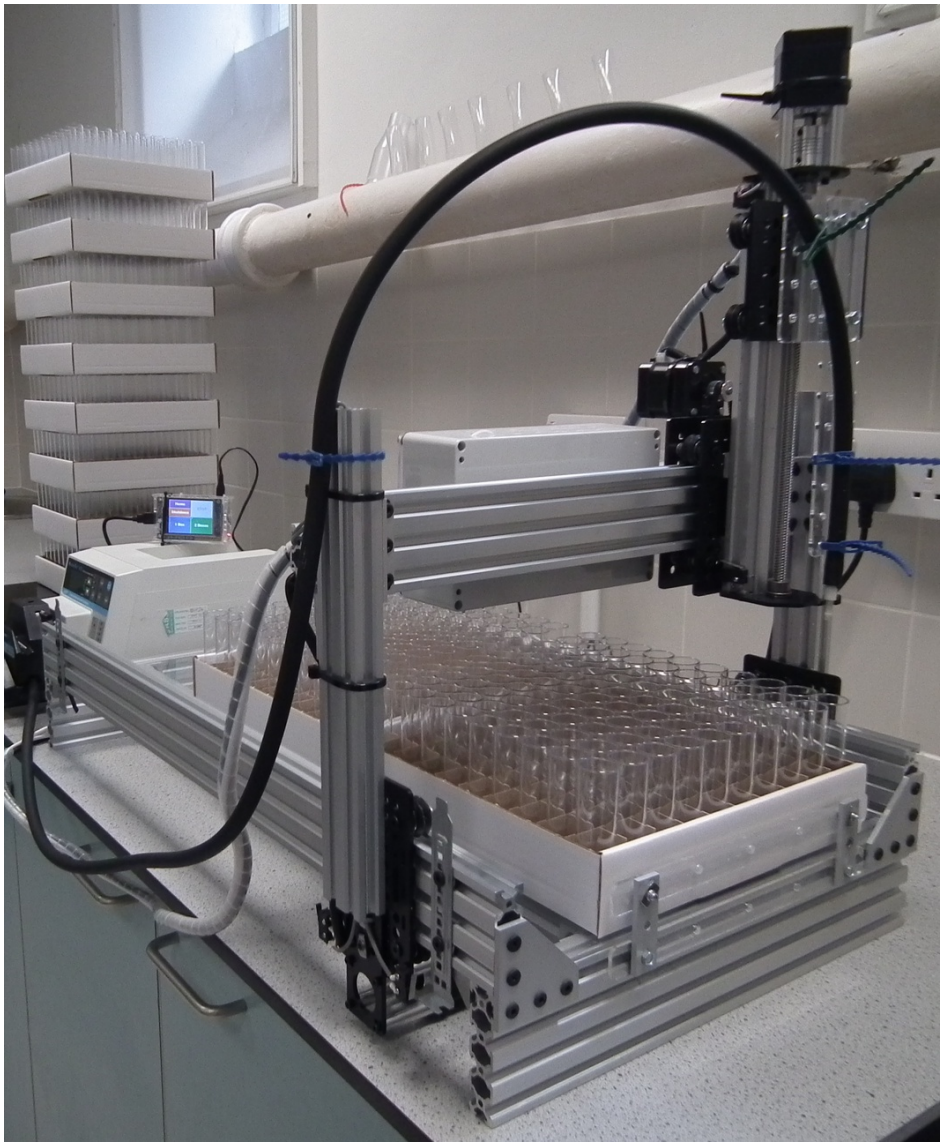
CNC Fly Food Dispenser

Matt Wayland

2017-07-03

Contents

Preface



Github

WaylandM/fly-food-robot

License

Contact

Matt Wayland

Colophon

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

Chapter 1

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.

Chapter 2

Grbl installation and configuration

2.1 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.

2.2 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.

