Valve and Laser Controller

Software Functional Requirements

By **Gary Twinn**

Contents

List of Figures

List of Tables

Overview

The valve and laser control software consists of a python (Python Software Foundaton, 2020) application to control the 12 valves of the Helium line and a single channel Transistor - Transistor Logic (TTL) output to switch on/off the Laser.

Valve	Description
1	^{4}He pipette input
2	^{4}He pipette output
3	^{3}He pipette output
4	^{4}He pipette input
5	port 1(not currently used)
6	ion pump
7	gas analyser
8	gallery A
10	laser cell
11	getter
12	buffer tank
13	turbo pump

Table 1: He line valves

The computer that controls the valves is a Raspberry Pi 4B (Raspberry PI Foundation, 2020), it uses a 40 way connector to break out the required 12 GPIO pins and connect to the driver boards.

Each valve is controlled by a dedicated GPIO line on the raspberry Pi computer. The valves run on 24V DC and the Pi uses a 3.3V signal, so a driver circuit is required to step up the voltage and current from the Pi.

The computer is controlled via a RESTful API listening on port 80 for a valid json message.

The software contains logic to prevent the input and output valves on a pipette opening at the same time in the case of a mistake in the commands sent to the computer. (valve 1 and valve 2), (valve 3 and valve 4)

To simplify charging and unloading of pipettes a single command was implemented to load and unload each pipette. The pipette command contains a 0.5s delay between one valve closing and one opening.

The controller also operates a single 5V TTL line to switch on and off the laser. The Raspberry Pi outputs are 3.3v and TTL is based on 5V so a buffer was needed to step the voltage up from 3.3V to 5V.

A single command was also implemented to close all valves and switch off the laser in case of an issue.

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": ""}
```

Load pipette (n): (^{4}He = 1, ^{3}He = 2)

```
{
    "item": "pipetten",
    "command": "load"
}
```

Unload pipette (n): (^{4}He = 1, ^{3}He = 2)

```
{
    "item": "pipetten",
    "command": "unload"
}
```

Get Valve Status: (for all valves and laser)

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

Laser On

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

Laser Off

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

Output Messages

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


