Reference Manual for the Implementation of the Control System for a Pneumatic Valve over the TCP/IP Protocol

Miguel Simão (miguel.simao@uc.pt)

Collaborative Robotics Group
Department of Mechanical Engineering
University of Coimbra

Abstract

Documentation for the set up that actuates a pneumatic valve at the base of the IIWA. The end goal is to commutate air pressure between the two air inputs according to a control message to be sent over a TCP/IP socket. On the first section, we show how the current set up can be used and a list of things that may be implemented in a future version. The second section describes in detail the hardware used in each of the subsystem of the system. The last section concerns both the network configuration and software used on the controller. It also describes how the system can be replicated, updated and extended.

Contents

1 Introduction

1.1 Usage

The RPi server accepts several connections over the TCP/IP protocol, for which you need to provide an IP address and a port. As of now, the RPi has a static IP address in the wi-fi network that you should only use if you're trying to connect over wi-fi:

```
192.168.1.199
```

This IP should be valid forever. The port used is 1080. If you're trying to connect from a computer in the lab's wired network, the temporary IP address you should use is (Ethernet):

```
172.16.22.197
```

at the same port. This is the IP of the router and it will redirect the connection to the target RPi only if you use this port. Otherwise, you have to change some network configurations..

→ See Section 3.3

! → For now, the IP address of the router is dynamic, but it will be given a static IP to be defined in the close future.

You can turn on the RPi just by powering it with a μUSB cable. The server is running when the large red LED is bright. The following commands are available:

1. O10 - [Default] Turns on "Air 1" and locks tool
2. O11 - Turns on "Air 2" and unlocks tool
3. reboot - reboots the RPi
4. shutdown - shuts down the RPi

Every command should include a terminator that is CR/LF (carriage return and line feed). The first character of the output is the letter "O", for Output. The next character is the output number, which at this time is just the "1". The final character sets the value ON ("1") or OFF ("0") of the digital output @24VDC.

1.2 "To do" List

List of things to implement on the future:

LOW PRIORITY
1. Maybe change to a connectionless type of sockets (`SOCK_DGRAM`)

HIGH PRIORITY
2. Implement ACPI to the RPi to prevent damage

HIGH PRIORITY
3. Add a manual override to the valve control (lock/unlock)

HIGH PRIORITY
4. Configure RPi to connect to the local network of the robot through its Ethernet port

2 Hardware

In this section we describe the different subsystems that can be defined in this system. We also describe their purpose and the relationships between distinct subsystems.

2.1 Material List

To begin with, you have access to the following parts:

1. Pneumatic valve with electric control
2. 24VDC power supply
3. Relay board (breakout)
4. 5VDC regulated power supply A regular USB power supply of at least 2.5A suffices.
5. Raspberry Pi 2
6. USB Wi-fi adapter
7. 1 Red LED
8. 1 100Ω resistor
9. 2 Yellow LEDs
10. 2 1500Ω resistors

2.2 Actuation Subsystem

The actuation subsystem is composed by the commutation pneumatic valve. It interfaces with the control subsystem. It moves the valve between the following two positions:

Ouputs:

1. P2 – Pneumatic output 2
2. P4 – Pneumatic output 4

It requires as input the 24 VDC signals and a GND connection (referenced to the input signals):

Inputs:

1. S1 – 24V signal 1 (S1)
2. S2 – 24V signal 2 (S2)
3. GND24 – Ground

The input signal S1 activates P2 and S2 activates P4.

When you turn on the system, Air 1 is active and locking the tool.

A real representation in shown in fig. 1. In this figure, P1 is the pressured air input from its source, P3 and P5 are sinks, P12 and P14 are connected with P1 and assist the actuation of the valve. P2 is connected to `Air 1` to the robot and P4 to `Air 2`. When the system is on, `Air 1` is the normally pressured line. If the system is off, both air line are locked.

