Oxide Line - Valve 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).

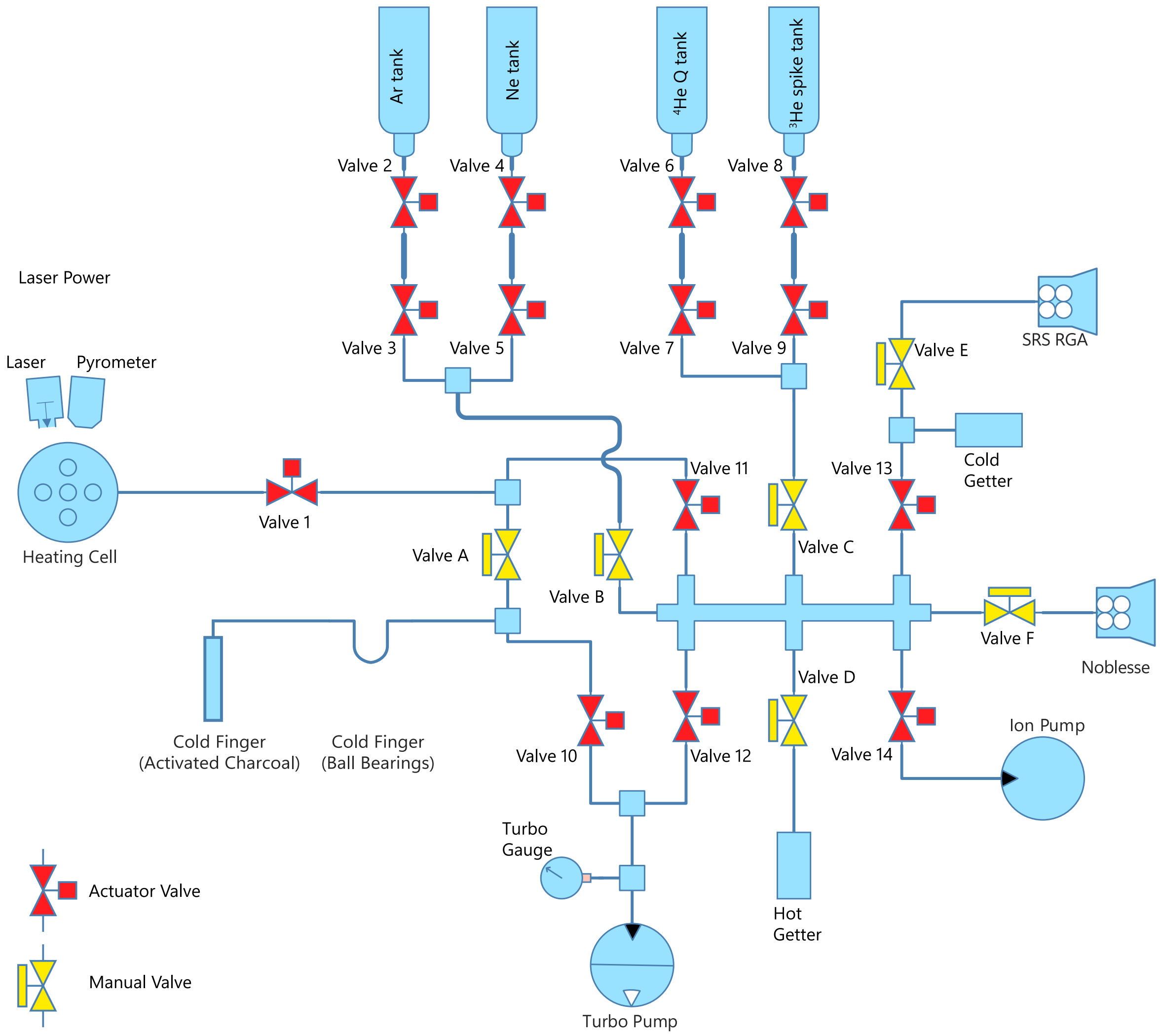


Figure 1: The UCL Oxide Nobel Gass Extraction Line

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.

