Pump Reader Controller

Software Functional Requirements

By **Gary Twinn**

Contents

List of Figures

List of Tables

Overview

The Pump Reader control software consists of a python (Python Software Foundation, 2020) application to read the pressures from the Vacuum Gauges and Pyrometer. The application is written to use the Flask web interface to provide an HTTP Status page as well as serving requests via the API (Application Programming Interface)

The computer that reads the gauges is a Raspberry Pi 4B (Raspberry PI Foundation, 2020), it uses 4 x CH340 USB to RS232 controllers to access the gauges and pyrometer.

It has an internal scheduler that poles each gauge every 4 seconds with a query to read back the pressure. The pyrometer is polled the same way for temperature. There are also two commands that can be sent to the Pyrometer in order to switch on and off the range finding lasers.

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

Query Messages

Json messages in the following formats are accepted:

Read Pressures:

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

Read Pyrometer Temperature:

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

Command Messages

Json messages in the following formats are accepted:

Switch on pyrometer laser:

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

Note, the laser ‘on’ request will also set a time to automatically switch off the laser after 60 seconds unless it is manually switched off earlier.

Switch off pyrometer laser:

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

Output Messages

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

```
[
    {
        "pressure": 4.17e-08,
        "pump": "turbo"
    },
    {
        "pressure": 0.000691,
        "pump": "tank"
    },
    {
        "pressure": 1.4e-09,
        "pump": "ion"
    }
]
```

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

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

```
{
    "temperature": 385,
    "laser": "off",
}
```

Temperature will be °C in floating point notation when connected to the pyrometer. When disconnected it will return a zero.

Web interface

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

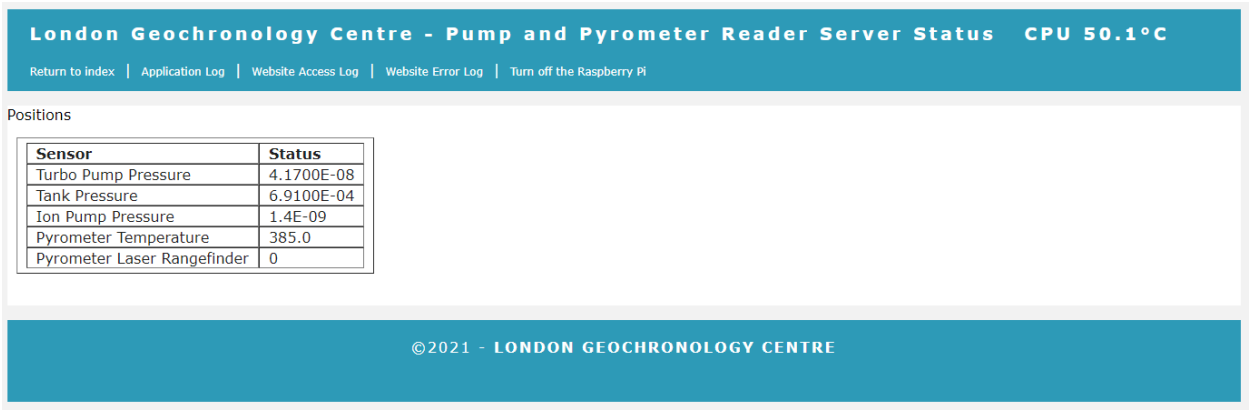

Figure 1: Web status page

Interfaces

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.

