

# Building a better Sketch-Y-Etch

Sensored Hacker

April 27, 2025

## Abstract

Initial versions of the sketch-y-etch had issues related to haphazard planning, 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 and 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.
- **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<sup>2</sup>C. 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<sup>2</sup>C data line)
- **Yellow:** SCL (I<sup>2</sup>C clock line)

As a mnemonic, you can remember that *yellow* (like the sun on 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.

## Download the Code

If you have not yet installed Python 3, download and install it now. You will also need the `numpy` and `smbus2` libraries, which can be installed with:

```
pip install numpy smbus2
```

Download the source repository for the Sketch-Y-Etch project, available at:

<https://github.com/FOSSBOSS/SketchyEtch>

Once you have the repository, extract it if necessary, and navigate to the `v2_WIP` folder. In your IDE of choice, try running `SketchyV2_WIP.py`.

If you encounter import errors, you may be missing required libraries, or your Python 3 path may be incorrect.

```
l@cker:~/SketchyEtch/V2_WIP$ python3 SketchyV2_WIP.py
Traceback (most recent call last):
  File "/home/l/SketchyEtch/V2_WIP/SketchyV2_WIP.py", line 23, in <module>
    from smbus2 import SMBus, i2c_msg
ModuleNotFoundError: No module named 'smbus2'
l@cker:~/SketchyEtch/V2_WIP$
```

Figure 1: Missing `smbus2` library error example.

## Embrace Complexity

This project has many possible configurations. Here are two basic setups you might consider:

1. Connect a computer to the Sketch-Y-Etch, then to a display screen.
2. Connect a laptop's HDMI output directly to the Sketch-Y-Etch.

Many possibilities and functionalities exist at this stage. The zeitgeist of this project is to embrace experimentation, and creativity with flexibility. Take some time to consider how you want to configure your Sketch-Y-Etch.

While video pass-through is an option, if you have multiple display outputs, you may route your controls in various ways. You might choose to embed the Sketch-Y-Etch into a TV, build a standalone control panel, make multiple controllers on one computer, use a projector, or simply connect via a laptop using HDMI as a control bus only.

Is HDMI strictly required? Not necessarily. HDMI is modern and widely available, but it is not the only option. Other video or data interfaces could be adapted with additional development effort. Adoptions may change the path of these instructions, which should conceptually work regardless of your desired implementations. I am however not going to write a new tutorial for every conceivable build path, or video technology in existence.

Please keep in mind: this project remains experimental. It is not intended as a polished, professional consumer product. While it is feasible to create a more plug-and-play experience, the practical setup will depend heavily on the specific hardware you have available. This flexibility can be confusing at first, but the key is to clearly define what you want to build before you start building it.

## Optional Steps

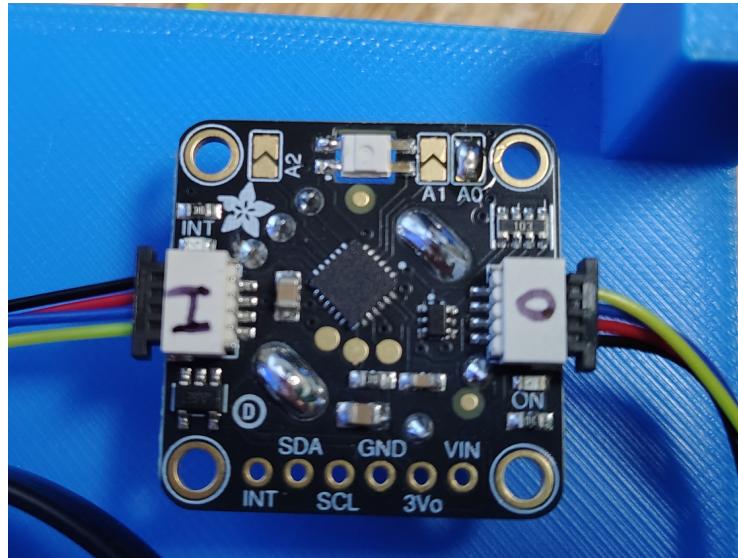
These steps are not strictly required but are highly recommended to simplify testing and setup. 3D printing the knobs, and experimental build platform may take approximately an hour of printing, depending on your printer. Print 4 knobs, and 1 table. Please note that the current knob.stl file is sized for the encoders used in this project, if you are using some other unspecified component you may need to adjust the models to meet your specific needs.

1. 3D print the `table.scad` build platform and install your devices onto it.
2. Drill or melt holes into the table to match your layout preferences.
3. Install the knobs by pushing them on to the encoder shafts. Using alternate knobs is also acceptable, we're just trying to make the encoders easy to use.
4. Test and verify all hardware and software functionality using the build platform before embedding the electronics into a TV.

