Building a better Sketch-Y-Etch

Sensored Hacker

April 27, 2025

Abstract

Initial versions of the sketch-y-etch had issues related to haphazard planing, and inconsistent use of parts. It has been two years since the initial creation filled the Bangor Makerspace with retro drawing for all, and we have heard many things about what makes the sketch-y-etch hard to use. Here in v2, we are going to try an fix the issues.

ISSUES:

- **Drawing Performance:** Performance was limited by encoder issues.
- **Build Quality:** V1 used available parts with no specific plan for integration, negatively affecting the overall experience.
- **Stability:** The wobble made drawing difficult. While the Sketch-Y-Etch has never tipped over, it feels like it could—and that can be fixed.
- **Portability:** Portability was never a design goal of V1 but can be improved in V2.
- **Documentation:** Existing documentation focused on the history and developer notes but lacked detailed build instructions.
- **Bill of Materials:** A BOM was non-existent in V1. Not everyone has a stockpile of miscellaneous computer parts.

Improvements

- **Improved Components:** A concerted effort was made to use higher-quality parts throughout the build.
- **Specialized Hardware:** A new conceptual design in v2 maintains compatibility with a wide range of hardware and platforms, while offering many new design options.
- **Focused Documentation:** We want you to build your own Sketch-Y-Etch! A detailed tutorial is now available to guide you through the process.

- **New Features:** More colors, adjustable pen size, dynamic scaling, and additional enhancements.
- **Bill of Materials:** All required parts can be easily acquired, with a total estimated cost of about $50.

Let's Build It!

To begin building your Sketch-Y-Etch, you will need a set of materials and tools. The Bangor Makerspace offers complete kits for purchase if you are interested, or you can source your own components independently.

Required Tools

Basic tools required for assembly include:

- Screwdrivers
- 3D printer
- Wire cutters
- Soldering station
- Drill and drill bits
- Personal protective equipment (PPE)

Electronic Components

The Bangor Makerspace developed the HDMI-TAP specifically for this project. While its use is highly recommended for simplicity and reliability, it is also possible to build a compatible alternative at relatively low cost.

This project utilizes Qwiic connectors for inter-device communication over I^2C. A common practice in electronics is to color-code wires according to their function. Although wire insulation color does not affect circuit operation, following a standard color convention greatly simplifies visual inspection and troubleshooting.

In this project, the following color scheme is used:

- **Red:** +5VDC (power)
- **Black:** Ground (common)
- **Blue:** SDA (I^2C data line)
- **Yellow:** SCL (I^2C clock line)

As a mnemonic, you can remember that *yellow* (like a sundial) represents the clock signal.

This color convention matches the standards used by Adafruit STEMMA QT and SparkFun Qwiic connect systems, utilized in this project.

Bill of Materials

Item	Part Number / Name	Qty	Notes
Rotary Encoder	Adafruit Seesaw Encoder	4	I2C, with built-in pullups
Display	HDMI Monitor	1	Any 1080p-capable screen
Cables	Assorted Qwiic Wires	5	Male-male and male-female
Knobs	Custom 3D-Printed Knobs	1	STL files available
Computer	Any with HDMI Output	1	Laptop, desktop, or SBC
HDMI-TAP	v2 w/qwiic connects	1	easy access HDMI i2c bus.

Optional Steps

These steps are not strictly required but are highly recommended to simplify testing and setup.

1. 3D print the `table.scad` build platform and install your devices onto it.
2. Test and verify all hardware and software functionality using the build platform before embedding the electronics into a TV.

Recommended Software

The only essential software requirement for the Sketch-Y-Etch is a working installation of Python 3:

- Python 3: `https://www.python.org`

Helpful Tools

The Sketch-Y-Etch was developed primarily on Linux using Python 3 alongside a variety of additional software tools:

- OpenSCAD (for 3D design): `https://openscad.org`
- Geany (lightweight code editor): `https://www.geany.org`
- KiCad (PCB design and schematic capture): `https://www.kicad.org`
- i2c-tools (I^2C communication utilities): `https://archive.kernel.org/oldwiki/i2c.wiki.kernel.org/index.php/I2C_Tools.html`

While the design should, in principle, work on any operating system, platform-specific nuances have not been extensively tested. The tools listed above reflect the preferences of the original developer. Although not strictly required, they are highly recommended for a smoother and more educational development experience.

References

- Turtle Graphics API (Python Standard Library): `https://docs.python.org/3/library/turtle.html`
- I^2C Implementation Guide (SparkFun): `https://learn.sparkfun.com/tutorials/i2c/all`
- Adafruit Seesaw Library: `https://github.com/adafruit/Adafruit_Seesaw`
- Component Datasheets: Included in this repository.

All references are provided for educational purposes. They are not required to complete the project but may offer valuable background information.