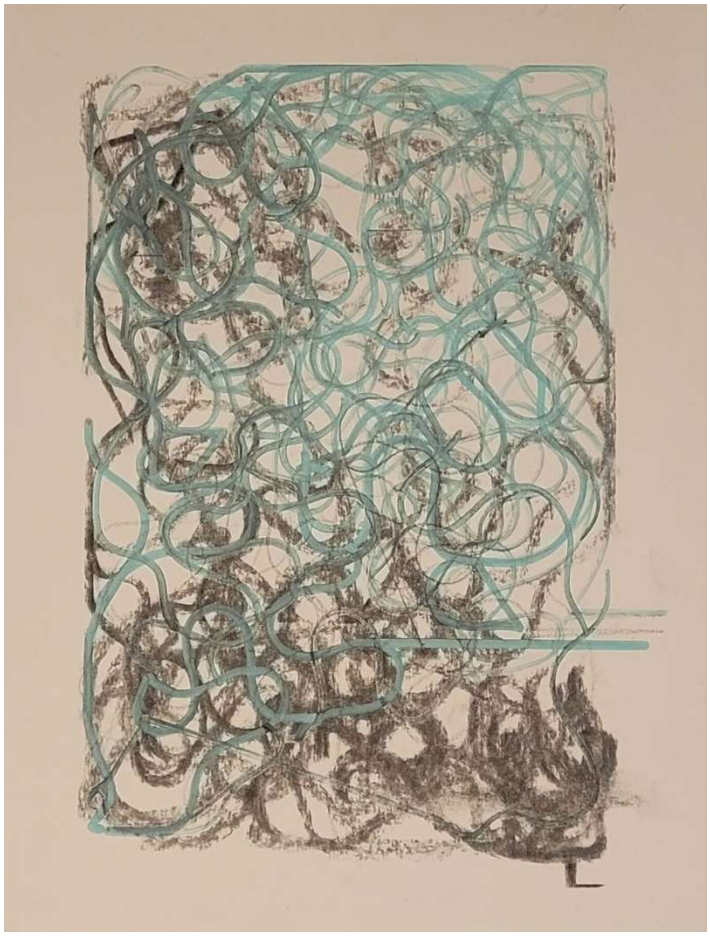


The Scribe

Technical Document

By Diego Oliver

User Manual

1. How to set up
2. UI quick guide
3. Common Errors

Technical Specifications

1. Connecting to the RPi5
2. File Structure
3. GitHub Repo
4. Adding a new model
5. Managing the Touchscreen Display
6. GRBL 2 Axis Firmware

USER MANUAL

How to set up

The Scribe requires two different power supplies to get started. The Blackbox motor controller needs a 24V, 3 A supply and the raspberry pi a 5V, 5A supply. Once connected, the raspberry pi will automatically begin boot. This should take a couple of minutes but eventually the following screen should appear.

