Oxide Line - Laser Controller

Software Functional Requirements

By **Dr Gary Twinn**

Contents

List of Figures

List of Tables

Overview

The new oxide extraction line is a configurable Nobel gas extraction line used as a test environment (Figure 1).

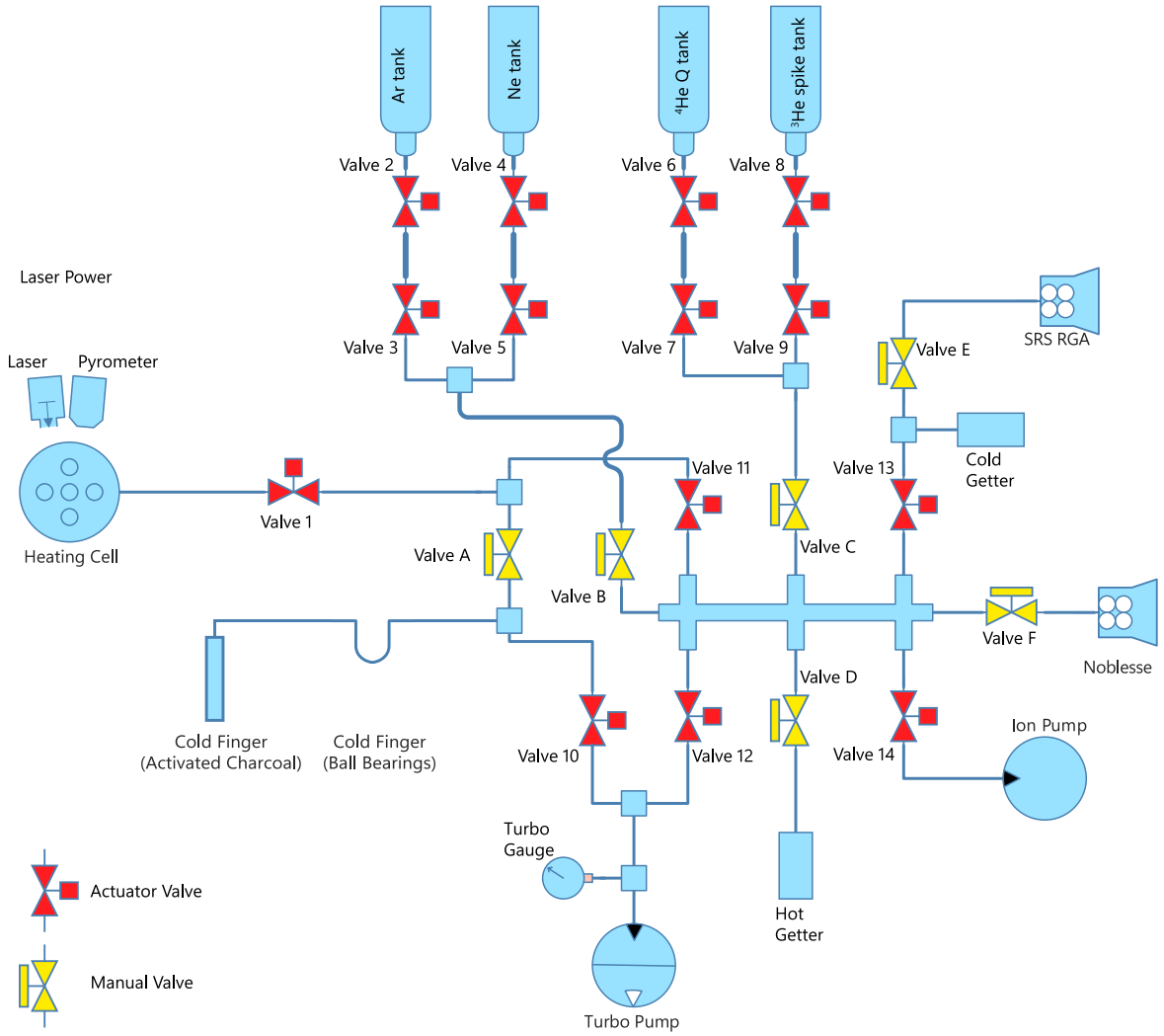

Figure 1: The UCL Oxide Nobel Gass Extraction Line

It consists of a laser/pyrometer system use to heat samples and a mass spectrometer used to measure the resultant gasses. The extraction line includes a 3He spike tank, a 4He standard tank as well as an Ar and Ne tank. There is a cryotrap which consists of two cold fingers that can be added into the line as well.

The laser control software includes a Python application for managing a LaserTree K60 450nm laser.

This application uses the Flask web interface to provide an HTTP status page and handle requests via the API (Application Programming Interface).