```
{
    "status": "ss",
    "valve": 1
},
{
    "status": "ss",
    "valve": 2
},
{
    "status": "ss",
    "valve": 3
},
{
    "status": "ss",
    "valve": 4
},
{
    "status": "ss",
    "valve": 5
},
{
    "status": "ss",
    "valve": 6
},
{
    "status": "ss",
    "valve": 7
},
{
    "status": "ss",
    "valve": 8
},
{
    "status": "ss",
    "valve": 10
},
{
    "status": "ss",
    "valve": 11
},
{
    "status": "ss",
    "valve": 12
},
{
    "status": "ss",
    "valve": 13
}
{
    "laser": "0",
    "status": "xx"
}
```

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 2	**GPIO 17**	11	12	**GPIO 18**	Valve 5
Valve 6	**GPIO 27**	13	14	**GND**	
Valve 10	**GPIO 22**	15	16	**GPIO 23**	Valve 1
	3v3	17	18	**GPIO 24**	Valve 8
	GPIO 10	19	20	**GND**	
Valve 7	GPIO 09	21	22	GPIO 25	
Valve 11	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**	Ready LED
Valve 3	**GPIO 13**	33	34	**GND**	
Valve 4	**GPIO 19**	35	36	**GPIO 16**	
Valve 13	**GPIO 26**	37	38	**GPIO 20**	Laser TTL
	GND	39	40	**GPIO 21**	Valve 12

Table 2: GPIO to valve assignments

Table 1 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.

Once the boot sequence has completed and the software has started the final command on the initialising function will be to light the “Ready LED” (GPIO12) to give a visual indication the Raspberry PI had booted and is ready to accept commands on the REST api.

Driver circuits and power supply

The Raspberry Pi 4 uses a 5v power supply provided via a USB-C connector. A 5V power supply rated at 3A will be required. The Raspberry Pi internal voltage and the outputs of the GPIO connectors is 3.3v.