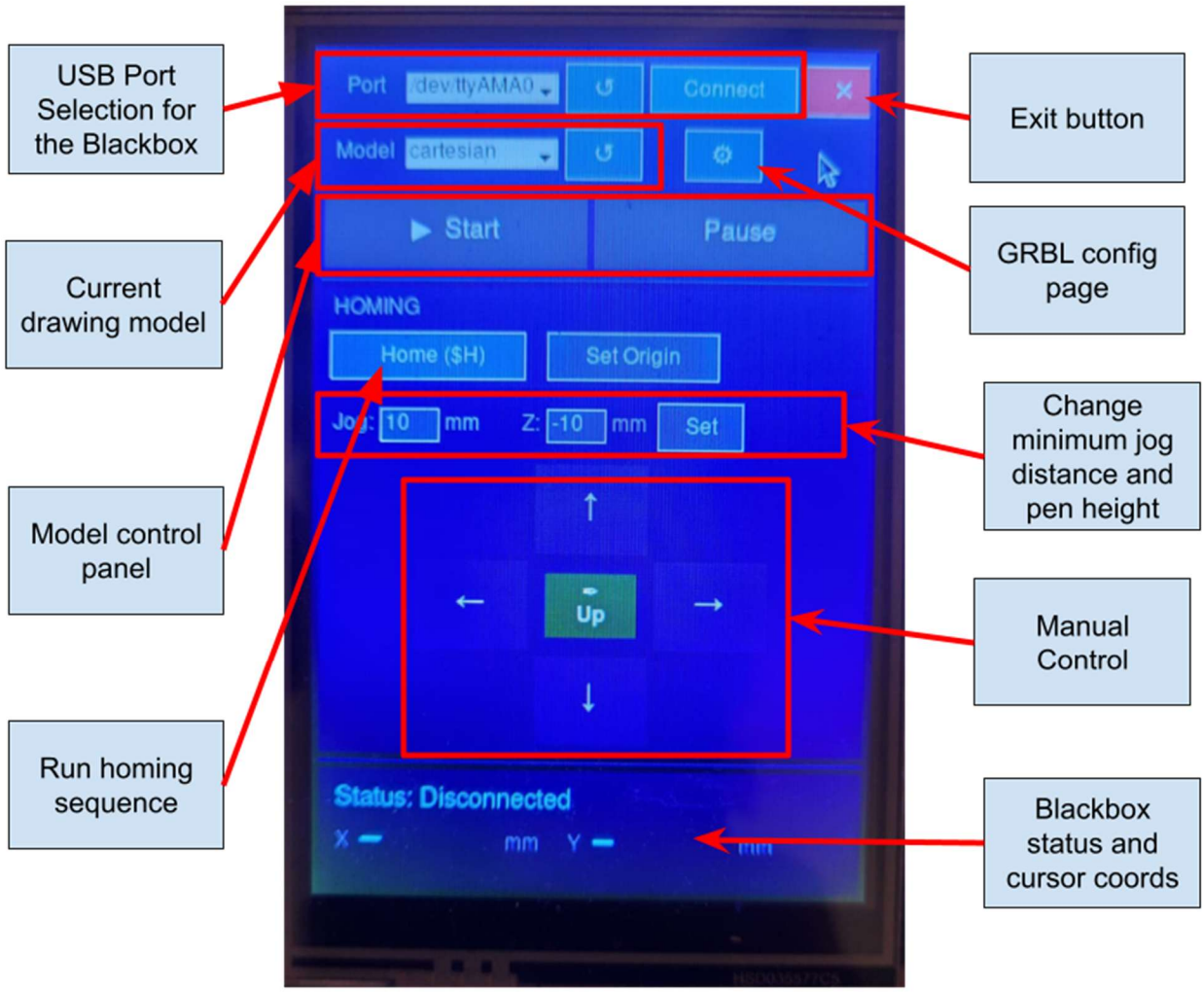
The scribe runs raspbian OS so the home screen resembles and acts like any other Linux computer. From here you can connect to the internet, surf the web, manage files and internal settings, but the focus should be the two executables in the desktop.

“Monitor” is a terminal based playback file useful when wanting to hear what the raspberry pi is receiving via its audio input.

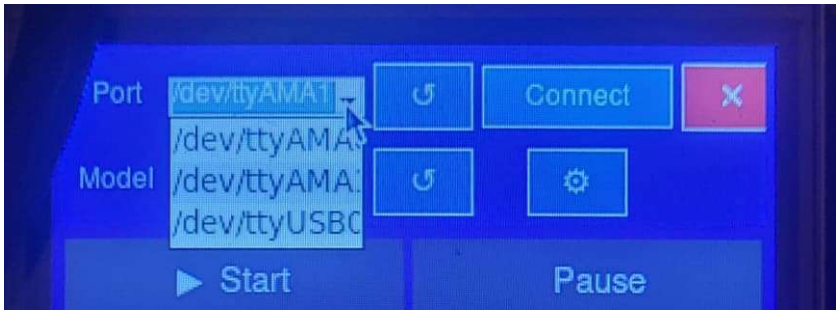
“Drawing Machine” opens the GUI to control the Scribe, more detailed information in the section below.