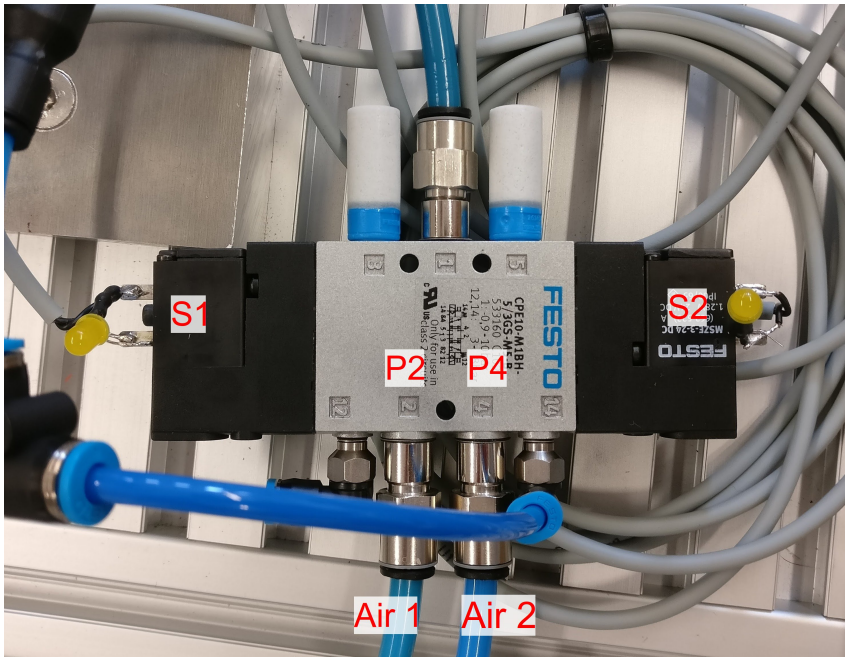

Figure 1: Actuation subsystem. Pneumatic valve with 3 positions and electrically-assisted control.

2.3 Control Subsystem

The control subsystem is composed of the relay and interfaces with the communications subsystem and the actuation subsystem.

→ See Section 2.2

The actuation valve is normally centred and requires two 24V signals to commutate between P2 and P4 (see section 2.2). To provide those, we use a single relay (fig. 2) to commutate a 24V source from a power supply, between the two signals S1 and S2. The control signal should be 5V but can be done with the 3.3V output level of the RPi.

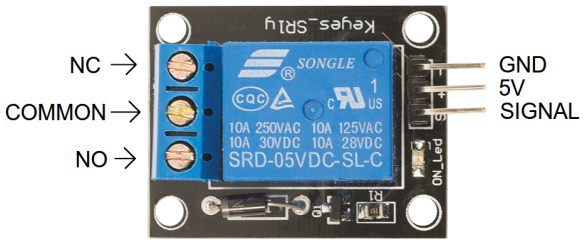

Figure 2: Board with one relay. NC stands for normally closed and NO for normally open.

When the RPi digital output is high (3.3V), the relay connects (left side of the picture) the common line to the Normally Open (NO) line. Otherwise, the Normally Closed (NC) line is active. The 5V (DC) input may come from the RPi 5V line and GND from the GND line on the RPi's General Purpose Input/Output (Raspberry Pi) (GPIO) pins.

Outputs:

1. S1 – 24V signal 1 (S1)
2. S2 – 24V signal 2 (S2)

Inputs:

1. S – Control signal (3.3V)
2. PWR24 – 24 VDC from source
3. PWR5 – 5 VDC from source
4. GND5 – Ground referenced to PWR5

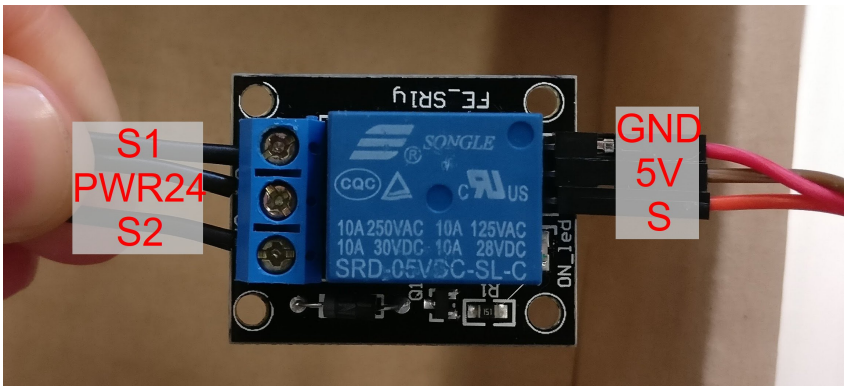

Figure 3: Control subsystem. Single relay board with 24V rail on the left side and the control side on the right.