Valve	Description
Valve A	Cryotrap Isolator
Valve B	Ar / Ne Tank Isolator
Valve C	Helium Tanks Isolator
Valve D	Hot Getter Isolator
Valve E	SRS Mass Spectrometer Isolator
Valve F	Noblesse Mass Spectrometer Isolator

Table 1: Manual Valves

Valve	Description
Valve 1	heating cell
Valve 2	Ar tank pipette input
Valve 3	Ar tank pipette output
Valve 4	Ne tank pipette input
Valve 5	Ne tank pipette output
Valve 6	4He Q tank pipette input
Valve 7	4He Q tank pipette output
Valve 8	3He spike tank pipette input
Valve 9	3He spike tank pipette output
Valve 10	turbo to cryotrap
Valve 11	input to manifold
Valve 12	turbo to manifold
Valve 13	SRS RGA
Valve 14	Ion Pump
Valve 15	spare

Table 2: Automated valves

Valve Controller Description

The valve and laser control software is a Python (Python Software Foundation 2023) application that manages up to 15 solenoid valves. The Raspberry Pi 4B (Raspberry PI Foundation 2020) controls these valves using a 40-way connector for the necessary GPIO pins and driver boards. Each valve operates via a dedicated GPIO line. Since the valves require 24V DC and the Pi uses 3.3V signals, a driver circuit steps up the voltage and current.

The system is managed through a RESTful API on port 80, accepting valid JSON messages. A 30 character API-Key is generated when the application is first run to secure the API.

Safety logic prevents simultaneous opening of pipette input and output valves (valve pairs 2-3, 4-5, 6-7). A single command can close all valves in case of an issue.

Additionally, the controller monitors two vacuum gauges using CH340 USB to RS232 controllers, polling each gauge every 4 seconds to read gas pressure.

Input Messages

Json messages in the following formats are accepted:

Open a single valve (nn):

```
{
    "item": "valvenn",
    "command": "open"
}
```

Close a single valve (nn):

```
{
    "item": "valvenn",
    "command": "close"
}
```

Close all valves:

```
{
    "item": "closeallvalves"
    "command": ""}
```

Get Valve Status: (for all valves)

```
{
    "item": "valvestatus",
    "command": ""
}
```

Read Pressures:

```
{
    "item": "getpressures",
    "command": "read"
}
```

Output Messages

Following a “valvestatus” command any valid command, the following data is returned: (ss = open or closed, xx = on or off)


```
{
    "status": "closed",
    "valve": 1
},
{
    "status": "closed",
    "valve": 2
},
{
    "status": "closed",
    "valve": 3
},
{
    "status": "closed",
    "valve": 4
},
{
    "status": "closed",
    "valve": 5
},
{
    "status": "open",
    "valve": 6
},
{
    "status": "open",
    "valve": 7
},
{
    "status": "open",
    "valve": 8
},
{
    "status": "open",
    "valve": 9
},
{
    "status": "open",
    "valve": 10
},
{
    "status": "closed",
    "valve": 11
},
{
    "status": "closed",
    "valve": 12
},
{
    "status": "closed",
    "valve": 13
},
{
    "status": "closed",
    "valve": 14
},
{
    "status": "closed",
    "valve": 15
}
```

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


```
{
    "pressure": 4.17e-08,
    "pump": "turbo"
},
{
    "pressure": 1.4e-09,
    "pump": "ion"
}
```


Pressures will be in floating point notation when connected to the gauge. When disconnected they will return zeros

Valve to GPIO assignment

Valve	Designation	Connector		Designation	Valve
	3v3	1	2	**5v**	
	GPIO 02	3	4	**5v**	
	GPIO 03	5	6	**GND**	
	GPIO 04	7	8	GPIO 14	
	GND	9	10	GPIO 15	
Valve 1	**GPIO 17**	11	12	**GPIO 18**	Valve 2
Valve 3	**GPIO 27**	13	14	**GND**	
Valve 4	**GPIO 22**	15	16	**GPIO 23**	Valve 5
	3v3	17	18	**GPIO 24**	Valve 6
	GPIO 10	19	20	**GND**	
Valve 7	**GPIO 09**	21	22	GPIO 25	
Valve 8	**GPIO 11**	23	24	GPIO 08	
	GND	25	26	GPIO 07	
	GPIO 00	27	28	GPIO 01	
	GPIO 05	29	30	**GND**	
	GPIO 06	31	32	**GPIO 12**	Valve 9
Valve 10	**GPIO 13**	33	34	**GND**	
Valve 11	**GPIO 19**	35	36	**GPIO 16**	Valve 12
Valve 13	**GPIO 26**	37	38	**GPIO 20**	Valve 14
	GND	39	40	**GPIO 21**	Valve 15