The laser requires a TTL control signal with a voltage higher than 4v for an ‘ON’ and a voltage lower than 1v for an ‘OFF’. A SN74HC125NE4 quad buffer integrated circuit (Texas Instruments, 1984) will be used to step the 3.3v signal up to a TTL signal for the Laser control (Figure 1).

The valves are opened by applying 24V DC to the solenoid, when the power is removed the valves will close against a spring. The assumption is that the maximum number of valves that could be actuated (during a pump-down cycle) would be 10 valves so the supply should be able to supply a minimum of 1.0A continuously. In order to provide this level of current reliably a 24V power supply with a continuous rating of 3.0A will be required.

In order to protect the Raspberry Pi from any voltage spikes that may be generated by the solenoids as the valves close, or a faulty component, an opto-isolator will be required to provide full electrical isolation between the Pi and the driver circuit.

The solenoids on the valves will be driven via Metal Oxide Field Effect Transistors (MOSFET). MOSFETs have a very low forward voltage once switched on and have a high-power capacity (Inchange Semiconductor, 2016).

As the valves are operated by inductive solenoids, when the power is removed from the solenoids there is a chance voltage spikes will be induced into the circuit so a Schottky diode will protect the MOSFET and opto-isolator.

Each valve driver board has 4 channels so the valve controller requires 3 driver boards to control the 12 valves (Figure 2).

