

Tutorial 13: Processing + Arduino

CSC 413/513 Summer 2025

June 19, 2025

Group 13

Colleen Cipriano

Chloë Farr

Soyun Lee

Frank Lu

Weiqi Long

Introduction	2
About Processing	2
How to Use the Etch-a-sketch System	3
Hardware Setup	3
Components	3
Wiring Power & User Input	4
Connecting Arduino + Processing	6
Code	6
Arduino Code	6
Processing Code	9
References	10
Contribution Table	11

Introduction

View our full project on [Github](#)

For this tutorial, our group will walk you through how to build a simple 'Etch-a-Sketch' using the components listed below. We will show how to wire the circuit as well as demonstrate how to build the real physical circuit. Finally, we'll walk you through how to connect Processing and Arduino to create a tangible Etch-a-Sketch.

About Processing

[Processing](#) is an open source IDE that can be used to design and code a wide range of projects with Arduino. Originally created to simplify programming for visual artists and designers, processing supports rapid development of interactive graphics, animations, and visualizations using a syntax similar to Java. Its simplicity and real-time rendering capabilities make it especially well-suited for creative and educational applications.

When combined with Arduino, Processing lets users create interactive systems that respond to real-world input. Arduino reads data from sensors, like joystick movements or button presses, and sends it to Processing over a serial connection. Processing then uses that data to update visuals on screen in real time, forming a smooth link between physical actions and digital feedback. In our project, for example, the joystick controls the drawing cursor, while a push button clears the screen. This setup creates a simple but engaging way to turn physical motion into creative expression.

Welcome to Processing!

Processing is a flexible software sketchbook and a language for learning how to code. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. There are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning and prototyping.

[Download](#)[Reference](#)[Donate](#)[Open Editor](#)

How to Use the Etch-a-sketch System

This system lets people draw on the screen using a joystick and a push button, just like the classic toy. Here is how to use it, once it is built:

Step 1: Move the Joystick to Draw

Gently push the joystick in different directions. Left /right to move the cursor horizontally and up/down to move it vertically. As you move the joystick, the system draws a continuous line on the screen, simulating sketching.

Step 2: Press the Button to Clear the Screen

When you want to start a new drawing, simply press the push button connected to the circuit. This will instantly clear the canvas and reset the cursor position to the centre.







Hardware Setup

Components

For this project, you will need the hardware components below.

Component	Qty.	Picture	Component	Qty.	Picture
-----------	------	---------	-----------	------	---------

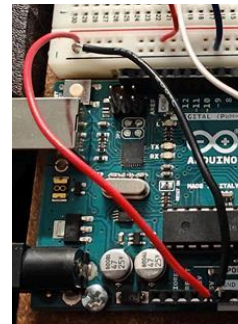
Tutorial 13: Processing + Arduino

Joystick	1		Push button	1	
Breadboard	1		Resistor (220 Ω)	1	
Arduino Uno R3	1		Wires	10	
			(5 male - male) (5 female - male)		

Wiring Power & User Input

- Connecting power (connect Arduino and breadboard)

Wire color	End 1	End 2	Purpose
red	Arduino +5V	Breadboard (+) rail	voltage
black	Arduino GND	Breadboard (-) rail	ground