The computer controlling the laser is a Raspberry Pi 4B. The laser unit has a Transistor-Transistor-Logic (TTL) input for switching the beam on and off. The beam power can be modulated using pulse width modulation, with a duty cycle ranging from 0 to 100% to manage the output power.

To enable the beam, the laser requires a +5v TTL output, but the Raspberry Pi GPIO uses a 3.3v Complementary Metal Oxide Semiconductor (CMOS) integrated circuit. To step up the signal voltage, an SN74HC125NE 4 buffer integrated circuit was used to convert the CMOS signal to TTL for laser control.

The system includes two safety interlocks: a key switch to arm the laser and a microswitch attached to the door of the laser enclosure to prevent the laser from firing if the access door is open.

The computer is accessed via a RESTful API listening on port 80 for valid JSON messages. A 30-character API key is generated when the application is first run to secure the API.

Query Messages

Json messages in the following formats are accepted:

Read Laser Status:

```
{
    "item": "laserstatus",
    "command": 1
}
```

The Raspberry Pi will return the laser power level, interlock status and if the laser if firing or not.

Command Messages

Json messages in the following formats are accepted:

Switch on laser:

```
{
    "item": "laser",
    "command": "on"
}
```

Switch off laser:

```
{
    "item": "laser",
    "command": "off"
}
```

Set the laser power to nn.n%

```
{
    "setlaserpower": nn.n
}
```

Change the default maximum time the laser can fire to nnn seconds (default is 300)

```
{
    "setlasertimeout": nnn
}
```

Output Messages

Following a “`laserstatus`” command, the following data is returned:

```
{
    "laser":0,
    "power":45.0,
    "key": 1,
    "door": 1
}
```

The "key" and "door" values are

ID	Value	Description
door	0	Door is open, laser disabled
door	1	Door is closed, laser can be armed if key = 1
key	0	Key is in position off, laser disabled
key	1	Key is in position on, can be armed if door = 1

Table 1: Laser status values

Web interface

As well as reading and setting the status of the laser via the RESTful API a read-only web interface that can be accessed directly from a browser and will be available to view the laser status, laser settings, application and web server logs.