2.4 Communications Subsystem

The communications subsystem is a Raspberry Pi with a socket server running that allows remote connections. It is composed by the RPi (fig. 4) and an USB Wi-fi adapter OR an Ethernet connection. Both interfaces can also be connected (Ethernet + Wi-fi). For reference, the GPIO pin layout is shown in fig. 5.

→ See Figure 5 to see GPIO header pin layout.

A real representation of the subsystem in shown in fig. 4. Power to the RPi is provided by the μUSB input, but it is also possible to power it from a regulated[1] 5V power supply to pin 2. Regarding outputs, power and ground to the control subsystem are derived from the 5V output at pin 4 and 6. The control signal `S` is provided by pin 7 (GPIO4). There's also a red indication LED connected to pins 26 (anode) and 25 (cathode).

Input/output list:

Ouputs:

1. S – Control signal for the control subsystem
2. PWR5 – 5VDC (limited amperage)
3. GND5

Inputs:

1. Network (USB Wi-fi or Ethernet)
2. PWR5 – 5VDC from source or USB
3. (Optional) GND5 – Ground, if USB is NOT used

[1] Careful! There's no circuit protection on the GPIO header, so if your supply peaks above 5V, you can fry your RPi.

Figure 4: Real set-up of the communications subsystem.

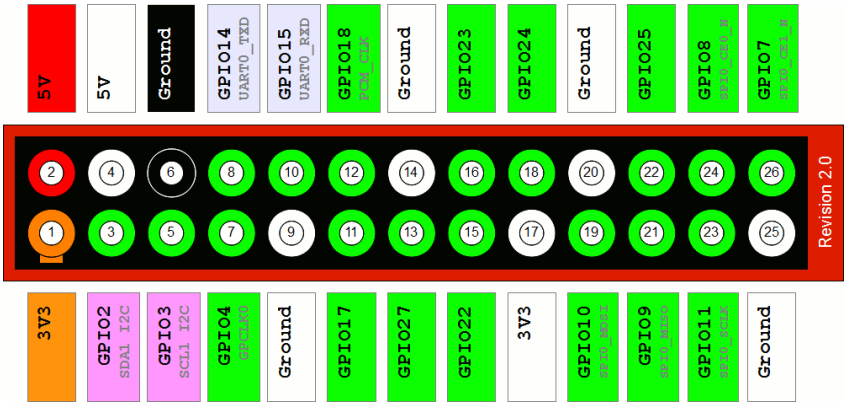

Figure 5: Raspberry Pi 2/3 GPIO layout.

2.5 Power Subsytem

There are two power rails in this system: 24 VDC and 5 VDC. There are separate power supplies for each, so the grounds are referenced individually. The 5 VDC supply is required for the Raspberry Pi and the control subsystem.

Ouputs:

1. PWR24 – 24 VDC source
2. GND24 – Ground for 24 VDC devices
3. PWR5 – 5 VDC source
4. GND5 – Ground for 5 VDC devices

Inputs:

1. Network (Wi-fi or Ethernet)
2. PWR5 – 5VDC from USB or, optionally, source
3. (Optional) GND – Ground, if USB is NOT used.

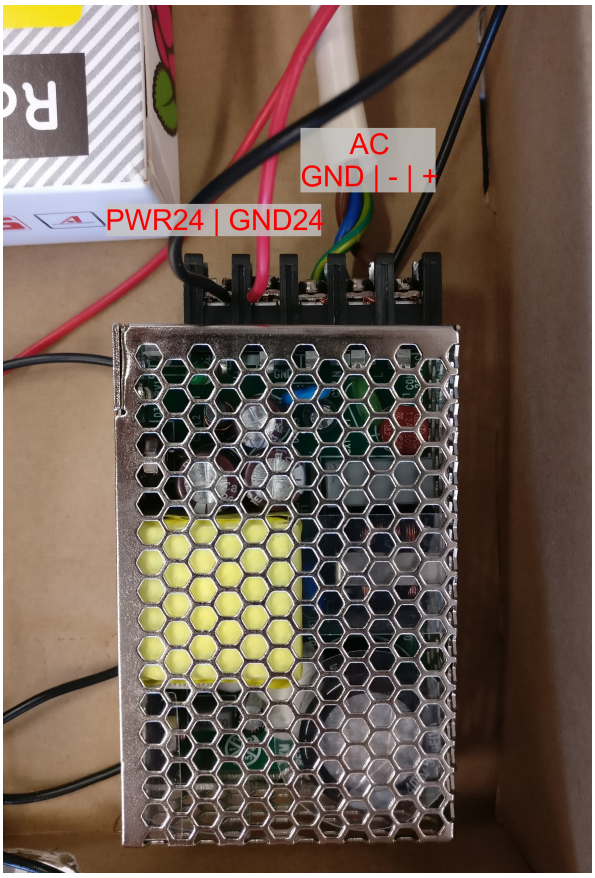

Figure 6: 24 VDC power supply.