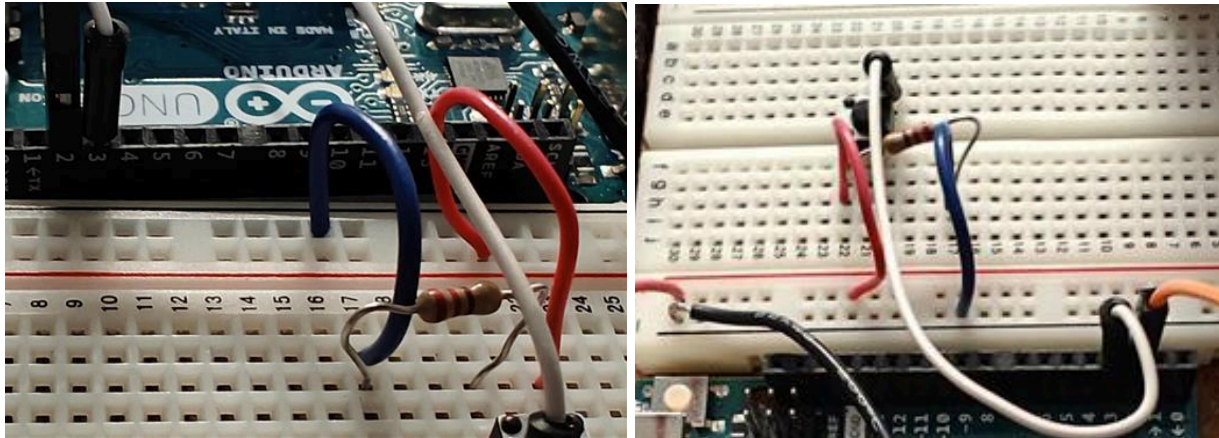


- User Input: [Push button](#)

The pushbutton is pressed by the user to clear the drawing and reset the 'pen' to the centre of the drawing window. To connect the pushbutton, you will need the pushbutton, 3 jumper wires, one resistor (we used 220 Ω , but may vary), the breadboard, and Arduino. The wire color in the table matches the color in the pictures. You can use whatever colors you choose.

Wire color	End 1 (joystick)	End 2	Purpose
red	breadboard H22	breadboard (+) rail	voltage

blue	breadboard I17	breadboard (-) rail	ground
white	breadboard D20	Arduino D3 female	Data (io pin)
Resistor 220 Ω	breadboard H20	Breadboard H17	Limit current



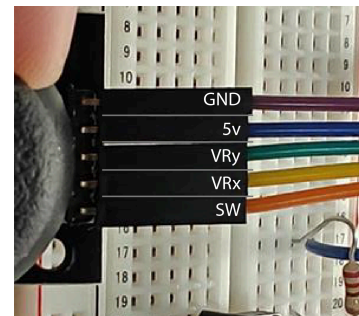
3. User Input: [Joystick](#)

The joystick is for the user to guide the 'pen' of the Etch-a-sketch. They can push it left-right (x input) and up-down (y input). For this part, you will need the joystick, 5 female-male jumper wires, the breadboard, and Arduino.

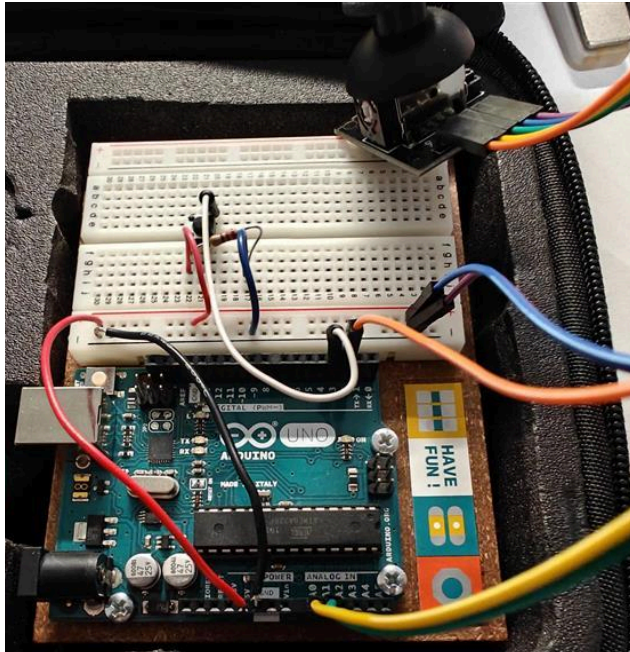
The joystick is wired with female-to-male jumpers so it's not stuck in the breadboard, making it easier to play around with inputs. The joystick pins connect to the female end of the jumper wires, and the male ends connect to the breadboard and Arduino as specified.

Note that the connection of SW - D2 pin is to get pushbutton input from the joystick. We have made this connection for facilitative purposes, but do not utilize the functionality. We instead chose to add a separate pushbutton, because using the one in the joystick poses too high of a risk of the user accidentally resetting their drawing.

