CNC Fly Food Dispenser

Matt Wayland

2017-07-01

Contents

Chapter 1

About

1.1 Overview

Figure 1.1

1.2 Github

1.3 Contact

Matt Wayland

1.4 Colophon

This book was produced using the **bookdown** package (Xie, 2017), which was built on top of R Markdown and **knitr** (Xie, 2015).

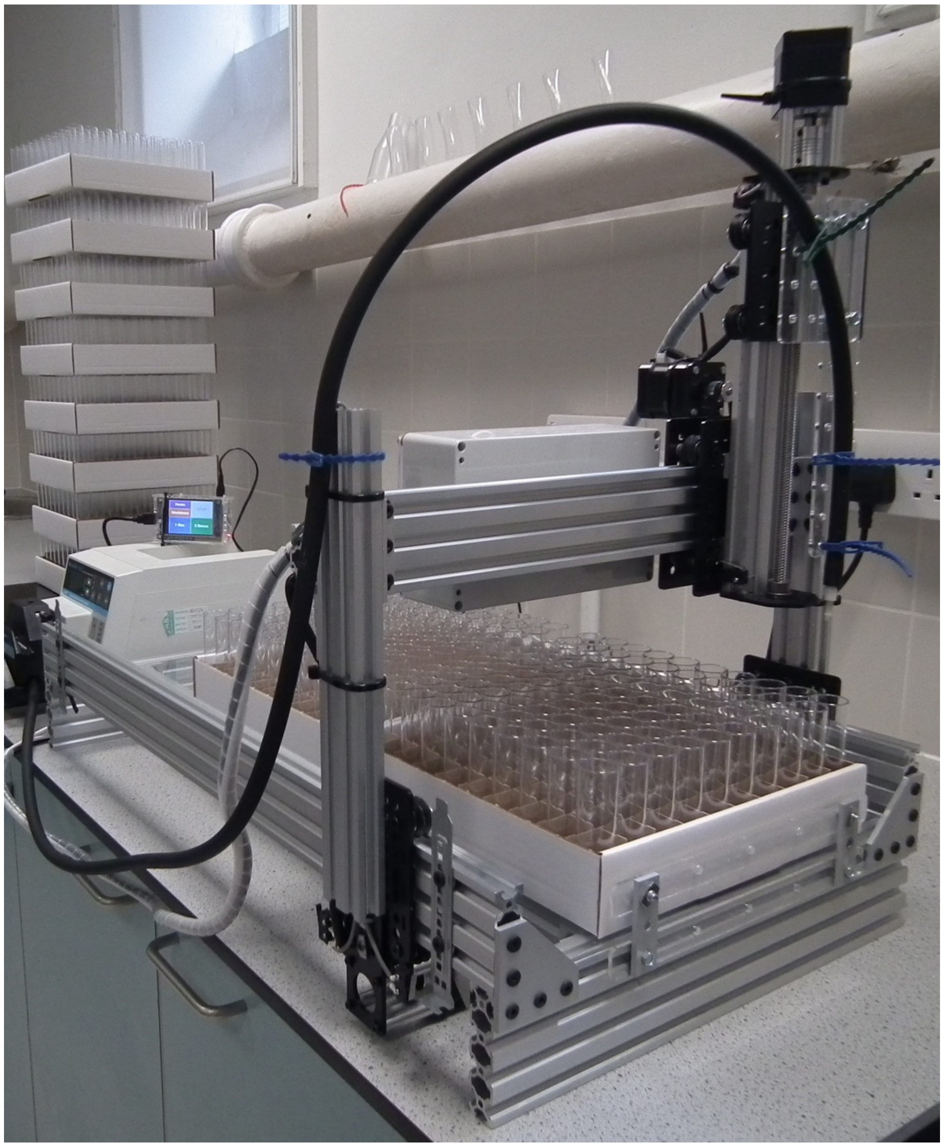


Figure 1.1: Robot

Chapter 2

Introduction

The fruit fly, *Drosophila melanogaster*, is one of the most important model organisms in biological research. Maintaining stocks of fruit flies in the laboratory is labour-intensive. One task which lends itself to automation is the production of the vials of food in which the flies are reared. Fly facilities typically have to generate several thousand vials of fly food each week to sustain their fly stocks. The system presented here combines a cartesian coordinate robot with a peristaltic pump. The design of the robot is based on the Routy CNC Router created by Mark Carew (http://openbuilds.org/builds/routy-cnc-router-v-slot-belt-pinion.101/), and uses belt and pully actuators for the X and Y axes, and a leadscrew actuator for the Z axis. CNC motion and operation of the peristaltic pump are controlled by grbl (https://github.com/gnea/grbl), an open source, embedded, high performance g-code parser. Grbl is written in optimized C and runs directly on an Arduino. A Raspberry Pi is used to generate and stream G-code instructions to Grbl. A touch screen on the Raspberry Pi provides a graphical user interface to the system. This manual explains how to install the required software and operate the robot. Instructions for building the hardware are available on DocuBricks.

A Raspberry Pi is used to generate and stream G-code to the Arduino. A touch screen on the Raspberry Pi provides the user interface; a resistive rather than capacitive touch screen was chosen so that it could be operated by a person wearing gloves.

knitr::include_graphics("images/system.jpg")

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 2. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter **??**.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 2.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 2.1.

