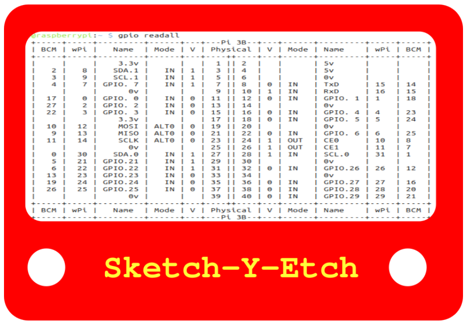
Sketch-Y-Etch a retrospective drawing experience for the masses

SensoredHacker0 @ The Bangor Makerspace

March 18, 2024

Abstract

The Sketch-Y-Etch is a pseudo clone of the classic Etch-a-sketch with a modern twist. Features include similar Cartesian drawing using knobs, with the addition of being able to change colors, lift the pen, erase segments, Erasing without shaking, saving files, and its huge. Written in python, utilizing standard hardware, and some open source projects, lets explore the Sketch-Y-Etch, and how you can create your very own.



```
@raspberrypi:~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 3B--+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5v      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 0 | IN   | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | IN   | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI | ALT0 | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO | ALT0 | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK | ALT0 | 0 | 23 || 24 | 1 | OUT  | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | OUT  | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3B--+---+------+---------+-----+-----+
```

History

The Sketch-Y-Etch is thematically based on the Classic Etch-A-Sketch. The Etch-A-Sketch is a mechanical drawing toy invented in 1960 by André Cassagnes of France and subsequently manufactured by the Ohio Art Company. Similar to the original, knobs are used to control x,y axis drawing. Sketch-Y-Etch how ever is a computerized project, and similarities end there.

Operation

Basic usage is intent on similar operation to the original Etch-a-sketch. Turn knobs, make pictures. Cycle through a set of colors with a push button. Momentarily stop drawing, and move your marker with the lift button. Erase it all

with the erase button. Save an svg [Scalable Vector Graphic] using the save button, and for display purposes, use the demo switch to run a demonstration mode which showcases various drawings, documentation, and other info.
To use the Sketch-Y-Etch drawing program with a standard keyboard use these left and right arrow keys to navigate on the +/-X plane. Up and Down keys navigate the +/- Y plane. Press "g" to change the color. Pressing "g" again will shift to the next color in the list. pressing "g" until the white color is selected enable the erase function, as the background is white. press "l" to toggle the lift pen function. Lifting the pen allows the cursor to be moved, without drawing. Press "c" to clear the screen. The cursor will reset to the center position, and erase everything which has been drawn. Press "s" to save the file, and "d" to enable demo mode. The Python script may require modification to save at an appropriate location.

Construction:

The Sketch-Y-Etch project is designed to run on standard computer hardware. Making A version using buttons and knobs is entirely optional, though in our opinion, much cooler.

Material needs: Two rotary encoders, four buttons. a micro controller capable of acting as a keyboard, a computer of some sort, and a TV.

This project utilizes a raspberry Pi 3b+, and a Samsung ln26A330j1 30 inch TV. Any TV, and computer of your choice should work. This model TV, has wide bevels, and lots of internal space where hardware can be embedded. If you can not get a TV with space to embed various additional electronics, because many modern TVs do not have bevels, you could create a control panel, or make a box to contain your additional electronics.

Software Resources Used:

OpenScad to 3D model the Knobs.
The micro controller interface was coded in Arduino IDE using C.
The drawing program is in Python3, using the turtle library.

Programming:

Programming for the Current version of Sketch-Y-Etch is to use a keyboard supporting micro controller, and Python for graphics. The Python3 code listens for key commands, and the micro controller provides them via the IO. A standard keyboard would also work. Source Code is available at:
https://github.com/FOSSBOSS/SketchyEtch

Tribulations of the build:

This project went through many revisions. The initial version had a 6 foot by 8 foot screen and stood nearly 10 feet tall. It was also heavy, and impossible to move.

Raspberry Pies have GPIO.. why did you not use that for IO? Several versions were created to utilize Raspberry PI GPIO. Testing revealed latency problems, that we did not want to deal with. Also, utilizing standard hardware maximizes the number of devices, and subsequent variants which can be built.

Initially a prototype was built in approximately 6 hours. The encoders worked to draw as expected but when the code was adapted to facilitate a full screen mode, a lag would develop rapidly as drawing occurs. Part of this was found to be the use of the raw-turtle python library using too much memory. Raw-turtle had been implemented to solve an issue related to listening for multiple GPIO inputs, which it did successfully address, however solving the associated memory leaks proved exceedingly difficult.

The second revision implemented the idea of using keystrokes to control navigation, which allowed us to share the Sketch-Y-Etch concept with other developers interested in the project, without forcing them to have a Raspberry PI, and custom electronics.

Eventually, we opted to ditch Raspberry Pi GPIO, raw-turtle, and the various other derivative projects. The version utilizing keystrokes to process information was by far the best performing model, so we went with that.

Embedding electronics in the TV had a few revisions. The Second time the TV had to be disassembled, we realized because of the knobs and buttons, laying the TV down on its face was no longer an option. In standard disassembly of this model TV, the stand is removed first, then the rear cover. Inability to lay the TV down made handling the project difficult. We promptly modified the bolts used to secure the stand to the TV, such that they can remain attached when removing the rear cover.

Annex of variations:

The Sketch-Y-Etch was developed on a variety of Raspberry Pies, and an assortment of TVs & monitors. We chose a Raspberry PI 3b+ as a minimum operational specification. Should you build your own version, and encounter glitches, or various nuances, The Bangor Makerspace is certainly interested in offering assistance. The Sketch-Y-Etch was developed 100% on Linux, and has no other platform testing to date. Given how the Sketch-Y-Etch program operates, we see no specific reason it can not be adapted to any other platform; we simply haven't tested this.