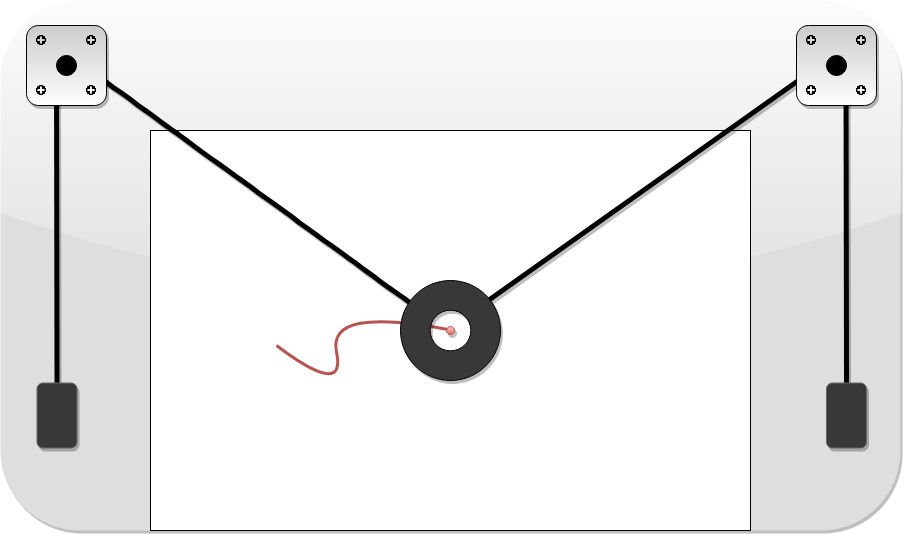
Interactive Programmable Arts Display

V Plotter Installation and User Manual



Version History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Description** | **Author** |
| 1 | 24/10/2019 | Initial creation | Zane Drysdale – [zjdrysdale@gmail.com](mailto:zjdrysdale@gmail.com) |
| 2 | 26/10/2019 | Firmware and software Instructions update | Zane Drysdale – [zjdrysdale@gmail.com](mailto:zjdrysdale@gmail.com)  Corey Page –  Corey.t.page@outlook.com.au |
| 3 | 27/10/2019 | 5.1 .1 Values update | Corey Page –  Corey.t.page@outlook.com.au |
| 4 | 27/10/2019 | Team Signoff | Zane Drysdale – [zjdrysdale@gmail.com](mailto:zjdrysdale@gmail.com)  Corey Page –  [Corey.t.page@outlook.com.au](mailto:Corey.t.page@outlook.com.au)  Franklin Moon – [Franklin.m.moon@gmail.com](mailto:Franklin.m.moon@gmail.com)  Zinan Chen– |

Contents

[1 Introduction and Purpose 4](#_Toc23339945)

[2 Hardware 4](#_Toc23339946)

[2.1 Components 4](#_Toc23339947)

[2.1.1 Arduino UNO 5](#_Toc23339948)

[2.1.2 Stepper Motor 5](#_Toc23339949)

[2.1.3 Power Supply 5](#_Toc23339950)

[2.2 Hardware Setup 5](#_Toc23339951)

[2.2.1 Writing Head 5](#_Toc23339952)

[2.2.2 Stepper Motors 5](#_Toc23339953)

[2.2.3 Servo Motor 6](#_Toc23339954)

[2.2.4 Arduino and Stepper Motor Controller 6](#_Toc23339955)

[2.2.5 Power Supply 6](#_Toc23339956)

[2.3 Modifications or Customizations 7](#_Toc23339957)

[3 Firmware 7](#_Toc23339958)

[3.1 Requirements 7](#_Toc23339959)

[3.2 Installation – Custom Firmware 7](#_Toc23339960)

[3.3 Modifying firmware from source 8](#_Toc23339961)

[4 Software 9](#_Toc23339962)

[4.1 Requirements 10](#_Toc23339963)

[4.2 Installation 10](#_Toc23339964)

[4.3 Run from Source 10](#_Toc23339965)

[4.4 Compile from Source 11](#_Toc23339966)

[5 Configuration 12](#_Toc23339967)

[5.1 Configure Polargraph application 12](#_Toc23339968)

[5.1.1 Physical Attributes 12](#_Toc23339969)

[5.1.2 COM Configuration 14](#_Toc23339970)

[6 User Guide 14](#_Toc23339971)

[6.1 Navigating the Application 14](#_Toc23339972)

[6.1.1 Input: 14](#_Toc23339973)

[6.1.2 Setup 15](#_Toc23339974)

[6.1.3 Trace 15](#_Toc23339975)

[6.1.4 Queue 16](#_Toc23339976)

[6.2 Drawing 16](#_Toc23339977)

[6.2.1 Queue 16](#_Toc23339978)

[6.2.2 Homing 16](#_Toc23339979)

[6.2.3 Manually Moving 17](#_Toc23339980)

[6.2.4 Draw a Vector 17](#_Toc23339981)

[6.2.5 Draw a Bitmap 17](#_Toc23339982)

[6.2.6 Draw shaded pixels from an Image 17](#_Toc23339983)

[6.2.7 Trace lines from an Image 18](#_Toc23339984)

[7 Troubleshooting 18](#_Toc23339985)

[8 Glossary 18](#_Toc23339986)

[9 Appendices 19](#_Toc23339987)

[9.1 Appendix 1: Additional Resources: 19](#_Toc23339988)

[9.2 Appendix 2: Protoneer Stepper Motor 19](#_Toc23339989)

[9.3 Appendix 3: Motor Wiring Diagram 20](#_Toc23339990)

# Introduction and Purpose

A V Plotter is a drawing robot which uses two motors to move a hanging writing implement across a drawing surface.

The purpose of this document is to provide a complete end-to-end guide for the creation of the V Plotter, including hardware requirements as well as software installation, configuration, usage, and troubleshooting.

This complete solution is reliant upon three distinct yet interrelated subsets. The **hardware** which is the physical canvas, motor and electronics setup, the Windows/Linux **software** which handles end user input, and the Arduino **firmware** which interprets commands from the software and controls the motors. This document details the installation and configuration of all three of these subsets as part of the overall solution.

This document is targeted towards anyone who wishes to set up the V Plotter and has basic computing knowledge. This guide presumes that you are setting up the V Plotter with the hardware as provided, however, additional notes cover alternative implementations or possible modifications or options.

**Some steps require caution, this information is in red.**

# Hardware

The physical aspects of the V Plotter consist of the motors and the board that they are mounted to, as well as the belts and physical electrical and computer hardware. The below diagram details at a high level the basic physical components:

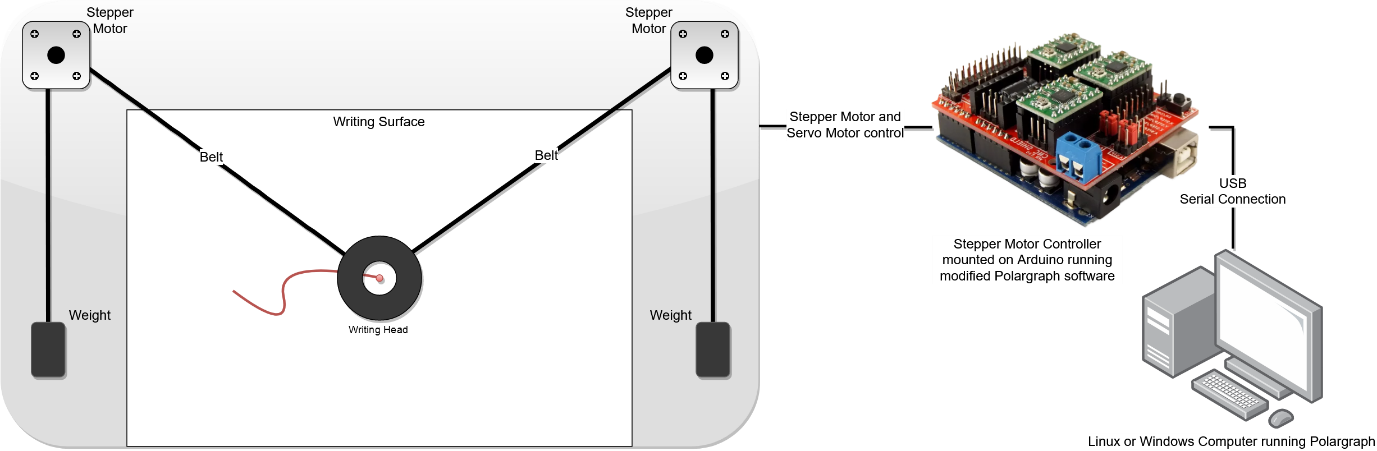
The

Figure 1: Overall V Plotter setup

## Components

The V Plotter as delivered consists of the following required and supplied hardware components:

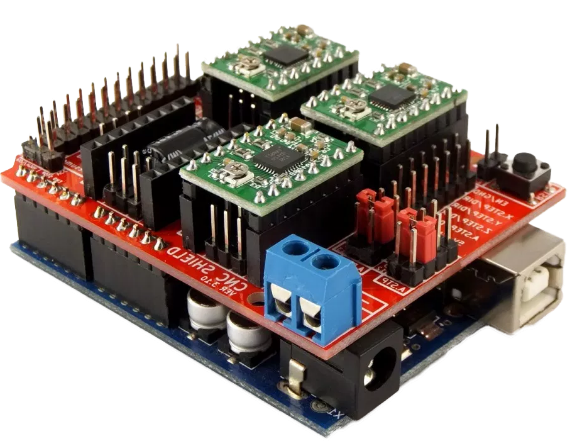
* Machine surface – overall surface that the motors are mounted to
* Writing surface
* 2 NEMA stepper motors. 200 steps per revolution.
* Official Arduino UNO loaded with modified Polargraph firmware
* Stepper Motor Controller – Specifically the Protoneer CNC Shield V3: <https://blog.protoneer.co.nz/arduino-cnc-shield>
* 12V 1 Amp DC Power supply
* Belts and suitably toothed and spindled gears
* Microsoft Windows based computer running Polargraph
* 3D printed motor mounts
* Associated wiring for motors

The main components are further detailed as follows:

### Arduino UNO

The Arduino is responsible for translating the commands sent over USB from the Polargraph application into electrical signals to the stepper motor controller. As such, the Arduino and stepper motor controller are highly dependent upon one another.

### Stepper Motor

The stepper motor driver is responsible for actuating the motors as required, and electronically separating logic and power for safety and durability. The Protoneer stepper motor is typically used for three-axis CNC machines. The Protoneer stepper motor provided is designed to be directly compatible with an Arduino UNO, and sits directly on top, interfacing with the Arduino outputs directly, as shown in Figure 2:

### Power Supply

Figure . Stepper Motor Controller mounted on top of an Arduino UNO

Responsible for providing power to the stepper motor controller. Simply plugs into the DC female terminal on the Arduino, also visible in Figure 2.

## Hardware Setup

This section details the physical installation of the components.

The physical installation is provided pre-assembled and will require minimal setup or extra assembly. However, if assembly is required:

### Writing Head

The writing head is the device which sits hung across the drawing surface. It holds the pen or other writing implement and is supported and moved by the belts.

1. Place the pen into the centre of the writing head and tighten the threads to secure it in place. It is imperative that the tip of the pen only minimally protrudes from the underside of the writing head to improve accuracy
2. Affix the belts around the mounts on the corners of the writing head
3. Mount the servo motor into the top of the writing head

### Stepper Motors

The motors are responsible for moving the writing head by lengthening or shortening the amount of belt between the motor and writing head.

1. Ensure that the motors are bolted to their mounts
2. Bolt each mount to the machine surface
3. Plug in the motor-end of the supplied wiring
4. Run the belt atop the gear to form the overall ‘V’ shape, as in Figure 1
5. Affix the weights to the end of the belt

### Servo Motor

The servo motor is responsible for raising and lowering the pen off the page. It is connected to the stepper motor controller in our solution. Refer to Appendix 3 for the wiring diagram.

1. Plug the RED wire of the servo into the +5V output on the Protoneer. This is the power for the servo.
2. Plug the BROWN wire of the servo into the BLACK side of the *CooEn* pin on the Protoneer. This is the ground for the servo.
3. Plug the ORANGE wire of the servo into the WHITE side of the *CooEn* pin on the Protoneer. This is the signal for the servo.

### Arduino and Stepper Motor Controller

As detailed in their sections above, the Arduino and stepper motor controller are responsible for receiving commands over USB and driving the motors. The Protoneer stepper motor controller sits directly atop the Arduino as shown in Figure 2. Refer to Appendix 3 for the wiring diagram.

1. Sit the Protoneer stepper motor controller atop the Arduino as depicted in Figure 2.
2. Plug the other end of the previously attached motor wires into the stepper motor controller X and Y outputs, such that the left motor is plugged into the ‘X’ output, and the right motor is plugged into the ‘Y’ output. Take note of the direction that the cables are plugged in; the lip on the connector is facing the motor shields of their respective outputs, as visible below in Figure 3

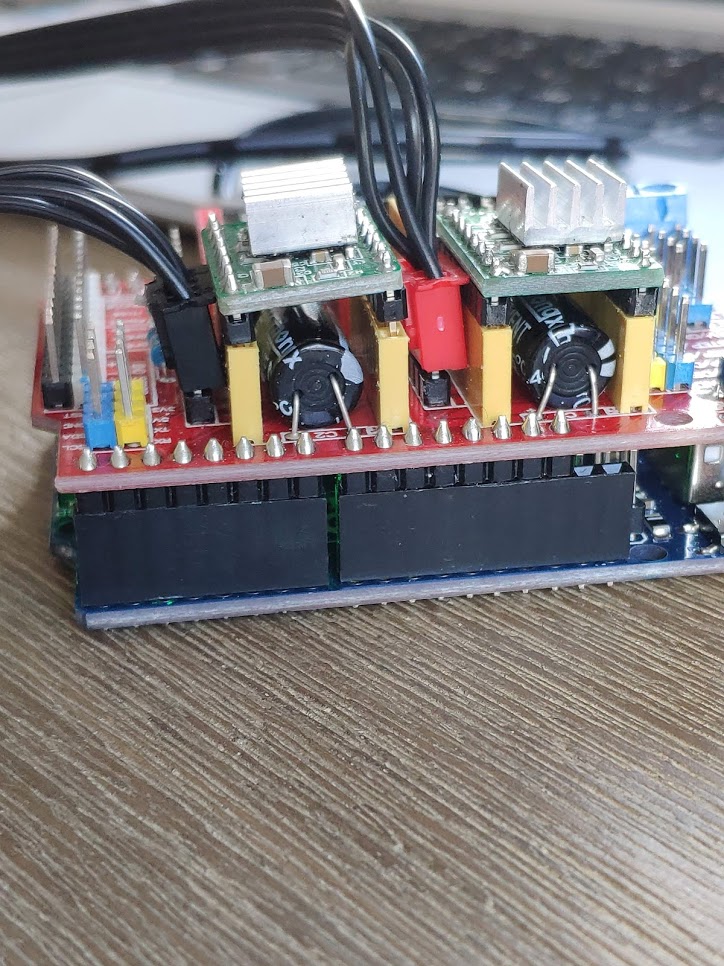


Figure 3: Motor wiring on stepper controller

### Power Supply

Caution: Although the supplied power supply has a low risk of electric shock, be careful to ensure that it is not shorted, it could hurt or even worse break something.

Caution: Never cut power to the stepper motor controller while the V Plotter is in operation or holding the motors’ position, as it can damage the stepper motor controller.