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

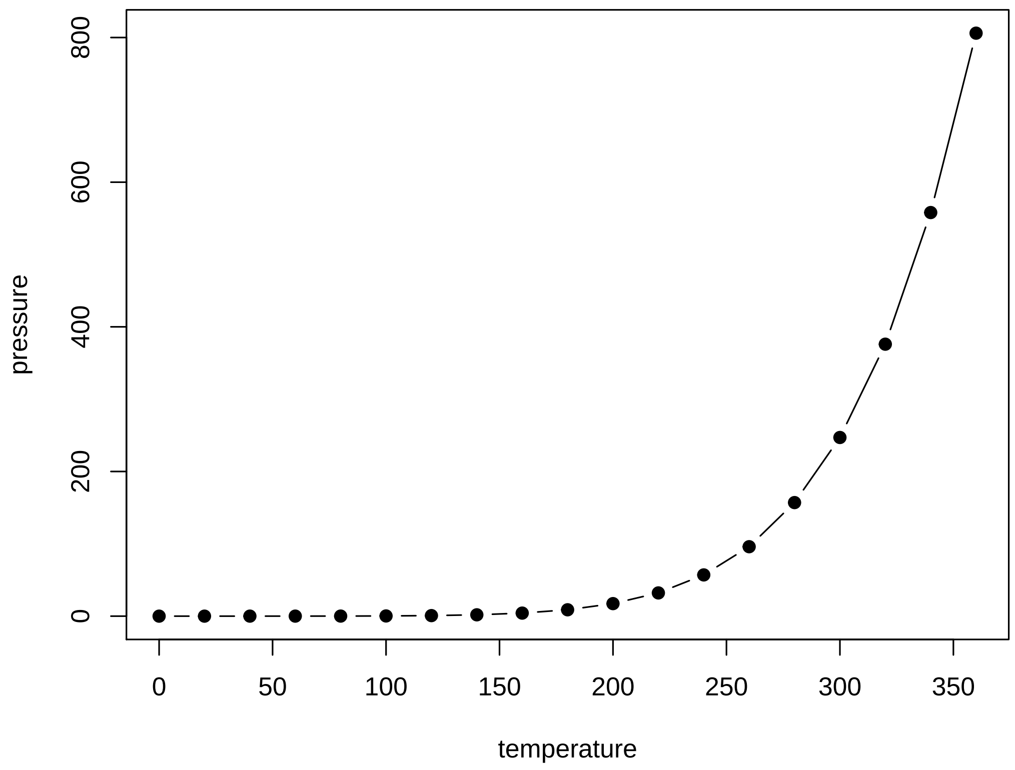


Figure 2.1: Here is a nice figure!

Table 2.1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

Chapter 3

Grbl installation and configuration

3.1 Overview

CNC motion control is provided by grbl (https://github.com/gnea/grbl), an open source, embedded, high performance g-code parser. Grbl is written in optimized C and runs directly on an Arduino. This is used in conjunction with the gShield (formerly known as grblshield) which provides the hardware drivers for the stepper motors. Grbl sends out TTL signals on pins A3 and 13 or the Arduino to control coolant flow and spindle direction, respectively. Here these signals are used to remotely control a peristaltic pump.

3.2 Flashing Grbl to Arduino

To flash Grbl to the Arduino you will need a computer with the latest version of the Arduino IDE installed. The following instructions for flashing Grbl to the Arduino are taken from: https://github.com/gnea/grbl/wiki/Compiling-Grbl

NOTE: Before starting, delete prior Grbl library installations from the Arduino IDE. Otherwise, you'll have compiling issues! On a Mac, Arduino libraries are located in ~/Documents/Arduino/libraries/. On Windows, it's in `My Documents\Arduino\libraries`.

1. Download the Grbl source code.

- Open the following page in your web browser: https://github.com/gnea/grbl
- Click on the `<>Code` Tab
- Click the `Clone or Download` green button on the Grbl home page.
- Click the `Download ZIP`
- Unzip the download and you'll have a folder called `grbl-XXX`, where `XXX` is the release version.

2. Launch the Arduino IDE

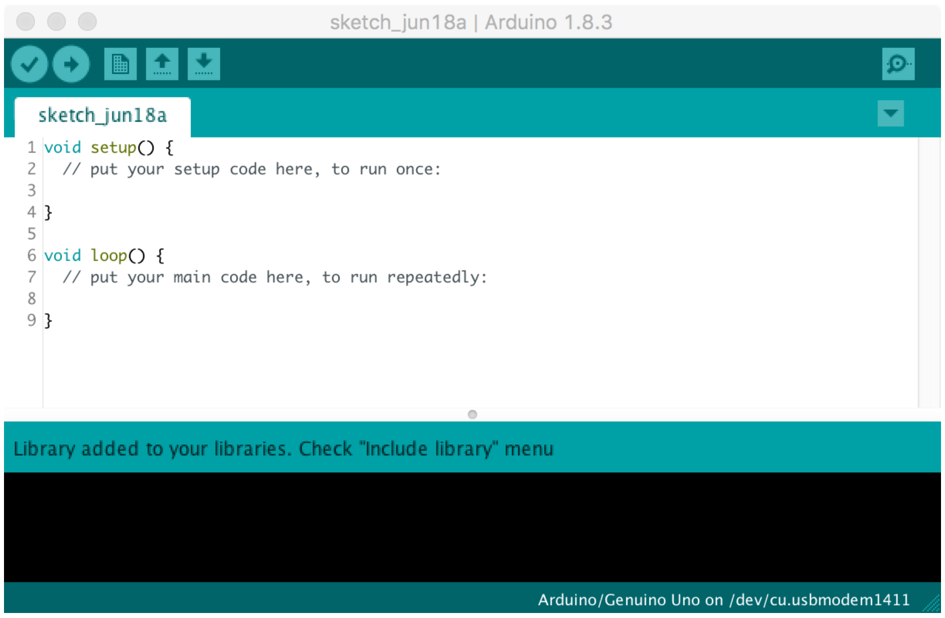
- Make sure you are using the most recent version of the Arduino IDE!

