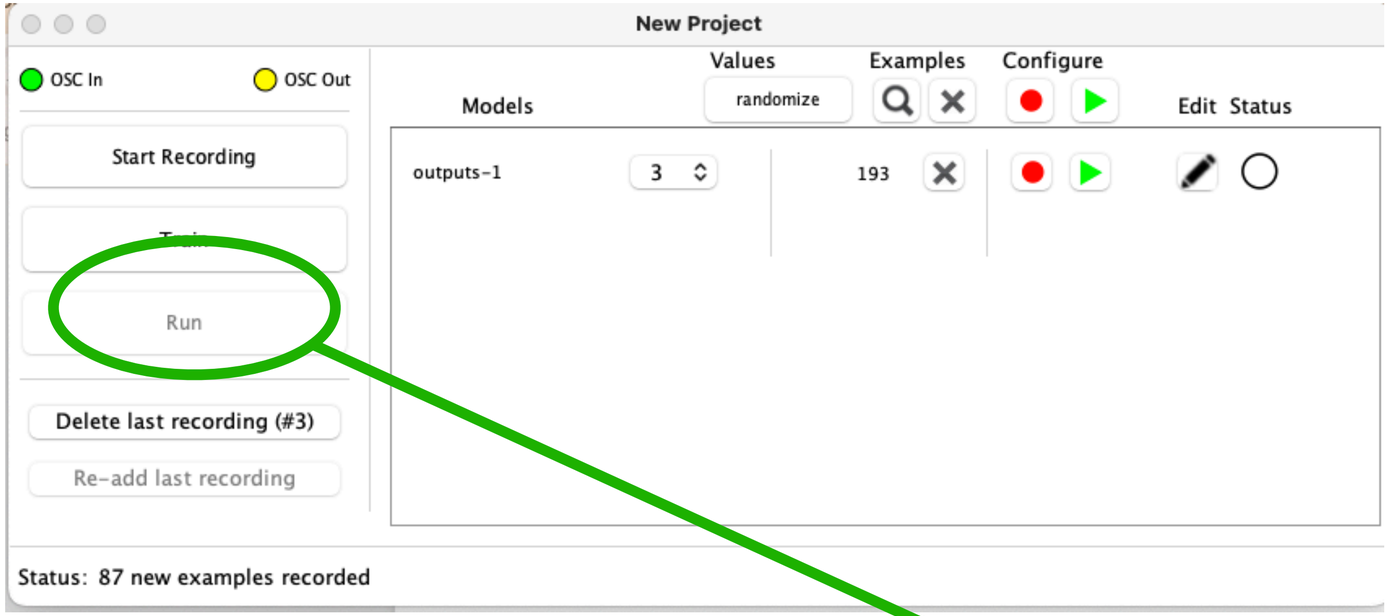


Configure classification

http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types

STEP 3

- Train the Module:
- Click Train



Configure classification

http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types

STEP 3

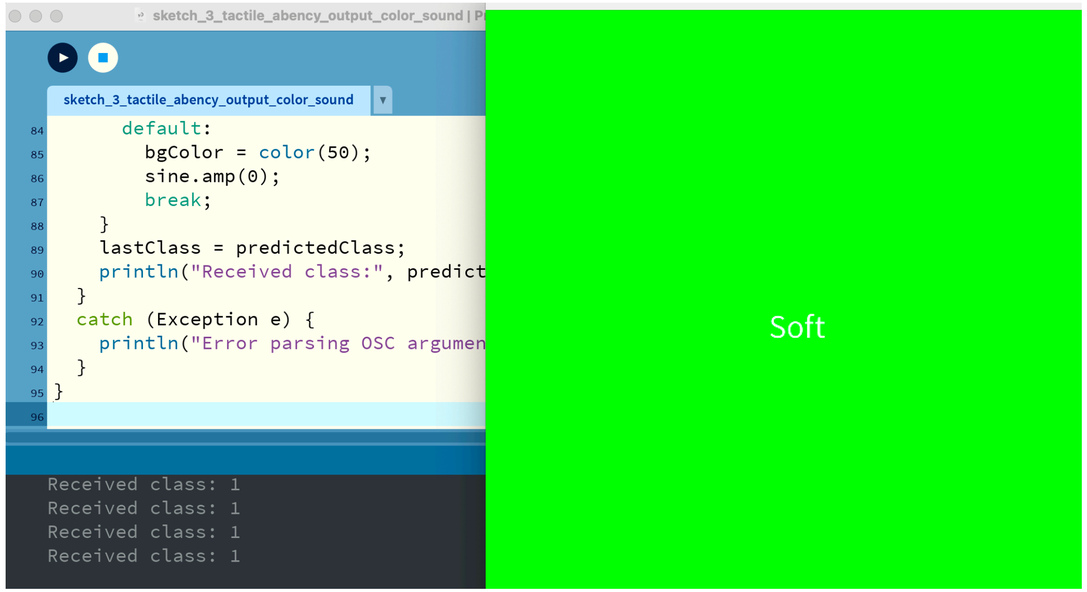
- Train the Module:
- Click run and pressure the sensor/

