# Aim

This document details how to configure the EleksLaser A3-pro in order to control it via python.

Sources of information include the following:

1. <https://cadduino.wordpress.com/2013/11/18/testing-grbl-in-arduino-board-without-the-motors/>
2. <https://github.com/jandelgado/eleksmaker_a3>
3. <https://github.com/grbl/grbl/wiki/Configuring-Grbl-v0.9>
4. <https://wiki.shapeoko.com/index.php/G-Code>

# Glossary of terms

|  |  |
| --- | --- |
| Term | Description |
| ELEKSROM | Software made by Eleksmaker to upload firmware to their board |
| ELEKSCAM | Software made by Eleksmaker to control their board |
| MCU | Microcontroller unit |
| EleksLaser A3-pro | Name of the 2D robot apparatus |
| G-code | Programming language used to control the board |
| Grbl | G-code parser that runs on an Arduino |

# Introduction

The Elekslaser A3-pro uses an Arduino Nano board as the MCU (microcontroller unit) with an ATMega328p processor.

There are multiple ways to control the EleksLaser board, a GUI based approach and a command line approach are listed below.

NOTE: Ensure that the board is plugged in and the button is pressed down.

# Manufacturer software

The manufacturer software consists of loading the ELEKSROM firmware via the ELEKSROM application and then using the ELEKSCAM software to control the device.

In the GUI for the ELEKSCAM we must click the “install driver” button.

Once the driver is installed the motors can be driven by the software.

# Grbl and Arduino

This approach allows us to control the board by sending commands down the COM serial port. To set this up we perform the following:

1. Download Arduino IDE
2. Go to <https://github.com/gnea/grbl/releases> and download the latest release source code.
3. Unzip the grbl folder and place it in the libraries folder of the installed Arduino software.
4. Rename the folder to GRBL.
5. Open up Arduino IDE and select File 🡪 Examples 🡪 Grbl, select GrblToArduino
6. Click Tools 🡪 Board, select Arduino Nano
7. Click Tools 🡪 Processor, select the old ATMega328p bootloader.
8. Close the Arduino and then install the GrblController 3.5.1 software
9. Click connect and then press some arrows. NOTE: Ensure that the button on the Board is pushed down!

## Limit switches

The setup as provided by Eleksmaker does not include limit switches, as a result the device cannot calibrate exactly where it is. A limit switch needs to be wired up manually.

## Grbl

Almost every interaction with Grbl is performed by sending a string of characters followed by enter. Grbl will then process the string and execute it accordingly, it will then reply back with a response to tell us how it went.

Every G-code block sent to Grbl and Grbl system commands will respond with how it went. There are a bunch of responses.

## G-code

G code consists of commands which start with mostly either G or M followed by parameters.

G code is not compiled, the Arduino runs an interpreter which receives it line by line and executes it.

The table below contains some basic G-code commands that are commonly used.

### System commands

System commands are different from regular commands, they start with a $ sign.

|  |  |
| --- | --- |
| Command | Description |
| $ | Help command |
| $$ | View grbl parameters |
| $H | Runs the homing cycle (brings the device back to a predefined place) |
| $X | Kill alarm lock (sometimes the device can get into alarm lock state) |
| ? | Current status |

All grbl parameters (viewable by $$ command) are persistent between shutdowns.

### Other commands

|  |  |
| --- | --- |
| Command | Description |
| G0 | Rapid movement |
| G1 | Linear movement to specified coordinates |
| G2 | Clockwise arc movement to specified coordinates |
| G3 | Counterclockwise arc movement to specified coordinates |
| G4 | Dwell, pauses the command queue and waits for a period of time. |
| G10 | Set work coordinate origin. This setting is persistent and expects the user to follow good practices and not manually move the machine. Machine should only be moved by jogging it. |
| G54-G59 | Work coordinate systems |
| G90 | Switch to absolute distance moved – Coordinates are now relative to the origin of the currently active coordinate system (as opposed to the current position) G0 X-10 Y5 will move the position 10 units to the left and 5 above the origin X0,Y0 |
| G91 | Switches to incremental mode – Coordinates are now relative to the current position, with no consideration for machine origin. G0 X-10 Y5 will move to the position 10 units to the left and 5 above the current position. |

### G28 and G30

G28 and G30 are not WCS settings. They are persistent, stored positions that we can send the machine to with a single command.

G28.1 and G30.1 are used to store the current machine position in absolute machine coordinates.

We can set the G28 position by moving the machine to the position we wish to set and issuing G28.1, the same for G30. These commands do not take any value.

NOTE: We can specify an additional, intermediate position by adding X, Y or Z values to the command. These coordinates are in the current WCS, not absolute machine values. E.g. We could say G28 Z10, this would move to Z to 10 in the current WCS and then move X, Y, Z to the saved position.

NEXT STEPS:  
Finish reading <https://wiki.shapeoko.com/index.php/G-Code> about the work coordinate system (Heading “using the work coordinate system)  
Also finish writing for G10 command on the same webpage

### Work coordinate system

Once we run the homing cycle our machine absolute zero will be set.

Once we have jogged the machine to our zero location, we can set a WCS by using the following command:

G10 P[1-9] L20 X [offset] Y [offset] Z [offset]

P[1-9] is used to select the work coordinate system to change. P1=G54, P2=G55 etc.

Grbl only has access to 6 of the 9 work coordinate systems.

L20 means that the offsets can be set in the new coordinate system. E.g:  
G10 L20 P1 X5 means that in the current location, for coordinate system P1 the current x value in this new coordinate system is 5.

#### Workflow

For the first time we:

* Home the machine.
* Jog to where we want our WCS 0,0,0 to be
* Say G10 L20 P1 X0 Y0 Z0
* GRBL calculates the offsets from machine origin and stores them in non-volatile memory (EEPROM)

Every other time we simply home for machine