3. Load Grbl into the Arduino IDE as a Library.

- Click the `Sketch` drop-down menu, navigate to `Include Library` and select `Add .ZIP Library`.
- **IMPORTANT:** Select the `Grbl` folder ***inside*** the `grbl-XXX` folder, which **only** contains the source files and an example directory.

- If you accidentally select the `.zip` file or the wrong folder, you will need to navigate to your Arduino library, delete the mistake, and re-do Step 3.

4. Open the `GrblUpload` Arduino example.

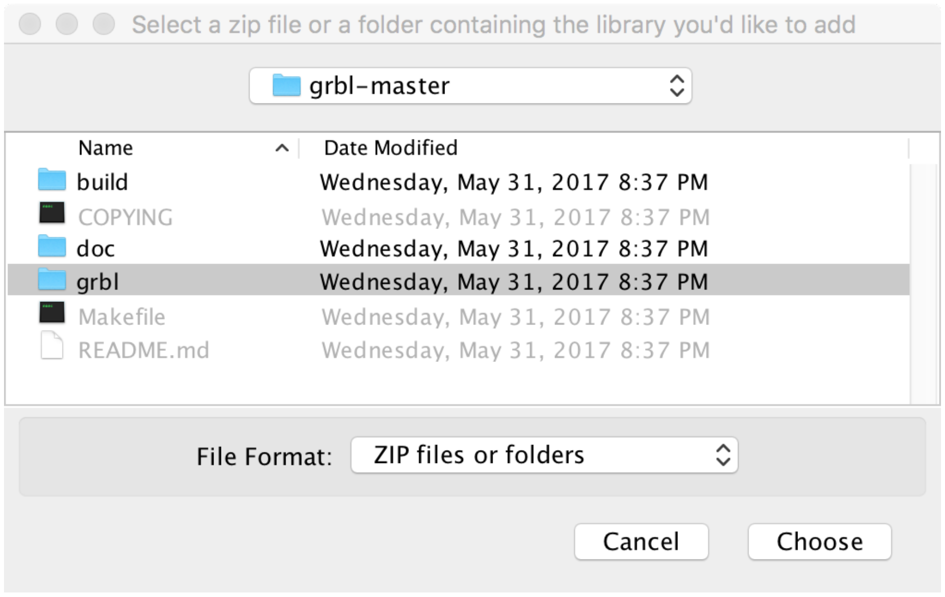


```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Library added to your libraries. Check "Include library" menu

Arduino/Genuino Uno on /dev/cu.usbmodem1411

Figure 3.1: Arduino IDE


Select a zip file or a folder containing the library you'd like to add
grbl-master

Name	Date Modified
build	Wednesday, May 31, 2017 8:37 PM
COPYING	Wednesday, May 31, 2017 8:37 PM
doc	Wednesday, May 31, 2017 8:37 PM
grbl	Wednesday, May 31, 2017 8:37 PM
Makefile	Wednesday, May 31, 2017 8:37 PM
README.md	Wednesday, May 31, 2017 8:37 PM

File Format: ZIP files or folders

Figure 3.2: Loading Grbl library into the Arduino IDE

- Click the `File` drop-down menu, navigate to `Examples->Grbl`, and select `GrblUpload`.

5. Compile and upload Grbl to your Arduino.

- Connect your computer directly to the Arduino using the USB cable.
- Make sure your board is set to the Arduino Uno in the `Tool->Board` menu and the serial port is selected correctly in `Tool->Serial Port`.
- Click the `Upload`, and Grbl should compile and flash to your Arduino! (Flashing with a programmer also works by using the `Upload Using Programmer` menu command.)

3.3 Check serial connection to Grbl

NOTE: Before powering up the gShield and motors, check that the actuator carriages for all three axes are approximately centred. Initially we do not know in which direction the actuator carriages will travel when G-code commands are issued, so positioning each in the middle of its range reduces the risk of collisions with the end stops.

1. Open serial monitor in Arduino IDE

- Click `Tools` drop-down menu, and select `Serial Monitor`
- Note that line-ending is set to `Carriage return` and baud rate is set to `115200`

2. Try issuing a G-code command.

- Type `?` and hit return. *This command will report the current position; as we have just started the system up all axes will be at 0.000.