* Plug the output of the power supply into the female DC power input of the Arduino, as visible in Figure 2

## Modifications or Customizations

The V Plotter is designed to be easily modular and flexible to alternative physical implementation with only a software reconfiguration required. However:

* Increasing the size of the machine considerably may require motors with additional torque and larger sprockets, and possibly in turn a higher-output power supply.
* Alternative stepper motor controllers are possible; however, as detailed in the below section, the Arduino firmware may require customization to modify the output pins to match the alternative stepper motor controller.

# Firmware

The firmware is responsible for receiving the commands from the Polargraph application running on a computer and sending the correct electrical signals to drive the stepper motor controller. It interprets ASCII commands sent over the USB connection from the Polargraph application running on a computer.

This solution uses the *Polargraph* firmware and associated end user application from polargraph.uk. This step covers the firmware half.

The Polargraph firmware expects an alternative stepper motor controller and therefore our solution includes a modified version of the firmware that supports the Protoneer stepper motor controller. 3.2 details how to install this modified firmware, or you can follow 3.3 to modify the firmware from source.

This step details how to modify and load the firmware on the Arduino if required. As delivered, the Arduino is already loaded with the modified firmware.

## Requirements

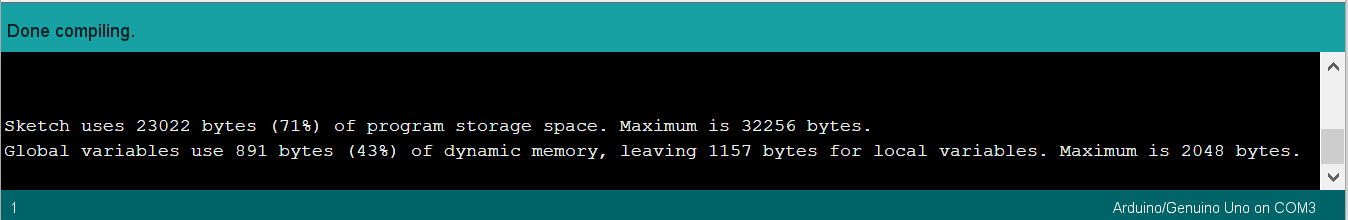
The modified firmware as delivered is intended for an Arduino UNO and the Protoneer stepper motor controller pairing. The firmware can be installed from a Windows, Linux, or Mac computer. The modifications made to the source include:

* Manual configuration of the stepper motor output pins on the Arduino to match the expected inputs on the Protoneer stepper motor controller.
* Manual configuration of the Pen lift pin, to allow the signal to be passed through to Protoneer stepper motor controller via the “Coolant Engage” pin, using the A3 pin on the Arduino.

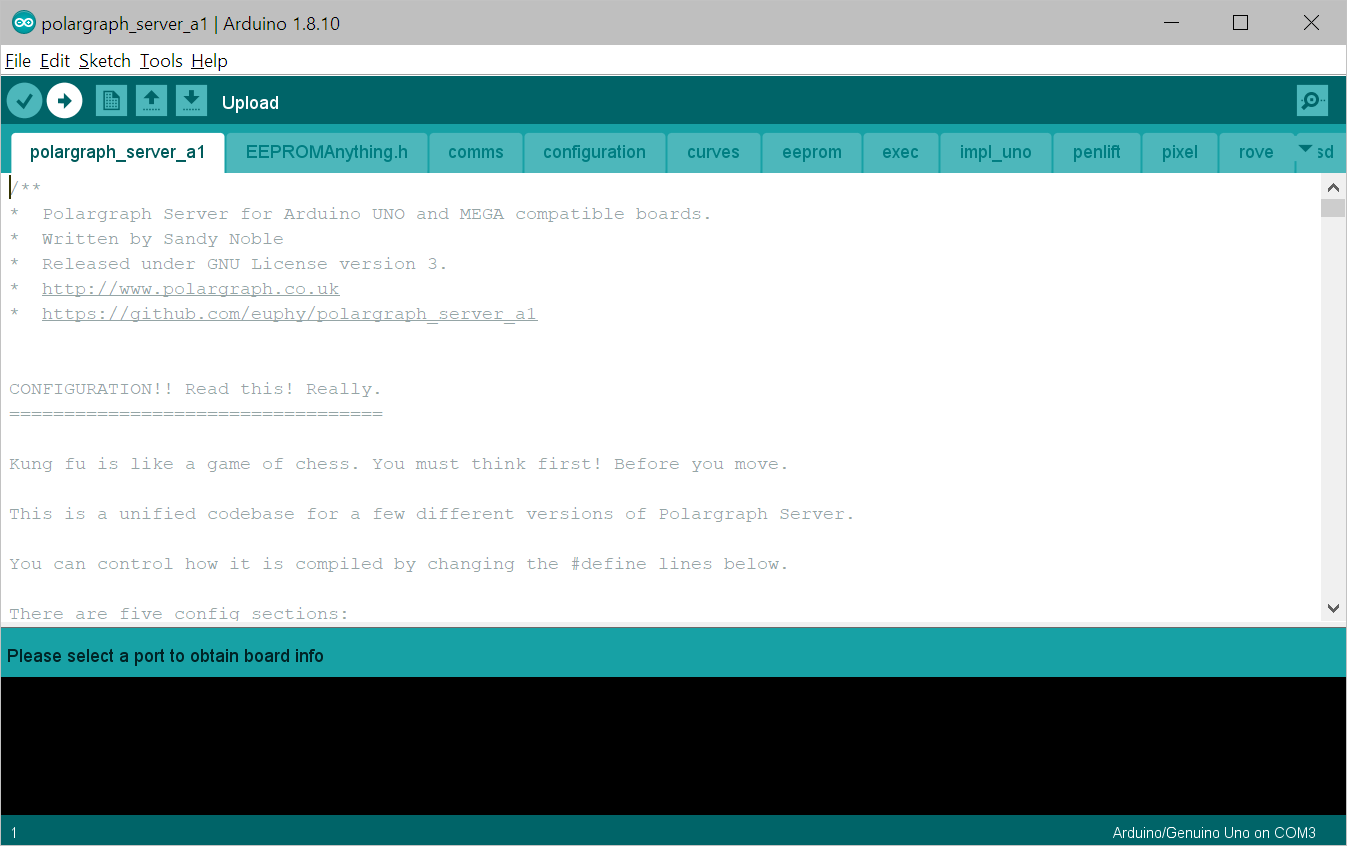
It would be best to clone or download the IPAD repo (<http://github.com/zanedrys/IPAD>) to ensure that you have the firmware and software required for the following steps.

## Installation – Custom Firmware

1. Firstly, install the Arduino IDE from arduino.cc for your platform
2. Download the modified firmware from GitHub: <https://github.com/zanedrys/IPAD/tree/master/Polargraph/Polargraph%202017-11-01/modified%20arduino-source>, or even better clone the repository for future use.
3. Included are the Arduino libraries that are required and known to work to compile the source code. Copy the files in the “libraries” folder into your Arduino Sketch directory, or alternatively add them to your Libraries from the Arduino IDE.
4. Open the “*polargraph\_server\_a1*” sketch in the Arduino IDE.
5. Verify the sketch to ensure that it compiles:



1. Plug the Arduino into your computer with a USB cable, and wait until the drivers are installed.
2. Within the Arduino IDE, check that your Arduino is connected. Under ‘*Tools’* -> ‘*Port’*, the Arduino should be listed.
3. Click the arrow to upload the firmware to the Arduino:



1. The Arduino IDE should tell you that the firmware was successfully flashed.

## Modifying firmware from source

This step is not necessary if you flashed the custom firmware from the previous step.