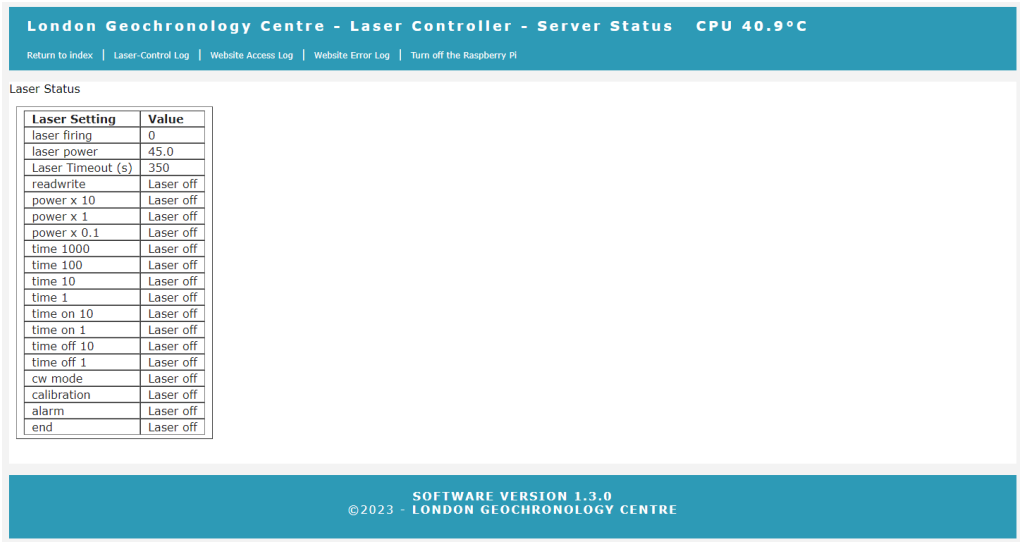

Figure 2: Web status page

Interfaces

TTL convertor Schematic

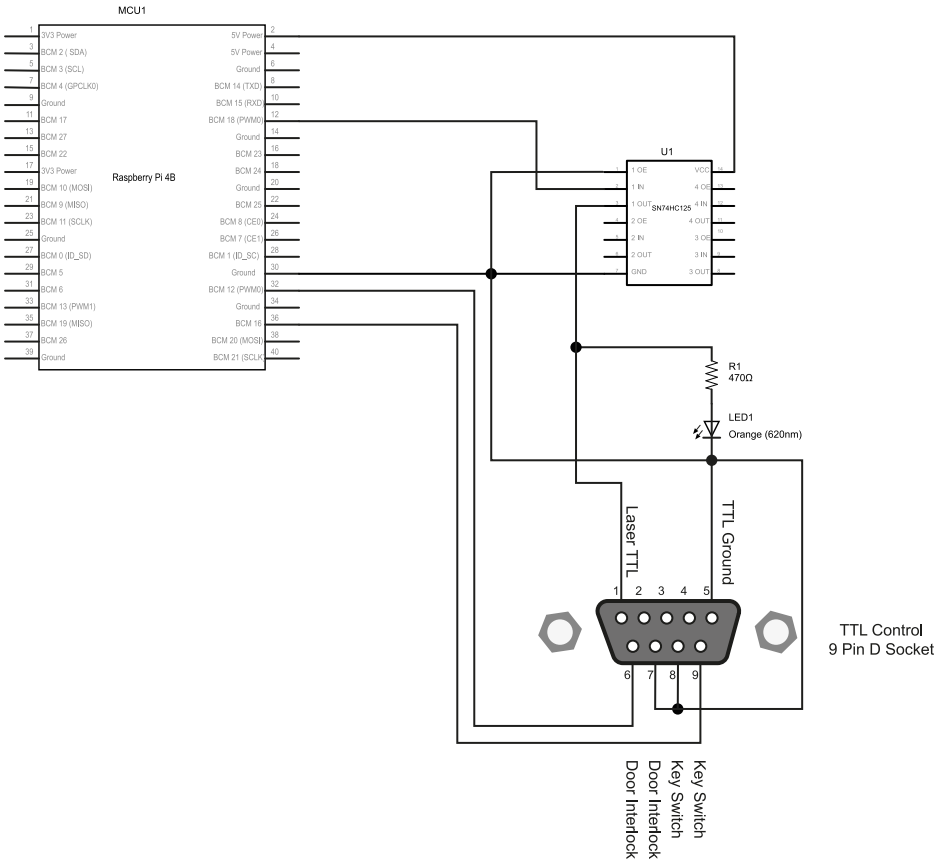

Figure 3: Laser controller TTL convertor and interlock configuration schematic

Raspberry Pi Operating System Installation

Operating system

Use BalenaEtcher to install the latest version of the bookworm-lite 64bit operating system onto a 32Gb MicroMMC card.

Run the `sudo raspi-config` command to:

- enable ssh
- disable Serial
- disable 1^2C bus
- set the password on the pi user account to something other than “raspberry”
- change the GPU memory to 16Gb

Raspberry Pi Software installation

Run Operating System Updates

Run `sudo apt update`
Run `sudo apt upgrade`

Install git and gh

Run `sudo apt install git`
Run `sudo apt install gh`
Run `gh auth login`

- What account do you want to log into? `GitHub.com`
- What is your preferred protocol for Git operations? `HTTPS`
- Authenticate Git with your GitHub credentials? (Y/n) y
- How would you like to authenticate GitHub CLI? `Paste an authentication token`

Nginx installation

Run `sudo apt install nginx`

Configure git and download the valve controller code

Create a GitHub folder
Run `mkdir github`
Run `cd github`

Clone the repo
Run `git clone https://github.com/westerlymerlin/UCL-Oxide-Laser-Controller.git`

Copy the files to the home folder
Run `cp -r ~/github/UCL-Oxide-Laser-Controller/* ~/`

Set the execute flag on the scripts
Run `chmod 755 ~/bin/*`

Copy the Raspberry pi config files to the etc folder
Run `sudo cp -r ~/raspberry-pi/etc/* /etc`

Reboot the Raspberry pi
Run `sudo reboot`

After the reboot a warning banner will appear at ssh logon.

Create a python venv

Run `python3 -m venv .venv`

Change to the venv

Run `source activate`

Install the Required Python Libraries

Run `pip install -r requirements.txt`

Gunicorn for Python service

Run `sudo systemctl enable gunicorn`

Nginx configuration

Change directory to `/etc/nginx/sites-enabled/`
Run `sudo rm default`
Run `sudo ln -s /etc/nginx/sites-available/laserdata`

Reboot the Raspberry pi
Run `sudo reboot`

If flask is installed, the python files are in the `/home/pi` directory, gunicorn3 is installed and configured and nginx is installed and configured the web service should be running, and the site will be accessible on http://*ip address of the server*

References

Python Software Foundation (2023) *Python 3 Programming Language*. Online. Available online: https://www.python.org [Accessed March 2023].

Raspberry PI Foundation (2020) *Raspberry PI Model 4B Reference*. Available online: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/ [Accessed July 2020].

Texas Instruments (1984) *SN74HC125NE4 Datasheet*. Available online: https://www.ti.com/lit/ds/symlink/sn74hc125.pdf [Accessed July 2020].