See how the classes changes according with The trained module

Open: sketch_2_tactile_agency_output_color

Processing output

```
import oscP5.*;
import netP5.*;

OscP5 osc;
int lastClass = 0;
color bgColor = color(50);

void setup() {
  size(600, 600);
  // Bind OscP5 to port 12000 to receive from Wekinator
  osc = new OscP5(this, 12000);
  background(bgColor);
  println("Listening for OSC on port 12000");
}

void draw() {
  background(bgColor);
  fill(255);
  textAlign(CENTER, CENTER);
  textSize(32);
  switch(lastClass) {
    case 1:
      text("Soft", width/2, height/2);
      break;
    case 2:
      text("Medium", width/2, height/2);
      break;
    case 3:
      text("Hard", width/2, height/2);
      break;
    default:
      text("Waiting…", width/2, height/2);
      break;
  }
}

// This method is called whenever an OSC message arrives
void oscEvent(OscMessage msg) {
  // Debug: print incoming address and typetag
  println("Incoming OSC address: " + msg.addrPattern());
  println("Typetag string: " + msg.typetag());

  // Check that the address matches exactly what Wekinator is sending
  if (msg.checkAddrPattern("/wek/outputs")) {
  String tags = msg.typetag();
  try {
    int predictedClass = 0;

    // If Wekinator sends an integer classification (typetag “i”)
    if (tags.equals("i")) {
      predictedClass = msg.get(0).intValue();
      println("Parsed integer class: " + predictedClass);
    }
    // If Wekinator sends a float (typetag “f”), convert/round it
    else if (tags.equals("f")) {
      float rawVal = msg.get(0).floatValue();
      predictedClass = round(rawVal);
      println("Parsed float class (rounded): " + predictedClass);
    }
    // Otherwise, print an error
    else {
      println("Unexpected typetag: " + tags);
      return;
    }

    // Update background color based on predicted class
    switch(predictedClass) {
      case 1:
        bgColor = color(0, 255, 0);   // Green for “Soft”
        break;
      case 2:
        bgColor = color(255, 255, 0); // Yellow for “Medium”
        break;
      case 3:
        bgColor = color(255, 0, 0);   // Red for “Hard”
        break;
      default:
        bgColor = color(50);          // Default gray
        break;
    }
    lastClass = predictedClass;
  }
  catch (Exception e) {
    println("Error parsing OSC argument:", e.getMessage());
  }
 }
 else {
  println("Ignored OSC address: " + msg.addrPattern());
 }
}
```



```
import oscP5.*;
import netP5.*;
import processing.sound.*;

OscP5    osc;
SinOsc   sine;
int      lastClass = 0;
color    bgColor   = color(50);

void setup() {
  size(600, 600);

  // 1) OSC listener on port 12000
  osc = new OscP5(this, 12000);
  println("Listening for OSC on port 12000");

  // 2) Sound setup: start the sine oscillator at zero amplitude
  sine = new SinOsc(this);
  sine.freq(440);    // default pitch
  sine.amp(0);       // silent initially
  sine.play();       // start oscillator

  background(bgColor);
}

void draw() {
  background(bgColor);
  fill(255);
  textAlign(CENTER, CENTER);
  textSize(32);
  switch (lastClass) {
    case 1:
      text("Soft", width/2, height/2);
      break;
    case 2:
      text("Medium", width/2, height/2);
      break;
    case 3:
      text("Hard", width/2, height/2);
      break;
    default:
      text("Waiting…", width/2, height/2);
      break;
  }
}
```

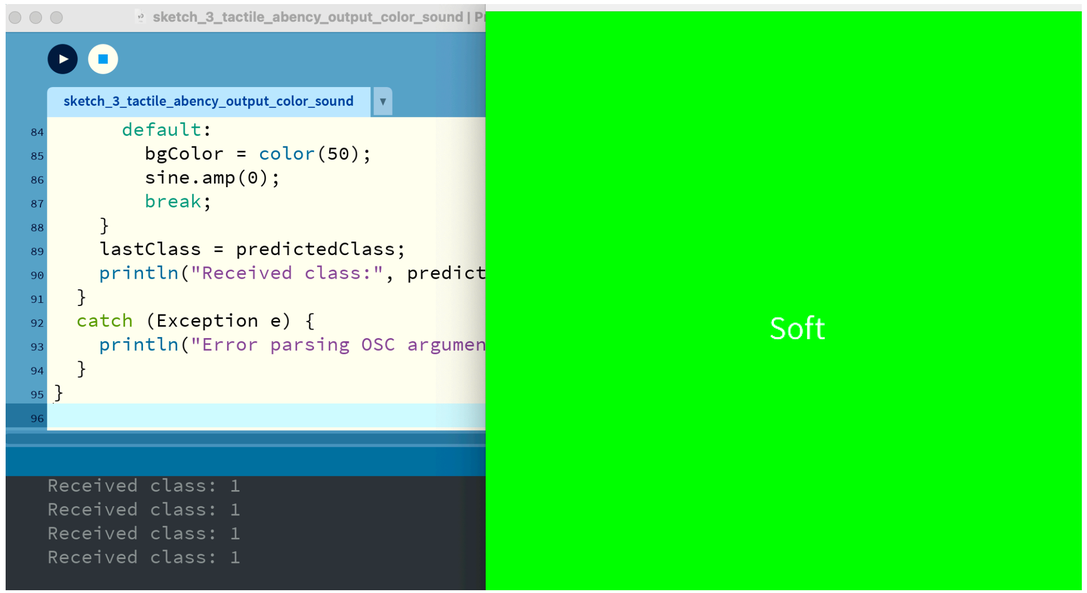
```
void oscEvent(OscMessage msg) {
  if (!msg.checkAddrPattern("/wek/outputs")) return;

  String tags = msg.typetag();
  try {
    int predictedClass = 0;

    if (tags.equals("i")) {
      // Wekinator sent an integer class
      predictedClass = msg.get(0).intValue();
    }
    else if (tags.equals("f")) {
      // Wekinator sent a float; round it to get the class index
      float rawVal = msg.get(0).floatValue();
      predictedClass = round(rawVal);
    }
    else {
      return; // unexpected typetag
    }

    // Update background color and sine parameters
    switch (predictedClass) {
      case 1:
        bgColor = color(0, 255, 0);   // green
        sine.freq(220);               // lower pitch for “Soft”
        sine.amp(0.2);                // set amplitude
        break;
      case 2:
        bgColor = color(255, 255, 0); // yellow
        sine.freq(440);               // medium pitch for “Medium”
        sine.amp(0.4);
        break;
      case 3:
        bgColor = color(255, 0, 0);   // red
        sine.freq(880);               // higher pitch for “Hard”
        sine.amp(0.6);
        break;
      default:
        bgColor = color(50);          // default gray
        sine.amp(0);                  // mute
        break;
    }
    lastClass = predictedClass;
    println("Received class:", predictedClass);
  }
  catch (Exception e) {
    println("Error parsing OSC argument:", e.getMessage());
  }
```