Table 3: GPIO to valve assignments

Table 2 shows GPIO channels on a Raspberry Pi 4B, channels in green are available and remain at 0v during the Pi boot up sequence, it is important that no lines are used that could cause the laser to turn on or a valve to open before the software is ready.

Driver circuits and power supply

The Raspberry Pi 4 operates with a 5V power supply delivered through a USB-C connector. A power supply rated at 3A is necessary for optimal performance. The internal operating voltage of the Raspberry Pi and the output voltage of the GPIO connectors are both 3.3V.

The valves are activated by supplying 24V DC to the solenoid. Upon removal of the power, the valves will close due to a spring mechanism. It is assumed that up to 10 valves may be actuated simultaneously during a pump-down cycle; therefore, the power supply must be capable of delivering at least 1.0A continuously. To ensure reliable provision of this current, a 24V power supply with a continuous rating of 3.0A is required.

To safeguard the Raspberry Pi from potential voltage spikes generated by solenoids during valve closure or due to a malfunctioning component, the use of an opto-isolator is recommended. This will ensure complete electrical isolation between the Raspberry Pi and the driver circuit.

The solenoids on the valves will be operated using Metal Oxide Field Effect Transistors (MOSFETs). MOSFETs exhibit a very low forward voltage once activated and possess a high power capacity (Inchange Semiconductor 2016).

Since the valves are operated by inductive solenoids, removing power from the solenoids can potentially induce voltage spikes into the circuit. Therefore, a Schottky diode is used to protect the MOSFET and opto-isolator from any potential damage.

Each valve driver board has 5 channels, so the valve controller requires 3 driver boards to control the 15 valves (Figure 2).

The driver circuit must be enclosed in a housing that includes external connectors for the mains supply, Ethernet cable, and 12 valve cables. Ventilation holes are necessary to prevent the Raspberry Pi, power supplies, or driver MOS-FET transistors from overheating. Additionally, indicator lights should display the status of the 5V PSU for the Raspberry Pi and the 24V valve PSU.