If you wish to modify the firmware yourself, the Polargraph firmware can be modified to work with a variety of stepper motor controllers:

* The Adafruit Motorshield v1 and v2
* Generic serial stepper drivers
* Four-wire signal amplifiers

By default, the source Polargraph firmware is designed to be used with an Adafruit stepper motor. The Protoneer stepper motor controller used in this solution is a generic serial stepper driver at core. Therefore, the source firmware must be modified to work with the Protoneer. Below are the steps required to modify the firmware from source with examples for the Protoneer controller in particular:

1. Download the source Polargraph firmware from GitHub: <https://github.com/euphy/polargraphcontroller/releases/>
2. Install the Arduino IDE from arduino.cc for your platform
3. Included within the “*arduino\_source*” directory are all the files required. Copy the files in the “*libraries*” folder into your Arduino IDE Sketch directory, or alternatively add them to your Libraries from the Arduino IDE
4. Open the “*polargraph\_server\_a1.ino*” sketch in Arduino IDE. It is commented well with instructions. Instructions specifically for the Protoneer follow:
5. Specify the custom motor driver:
   1. Comment out lines within section 3i, as this solution does not use the Adafruit:

// i. Adafruit Motorshield v1. The original, and still the best.

// #define ADAFRUIT\_MOTORSHIELD\_V1

// #include <AFMotor.h>

* 1. Uncomment out lines within section 3ii, as this solution uses a discrete stepper driver:

// iii. Using discrete stepper drivers? (eg EasyDriver, stepstick, Pololu gear)

// ----------------------------------------------------------------------------

// Don't forget to define your pins in 'configuration.ino'.

#define SERIAL\_STEPPER\_DRIVERS

* 1. Change the output pin for the Pen Lift to **A3** as in the following block of code.

// Pen raising servo

Servo penHeight;

const int DEFAULT\_DOWN\_POSITION = 90;

const int DEFAULT\_UP\_POSITION = 180;

static int upPosition = DEFAULT\_UP\_POSITION; // defaults

static int downPosition = DEFAULT\_DOWN\_POSITION;

static int penLiftSpeed = 3; // ms between steps of moving motor

const byte PEN\_HEIGHT\_SERVO\_PIN = **A3**; //Change the servo pin to A3, to pass through to the coolant enable pin.

boolean isPenUp = false;

* 1. Configure the Arduino output pins in ‘*configuration.ino’* as follows, to match the Protoneer. To see where we got these pins are from, refer to Appendix 2 for the Protoneer pinout:

#ifdef SERIAL\_STEPPER\_DRIVERS

#define MOTOR\_A\_ENABLE\_PIN 8

#define MOTOR\_A\_STEP\_PIN 2

#define MOTOR\_A\_DIR\_PIN 5

#define MOTOR\_B\_ENABLE\_PIN 4

#define MOTOR\_B\_STEP\_PIN 3

#define MOTOR\_B\_DIR\_PIN 6

AccelStepper motorA(1,MOTOR\_A\_STEP\_PIN, MOTOR\_A\_DIR\_PIN);

AccelStepper motorB(1,MOTOR\_B\_STEP\_PIN, MOTOR\_B\_DIR\_PIN);

#endif

1. Follow the instructions for *3.2 Installation – Custom Firmware* to install the modified firmware.

# Software

This solution uses the *Polargraph* firmware and associated end user application from polargraph.uk. This step covers the installation of the application half.

The application is responsible for providing an end-user graphical interface and sending commands to the firmware running on the Arduino. It can load vector and bitmap images.

## Requirements

The Polargraph application is written in Processing 2.2.1 and requires the Java Runtime Environment to run.

Processing is compatible with Windows (32/64 bit), Linux (32/64 bit), and Mac OSX. Therefore, Polargraph can only run on those platforms.

## Installation

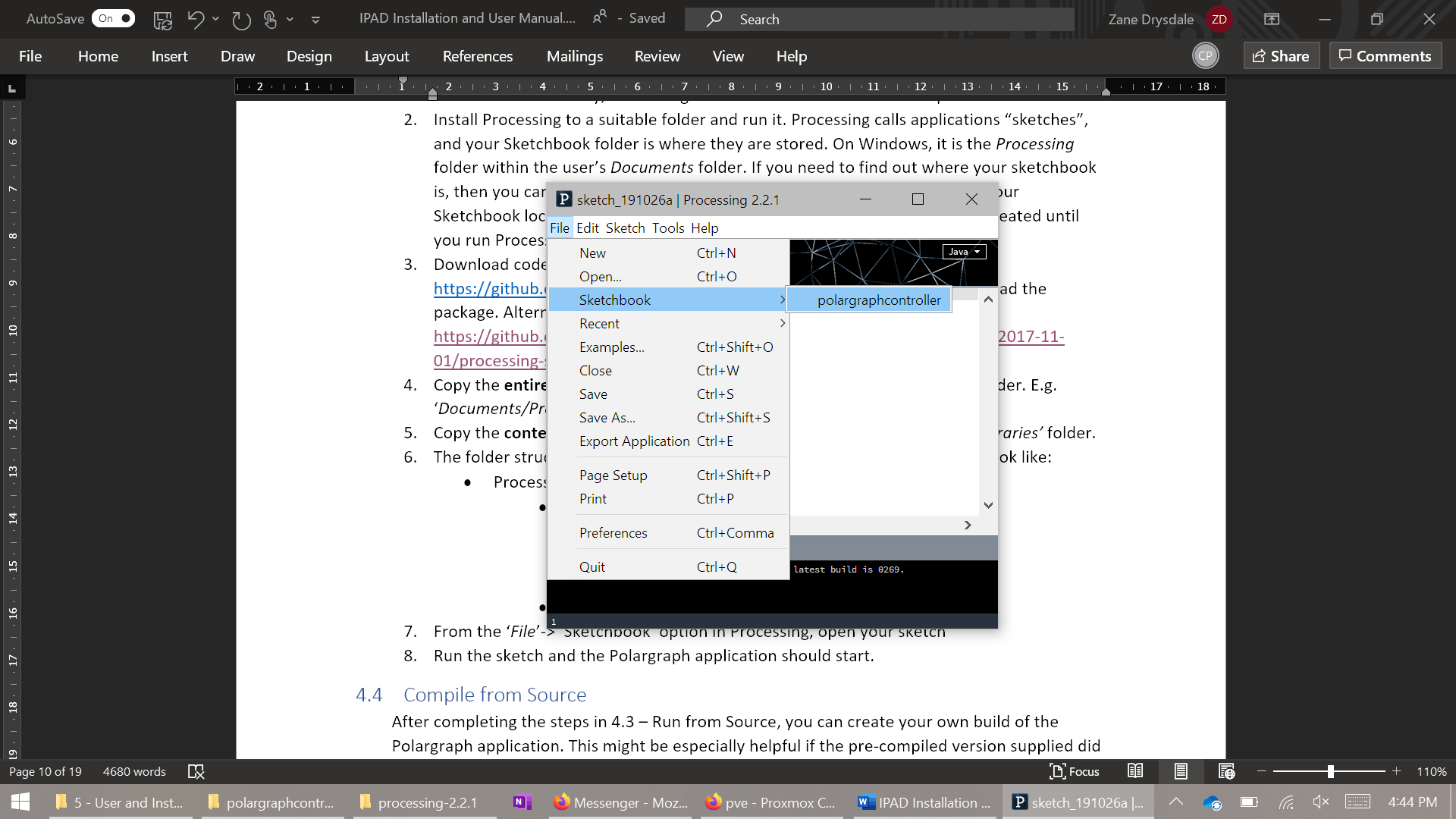
Included in the GitHub repository are pre-compiled versions of the Polargraph application: “IPAD\Polargraph\Builds”.