Open: sketch_3_tactile_agency_output_color_sound

Processing

output

Add sound

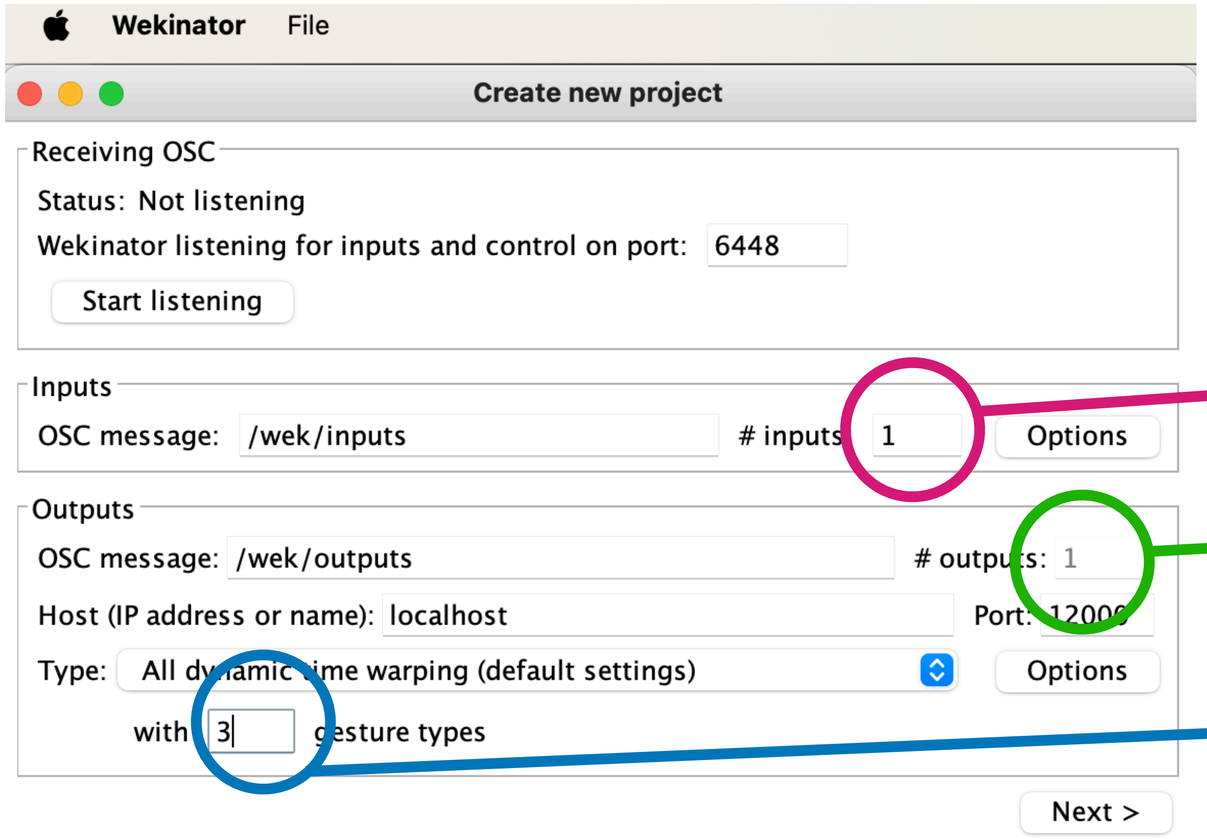


Training Patterns Wekinator

Let's record a pattern!

Dynamic time warping event outputs: Use this output type when you want Wekinator to recognize patterns over time. That is, you want Wekinator to look for a particular pattern (or multiple patterns) of how the inputs are changing over time, and tell you when a pattern is spotted and which one it was.

http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types



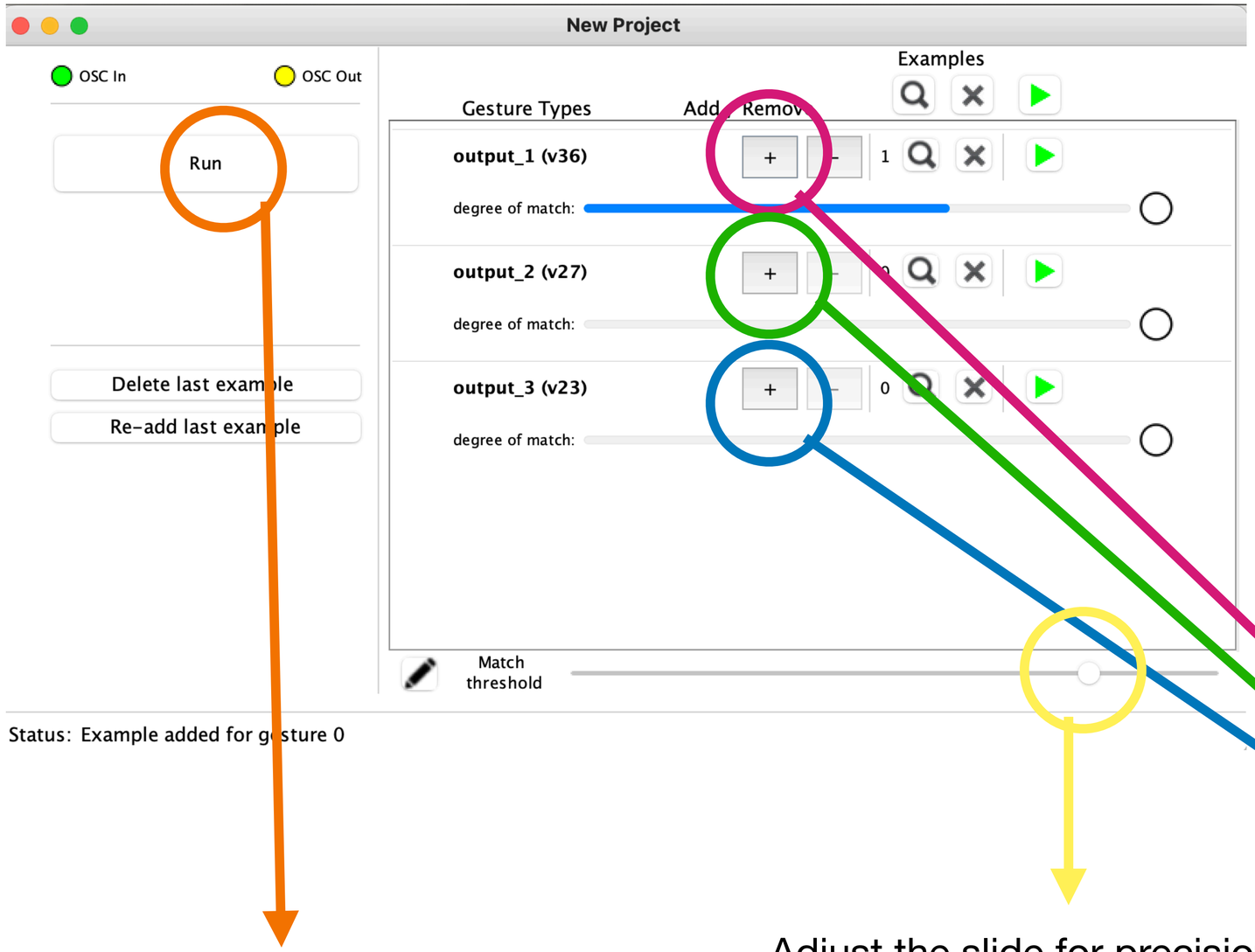
Our fsr sensor

Our output

The number of patterns we want to record

Training Patterns Wekinator

http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types



Hold + to record a gesture
(Repeat 3 times)