3. Now try moving actuators

- To move in the x-axis type `x5` and hit return. Make a note of the direction in which the actuator carriage moves. N.B. this command tells Grbl to move to the x coordinate that is 5 units from the origin, it is not equivalent to telling the robot to move 5 units in the x-axis.
- To move in the opposite direction along the x-axis type `x-5` and hit return.
- To return to the starting point, use `x0`
- Repeat for the other axes, replacing the x in the commands with y or z. Make a note of the direction the actuator carriages move with each command.

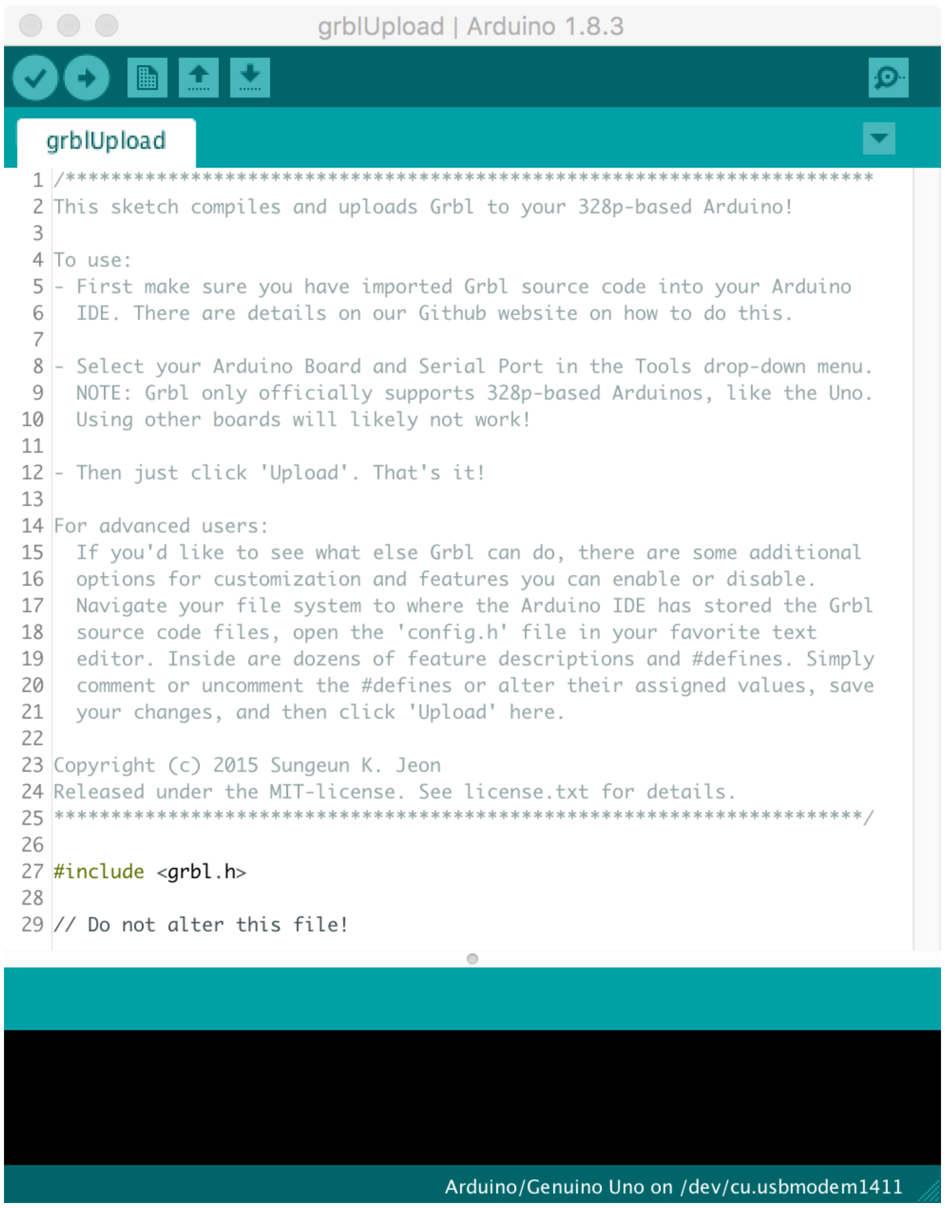
3.4 Grbl configuration

3.4.1 Read current configuration

- The `$$` command will report Grbl's current configuration.
- Descriptions of these settings can be found here: https://github.com/gnea/grbl/wiki/Grbl-v1.1-Configuration
- These settings will be modified in subsequent steps.

3.4.2 Check directionality of each axis.

- At present the origins of all three axes are mid-way along each actuator, because this was the position of the actuator carriages when the system was started.
- Make sure actuator carriages are at their current origin by entering this command: `x0y0z0`
- Enter the command: `x5`. The x-axis carriage should move from right to left (orientation of robot is shown in figure 3.7. If it doesn't, make a note that it will need to be inverted.



```
/***********************************************************************
This sketch compiles and uploads Grbl to your 328p-based Arduino!

To use:
- First make sure you have imported Grbl source code into your Arduino
  IDE. There are details on our Github website on how to do this.

- Select your Arduino Board and Serial Port in the Tools drop-down menu.
  NOTE: Grbl only officially supports 328p-based Arduinos, like the Uno.
  Using other boards will likely not work!

- Then just click 'Upload'. That's it!

For advanced users:
  If you'd like to see what else Grbl can do, there are some additional
  options for customization and features you can enable or disable.
  Navigate your file system to where the Arduino IDE has stored the Grbl
  source code files, open the 'config.h' file in your favorite text
  editor. Inside are dozens of feature descriptions and #defines. Simply
  comment or uncomment the #defines or alter their assigned values, save
  your changes, and then click 'Upload' here.