1. Copy the suitable folder for your system to a suitable location. The binaries are available in Linux and Windows, each in 32-bit or 64-bit versions. The Windows 64-bit version includes its own Java Runtime Environment, otherwise ensure that you have the Java Runtime Environment installed.
2. Run the suitable binary for your system.
3. After a minute or so on first run the application should start. If it doesn’t and Java is installed, it may be necessary to run it from source. Refer to section 4.3 below:

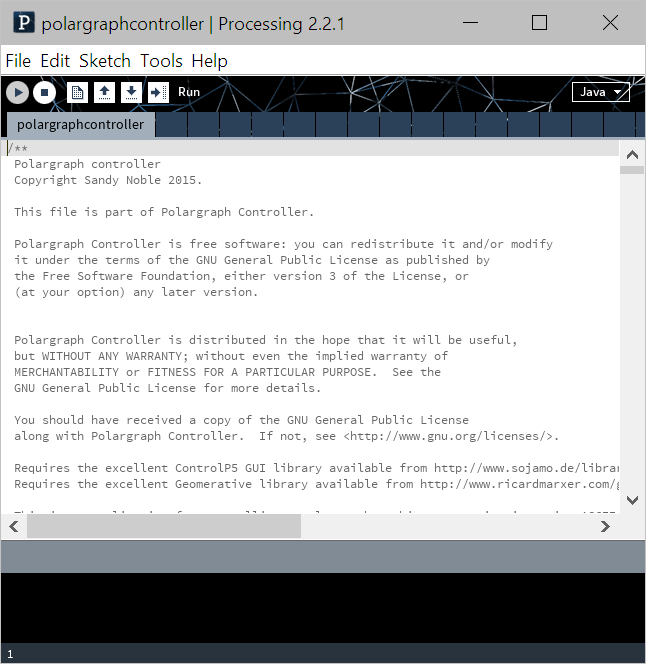
## Run from Source

Java, Windows, and Processing may not work as expected on some machines. In this case the Polargraph software can be run from source code. This method can also be used if the Polargraph software receives an update, or if you wish to modify the source code or debug the application.

1. Download Processing 2.2.1 from [processing.org](https://processing.org/download/?processing). Ensure to download the version with Java included. Alternatively, Processing 2.2.1 is included in the GitHub repo.
2. Install Processing to a suitable folder and run it. Processing calls applications “sketches”, and your Sketchbook folder is where they are stored. On Windows, it is the *Processing* folder within the user’s *Documents* folder. If you need to find out where your sketchbook is, then you can open Processing, and go *file->preference*s and it'll tell you your Sketchbook location at the top of the dialog box. The folder might not get created until you run Processing once.
3. Download code from the repo <https://github.com/euphy/polargraphcontroller/releases/latest> and download the package. Alternatively download the version from our repo: <https://github.com/zanedrys/IPAD/tree/master/Polargraph/Polargraph%202017-11-01/processing-source>
4. Copy the **entire** ‘*polargraphcontroller’* folder into your Processing sketch folder. E.g. ‘*Documents/Processing/polargraphcontroller’*
5. Copy the folders **within** the ‘[Processing libraries](https://github.com/zanedrys/IPAD/tree/master/Polargraph/Polargraph%202017-11-01/processing-source/Processing%20libraries)’ folder into the Processing ‘l*ibraries’* folder, not the folder itself.
6. The folder structure therefore after these changes (shown in bold) should look like:
   * Processing/
     + libraries/
       - **ControlP5/..**
       - **diewald\_CV\_kit/..**
       - **geomerative/..**
     + **polargraphcontroller/**
7. From the ‘*File*’-> ‘Sketchbook’ option in Processing, open the *polargraphcontroller* sketch:



1. Run the sketch and the Polargraph application should start.

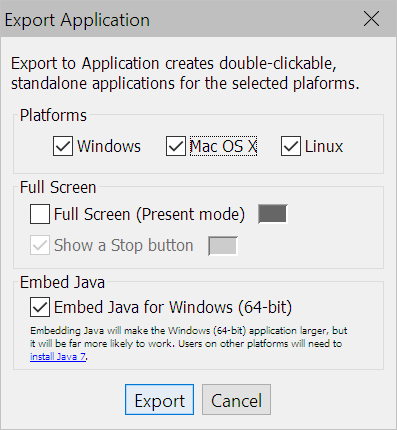


1. Congratulations, the Polargraph application should now be running.

## Compile from Source

After completing the steps in 4.3 – Run from Source, you can create your own build of the Polargraph application. **This might be especially helpful if the pre-compiled version supplied did not start, and you had to run the application from source as above.**

1. Open the sketch in Processing as detailed in 4.3 – Run from Source
2. Go *‘File’* -> *‘Export Application’:*
3. Choose the platforms to compile for, and whether to include Java in the Windows 64-bit build if applicable. Click Export:



# Configuration

Now that the hardware is set up, the Arduino is running the firmware, and the software is running on computer, it is time to set up the Polargraph software to finally draw. Below is a checklist to go through to ensure that the V Plotter is ready.

Pre-Flight Checklist:

1. Check that the board and surface is set up, and the belts and writing head are set up.
2. Check that the motors are correctly plugged into the motor controller.
3. Check that the stepper motor is correctly plugged into the top of the Arduino.
4. Ensure that the Arduino is connected to your computer over USB, and that it is powered on.
5. Plug in the power adaptor and turn the switch on.

## Configure Polargraph application

### Physical Attributes

The Polargraph application needs to know the following information about the physical setup to draw accurately. This information is entered into the ‘Setup’ page, as shown in 6.1.2 – Setup Page.

* **Steps per Rev**: How many steps your stepper motors require to make a complete revolution. As delivered, the motors require **200** **steps** per revolution.
* **mm per rev**: How much belt is let in or out per revolution of the stepper motor. As delivered, this is **50 mm**. To calculate this yourself simply find the circumference of your sprocket.
* **Motor Step Multiplier**: How many micro-steps your motor controller makes per full step of the motor. Stepper motor controllers can increase the resolution of your motors by using micro stepping. As delivered, the Protoneer is **not** performing micro stepping.
* **Machine Width in mm:** The physical width of the machine, refer to Figure 4: Machine Dimensions. As delivered, this is **724 mm.**
* **Machine Height in mm:** The physical height of the machine. Refer to Figure 4: Machine Dimensions. As delivered, this is **480 mm.**
* **Page Width:** The width of the drawing area available to be drawn on. Naturally, this must be smaller than your machine width. Allow for some buffer as the V Plotter will be increasingly less accurate the closer it attempts to draw to the motors. As delivered, this is **600mm**
* **Page Height.** The height of your page. This must also be small than your machine height and have buffer between the height of the motors and the top of the page. As delivered, this is **400mm.**
* **Page Pos X:** Where the page sits horizontally. Polargraph can set this to be in the centre of your machine. As delivered, this is **150 mm**.
* **Page Pos Y:** How far below the machine top that the page sits. Again, this cannot be too close to the top of the machine. As delivered, this is **130 mm.**
* **Home Pos X/Y:** Where the physical position of the writing implement is aligned with the start position. This is required so that Polargraph knows where the writing head is. Polargraph can automatically set this to be the top centre of your page. It can be wherever you like, if you can accurately manually place the writing head in that location.
* **Motor Max Speed:** The maximum speed that the machine will move the motors. Have this too low, and the motors will move very slowly, and your drawing will take a long time to complete. Have this too high, and you may encounter the belt hopping on the sprocket, which will ruin your image. **100**
* **Motor Acceleration:** How quickly the motors will accelerate up to the Motor Max Speed. Have this too high and the belt may hop on the sprocket while the machine is accelerating from a stop, or from changing direction. Have this too low, and your machine may never reach its Motor Max Speed. **80**
* **Pen Up Position:** The angle that the machine will hold the servo at when the pen is up. **70mm**
* **Pen Down Position:** The angle that the machine will hold the servo at when the pen is down. **187 mm**
* **Pen Thickness:** Determines how many lines the V Plotter can fit into an area. If you have a pixel that is 20mm square, and you have a 1mm pen tip, then you can only fit a maximum of 20 lines in before it's at its maximum density. Adding more ink then won't make it any darker. You can change the pen width on the setup tab, by changing the value of pen tip size and clicking send pen tip size. The tip size is not saved in the machine, it needs to be resent every time the machine is restarted, which is why the value is pre-loaded in the queue when you restart the controller.