Our output

Adjust the slide for precision

Click Run to run the model

Open: sketch_4_tactile_agency_dynamic_color

PROCESSING CODE

```
import processing.serial.*;
import oscP5.*;
import netP5.*;
import processing.sound.*;

Serial      myPort;
OscP5       oscSender;      // sends FSR → Wekinator (port 6448)
OscP5       oscReceiver;    // listens Wekinator → Processing (port 12000)
NetAddress  wekinatorAddr;
SinOsc      sine;           // for audio feedback

int         lastClass = 0;
color       bgColor   = color(50);
String      leftover  = "";

void setup() {
  size(600, 600);

  // — OSC SENDER: no listening port, just sends to Wekinator on 6448 —
  oscSender     = new OscP5(this, 0);
  wekinatorAddr = new NetAddress("127.0.0.1", 6448);
  println("oscSender ready → sending to 127.0.0.1:6448");

  // — OSC RECEIVER: listen for Wekinator output (three distances) on 12000 —
  oscReceiver = new OscP5(this, 12000);
  println("oscReceiver listening on port 12000 for /wek/outputs");

  // — Sound setup: a sine oscillator, initially silent —
  sine = new SinOsc(this);
  sine.freq(440);    // default pitch
  sine.amp(0);       // start muted
  sine.play();       // begin playing (silent until amp > 0)

  // — SERIAL SETUP: read from Arduino (FSR on A0) —
  printArray(Serial.list());
  // Change the index if your Arduino appears at another slot
  String portName = Serial.list()[1];
  myPort = new Serial(this, portName, 9600);
  myPort.clear();

  background(bgColor);
}
```

```
void draw() {
  background(bgColor);
  fill(255);
  textAlign(CENTER, CENTER);
  textSize(32);
  switch (lastClass) {
    case 1: text("Gesture 1", width/2, height/2); break;
    case 2: text("Gesture 2", width/2, height/2); break;
    case 3: text("Gesture 3", width/2, height/2); break;
    default: text("Waiting…", width/2, height/2); break;
  }

  // Poll serial buffer and accumulate into leftover
  while (myPort.available() > 0) {
    String chunk = myPort.readString();
    if (chunk != null) leftover += chunk;
  }
  // Extract complete lines ending in '\n'
  int idx;
  while ((idx = leftover.indexOf('\n')) >= 0) {
    String line = leftover.substring(0, idx).trim();
    leftover = leftover.substring(idx + 1);
    if (line.length() > 0) sendToWekinator(line);
  }
}

// Send one FSR reading (float) to Wekinator at "/wek/inputs"
void sendToWekinator(String line) {
  float val;
  try {
    val = float(line);
  } catch (Exception e) {
    println("Cannot parse sensor value:", line);
    return;
  }
  println("FSR→", val);
  OscMessage msg = new OscMessage("/wek/inputs");
  msg.add(val);
  oscSender.send(msg, wekinatorAddr);
}
```

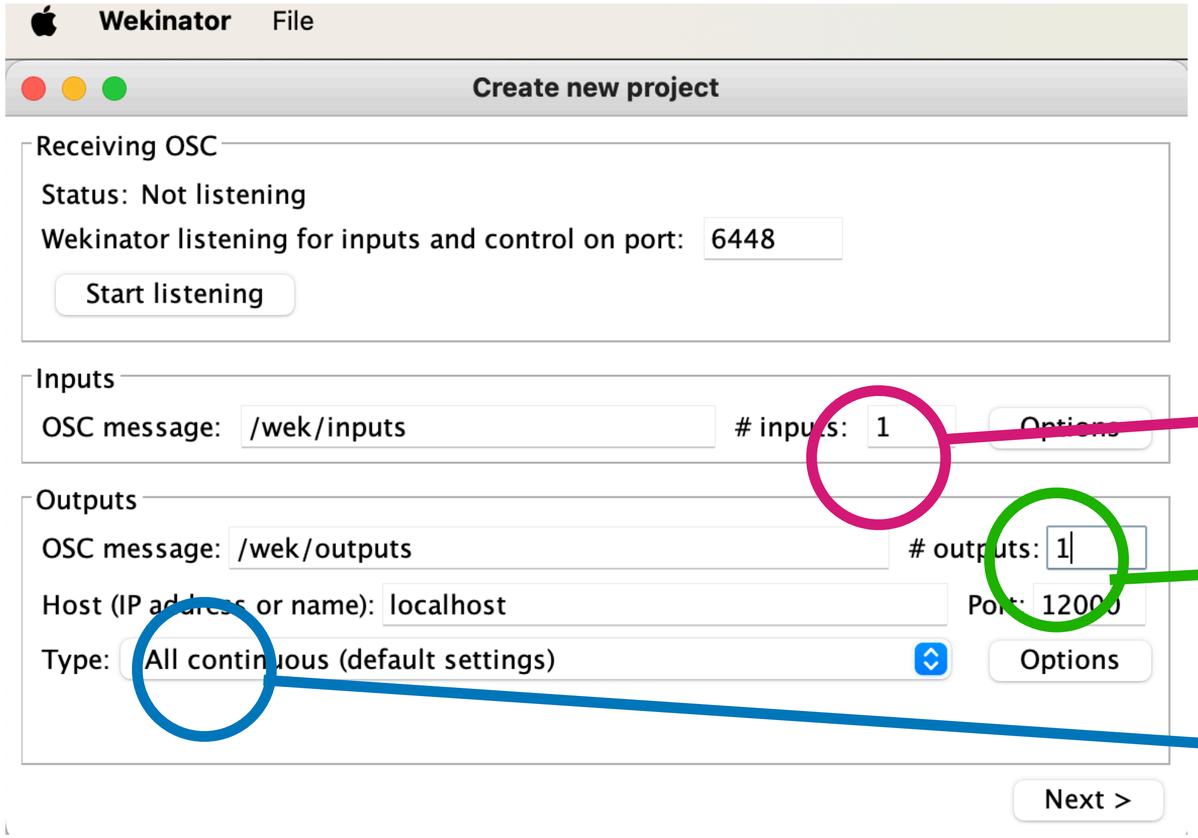
```
// Send one FSR reading (float) to Wekinator at "/wek/inputs"
void sendToWekinator(String line) {
  float val;
  try {
    val = float(line);
  } catch (Exception e) {
    println("Cannot parse sensor value:", line);
    return;
  }
  println("FSR→", val);
  OscMessage msg = new OscMessage("/wek/inputs");
  msg.add(val);
  oscSender.send(msg, wekinatorAddr);
}

// Called whenever an OSC arrives on port 12000
// We expect three floats: Arg[0], Arg[1], Arg[2]
void oscEvent(OscMessage msg) {
  if (!msg.checkAddrPattern("/wek/outputs")) return;

  int numArgs = msg.arguments().length;
  if (numArgs < 3) {
    println("⚠️ Expected 3 args but got " + numArgs);
    return;
  }
```

Let’s record continuous!