Copyright (c) 2015 Sungeun K. Jeon
Released under the MIT-license. See license.txt for details.
***********************************************************************/

#include <grbl.h>

// Do not alter this file!
```

Arduino/Genuino Uno on /dev/cu.usbmodem1411

Figure 3.3: GrblUpload example file

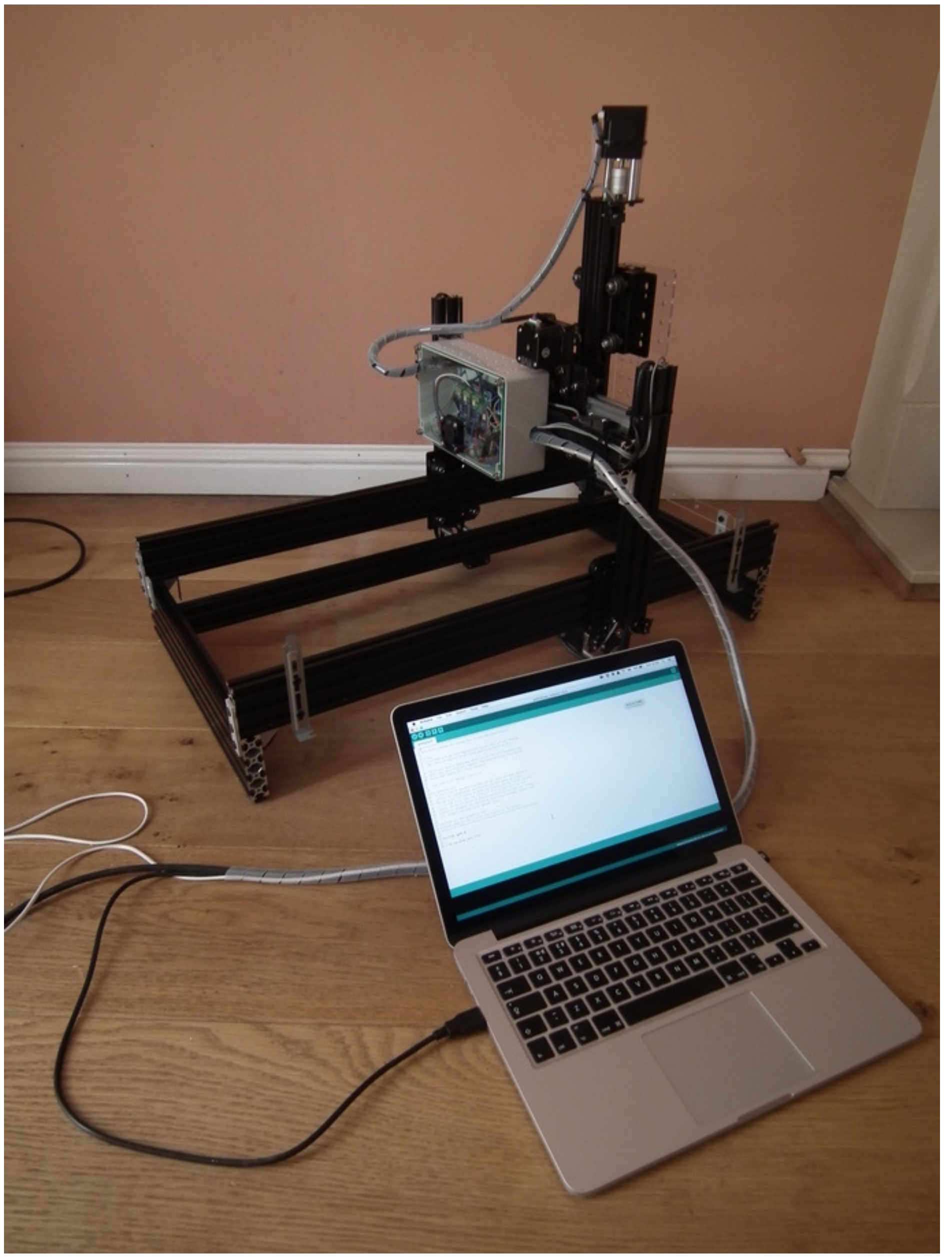


Figure 3.4: Laptop connected directly to Arduino

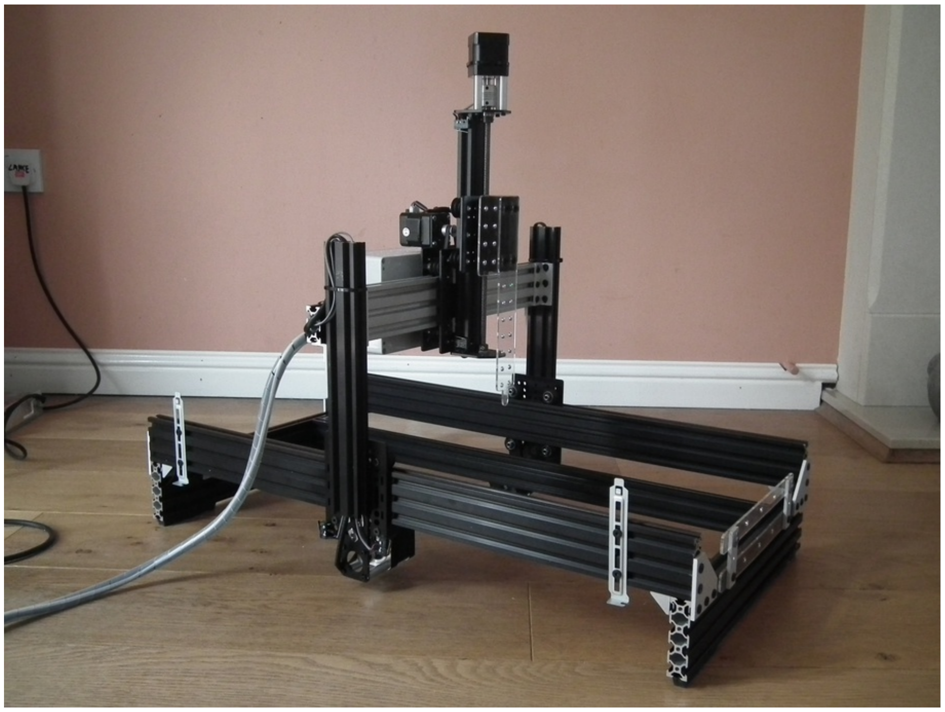
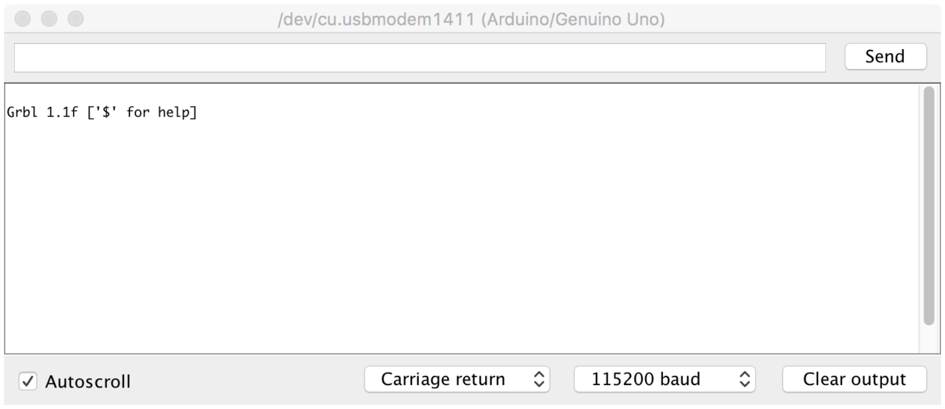


Figure 3.5: Actuator carriages centred in preparation for powering-up motors for first time