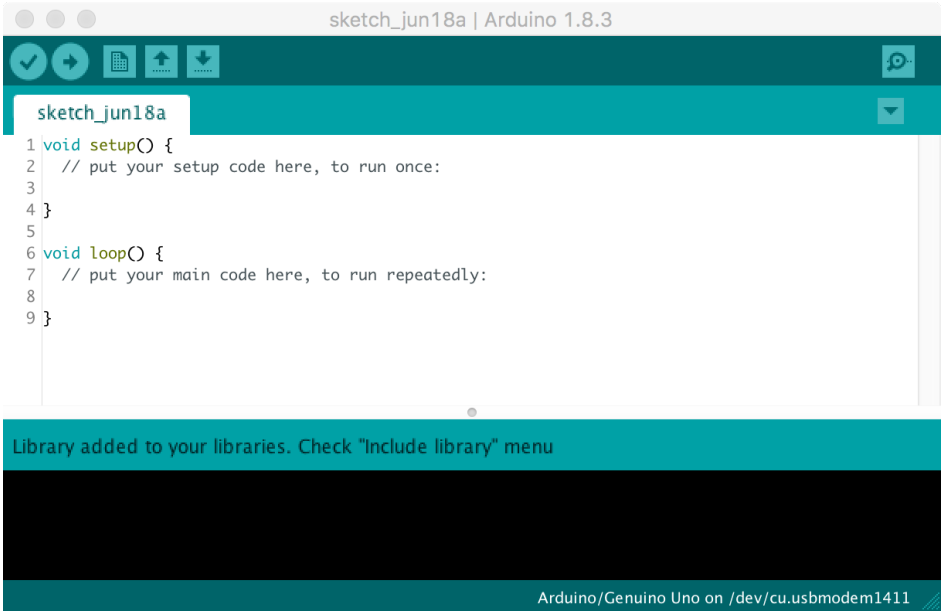


Figure 2.1: Arduino IDE

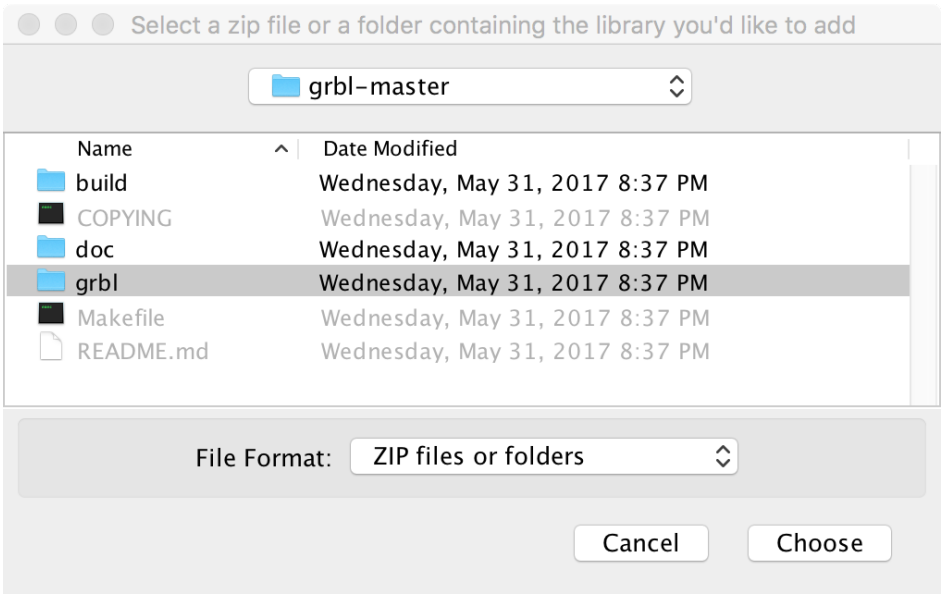


Figure 2.2: Loading Grbl library into the Arduino IDE

- 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.)

2.3 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.

2.4 Grbl configuration

2.4.1 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.