2) Numeric outputs: These are numeric values. There are two types of numeric outputs: Real-valued (“continuous”) numeric outputs can take on any number value (possibly limited to a certain range). Integer-valued numeric outputs can take on any integer value. We choose 0.0, 0.5, 1.0

Training Continuous Wekinator

http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types

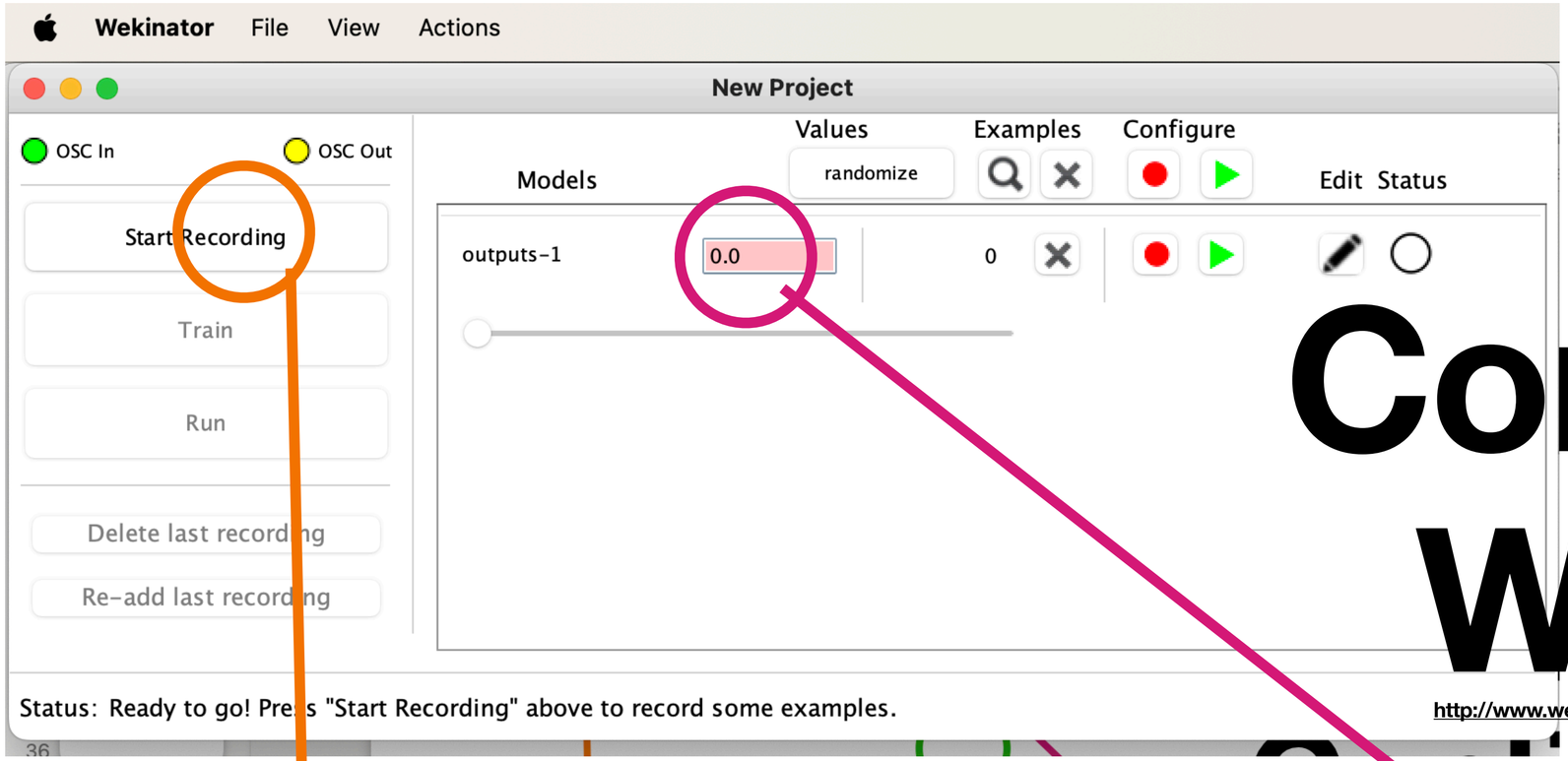