The driver circuit will need to be housed in an enclosure that has external connectors for the mains supply, ethernet cable and connectors for the 12 valve cables. It must have ventilation holes to prevent the Raspberry Pis, power supplies or driver MOS-FET transistors overheating. Indicator lights should show the status of the 5v PSU for the Raspberry Pi, the 24V valve PSU and the software ready light.

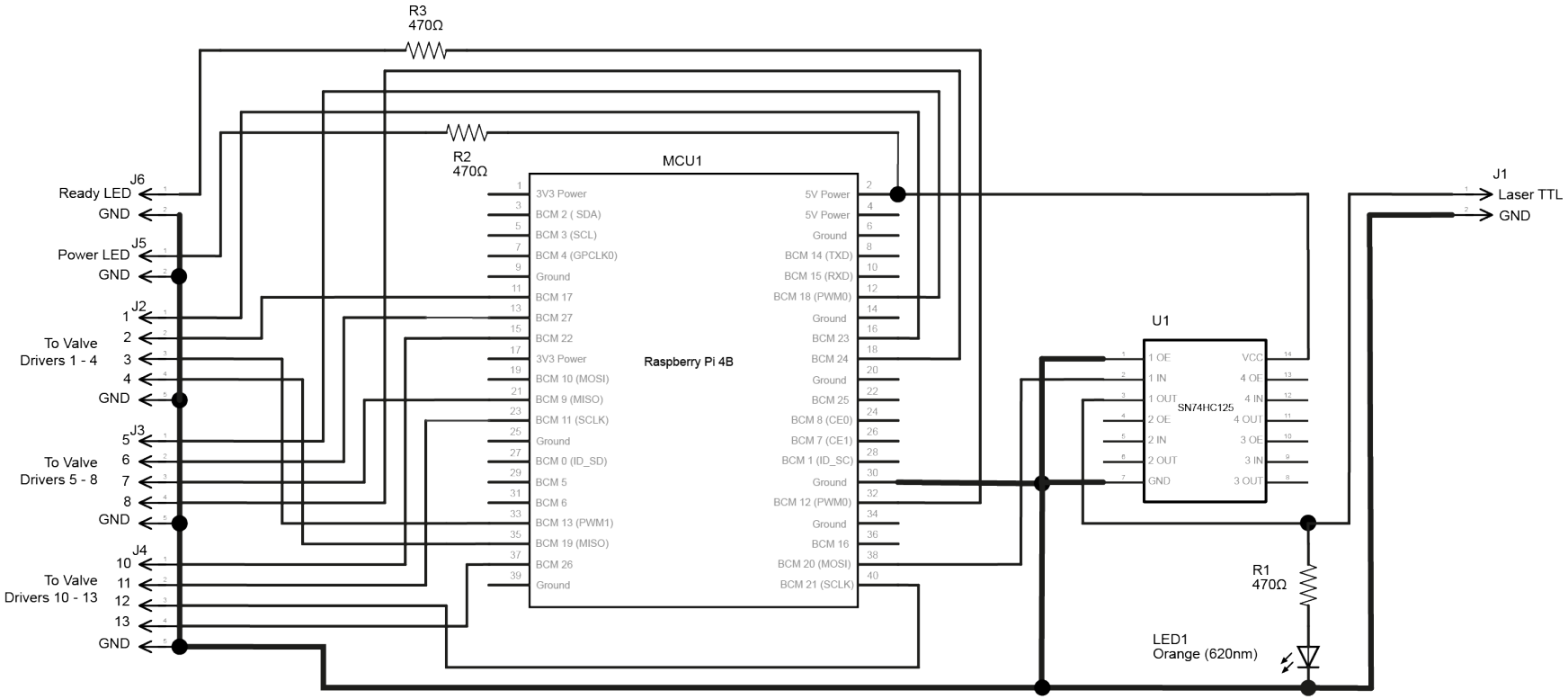


Figure 1: Schematic for Raspberry Pi connections and TTL buffer for Laser

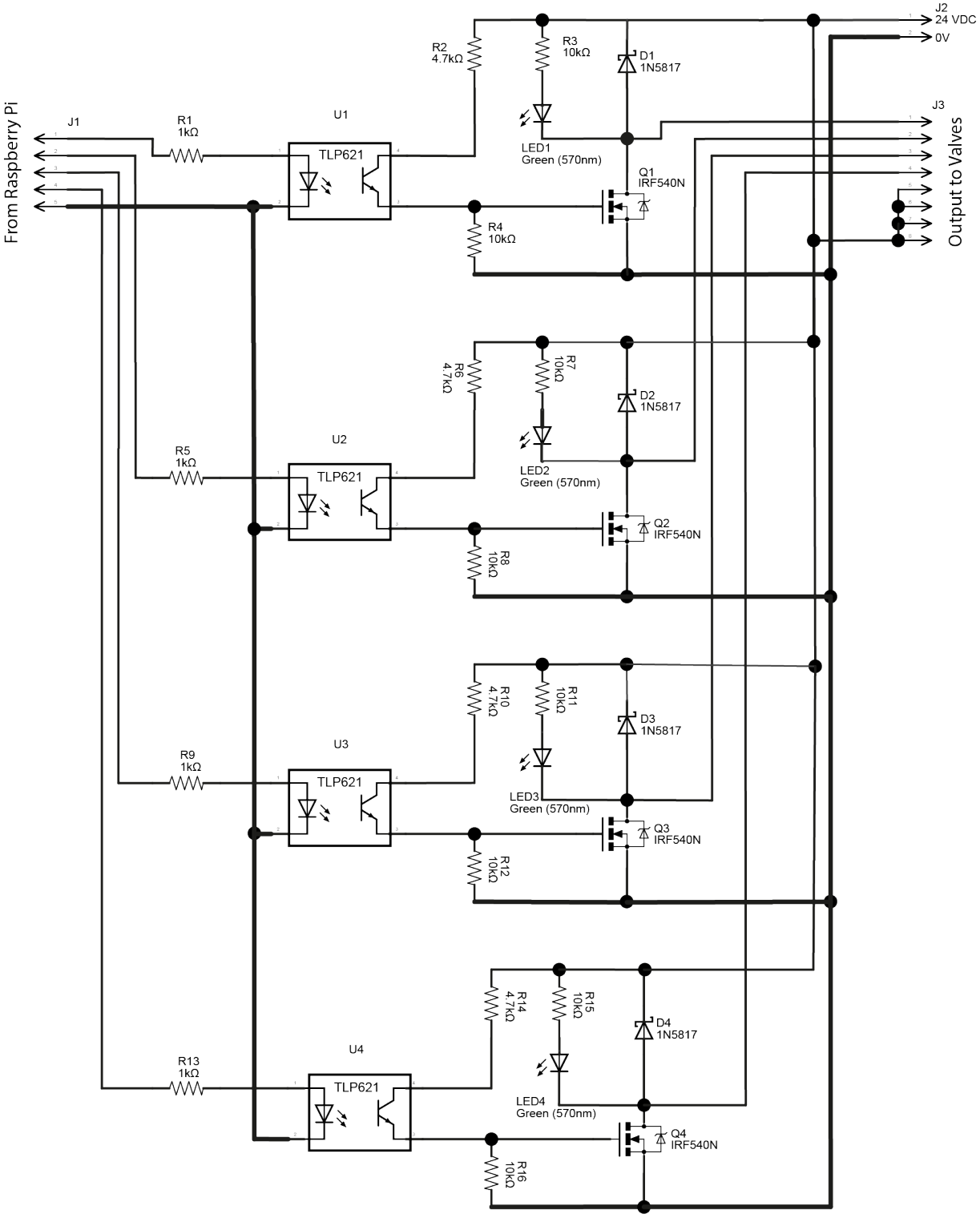
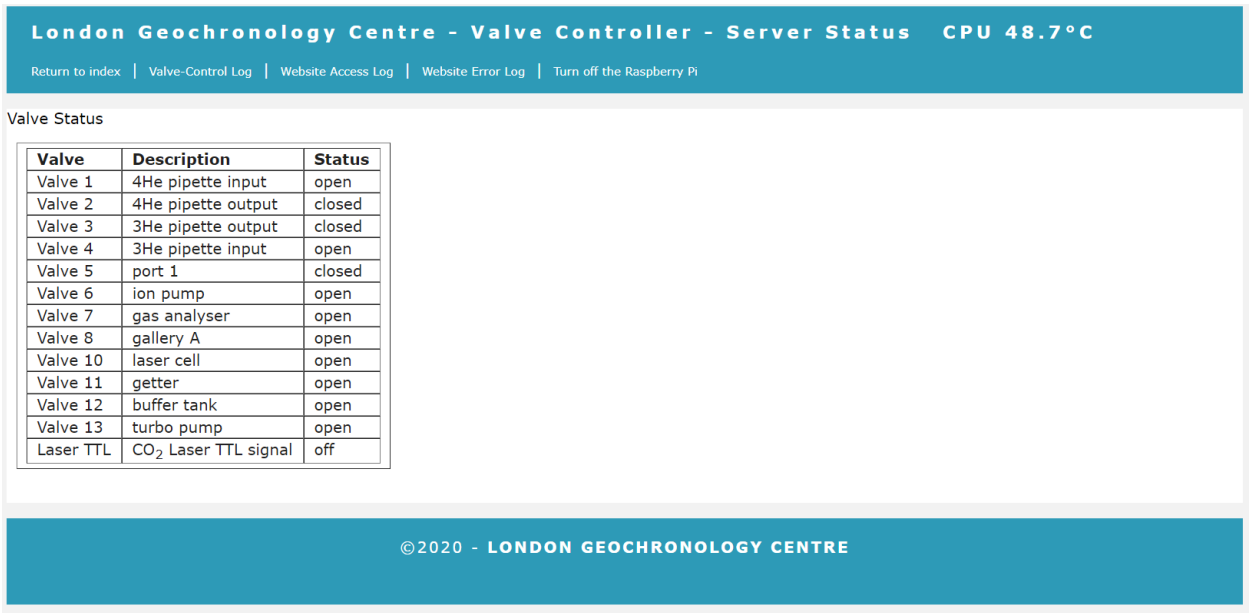


Figure 2: Schematic for a valve driver board

Web interface

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



Valve	Description	Status
Valve 1	4He pipette input	open
Valve 2	4He pipette output	closed
Valve 3	3He pipette output	closed
Valve 4	3He pipette input	open
Valve 5	port 1	closed
Valve 6	ion pump	open
Valve 7	gas analyser	open
Valve 8	gallery A	open
Valve 10	laser cell	open
Valve 11	getter	open
Valve 12	buffer tank	open
Valve 13	turbo pump	open
Laser TTL	CO_2 Laser TTL signal	off

Figure 3: Web status page

Interfaces

Valve Cable Specification

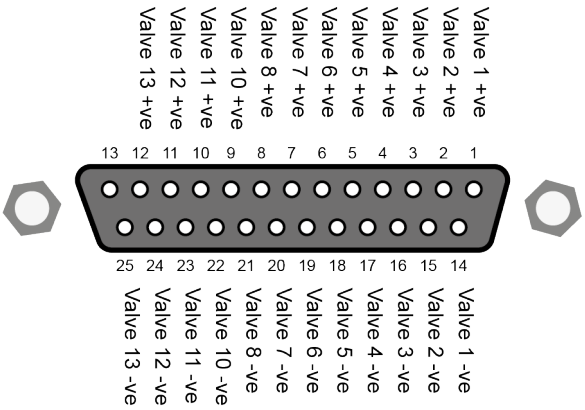


Figure 4: Valve controller D plug assignments

A 25 pin D plug connected to 12 individual cables, each terminated with an SMC 24v latching power plug.

Laser TTL Cable specification

The connection from the back of the housing to the valves is a standard 9 pin D Socket

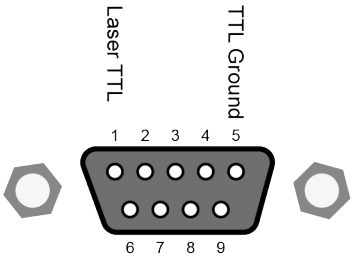


Figure 5: Laser TTL D Plug Assignment

A 9 way D plug linked to a female SMB mini coaxial connector with RG 174/U cable is required, pin 1 of the D plug should be connected the centre tap of the SMB connector.

Operating System Installation