Wire color	Joystick Pins	Breadboard/Arduino	Purpose
purple	+5v	breadboard (+) rail	voltage
blue	GND	breadboard (-) rail	ground
green	VRy	Arduino A1 female	y-input
yellow	VRx	Arduino A0 female	x-input
orange	SW	Arduino D2 female	Pushbutton



Completed hardware setup:



Click [here](#) to jump to the Arduino code

Click [here](#) for a demonstration video for component wiring.

Connecting Arduino + Processing

Workflow:

User input is read through the Arduino and initially processed by the `CSC4513_etchasketch.ino` script (provided [here](#) as plaintext). The user provides input via the joystick or pushbutton, actions are logged to the Serial Monitor. When the joystick is moved, the Serial Monitor outputs the updated cursor position in `x,y` format (e.g. `10,20`). When the pushbutton is pressed to clear the screen, the Serial Monitor outputs `clear 0,0`. The Processing script `sketch_250616a.pde` (provided [here](#) as plaintext) reads these coordinates from the Arduino via the Serial Port and draws lines to the new positions. If it receives `clear`, the Processing display is reset, along with the coordinates to `0,0`.

How to Run:

1. Open `CSC4513_etchasketch.ino` in the Arduino IDE.
2. Under 'Tools', select your board and the correct port.
3. Upload your sketch to the Arduino.
4. To verify input, open Serial Monitor (Tools → Serial Monitor), and set the baud rate to 9600. Move the joystick around, and press the pushbutton to check that input is logged correctly.

Once you have uploaded the Arduino sketch and are satisfied with the input behaviour and Serial output, you can close the Serial Monitor and open the Processing script. In the Processing code, ensure that the port name in the line matches your Arduino's connected port like: `String portName = "/dev/cu.usbmodem1101";`. Port name can be found on Arduino IDE →

Tools → Port. Once the port name is correct, run the Processing. The drawing should appear in the Etch-a-sketch in response to joystick output.

Points to note:

- We set the Serial Monitor baud (port) 9600 in **both scripts**. If you change this value in one, you must update the other to match.
- You do **not** need to keep the Serial Monitor open on the Arduino once you start the Processing sketch. Processing reads directly from the serial port, and the Monitor must be closed for it to work.

Code

Arduino Code

```
#include <stdio.h>

// ARDUINO TO PROCESSING
// This sketch reads analog inputs from one joystick (X, Y) and a push button (Clear).
// It sends Serial data to Processing whenever the cursor moves.
//
// Data format:
//   - On movement: "x_cur,y_cur\n"
//   - On clear button press: "clear\n"
//
// Baud Rate: 9600 (match in Processing)

// PIN CONFIGURATION
const int x_pin = A0; // JOYSTICK VRX
const int y_pin = A1; // JOYSTICK VRY
const int clear_button = 3;

// POSITION TRACKING
int x_cur = 0;
int y_cur = 0;
int x_prev = 0;
int y_prev = 0;

// STATE FLAG
bool clear_requested = false;

void setup() {
  // Initialize pins; Put analog pins in input mode
  pinMode(x_pin, INPUT);
  pinMode(y_pin, INPUT);

  // Put digital pin in input mode (default: high)
  pinMode(clear_button, INPUT_PULLUP);
}
```


Tutorial 13: Processing + Arduino

```
// Start Serial communication
Serial.begin(9600);
}

// Send current position or clear command over Serial
/*
In Processing:

String data = myPort.readStringUntil('\n');
// Parse 'data' for "clear" or x,y coordinates
*/
void print_pos() {
    if (clear_requested) {
        Serial.println("clear"); // Send clear command
        delay(200);
        clear_requested == false;
    }
    if (x_cur != 0 || y_cur != 0) {
        Serial.print(x_cur);
        Serial.print(",");
        Serial.println(y_cur); // Send position as x,y
    }
}

void loop() {
    // Placeholders
    x_prev = x_cur;
    y_prev = y_cur;

    // Read from potentiometers connected to analog pins
    int x_read = analogRead(x_pin);
    int y_read = analogRead(y_pin);

    const int centre = 512;
    const int th = 20;

    // Update current position based on new joystick position
    if (x_read < centre - th){
        x_cur -= 10;
    } else if (x_read > centre + th){
        x_cur += 10;
    }
    if (y_read < centre - th){
        y_cur -= 10;
    } else if (y_read > centre + th){
        y_cur += 10;
    }

    // Keep cur within bounds
    x_cur = constrain(x_cur,0,350);
    y_cur = constrain(y_cur,0,350);
}
```

```
// Invert value of button (FALSE = pushed, TRUE = not pushed)
if(digitalRead(clear_button) == HIGH && clear_requested == false){
    clear_requested = true;
    x_cur = 0;
    y_cur = 0;
}
else {
    clear_requested = false;
    Serial.println("no clear");
}

// Print new position to serial monitor if position values have changed
if ((x_prev != x_cur || y_prev != y_cur) || clear_requested) {
    print_pos();
}

// Wait for short time before looping
delay(100);
}
```