2.4.2 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`

Figure 2.3: GrblUpload example file

Figure 2.4: Laptop connected directly to Arduino

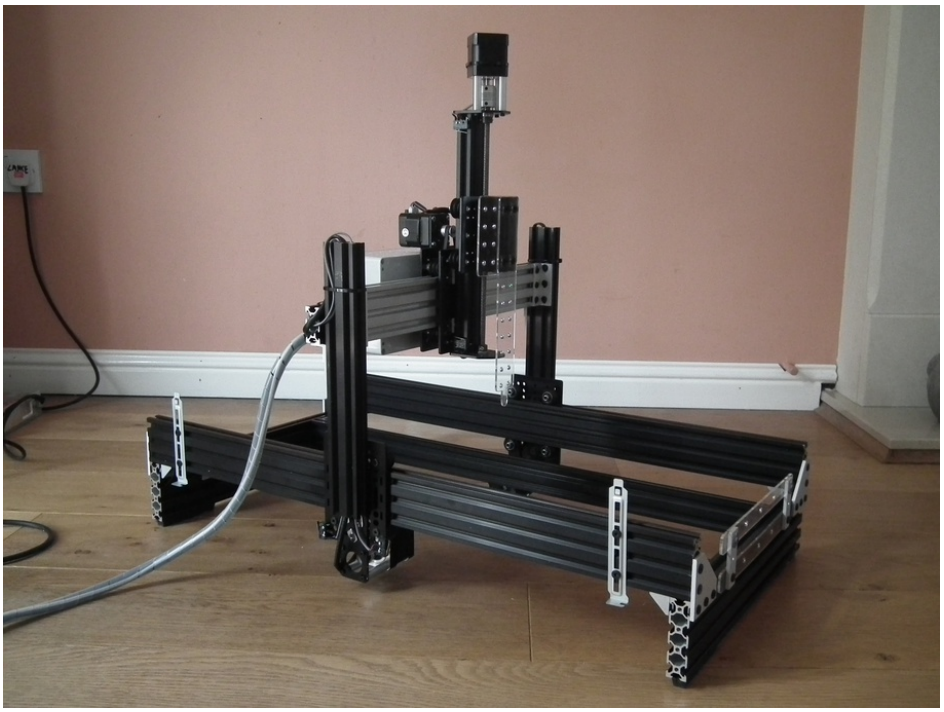


Figure 2.5: Actuator carriages centred in preparation for powering-up motors for first time