Our fsr sensor

Our output

Choose continuous

Training Continuous Wekinator

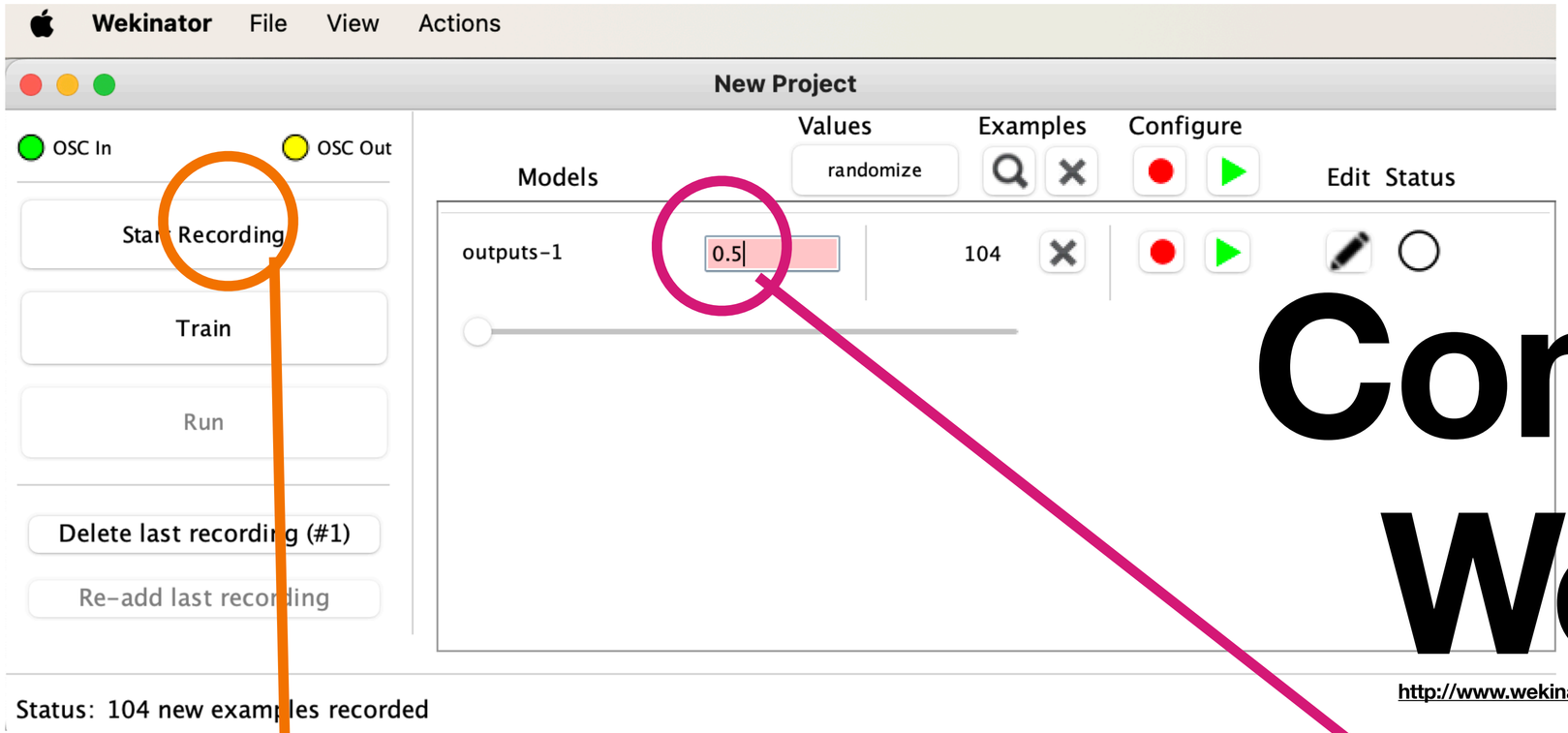


http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types

Change to 0.0

Click “Start recording”, repeat several times

Training Continuous Wekinator



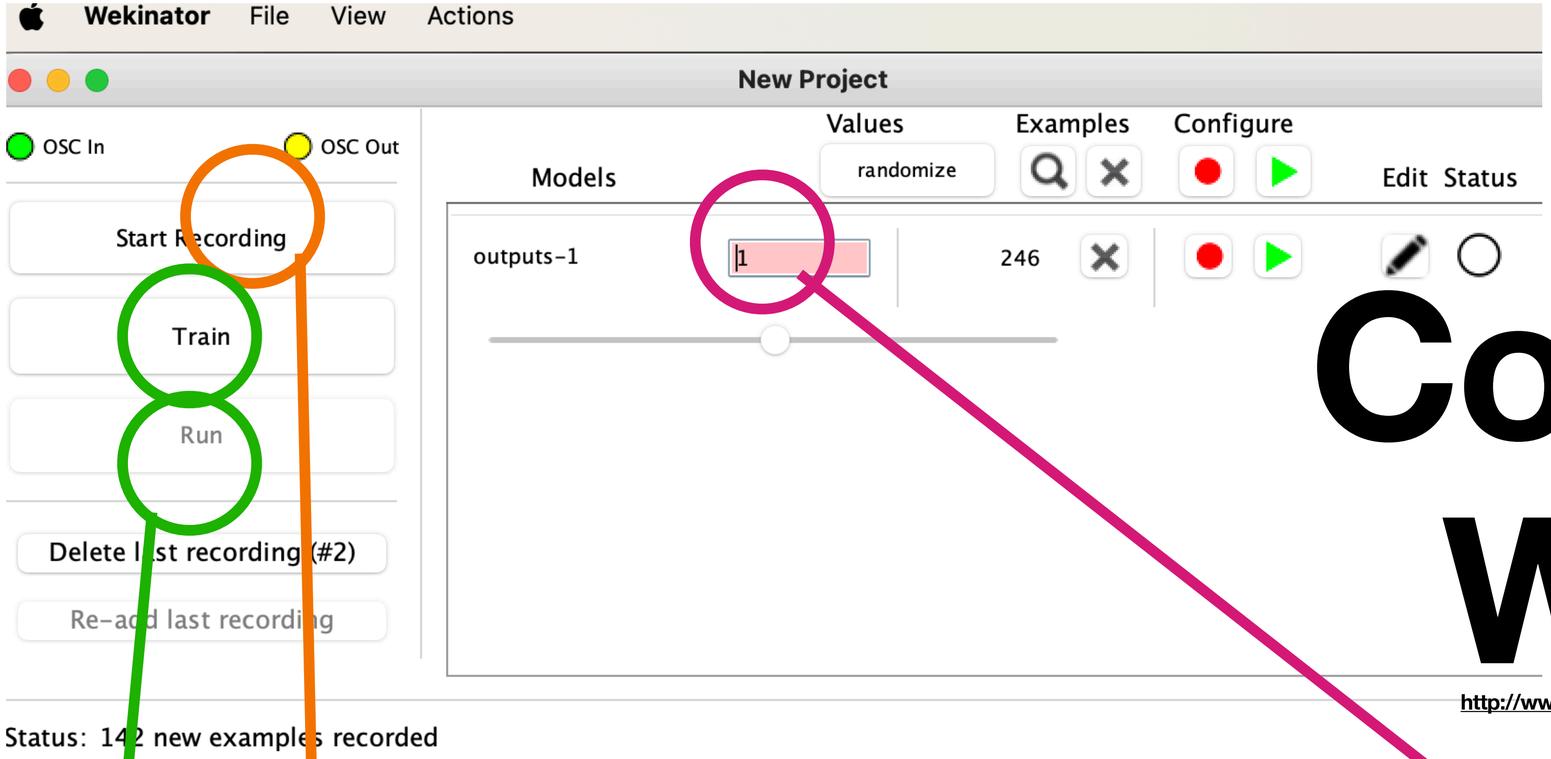
http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types