2.6 Display subsystem

This subsystem is responsible for giving feedback to the user about the status of important signals in the different subsystems. For now, the only indicators we use are LEDs, which are:

1. Valve P2 status
2. Valve P4 status
3. RPi server status

All of the LEDs are connected in a circuit that is simplified in fig. 7. One yellow LED connects S1 to GND and another connects S2 to GND (see fig. 1). Since these poles have a voltage of 24V, the resistors were calculated to be of 1500Ω.

A red LED, is connected on the RPi's GPIO on the pin GPIO07 (3.3V, fig. 5) with a 100Ω resistor.

2.7 Full Setup

Schematically, the whole system is represented in fig. 8.

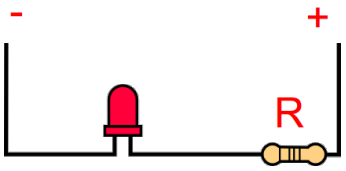

Figure 7: Simplified circuit diagram used to power an LED.

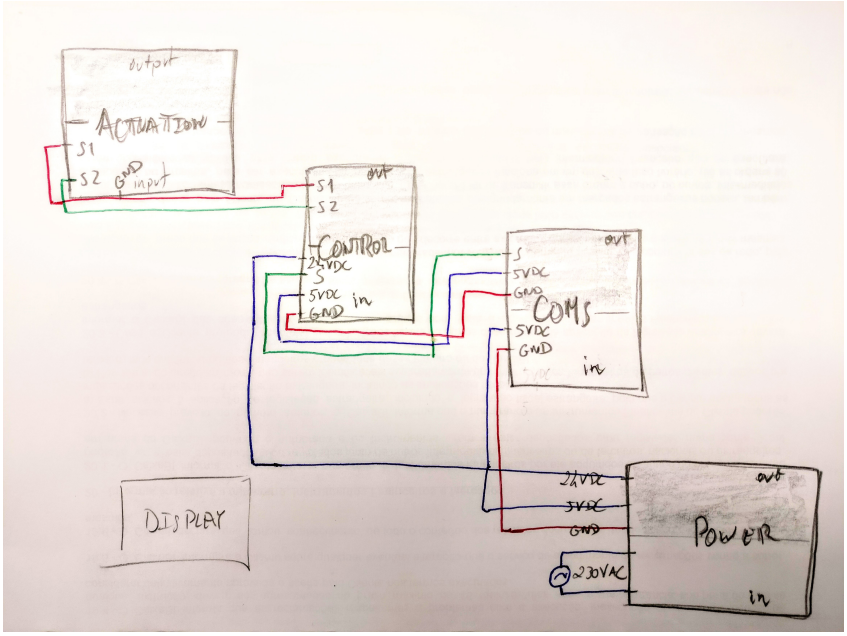

Figure 8: Subsystems' interactions.

3 Software

3.1 Operating System and Remote Access

OS Configuration

The Raspberry Pi is running a light version of Raspbian, Raspbian Jessie Lite[2]. This image does not have desktop environment, so all the configuration has to be done over a terminal. Configuration steps: [3]

1. Install OS image on the RPi's SD card
2. Connect a keyboard, Wi-fi adapter and screen to the RPi
3. Log into the OS with root user (pi)

   User name: *pi*

   Password: *raspberry*
4. Run `sudo raspi-config`

   Change the keyboard layout to *Portuguese* (all default)

   Enable SSH
5. Set network settings, both for wi-fi and Ethernet

[3] https://www.raspberrypi.org/downloads/raspbian/

6. Install `git`:

```
sudo apt-get update
sudo apt-get install git
```

At this point, the RPi is ready to run Python 2 code. All the following configuration can be done remotely.

We can access the RPi mainframe from another computer – preferably Linux or Windows with Putty – over the SSH protocol using the command:

```
ssh pi@RPI_IP_ADDRESS -p RPI_SSH_PORT
```

→ See section 3.3

where `RPI_PI_ADDRESS` is the IP address of the RPi and `RPI_SSH_PORT` is the SSH port allocated to the RPi (default: 20). There should be a SSH port for each of the devices in the local network, see section 3.3.