UI quick guide

The diagram below details all the different UI elements on the Scribes control panel.



The first thing you're going to want to do is connect the raspberry pi to the Blackbox motor controller. You can do this by using the Port dropdown menu and selecting the corresponding one. It should how up as "/dev/ttyUSBC...". If it doesn't show up, try hitting the refresh button and ensuring a proper cabled connection between devices.



At this point, you should always run the homing sequence for the Scribe to get its footing. Model controls should be disabled and manual controls limited until it has finished. Running home not only helps the drawing machine figure out its location in space but also puts in place its movement boundaries which makes it essential from a safety standpoint.

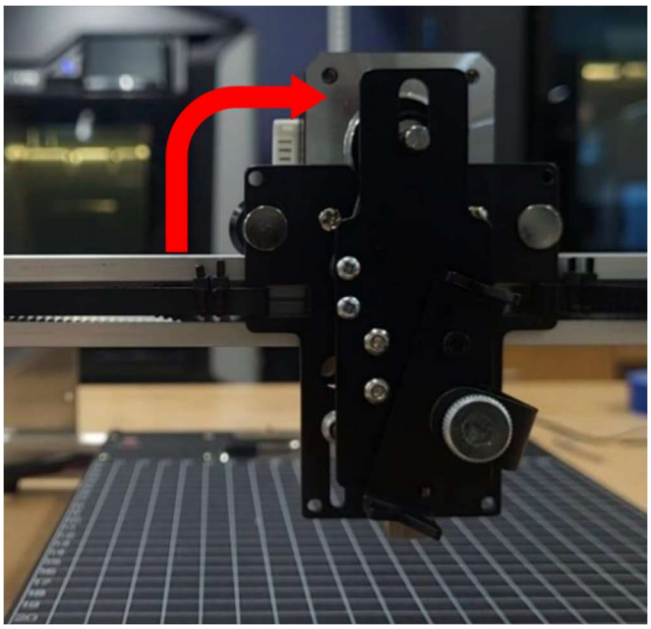
Once done, all controls should be enabled, and you can go ahead and either use the Scribe manually or run a custom drawing model automatically.

Common Errors

Listed below are some common errors that can come up when using the scribe and how to deal with them.

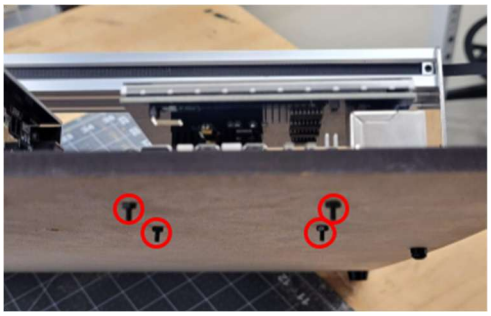
1. Z Axis stuck in the down position

   Sometimes, the pen can get stuck further down then originally intended due to the machine pausing/stopping a job halfway through a grbl command. This will prevent the pen from properly lifting back up and staying stuck in its down position. To fix this, all you need to do is manually move the top of the pen holder backwards in order to remove any slack added and the pen should pop back into place.



2. Raspberry Pi bottom screws

   The bottom four M2 screws holding down the raspberry pi can become loose after repeated use causing the nuts to fall off. This is unfortunately just an issue of my design but its still something to keep in mind moving forward. Occasionally, double check to see if all nuts and screws are still in place. The Wondry keeps these in stock so replacement shouldn't be hard if needed.



3. Emergency shut off

   If the Scribe ever moves past its software boundaries and begins repeatedly hitting one of the edges, you can use the power button on the side of the Blackbox as an emergency shut off. This should cut all power going to the stepper motors and prevent them from moving any farther

4. Random software bugs

   For any other unaccounted bugs that may pop up in the software, closing the drawing machine control panel and opening it back up should be enough to resolve most of them. If issues persist, resetting the raspberry pi itself could also prove to be helpful.

TECHNICAL SPECIFICATIONS

Connecting to the RPi5

The main way to connect to the raspberry pi is by setting up a wired SSH connection from your laptop using a standard ethernet cable.