## Wiring

Assuming you are following along with building the test platform first for testing, and your 4 rotary encoders are now installed, flip the table to view the encoders for wiring. The encoders have directionality, which may be counter intuitive. There is an input side, and an output side. If looking at the underside of the encoder, with the LED at the top, or away from you, the input port is on the

left, and the output qwiic connect port is on the right. You may opt to label the ports I for input and O for output. When connecting the encoders, you will connect from the output of the first encoder to the input of the next encoder in line. Here, you can observe the exposed A0, A1, and A2 pads. These pads



control the I<sup>2</sup>C addresses your encoders use to communicate on the bus. By default, all pads are disconnected, which is how the encoders are shipped from the factory. The default I<sup>2</sup>C address is 0x36.

By bridging these pads with solder, you manually set the device's address. You can attach your encoders in any order; however, each encoder must have a unique address. Up to eight address combinations are possible. Please note that whichever addresses you assign must be configured correctly in the software to ensure that each encoder performs its designated function.

Once you have assigned your 4 distinct addresses by soldering, record those addresses, and their respective functions per your board layout. You will need this information when you go to adapt the SketchyEtch.py code to your specific installation.

## More wiring

### Inspect

Inspection is an important step in every electronics project. Mistakes happen, the world is a distracting place, and the fragility of human existence can lead to some unfortunate side effect. Your electronics projects do not need to suffer from this. Look for loose wires, stray strands of wire poking out of terminals,

ADDR	A0	A1	A2
0x36	H	H	H
0x37	L	H	H
0x38	H	L	H
0x39	L	L	H
0x3A	H	H	L
0x3B	L	H	L
0x3C	H	L	L
0x3D	L	L	L

Figure 2: Encoder addresses `Empty pad is high by default`

connectors ajar, messy incoherent wiring... if you can spot problems before they start, it will be safer, and less confusing later. If everything on your test rig looks good, try plugging it in.

projects rarely work perfectly the first time around, even with expertly drafted and bespoke instructions.

Plug your test rig in. Fire? unplug it.  
No fire? no smells?

Great!! you have passed the first step of electronics tinkering! Did your externally attach monitor resume video playback? Did the power lights on the encoders illuminate?

If every thing seems to be working, then lets proceed. If you start to smell something, or your computer begins to behave erratically, I would suspect the newly attached experimental electronics project, and unplug it as best practice.

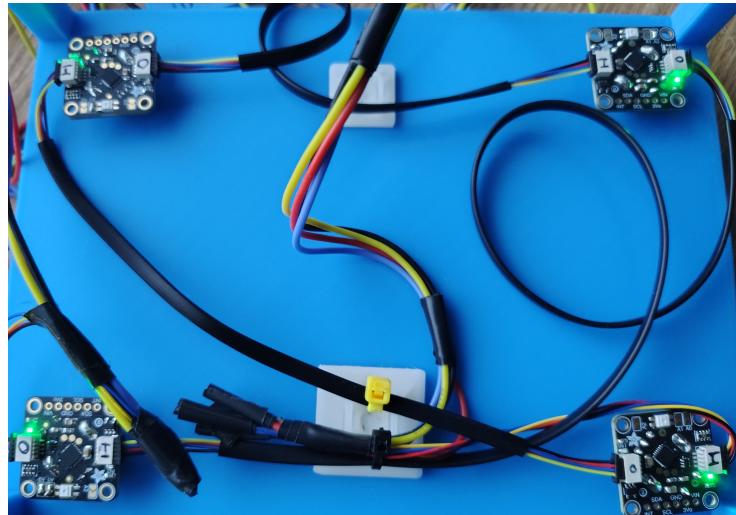


Figure 3: Lights on! no fire!

## Install Hardware

## Test Hardware

## Adapt the code

## Play Time!

## Troubleshooting

## Contact

1. You love this project, and want to contribute!
2. You have commentary about this document, project, code and want further detail.
3. Complaints.

## 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<sup>2</sup>C communication utilities ): [https://archive.kernel.org/oldwiki/i2c.wiki.kernel.org/index.php/I2C\\_Tools.html](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>
- Smbus2 (Python I<sup>2</sup>C library) : <https://pypi.org/project/smbus2>
- I<sup>2</sup>C Implementation Guide (SparkFun): <https://learn.sparkfun.com/tutorials/i2c/all>
- Adafruit encoder pinouts <https://learn.adafruit.com/adafruit-i2c-qt-rotary-encoder/pinouts>
- Adafruit Seesaw Library: [https://github.com/adafruit/Adafruit\\_Seesaw](https://github.com/adafruit/Adafruit_Seesaw)
- Component Datasheets: Included in this repository.
- The Book of I<sup>2</sup>C by Randal Hyde <https://nostarch.com/book-i%C2%B2c>

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