Change to 0.5

Click “Start recording”, repeat several times

Training Continuous Wekinator

http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types



Train + Run

Change to 1

Click “Start recording”, repeat several times

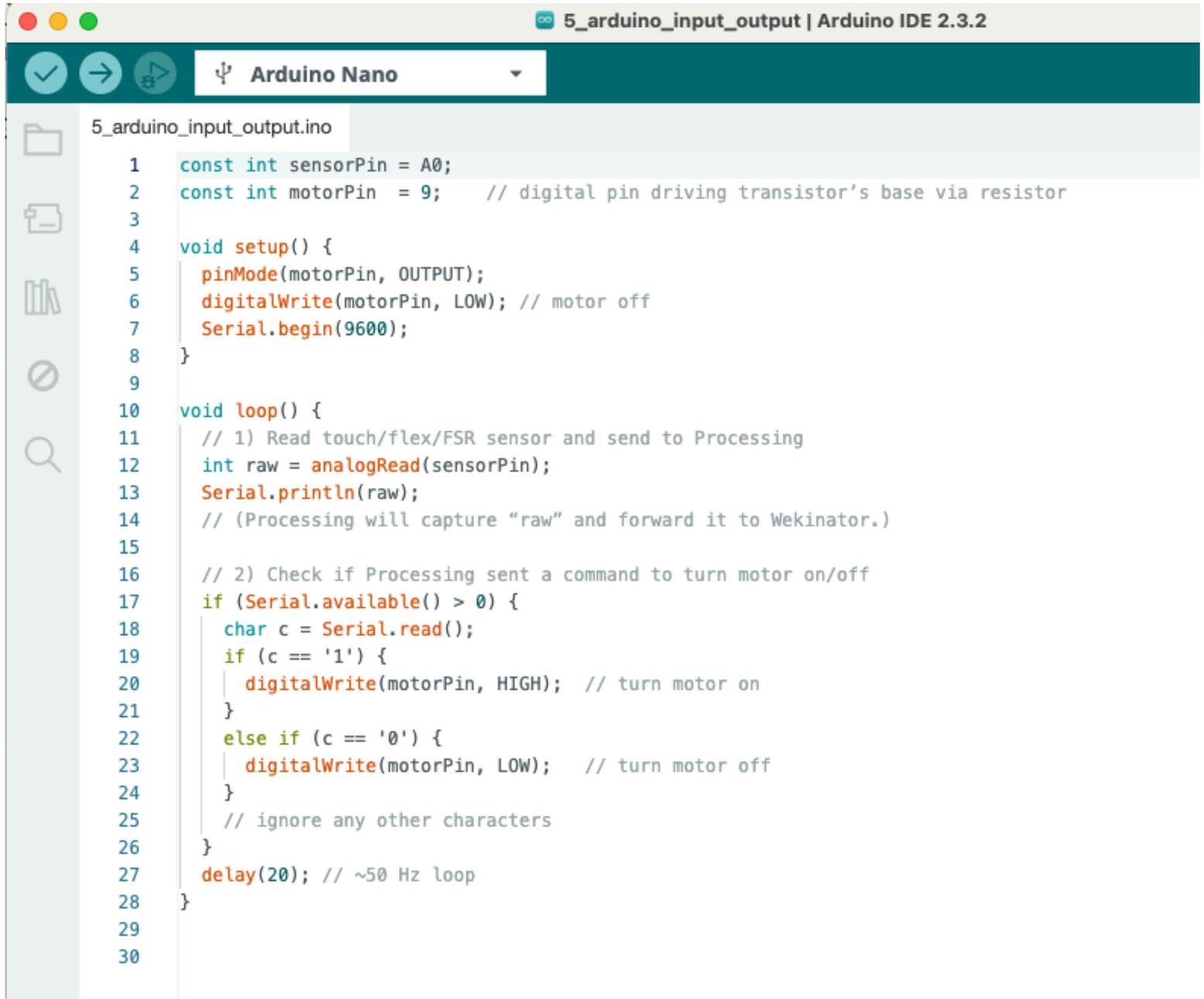
Open: sketch_5_tactile_agency_continous_color_sound

PROCESSING CODE

```
import processing.serial.*;
import oscP5.*;
import netP5.*;
import processing.sound.*;

Serial      myPort;
OscP5       oscSender;      // sends A0 → Wekinator (6448)
OscP5       oscReceiver;    // listens Wekinator → Processing (12000)
NetAddress  wekinatorAddr;
SinOsc      sine;

float       lastValue = 0;      // continuous output from Wekinator
String      leftover = "";

void setup() {
  size(600, 600);
  colorMode(HSB, 1.0);        // H, S, B all in [0..1]

  // 1) OSC sender to Wekinator on 6448 (no listening port here)
  oscSender     = new OscP5(this, 0);
  wekinatorAddr = new NetAddress("127.0.0.1", 6448);

  // 2) OSC receiver for continuous output on 12000
  oscReceiver = new OscP5(this, 12000);
  println("Listening for continuous output on 12000");

  // 3) Sound: sine oscillator, silent initially
  sine = new SinOsc(this);
  sine.freq(440);
  sine.amp(0);
  sine.play();

  // 4) Serial from Arduino (FSR on A0)
  printArray(Serial.list());
  String portName = Serial.list()[1];  // change index if needed
  myPort = new Serial(this, portName, 9600);
  myPort.clear();
}

void draw() {
  // Map lastValue (0..1) to hue (0=blue→~0.66, 1=red→0.0 or 1.0)
  float hueVal = lerp(0.66, 0.0, lastValue);
  background(hueVal, 1.0, 1.0);

  // Map lastValue to sine freq (220Hz→880Hz) and amplitude (0→0.5)
  float freqVal = lerp(220, 880, lastValue);
  float ampVal  = lerp(0.0, 0.5, lastValue);
  sine.freq(freqVal);
  sine.amp(ampVal);

  // Display numeric feedback
  fill(0, 0, 1);
  textAlign(CENTER, CENTER);
  textSize(32);
  text(nf(lastValue, 1, 2), width/2, height/2);
}

// Called when serial data arrives; send it to Wekinator
void serialEvent(Serial myPort) {
  // Read a full line ending in '\n'
  String line = myPort.readStringUntil('\n');
  if (line == null) return;
  line = trim(line);
  if (line.length() == 0) return;

  float sensorVal;
  try {
    sensorVal = float(line);
  } catch (Exception e) {
    return;
  }

d that raw sensor value as a single-element OSC array
  OscMessage msg = new OscMessage("/wek/inputs");
  msg.add(sensorVal);
  oscSender.send(msg, wekinatorAddr);
}

// Called when Wekinator sends its continuous float output
void oscEvent(OscMessage msg) {
  if (!msg.checkAddrPattern("/wek/outputs")) return;

  String tags = msg.typetag();
  if (tags.equals("f")) {
    float v = msg.get(0).floatValue();
    lastValue = constrain(v, 0, 1);  // ensure in [0..1]
    println("Got continuous:", lastValue);
  }
  else {
    println("Unexpected typetag:", tags);
  }
}
```