The raspberry pi's connection settings are as follow:

Device Name: filmpi

Username: filmpi
Password: password

Pinging the device "filmpi" from your laptop should give the response shown below.

```
PS C:\Users\diego> ping filmpi

Pinging filmpi.local [fe80::2ecf:67ff:fe34:4412%15] with 32 bytes of data:
Reply from fe80::2ecf:67ff:fe34:4412%15: time<1ms
Reply from fe80::2ecf:67ff:fe34:4412%15: time<1ms
Reply from fe80::2ecf:67ff:fe34:4412%15: time<1ms
Reply from fe80::2ecf:67ff:fe34:4412%15: time<1ms

Ping statistics for fe80::2ecf:67ff:fe34:4412%15:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

This let's us know we have established a stable connection with device "filmpi" and that our laptop and raspberry pi are able to communicate with one another. If this doesn't work, I'd recommend trying to ping the pi's specific ethernet address your computer assigns it.

We can then ssh into the raspberry pi by logging into the device "filmpi" under user "filmpi". This should then prompt us for the password.

```
PS C:\Users\diego> ssh filmpi@filmpi
filmpi@filmpi's password:
Linux filmpi 6.12.25+rpt-rpi-2712 #1 SMP PREEMPT Debian 1:6.12.25-1+rpt1 (2025-04-30) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Apr 27 06:00:44 2026 from 192.168.50.1
filmpi@filmpi:~ $
```

File Structure

~/LCD-show *Managing the touchscreen display
    /LCD-hdmi.py
    /LCD35-show.py

~/Desktop *How to set up
    /DrawingMachine.desktop
    /Monitor.desktop

~/Documents
    /drawing_machine_ctrl *GitHub repo
        /main.py
        /machine.py
        /gui.py
        /config.py *GRBL 2 Axis Firmware
        /models *Adding a new model
            /cartesian.py
            /plants_demo.py

    /misc *Miscellaneous programs
        /audio_capture
            /audio_capture.py
        /monitor.py
        /RMS_reader.py
        /usb_list.py

GitHub repo

The current GitHub repo for "drawing_machine_ctrl" can be found in the link below and pulled directly from the terminal. To push to the repo and update its contents you'll need access as a collaborator. Please feel free to contact me if needed.

https://github.com/dolive13/drawing_machine_ctrl

The repo has five main items: main, machine, gui, config, and models. These are what build and establish the interface the user utilizes to control the drawing machine as well as establish communication with the Blackbox motor controller. None of them, except for the config and models, should need to be modified, in fact I advise against it, yet their functions are as follow:

- main.py: bootup file for the program. Called once at the beginning to get everything started. This is the file that is ran when we execute "DrawingMachine.desktop".

- machine.py: thread that continuously communicates with the Blackbox motor controller. Used to queue, push, and reset any GRBL commands sent for the controller's firmware to interpret. Backend of the code.

- Gui.py: this is the thread running in parallel to machine.py responsible for the frontend elements the user interacts with. More information in “UI quick guide”.

- Config.py: this file holds the current GRBL settings pushed onto the Blackbox’s firmware. Used mainly to set the drawing boundaries, speed, and precision. Most rates should be left alone but if you need to recalibrate the mm to step distance for the machine or change the jogging speed it can be down from here. More info in how to do that in “GRBL 2 Axis Firmware”.

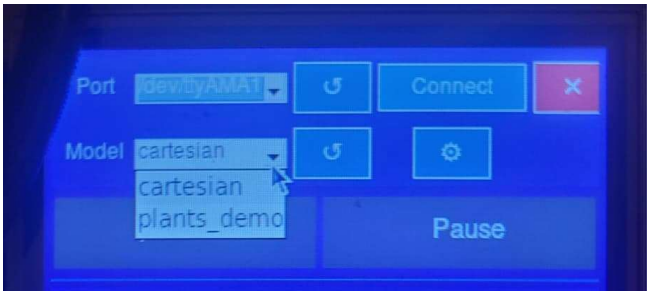
- /Models: This is the folder that holds all the different drawing files that the Scribe can execute. This is probably the only item that I recommend you modify since it’s the one in charge of uploading new programs to the machine. More details in how to do so in “Adding a new model”.

Adding a new model

As stated previously, this folder holds all the different drawing files that the Scribe can execute. By default, it already comes with two different files I’ve included.

- Cartesian.py: Draws an x, y coordinate plane in the middle of the page; used mainly for debugging purposes. A useful tool when trying to properly center a page in the drawing space.

- plants_demo.py: This is the plant to synth to drawing machine model I worked on with Rattner during my independent study with him. I recommend you use this as an example on how to build any other future models.



Changing between models Simply requires you to select a different one from the Model dropdown menu as shown above. This menu should populate automatically depending on the contents of the “/models” folder.

Miscellaneous programs

There are 4 different additional programs that should aid you in the use of the scribe:

```
filmpi@filmpi:~/Documents/misc $ ls
audio_capture  monitor.py  RMS_reader.py  usb_list.py
```

- /audio_capture: Within the audio_capture folder you'll find the audio_capture.py file. This is used to create an audio sample of whatever the Scribe's audio input is hearing. Depending on your input, you may have to reconfigure certain digital signal processing models like "plants_demo.py". To properly clean the signal and tailor it to whatever you need, it can be useful to create a quick capture of the sound in multiple different forms. This is why this file is in its own folder since it creates the files listed below when running.

  ```
  filmpi@filmpi:~/Downloads/audio_capture $ ls
  audio_capture.py  capture_dsp_float32.npy  capture_features.npz  capture_preview.wav  capture_raw_float32.npy
  ```

  Having these files and then passing them to an AI model to analyze has proven a quick and effective way to fine tune our signal processing so defiantly a useful tool to have in your back pocket.

- Monitor.py: This is a playback file that reproduces via the output jack whatever the raspberry pi is hearing via its unput. Easiest to run via its desktop executable.

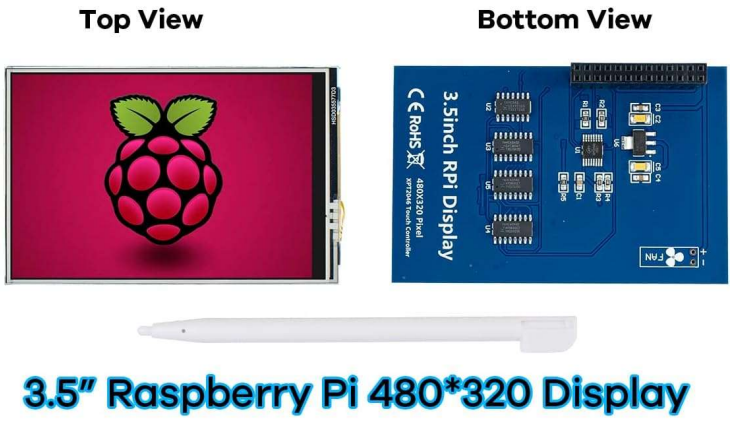
- RMS_reader.py: quick volume gauge running within the terminal.

- Usb_list.py: creates a list of all connected USB devices and in which ports.

Since these are supplementary programs that are not necessarily needed for the base use of the drawing machine, most of them don't have a quick access executable like "DrawingMachine.desktop". The only exception to this is monitor.py which runs via "Monitor.desktop". To run them you'll have to execute them manually via the terminal.

Managing the touchscreen display

The main interface for the Scribe is the following Waveshare 3.5 Inch 480x320 Touch Screen LCD SPI display.



In order to use it with the pi we pull the following GitHub repository from the developers team which give us a wide array of functionality: https://github.com/waveshareteam/LCD-show