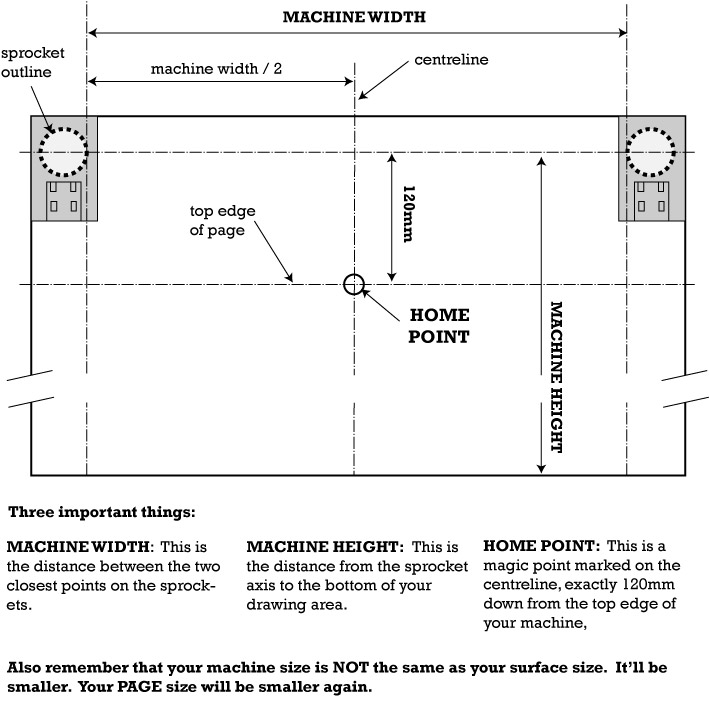
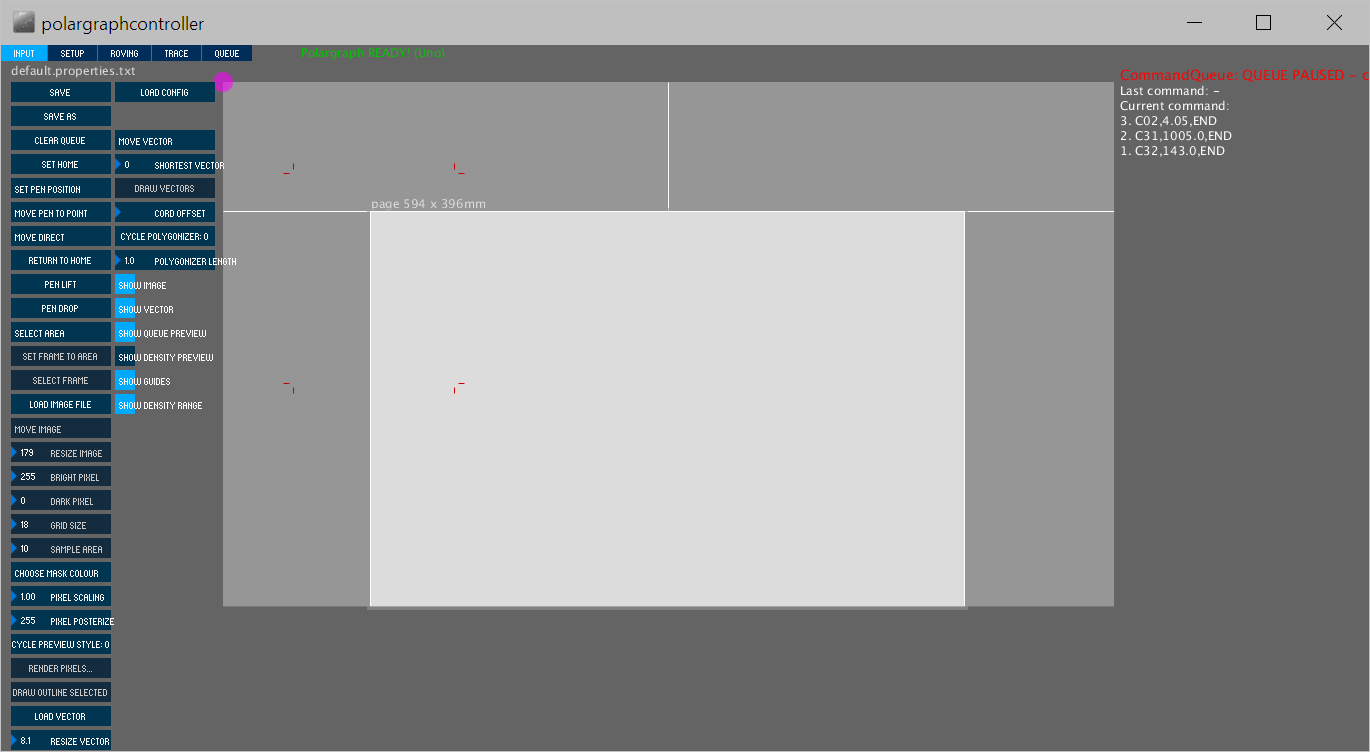


Figure 4: Machine Dimensions

### COM Configuration

The Polargraph application communicates with the Arduino firmware over a USB COM port. To set the COM port in the Polargraph application, go to ‘Setup’->’Serial Connection’, and choose your COM port. It can be best to restart Polargraph after changing this option.

The Polargraph application will display ‘Polargraph Ready! (Uno)’ when the Arduino is successfully connected:



# User Guide

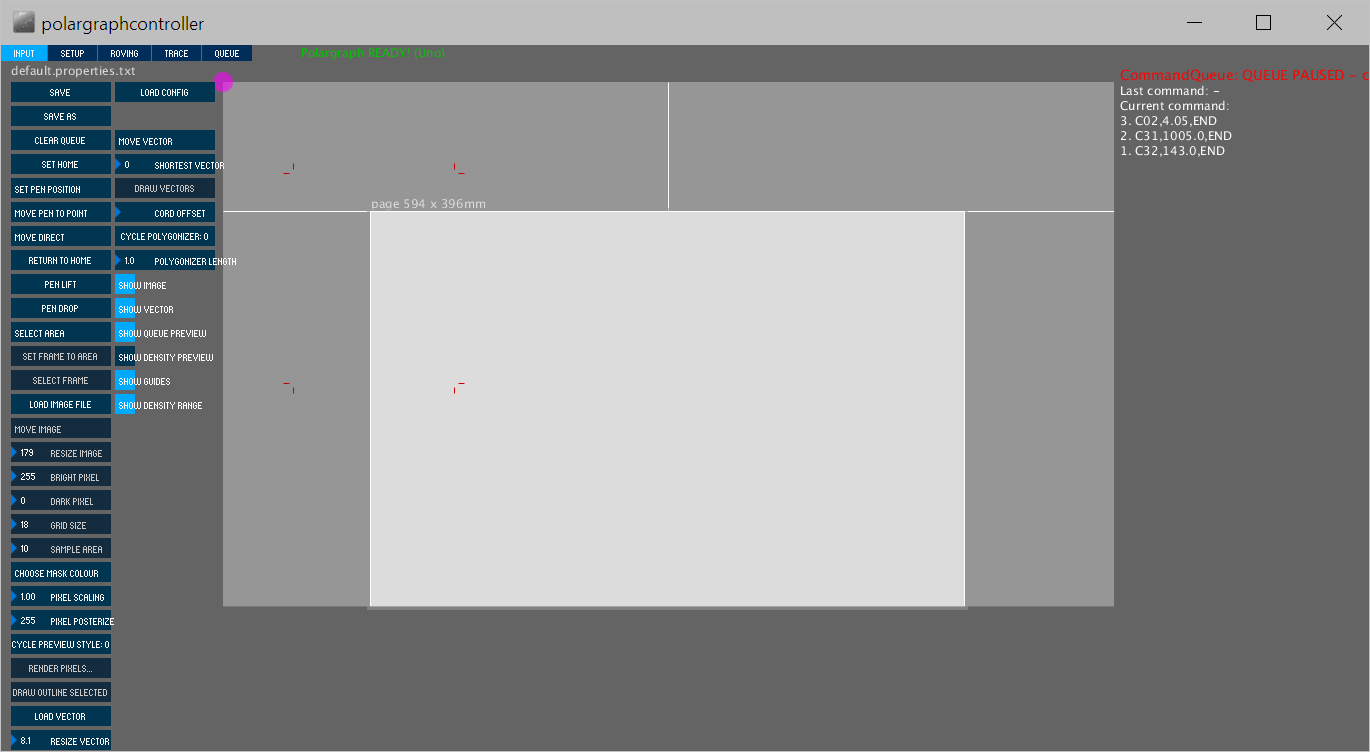
This section of the Manual will cover the usage of the application and presumes that the machine is set up with the information presented thus far.