Adding a Vibration motor To pin 9 Arduino

Open: 5_arduino_input_output



Adding an output Vibration motor

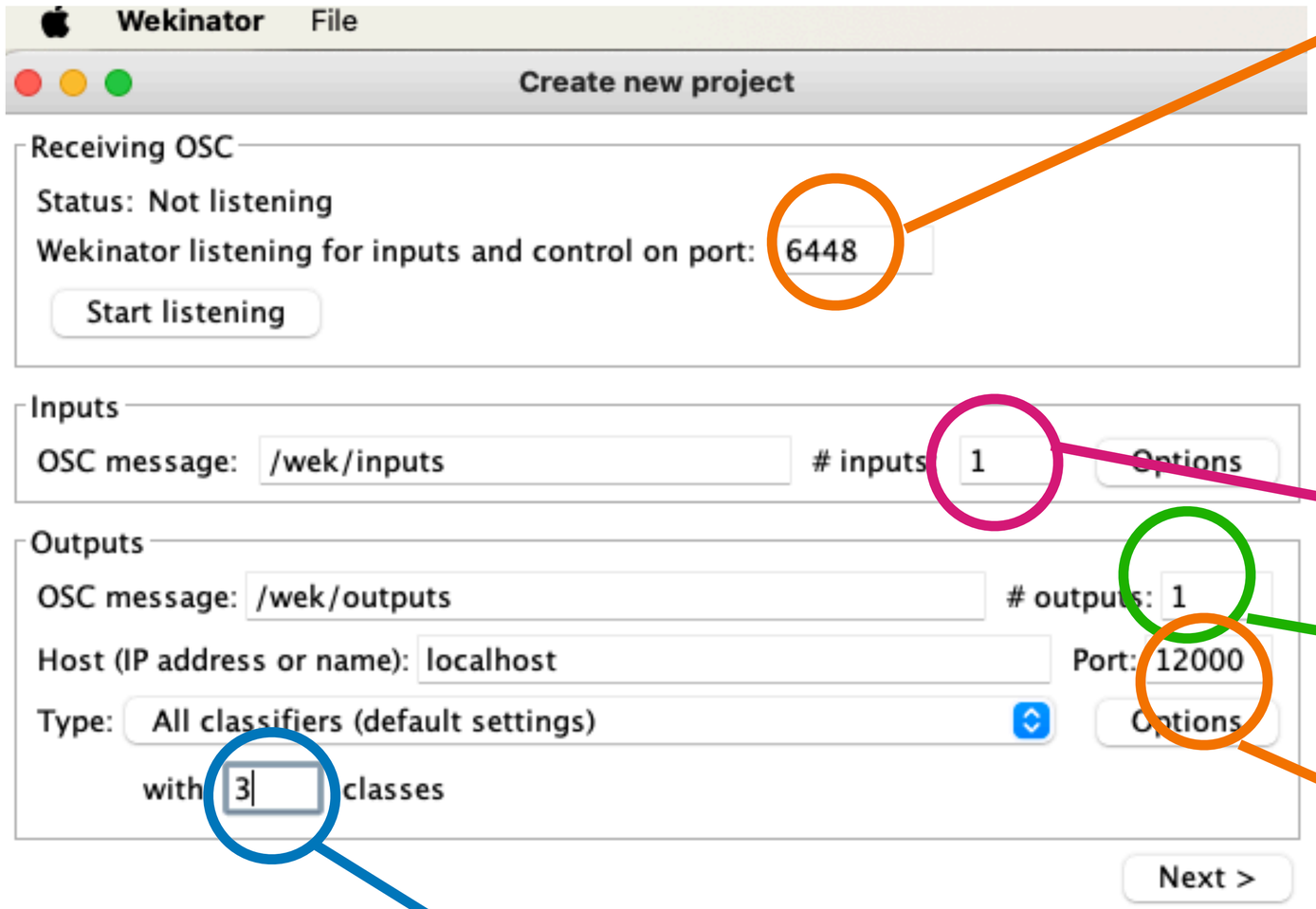
Choose your preferable configuration mode on wikanator

classification, dynamic or continuous

And train your model on wekinator

Configure According

http://www.wekinator.org/detailed-instructions/#Understanding_Wekinator8217s_output_types



Input port

Our fsr sensor

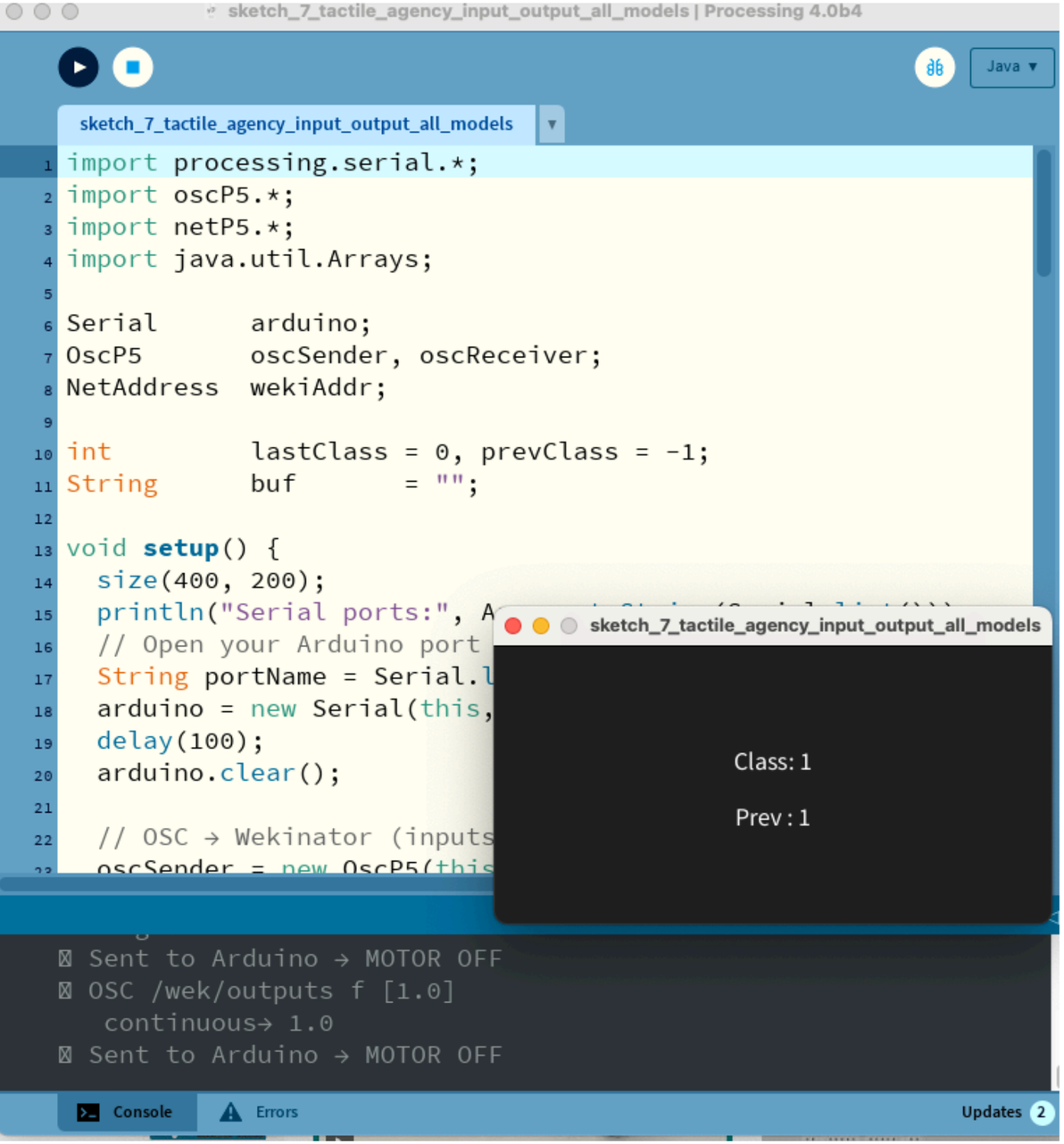
Our output

Output port

The number of classes (or sensor pressure):
1- soft
2- medium
3- hard

Open: sketch_7_tactile_agency_input_output_all_models

PROCESSING CODE



Can AI help Harmony at all??