```
filmpi@filmpi:~/LCD-show $ ls
boot                  LCD32-show      MHS35-show            PyMouse-1.0.tar.gz
DPI5_7_800_480-show   LCD35-show      MHS40C-show           python-xlib_0.23-2_all.deb
DPI7_1024_600-show    LCD55-show      MHS40-show            README.md
error_output.txt      LCD5-show       MIS35-show            rotate.sh
etc                   LCD7B-show      Mouse_Key.py          rpi-fbcp
LCD101H-show          LCD7C-show      MPI3508_480_320-show  system_backup.sh
LCD101S-show          LCD7H-show      MPI3508-show          system_config.sh
LCD101TMP-show        LCD7S-show      MPI3510-show          system_restore.sh
LCD101Y-show          LCD-hdmi        MPI4008-show          usr
LCD154-show           MHS24-show      MPI4009-show          xinput-calibrator_0.7.5-1_armhf.deb
LCD24-3A+-show        MHS32-show      MPI5001-show          xinput-calibrator_0.7.5+git20140201-1+b2_arm64.deb
LCD24-show            MHS35B-show     MPI5094-show          xserver-xorg-input-evdev_1%3a2.10.6-1+b1_armhf.deb
LCD28-show            MHS35IPS-show   NANO24-show           xserver-xorg-input-evdev_1%3a2.10.6-2_arm64.deb
```

Though it may seem overwhelming at first, there really are only two main commands you should be worried about:

- LCD35-show: when running this command, the raspberry pi connects directly to the LCD screen via its SPI ports and gives a clear image for us to use. Sometimes the display may show up with the wrong orientation, this can be changed by specifying by how much to rotate it when running the command

  For example:

  LCD35-show *initialize the display with the last saved settings

  LCD35-show 90 *initialize the display but changing the orientation 90°

- LCD-hdmi: disables the LCD display and re-enables the HDMI connection. It should be noted that while the touchscreen display is working the HDMI ports will be disabled and vice versa.

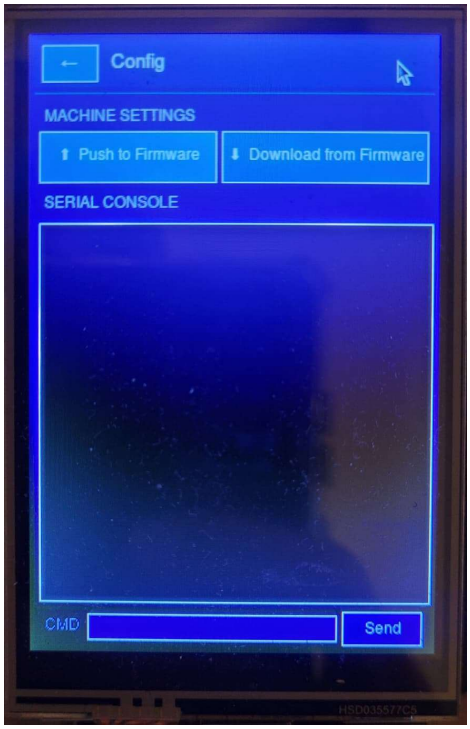
GRBL 2 Axis Firmware

The standard, open source GRBL firmware is made for more robust 3-axis CNC machines. In order to get it to work with our current setup I had to modify it a bit. The two biggest changes are the removal of a z axis homing sequence, meaning that when the machine runs $H it'll reset its x and y origin points but will ignore the Z axis. This is because our third axis is the pen and not really an additional plane.

The other change is the inclusion of the new command $Z which manually resets the machine coordinate of the z plane back to 0. When we pause or stop aa job halfway through, we wipe the commands queued inside the Blackbox. Sometimes, however, this wipe causes it to lose power momentarily. Since the Z axis is spring loaded it pulls the pen back to its original position causing a discrepancy between where the code thinks the pen is and where it landed. To account for this, we use the $Z command to reset its coordinate back to 0.

The forked GitHub repo with the applied changes can be found here:

https://github.com/dolive13/grbl_scribe.git



The firmware in the Blackbox should already be set, but in case you need to modify it I've created the config page above for quick edits. The download button updates the config file in the pi with whatever is currently sitting in the Blackbox. The push button does the exact opposite, pushing any edits we've manually done in the config all at once. I've also added a terminal in case you need to send individual gcode commands to the motor controller.