Operating system

Use BalenaEtcher to install the latest version of the buster-lite 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

Software installation

Run Operating System Updates

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

Install PIP3 for Python 3.x installation

Run `sudo apt install python3-pip`

Install Flask installation

Run `sudo pip3 install flask`

Download Valve Controller Application

Download from https://github.com/westerlymerlin/UCL-RPi-ValveController.git

Copy the python files to the Pi

- Copy the files from GitHub to `/home/pi/`
- Copy the folder templates to `/home/pi/templates`
- Copy the folder static to `/home/pi/statis`
- Create a folder `/home/pi/database`
- Create a folder `/home/pi/logs`
- `Copy the files /raspberry-pi/home/pi to /home/pi`

Nginx installation

Run `sudo apt install nginx`

Copy the Github file
`\raspberry-pi\home\pi\etc\nginx\nginx.conf`
to
`/etc/nginx/nginx.conf`

Copy the GitHub file
`\raspberry-pi\home\pi\etc\nginx\sites-available\icpmsdata`
to
`/etc/nginx/sites-available/icpmsdata`

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

Gunicorn for Python 3.x installation

Run `sudo apt install gunicorn3`

Copy the GitHub file
`\raspberry-pi\home\pi\etc\systemd\system\gunicorn.service`
to
`/etc/systemd/system/gunicorn.service`

Run `sudo systemctl enable gunicorn`

Run `sudo systemctl start gunicorn`

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 July 2020].

Python Software Foundaton (2020) *Python 3 Programming Language*. Online. Available online: https://www.python.org [Accessed July 2020].

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].