## Navigating the Application

The Polargraph application is used through the following three views:

### Input:

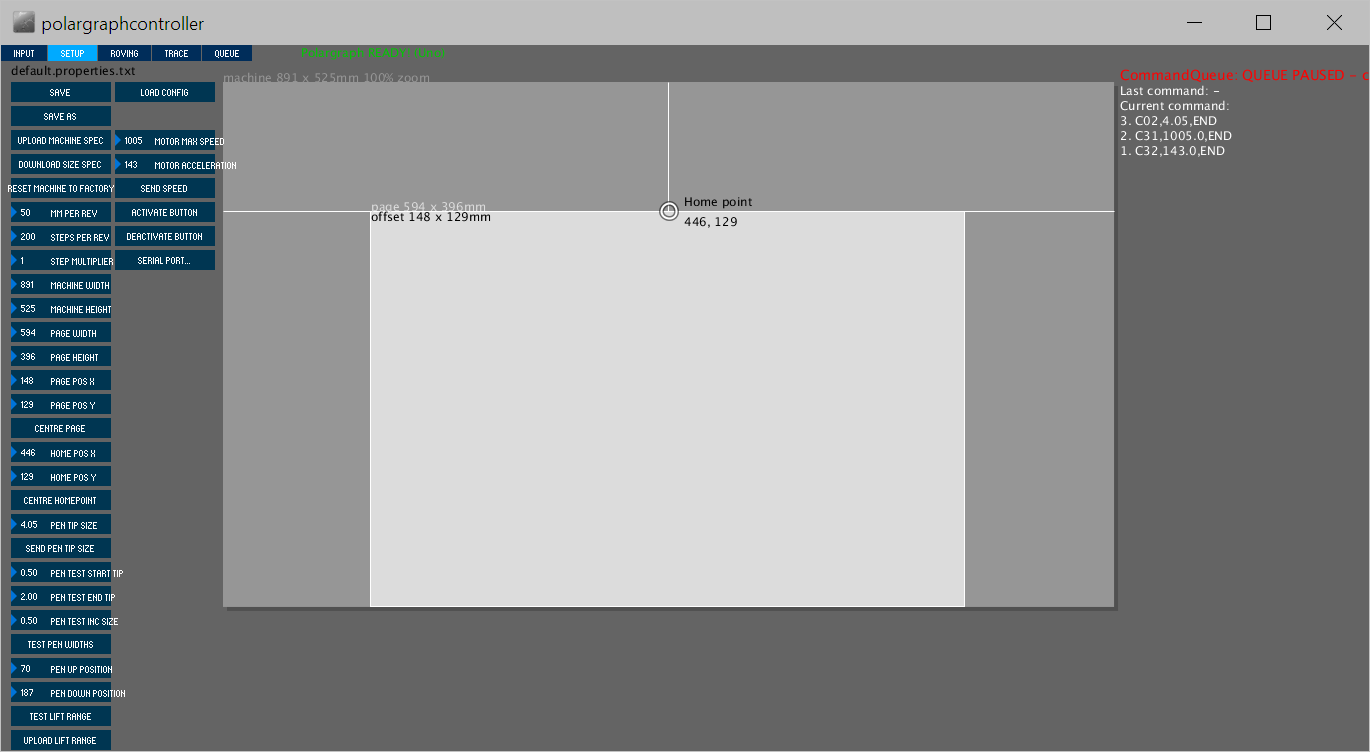
Input vectors, drawing and lines to be drawn.



### Setup

Sets up the machine configuration.

* Input the values from *5.1 – Configuring Polargraph Application* into this screen.
* Ensure to click ‘Upload Machine Spec’ with the queue running to send the updated settings to the Arduino.
* You can ‘Centre Page’ to place the page in the middle of the machine.
* You can ‘Centre Homepoint’ to place the home position at the top middle of the page.
* You can also export and load these settings with ‘Save As’ and ‘Load Config’.
* Ensure to click ‘Save’ to save these settings for the Polargraph application.

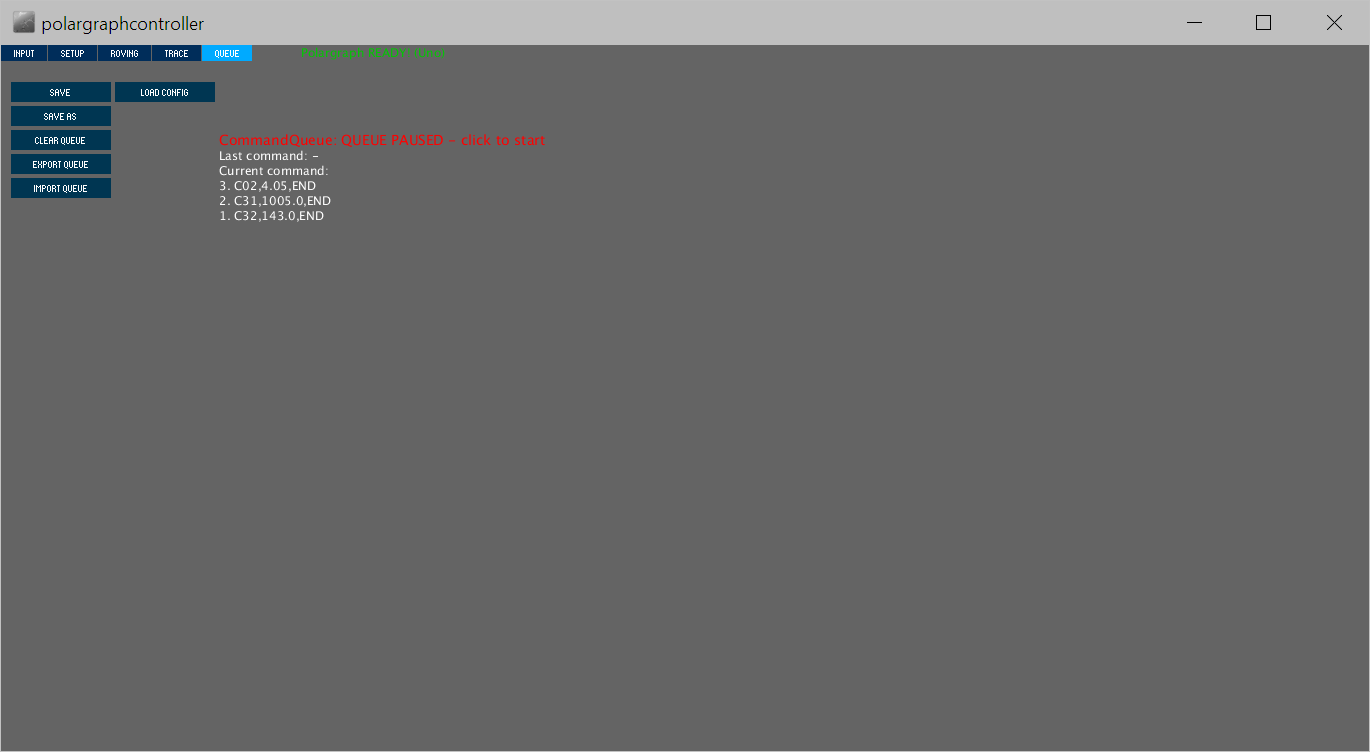


### Trace

Used to convert bitmaps to vectors.