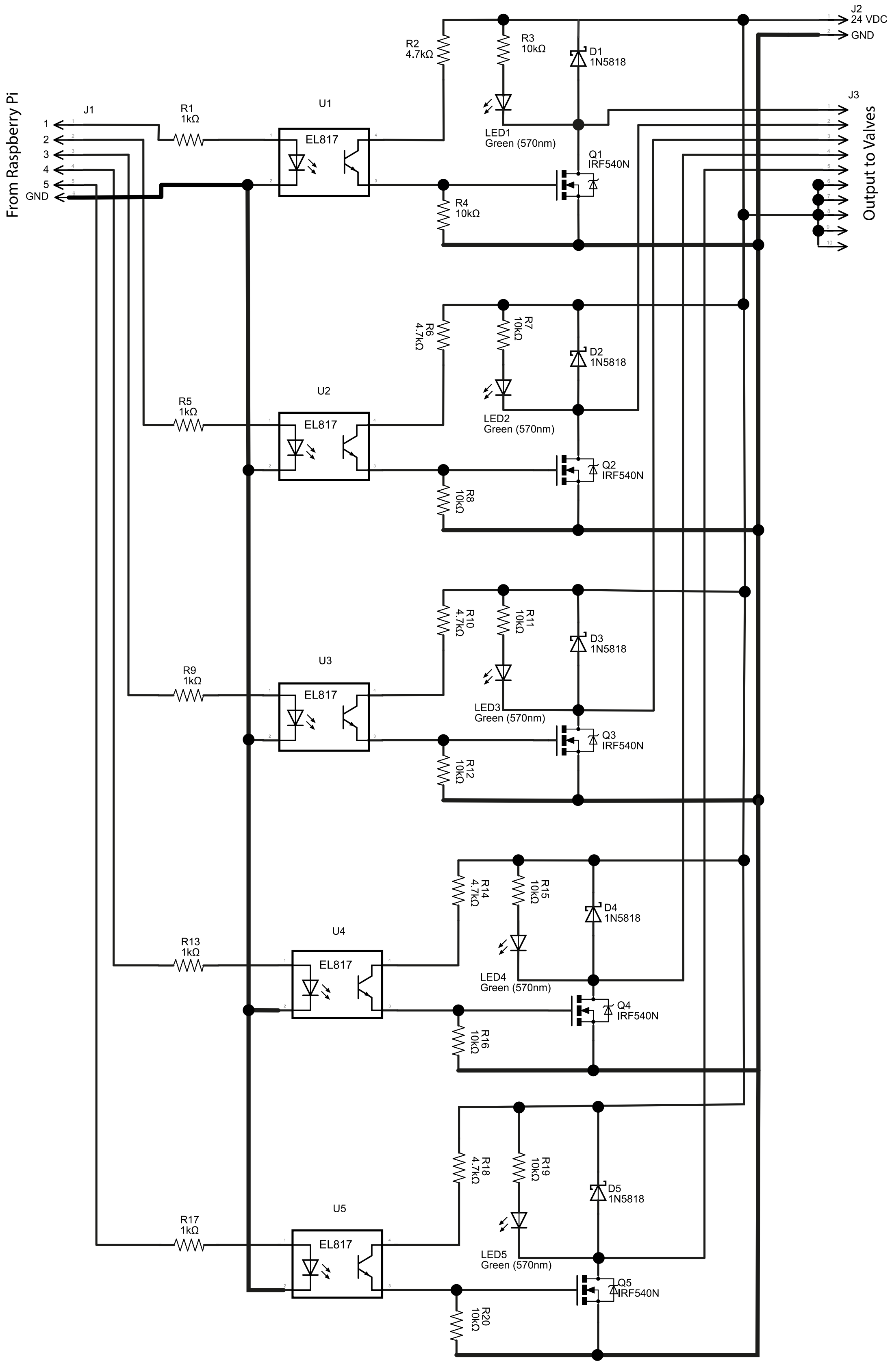
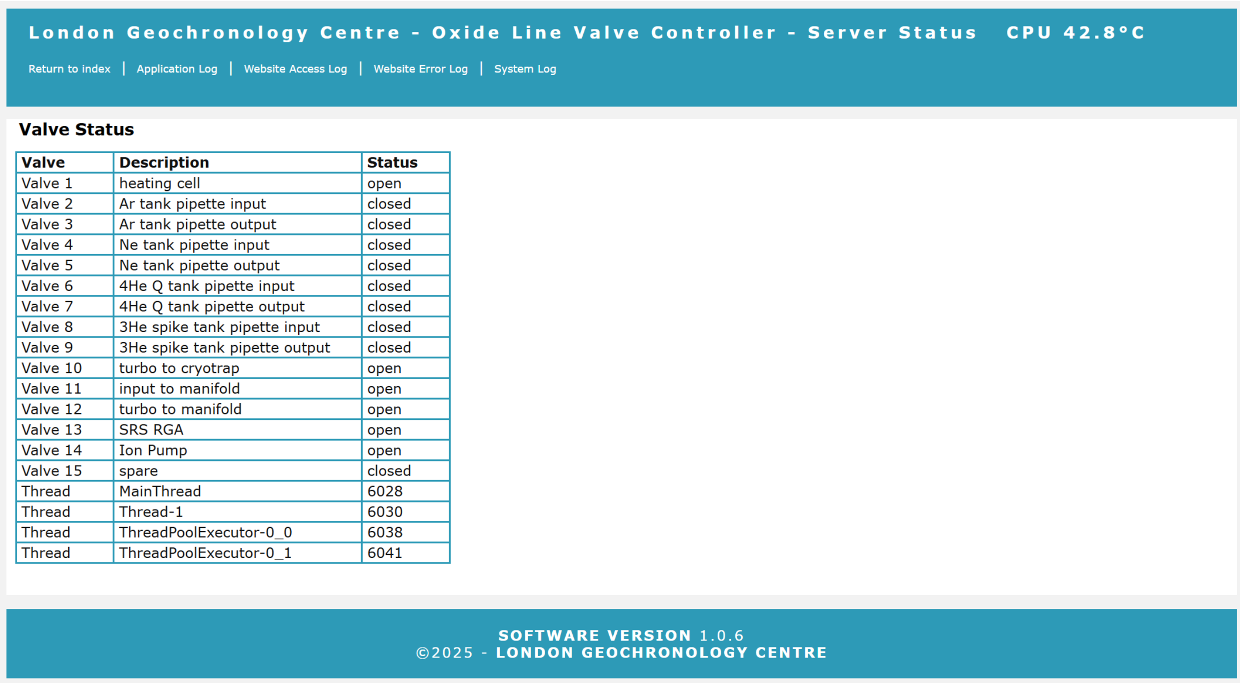