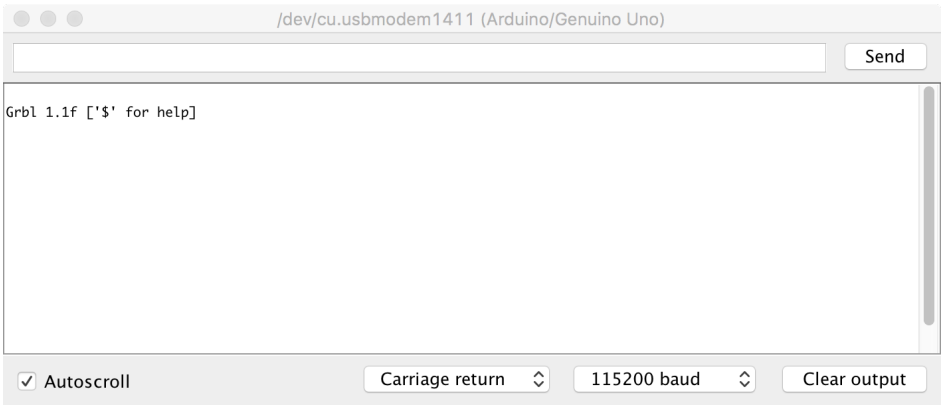


Figure 2.6: Arduino IDE Serial Monitor

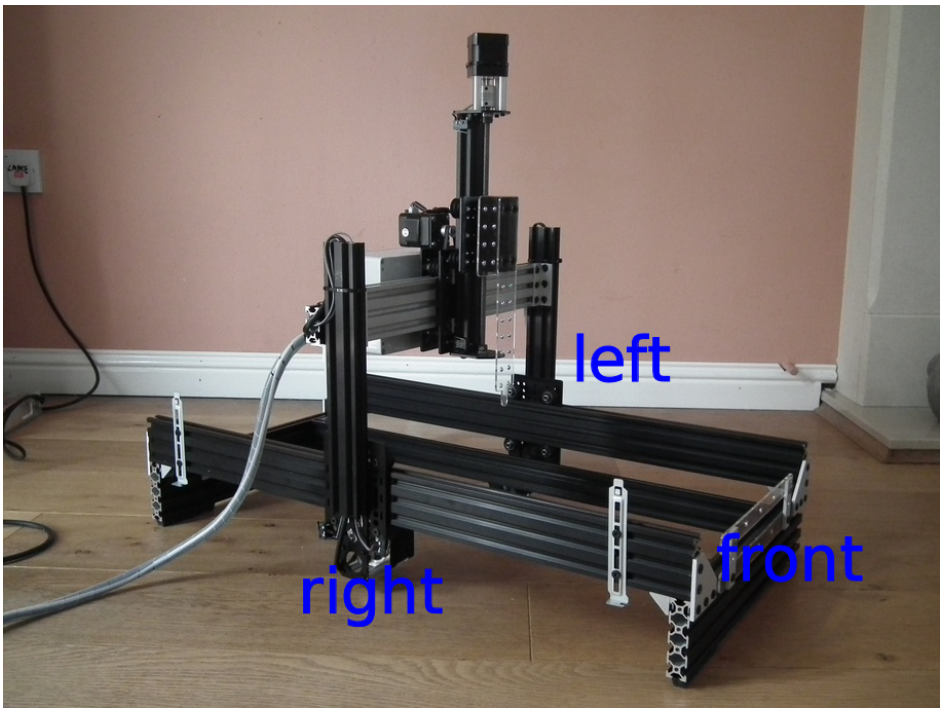


Figure 2.7: Orientation of robot.

- Enter the command: `x5`. The x-axis carriage should move from right to left (orientation of robot is shown in figure 2.7. If it doesn't, make a note that it will need to be inverted.
- 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 2.1. For example, to invert the direction of the X and Z axis actuators use the following command: `$3=5`

Table 2.1: 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

2.4.3 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
```

2.4.4 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:

```
$22=1
```

Initiate a homing cycle using the following command: `$h`. All actuator carriages should move to the origin of their axes. The origin of the cartesian coordinate system (home) for the robot is shown in figures 2.8 and 2.9

We also need to set **$24** homing feed rate and **$25** homing seek rate. Homing seek rate is the initial speed at which Grbl searches for the limit switches. Once it has them, it makes slower approach at the homing feed rate to get a more precise location for machine zero. We will set homing seek rate to 1000 mm/min `$24=100` and homing feed rate to 100 mm/min `$25=1000`.

At the end of a homing cycle each actuator carriage must be moved off its home limit switch, otherwise the hard limit will be triggered. The **$27** Homing pull-off (mm) specifies the distance required to clear the limit switches. For our robot we will use a value of 5mm:

```
$27=5
```

2.4.5 Motor step size

$100, **$101** and **$102** define [X,Y,Z] steps/mm. Suitable values for our stepper motors are:

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

If you are using a different type of stepper motor, the step size can be easily calculated by measuring how far each actuator moves in response to a G-code command. For example, we would calculate the step size for the X actuator as follows.

1. Using the serial monitor in the Arduino IDE, issue the following command to *home* the machine:

   ```
   $h
   ```

2. Read the current position of the machine, as reported by the Grbl controller:

   ```
   ?
   ```

   After homing the cartesian coordinates should be zero minus the homing pull-off:

- x = -5
- y = -5
- z = -5

3. Make a note of the physical position of the nozzle.

4. Issue a g-code command to move to the -100mm position on the x-axis:

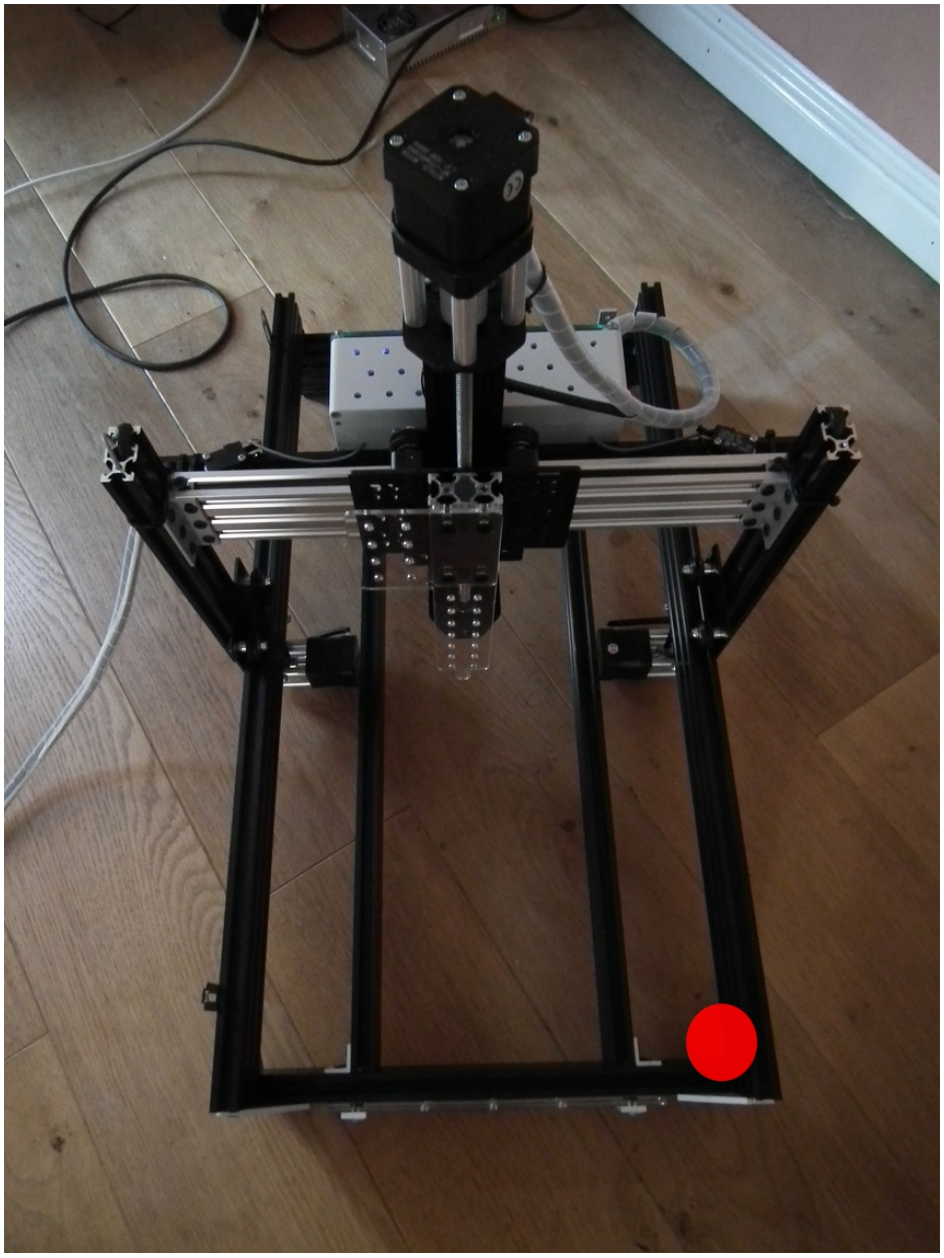


Figure 2.8: Origin of XY coordinate system.

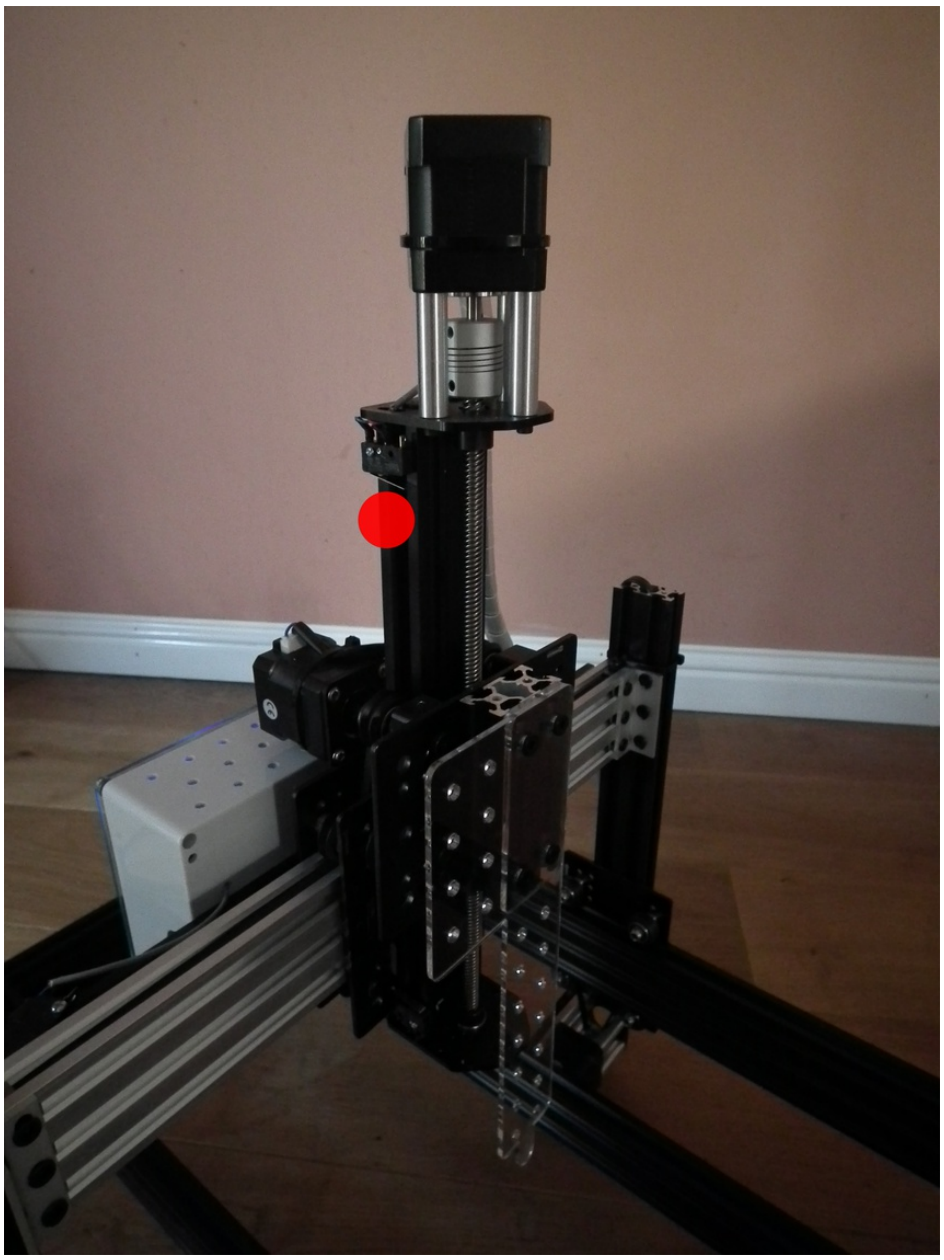


Figure 2.9: Origin of Z axis.