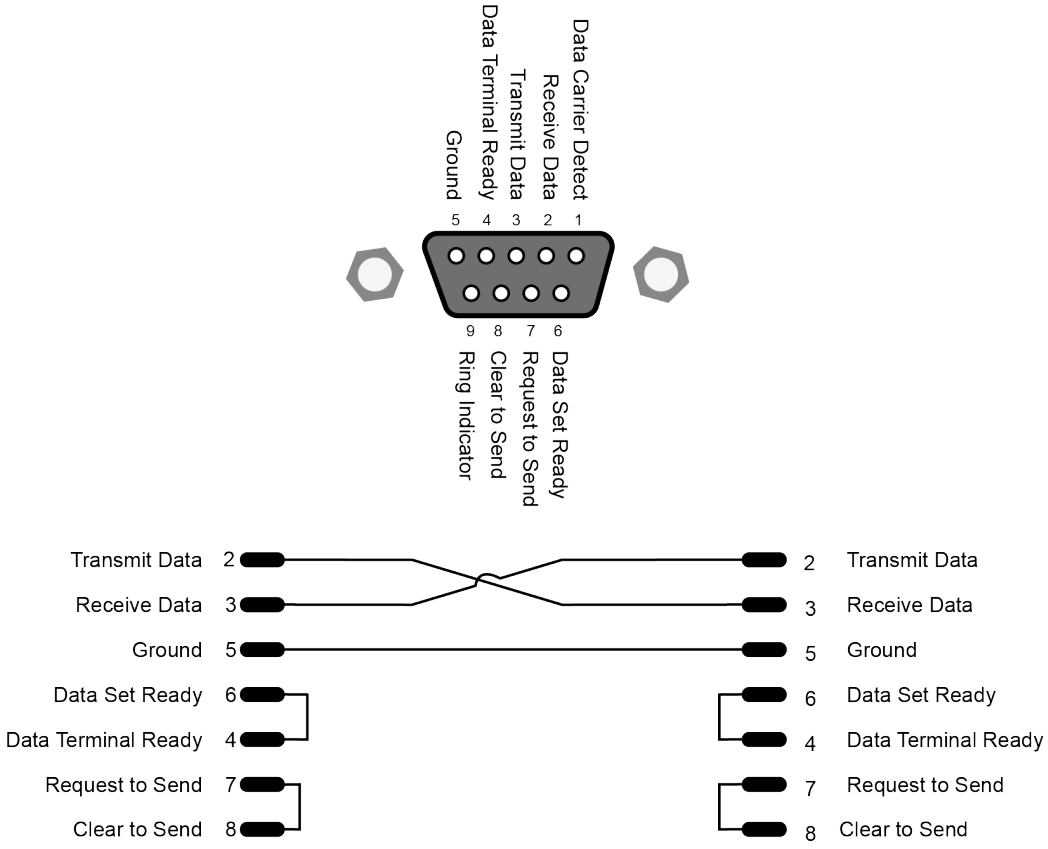

Figure 2: RS232 D Plug Assignment and null modem cable

Connections

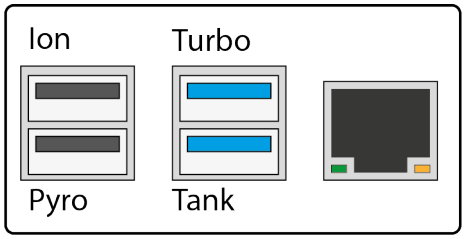

Figure 3: USB connections to the Pump Reader Raspberry Pi

Connection	USB	Serial Device	Equipment Connector
Turbo Pump	USB3 port 0	/dev/ttyUSB0	RS232 Plug (Male)
Tank Pump	USB3 port 1	/dev/ttyUSB1	RS232 Plug (Male)
Ion Pump	USB2 port 0	/dev/ttyUSB2	RS232 Socket (female)
Pyrometer	USB2 port 1	/dev/ttyUSB3	RS232 Socket (female)

Table 1: External Equipment Connections

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)	
MicroEpsilon Pyrometer	01 (Read Temperature)	
	05 (Read Laser Status)	
	A5 01 A4 (Switch Laser On)	
	A5 00 A5 (Switch Laser Off)	

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

Software installation

Run Operating System Updates

Run `sudo apt update`
Run `sudo apt full-upgrade`

Install PIP3 for Python 3.x installation

Run `sudo apt install python3-pip`

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-RPi-PumpReader.git`

Copy the files to the home folder
Run `cp -r ./github/UCL-RPi-PumpReader/*.sh  ./`

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.

Install the Flask libraries

Run `sudo pip install flask --break-system-packages`

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/icpmsdata`

Gunicorn for Python 3.x installation

Run `sudo apt install gunicorn3`
Run `sudo systemctl enable gunicorn`
Run `sudo systemctl start gunicorn`

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 (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].