Figure 2: Schematic for a valve driver board

Web interface

In addition to accessing the status of the valves via the RESTful API, a read-only web interface will be available. This interface can be accessed directly from a browser to view the valve status, as well as application and web server logs.



London Geochronology Centre - Oxide Line Valve Controller - Server Status CPU 42.8°C

Return to index | Application Log | Website Access Log | Website Error Log | System Log

Valve Status

Valve	Description	Status
Valve 1	heating cell	open
Valve 2	Ar tank pipette input	closed
Valve 3	Ar tank pipette output	closed
Valve 4	Ne tank pipette input	closed
Valve 5	Ne tank pipette output	closed
Valve 6	4He Q tank pipette input	closed
Valve 7	4He Q tank pipette output	closed
Valve 8	3He spike tank pipette input	closed
Valve 9	3He spike tank pipette output	closed
Valve 10	turbo to cryotrap	open
Valve 11	input to manifold	open
Valve 12	turbo to manifold	open
Valve 13	SRS RGA	open
Valve 14	Ion Pump	open
Valve 15	spare	closed
Thread	MainThread	6028
Thread	Thread-1	6030
Thread	ThreadPoolExecutor-0_0	6038
Thread	ThreadPoolExecutor-0_1	6041

SOFTWARE VERSION 1.0.6
©2025 - LONDON GEOCHRONOLOGY CENTRE

Figure 3: Web status page

Interfaces

9 Pin Valve Cable Specification

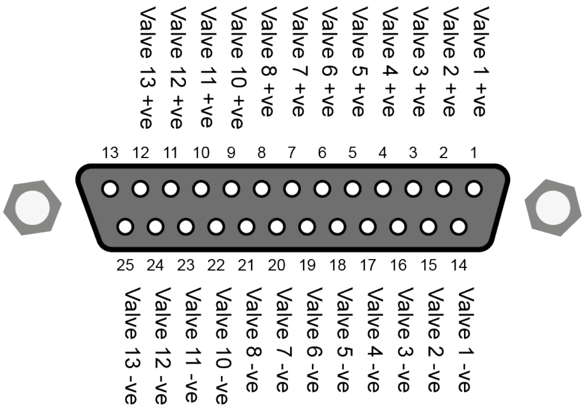


Figure 4: Valve controller 9 pin D plug assignments

Three 9 pin D plugs connected to 5 individual cables each, terminated with an SMC 24v latching power plug.

9 Pin RS232 Cable specification

The connection from RS232 port a standard 9 pin D Socket, a null modem switching pins 2 and 3 (receive and transmit) is used to connect to the gauges and pyrometer.