```
x-100
```

Keep decreasing the value of x until the x-actuator carriage almost meets the limit switch on the right hand side of the machine. The greater the distance moved, the more precise our calculation of step-size will be.

5. Query Grbl's machine coordinates:

   ```
   ?
   ```

6. Measure the physical distance travelled along the x-axis in millimetres.

7. Find **$100**, the currently configured step size for the x-axis:

   ```
   $$
   ```

8. Calculate the correct value for step-size:

```
current_step_size = steps/mm in current configuration
grbl_start = start position reported by Grbl controller (mm)
grbl_end = end position reported by Grbl controller (mm)
physical_distance = physical distance moved by actuator (mm)

steps/mm = -(curr_steps_per_mm * (end_pos_grbl-start_pos_grbl)) / physical_distance
```

2.4.6 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
```

2.4.7 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=100.000
```

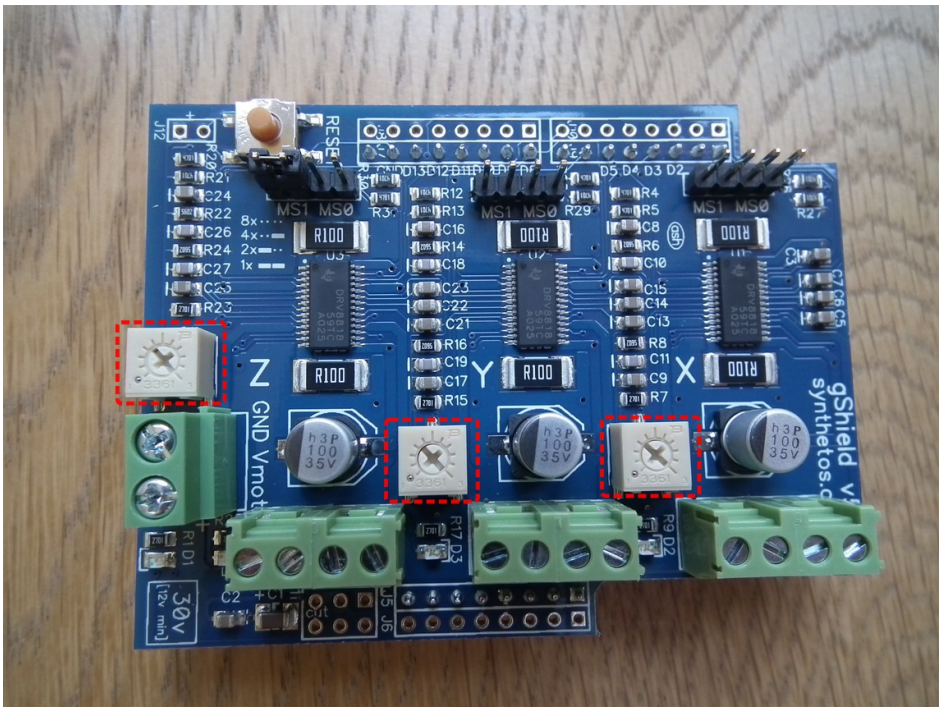


Figure 2.10: gShield trimpots

```
$25=1000.000
$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=500.000
$132=200.000
```

2.5 Setting motor current

The gShield has trimpots for adjusting the motor current of each axis, as shown in Figure 2.10.

Instructions for setting motor current are provided here: https://github.com/synthetos/grblShield/wiki/Using-grblShield#setting-motor-current

Chapter 3

Raspberry Pi setup

3.1 Install image

The first step is to install Adafruit's custom raspberry pi image on the micro SD card. The custom image is described here:

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

We want the classic version which boots into X by default, rather then the lite version that boots to the command line. The classic version can be downloaded from this link:

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

Instructions on installing images on SD cards can be found here:

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

3.2 Network configuration

The small screen of the pitft makes using most applications quite tricky. Therefore the first thing we should do after installing the image is configure networking, so that we can access the raspberry pi remotely using ssh.

To set a static IP address for the ethernet adapter, add the following lines to `/etc/dhcpcd.conf`:

```
interface eth0

static ip_address=192.168.1.3/24
static routers=192.168.1.254
static domain_name_servers=192.168.1.254
```

ip_address, **routers** and **domain_name_servers** should be set to values appropriate for your network.

To raspberry pi can then be accessed using ssh, *e.g.*:

```
ssh pi@192.168.1.3
```

The default password for the **pi** user account is **raspberry**

3.3 Serial connection

Connect the raspberry pi to the arduino using the usb cable.

3.4 Install minicom

Minicom is useful for manual control of the robot and for editing grbl settings. To install minicom run these two commands:

```
sudo apt-get update
sudo apt-get install minicom
```

Before we can use minicom we need to enable serial:

```
sudo nano /boot/config.txt
```

Change the last line of this file from

```
enable_uart=0
```

to

```
enable_uart=1
```

Once serial is enabled we can connect to the grbl controller running on the arduino by using:

```
sudo minicom -D /dev/ttyACM0 -b115200
```

3.5 Expand filesystem

Expand filesystem on micro SD card:

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

3.6 Install robot software

Make sure you are in pi's home directory:

```
cd
```

Download and unpack robot.tar.gz

```
curl -O https://raw.githubusercontent.com/WaylandM/fly-food-robot/master/raspberrypi/robot.tar.gz
tar xzvf robot.tar.gz
```

The **robot** directory contains two subdirectories: **nc** (g-code scripts) and **py** (python scripts).

To automatically launch the robot GUI when the raspberry pi starts up, we need to edit the autostart file for the pi user:

```
sudo nano /home/pi/.config/lxsession/LXDE-pi/autostart
```

Add the following line to autostart:

```
@/home/pi/robot/py/fly_gui.py
```

Chapter 4

Creating jobs in G-code

Running job without GUI - for testing

Chapter 5

Operation

5.1 Loading boxes of vials

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.