```
Grbl 1.1f ['$' for help]
```

✓ Autoscroll, Carriage return, 115200 baud

Figure 3.6: Arduino IDE Serial Monitor

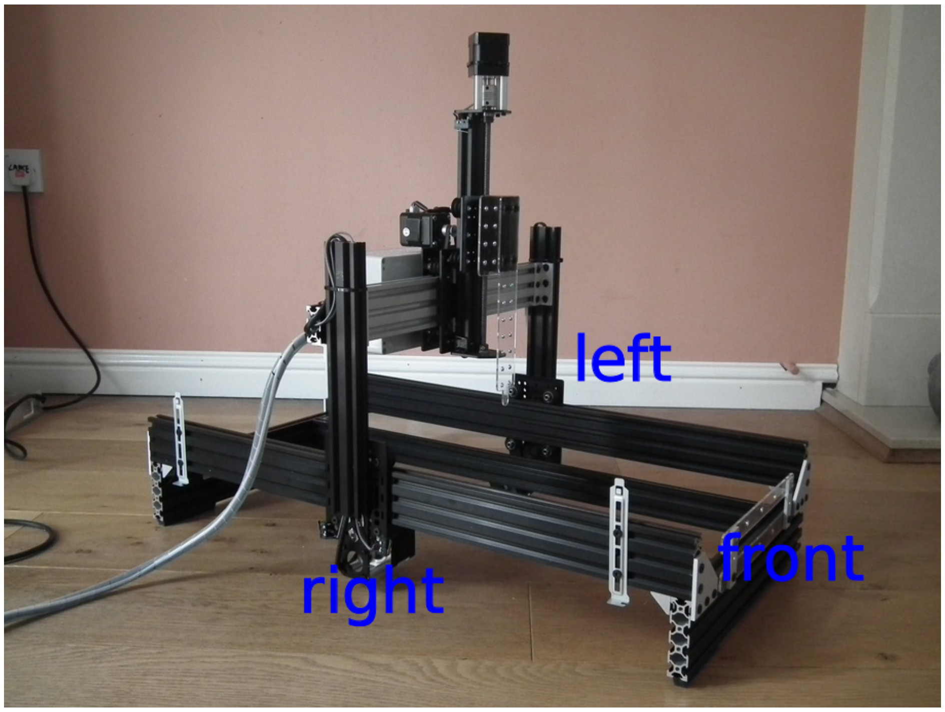


Figure 3.7: Orientation of robot.

- Enter the command: `y5`. The y-axis carriage should move forwards. If it doesn't, make a note that it will need to be inverted.
- Enter the command: `z5`. The z-axis carriage should move up. If it doesn't, make a note that it will need to be inverted.
- The direction of the actuators can be inverted using setting **$3**, the Direction port invert (mask). An appropriate value is selected from table 3.1. For example, to invert the direction of the X and Z axis actuators use the following command: `$3=5`

Table 3.1: Masks for direction port inversion.

Setting Value	Mask	Invert X	Invert Y	Invert Z
0	00000000	N	N	N
1	00000001	Y	N	N
2	00000010	N	Y	N
3	00000011	Y	Y	N
4	00000100	N	N	Y
5	00000101	Y	N	Y
6	00000110	N	Y	Y
7	00000111	Y	Y	Y

3.4.3 Activate hard limits

Hard limits are a safety feature to prevent the machine from travelling beyond the limits of travel. Grbl monitors the paired limit switches on each axis and if a switch is triggered it will immediately switch off all motors. Hard limits are activated by setting **$21** hard limits (boolean) to 1:

```
$21=1
```

3.4.4 Setup homing

The homing cycle is used to set the origin of the cartesian coordinate system used by the robot. During the homing cycle Grbl moves each actuator in the positive direction until the limit switches are triggered. The homing cycle is activated by setting **$22** homing cycle (boolean) to 1:

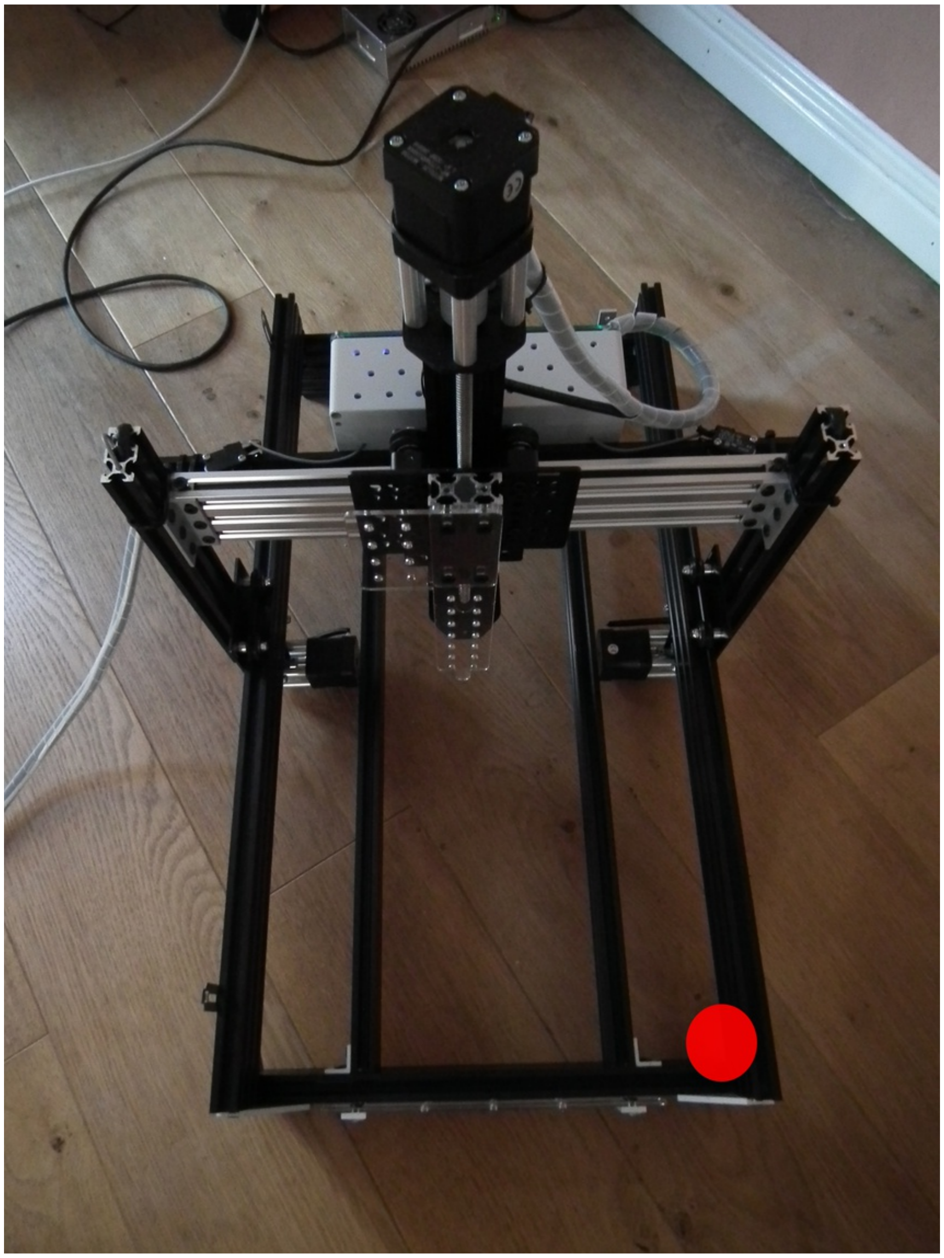


Figure 3.8: Origin of XY coordinate system.

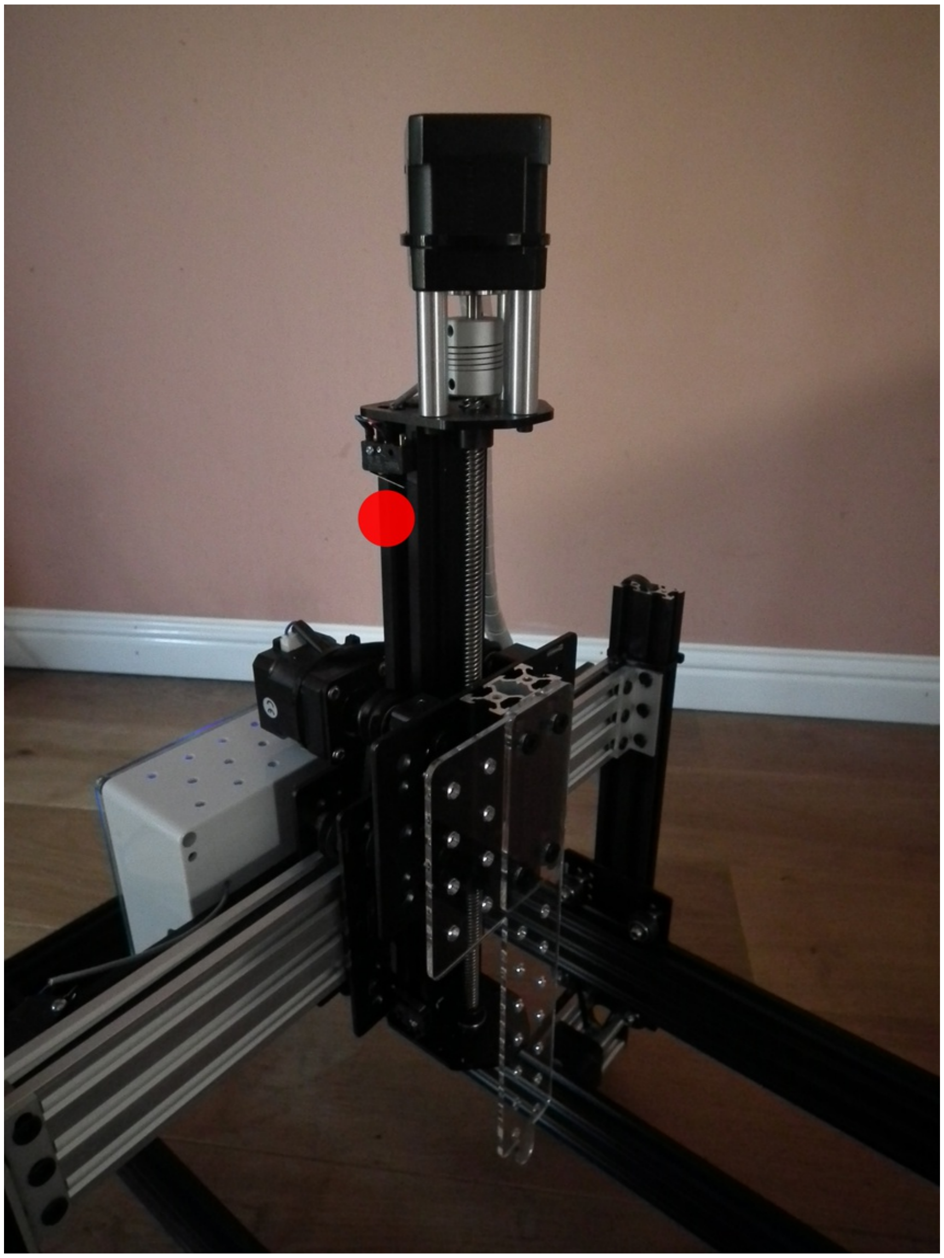


Figure 3.9: Origin of Z axis.