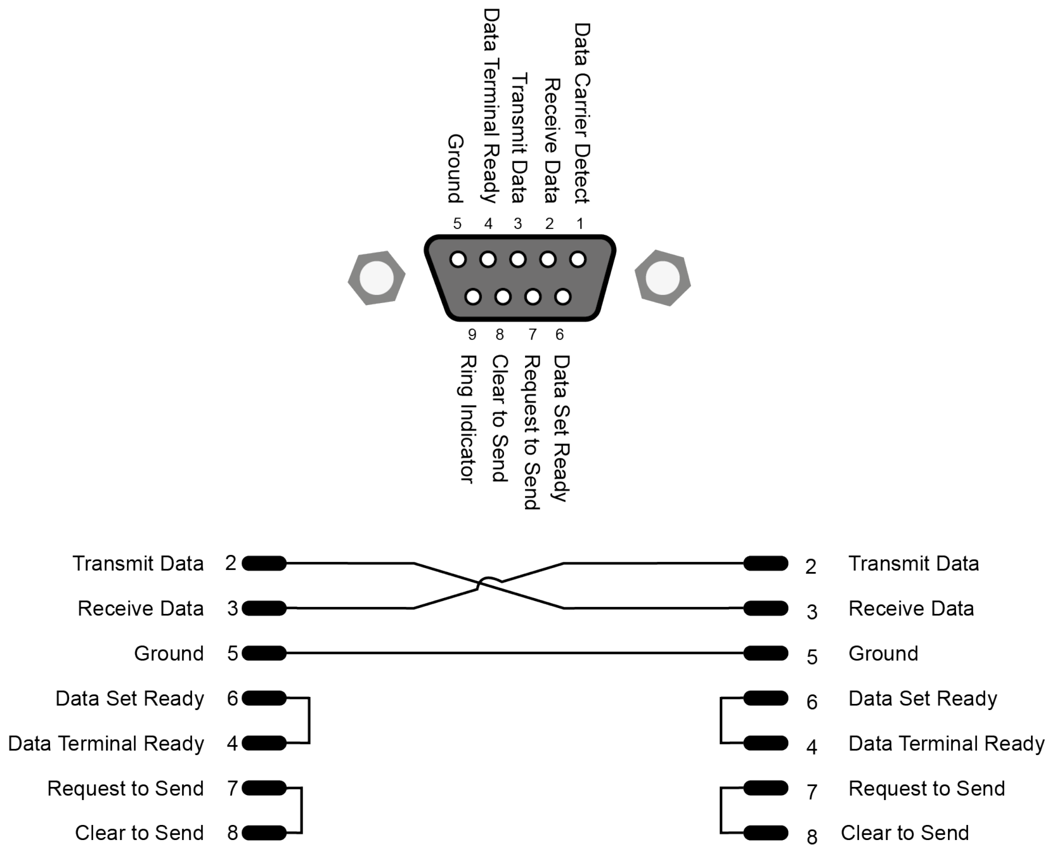


Figure 5: RS232 D Plug Assignment and null modem cable

USB Connections

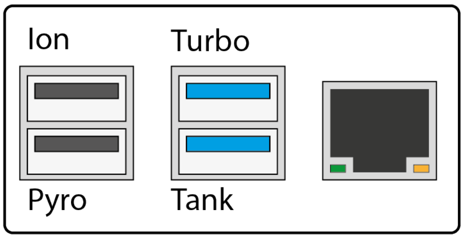


Figure 6: USB connections to the Valve Controller Raspberry Pi

Connection	USB	Serial Device	Equipment Connector
Ion Pump	USB3 port 0	/dev/ttyUSB0	RS232 Socket (female)
Turbo Pump	USB3 port 1	/dev/ttyUSB1	RS232 Plug (Male)

Table 4: External Equipment Connections

Gauge Data Request Commands

Equipment	First Command (Hexadecimal)	Second Command (Hexadecimal)
Pfeiffer Vacuum TPG 1 (Tank and Turbo)	50 52 31 0D (Pressure Gauge 1)	05 (Enquiry)
Gamma Vacuum SPe (Ion Pump)	7E 20 32 35 20 30 42 20 30 30 0D (read pressure from ID 05)	

Table 5: Commands to Vacuum Gauges

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`

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-Valve-Controller.git`

Copy the files to the home folder

Run `cp -r ~/github/UCL-Oxide-Valve-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`

Run `sudo systemctl start gunicorn`

Nginx installation

Run `sudo apt install nginx`

Change directory to `/etc/nginx/sites-enabled/`

Run `sudo rm default`

Run `sudo ln -s /etc/nginx/sites-available/valvedata`

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

Inchange Semiconductor (2016) IRF540N. Available online: http://www.iscsemi.com [Accessed.

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].