### Queue

Used to export and import the command queue.

## Drawing

### Queue

As visible in the screenshots above, there is a queue to the right. This list contains the list of commands that will be sent to the Arduino. On start-up, this queue is in a Paused state, and will not send commands to the Arduino. To un-pause it, simply click the Command Queue text.

### Homing

The V Plotter needs to be homed on start-up, so that Polargraph knows the physical location of the writing head. Manually move the drawing head to be over the Home location set in Polargraph, then on the ‘Input’ page, click ‘Set Home’. Ensure that the queue is running. The motors will both turn and lock in the current location.

### Manually Moving

You can manually move the writing head around the page and raise and lower the pen as desired.

* Click ‘Move Pen to Point’ and click on the canvas to move the pen to a point on the drawing surface.
* Click ‘Raise Pen’ and ‘Lower Pen’ to manually raise and lower the pen with the servo.
* Click ‘Return to Home’ to move the pen back to home.

### Draw a Vector

Vector images are well-suited to work with Polargraph. To draw a vector image, from the ‘Input’ page:

1. Set Frame:
   1. Click ‘Select Area’ and select the entire page.
   2. ‘Set Frame to Area’ to allow the vector to be drawn within the selected area.
2. Load Vector
   1. Click ‘Load Vector’ and choose your vector image.
3. Place Vector
   1. Click ‘Move Vector’ and click on the page where you want the vector to be drawn.
   2. Drag ‘Scale Vector’ to scale the vector to the intended size.
4. Choose Shortest Vector
   1. Increase the ‘Shortest Vector’ amount to increase the minimum length of line that is drawn. If it’s too low, the pen will move very slowly.
5. Draw!
   1. Click ‘Draw Vectors’ to load the commands to draw the vector into the queue.
   2. Ensure that your Queue isn’t paused, and the machine will start drawing the vector.

### Draw a Bitmap

There are two ways to draw bitmaps. It can detect and draw lines from an image, or it can draw pixels.

### Draw shaded pixels from an Image

This method samples an image, and then draws the pixels.

* Load and Sample Image:

1. Click ‘Load Image File’ and select your PNG or JPG image.
2. Click ‘Move Image’ to place the image where you want on the page.
3. Click ‘Resize Image’ to scale the image to the size that you want.
4. Click ‘Select Area’ to select the area of the canvas to sample. You will now have a preview of how the image will be sampled.
5. Adjust the ‘Dark Pixel’ and ‘Bright Pixel’ settings to limit the brightest and darkest pixels to be sampled.
6. Adjust the ‘Grid Size’ to choose how fine to sample the pixels.
7. Adjust the ‘Sample Area’ to choose the area of pixels to be sampled against the grid.

* Render Pixels
  1. Click ‘Render Pixels’
  2. Choose the direction to start from, as well as the shading style
  3. Click ‘Render’ and the commands will be generated and placed into the queue, ready to be printed.

### Trace lines from an Image

This method detects lines within an image, and then draws them.

* Load Image:

1. Click ‘Load Image File’ and select your PNG or JPG image.
2. Click ‘Move Image’ to place the image where you want on the page.
3. Click ‘Resize Image’ to scale the image to the size that you want.

* Trace Image
  1. Move over to the ‘Trace’ tab. You will see your image has now been traced.
  2. Adjust the ‘Blur’, ‘Simplify’ and ‘Posterise’ options to define the trace
  3. Click ‘Capture’ to capture the trace
  4. Click ‘Draw Capture’ to send the commands to the queue, ready to be printed.

# Troubleshooting

|  |  |
| --- | --- |
| **Issue** | **Solution** |
| 1. Commands aren’t running/V Plotter isn’t moving | * Check that the queue isn’t paused. * Refer to Issue 2. |
| 1. Polargraph “isn’t connected” error | * Change the COM port under ‘Setup’ to another port. Ensure to click OK. Restart Polargraph. |
| 1. V Plotter isn’t moving as expected: | * If the V Plotter movement appears horizontally inverted but not vertically inverted, the motors are plugged into the wrong outputs **and** have their wiring reversed.   + Swap the motors on the stepper motor to the opposite output and flip the connectors. * If the V Plotter movement appears vertically inverted but not horizontally inverted, the motors need to be swapped.   + Swap the motors on the stepper motor to the opposite output |
| 1. The belt is skipping teeth under acceleration | * Check that the weights are attached * Lower the “Acceleration” rate in the setup window of Polargraph. |
| 1. The belt is skipping teeth at speed | * Check that the weights are attached * Lower the “Max Speed” option in the setup window of Polargraph. |
| 1. V Plotter moves too slowly | * Increase your ‘Max Motor Speed’ and your ‘Motor Acceleration’ settings. * If it is moving slowly drawing vector images:   + Increase the “Shortest Vector” setting on the ‘Input’ page. |

# Glossary

|  |  |
| --- | --- |
| **Term:** | **Definition:** |
| V Plotter: | The overall solution. Uses two stepper motors to move a writing head. |
| Polargraph: | Refers to the specific software and firmware implementation used in the solution: <https://github.com/euphy/polargraph> |
| Arduino | Refers to the Arduino UNO, a small single-board microcontroller that in this implementation runs the Polargraph firmware. |
| Arduino IDE | The Integrated Development Environment used to write code and upload it to the Arduino. Runs on Windows, Mac OS X, and Linux. The Polargraph firmware is written and loaded within the Arduino IDE. |
| Processing | The sketchbook language that Polargraph is written in. |
| Stepper Motor | A motor that divides a full rotation into a number of steps (in this implementation, 200), allowing the position of the motor to be precisely commanded and known. |
| Stepper Motor Controller | The hardware that controls the stepper motor. In this implementation, a “Protoneer CNC Shield V3” which is controlled by the Arduino. |
| Machine Width | The height of the machine from the centre of the motors, to the base. |
| Machine Height | The width of the machine between the inside of the sprockets. |
| Page Width | The width of the usable drawing space. Allow buffer. |
| Page Height | The height of the usable drawing space. Allow buffer. |
| Home | An initial known point on the board that the Polargraph software bases its movement around. The writing head must be moved to this point. |

# Appendices

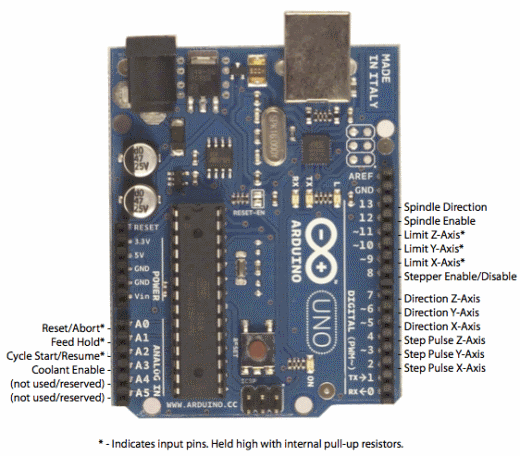
## Appendix 1: Additional Resources:

Our GitHub Repository: <https://github.com/zanedrys/IPAD>

The official Polargraph GitHub Repository: <https://github.com/euphy/polargraph>

## Appendix 2: Protoneer Stepper Motor

Board Layout imposed upon an Arduino UNO. Useful in the modification of the Arduino pins in the firmware:



## Appendix 3: Motor Wiring Diagram

The motor wiring used with the Stepper Motor Controller.

The servo is driven off the Coolant Enable Pin, which the Arduino drives through Pin A3.