```
$100=40
$101=40
$102=49.673
```

curr__steps__per__mm = steps/mm in current configuration start__pos__grbl = starting position reported by software end__pos__grbl = end position reported by software start__pos__physical = actual/physical start position end__pos__physical = actual/physical end position

steps/mm = (curr__steps__per__mm * (end__pos__grbl-start__pos__grbl)) / (end__pos__physical - start__pos__physical)

3.4.6 Feed rates and acceleration

$110, **$111** and **$112** set the maximum rates (mm/min) for the X, Y and Z actuators, respectively. We will use the following values:

```
$110=5000
$111=5000
$112=2500
```

Acceleration (mm/sec^2) is set to 50 for all axes:

```
$120=50
$121=50
$122=50
```

3.4.7 Summary of settings

```
$0=10
$1=25
$2=0
$3=5
$4=0
$5=0
$6=0
$10=1
$11=0.010
$12=0.002
$13=0
$20=0
$21=1
$22=1
$23=0
$24=25.000
$25=500.000
$26=250
$27=5.000
$30=1000
$31=0
$32=0
$100=40.000
$101=40.000
$102=49.673
$110=5000.000
$111=5000.000
$112=2500.000
```

```
$120=50.000
$121=50.000
$122=50.000
$130=200.000
$131=200.000
$132=200.000
```

3.5 fine adjustment of motors

Chapter 4

Raspberry Pi setup

https://learn.adafruit.com/adafruit-pitft-28-inch-resistive-touchscreen-display-raspberry-pi/easy-install

https://s3.amazonaws.com/adafruit-raspberry-pi/2016-10-18-pitft-28r.zip

https://www.raspberrypi.org/documentation/installation/installing-images/

```
sudo raspi-config
(expand filesystem)
sudo reboot
```

/etc/dhcpcd.conf

interface eth0

static ip_address=192.168.2.2/24 static routers=192.168.2.1 static domain_name_servers=192.168.2.1

Chapter 5

Creating jobs in G-code

Running job without GUI - for testing

Chapter 6

Trouble-shooting

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr.* Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2017). *bookdown: Authoring Books and Technical Documents with R Markdown.* R package version 0.4.