Processing Code

```
import processing.serial.*;

Serial myPort;
PImage frame;
PGraphics drawingLayer;

int x = 0;
int y = 0;
int x_prev = 0;
int y_prev = 0;

int canvasX = 175;
int canvasY = 170;
int canvasW = 935;
int canvasH = 660;

// initializes canvas, loads etch-a-sketch frame image, and creates a drawing layer
// and sets up serial communication with the Arduino code
void setup() {
    size(1280, 1044);
    frame = loadImage("EtchasketchController.png");
    drawingLayer = createGraphics(canvasW, canvasH);

    String portName = "/dev/cu.usbmodem1101";
}
```

Tutorial 13: Processing + Arduino

```
myPort = new Serial(this, portName, 9600);
myPort.bufferUntil('\n');
}

// renders the background, frame and overlays the drawing layer to the screen
void draw() {
  background(200);
  image(frame, 0, 0);
  image(drawingLayer, canvasX, canvasY);
}

// parses cursor position in "x,y" format or "clear"
// if "clear", it resets the canvas
// if cursor position is received, it draws a blackline from the previous to current
position on PGraphics layer
void serialEvent(Serial myPort) {
  String input = myPort.readStringUntil('\n');

  if (input != null) {
    input = trim(input);
    println("from arduino: " + input);

    if (input.equals("clear")) {
      drawingLayer.beginDraw();
      drawingLayer.background(255);
      drawingLayer.endDraw();
      x = y = x_prev = y_prev = 0;
    } else if (input.contains(",")) {
      String[] coords = split(input, ",");
      if (coords.length == 2) {
        try {
          x_prev = x;
          y_prev = y;

          x = constrain(Integer.parseInt(coords[0]) * canvasW/350, 0, canvasW);
          y = constrain(Integer.parseInt(coords[1]) * canvasH/350, 0, canvasH);

          drawingLayer.beginDraw();
          drawingLayer.stroke(0);
          drawingLayer.strokeWeight(2);
          drawingLayer.line(x_prev, y_prev, x, y);
          drawingLayer.endDraw();

        } catch (NumberFormatException e) {
          println("Invalid coordinate data: " + input);
        }
      }
    }
  }
}
```

References

- [1] Ben, "Connecting Arduino to Processing". SparkFun Learn.
<https://learn.sparkfun.com/tutorials/connecting-arduino-to-processing/all>
- [2] "Visualization with Arduino and Processing", Arduino Education.
<https://www.arduino.cc/education/visualization-with-arduino-and-processing/>
- [3] "How 2-Axis Joystick Works & Interface with Arduino + Processing". Last Minute Engineers.
<https://lastminuteengineers.com/joystick-interfacing-arduino-processing/>
- [4] Etch A Sketch Drawing Toy. PngEgg
<https://www.pngegg.com/en/png-wwenn/download>

Contribution Table

Team Member Name	Section
Colleen Cipriano	Powerpoint slides (1,2,3,4, 8, 11), What you need, Setup, References
Frank Lu	Section: how to use the system, Powerpoint slide (6)
Weiqi Long	Sections: About Processing (Writing), PowerPoint slide (10 - QR code), Build-up demonstration video recording
Chloë Farr	Sections: Hardware Setup, Connecting Arduino + Processing, Arduino Code
Soyun Lee	Sections: Connecting Arduino + Processing, Processing Code, Arduino Code - loop function