If the SSH connection was successful, you can manage the Pi. The first step is creating a known directory where to put the code into. Use the following code:

```
mkdir /python/socket_server
```

where you should replace `socket_server` by another name, if your following this documentation for another purpose.

3.1.1 Uploading or updating the code on the Pi

On the Pi, Python code is saved below the folder `python` on the home directory:

```
~/python
```

With individual code packages on the directories below:

```
~/python/socket_server
~/python/code_package_1
~/python/code_package_2
~/python/code_package_3
~/python/...
```

Copying files remotely

It is possible to copy the code remotely with the `scp` command. From your origin computer, run in one line:

```
scp -P RPI_SSH_PORT pi@RPI_IP_ADDRESS:~/python/code_package_x/FILE.py
/your/local/directory/FILE.py
```

! → which copies the `FILE.py` to the specified `/python/code_package_x directory`. If there's already a file there with the same name, it will be overwritten without warning.

From git

Alternatively, you can download code from a git repository[4]. You just need to run the command:

```
git clone https://github.com/MiguelSimao/RPI_Control.git
~/python/socket_server
```

which will copy the files inside `RPI_Control.git` to the `~/python/socket_server` directory.

[4] The main author's git repository (master branch) can be found on: https://github.com/MiguelSimao/RPI_Control.git

3.1.2 Running a script automatically after boot

For example, the socket server should run automatically after the OS boots. To do so, you need to follow two steps:

1. Create a launcher script
2. Add that script to `crontab`

The launcher script can be created in the home directory:

```
nano launcher.sh
```

In this file, you need to write the following code:

```
#!/bin/sh
# launcher.sh

cd /home/pi/python/socket_server
sudo python echo_server.py
```

`nano` is a command-line text editor, `/home/pi/python/socket_server` is the directory of the Python script and `echo_server.py` is the Python script you want to run. You can save and close the file by pressing `Ctrl+X`, followed by `Y` and `return` (Enter).

After that you need to make the file executable. You do so by running the following command:

```
chmod 755 launcher.sh
```

At this stage, you can test the launcher and script by running:

```
sh launcher.sh
```

Please note that if you need to edit the launcher file, you need to turn it back into a writable file using:

```
chmod u+w launcher.sh
```

The final step is writing a `crontab` entry to run this script after booting. Edit the `crontab` configuration by running:

```
sudo crontab -e
```

and add the following line at the end of the file:

```
@reboot sh /home/pi/bbt/launcher.sh &
```

This way, the RPi should be booting as usual and you should be able to `SSH` into it and do whatever management you need to do, while it is also running the server.

3.2 Socket Server

The code running the server on the RPi is on the file `echo_server.py`. This is a multi-thread server that allows several clients to be connected at the same time. The main thread loops listening for connections. The client threads – defined by `clientthread` – reads messages from the clients and translates them into the target GPIO header configuration (pulls pins high or low as needed).

Port: 1080 As is, this socket server works on the TCP channel and the port is the 1080. For the IP address, check section 3.3.

3.3 Network Configuration

The network has to be configured so that devices on the network can find the RPi. For wi-fi connections, the RPi needs to have a static IP address configured so that it can be found in the local network. To do so, follow the steps:

1. Connect any computer to the wi-fi network (ColRobot)
2. Access the router's main page at `192.168.1.1`
   Ask one of the lab's researchers for the credentials.
3. Go to `IP & MAC Binding / Binding Settings`
4. Add your device by MAC set its IP to `192.168.1.199` or lower
   The attributed IPs start at .199 and decrease for each added device (.198, .197, ...)

IP Address

At this point, you can access the RPi in the wi-fi network (not ethernet) using the static IP you set before.

3.3.1 Wired Connection

! To be implemented

If you're trying to connect thought a wired connection, the `RPI_IP` has to be set up manually to an IP in the robot controller's subnet.

3.3.2 Remote Communications

As remote, we mean connecting through the network to which the wi-fi router is connected. The router has a static IP address on the department's network – `ROUTER_IP` – and you can access the router through that IP from any computer connected to any of the Ethernet ports on the lab. If you want to send a command from a remote location (e.g., computer connected to the lab network, outside of the wi-fi network), first you need to connect to the router through `ROUTER_IP`. The router then decides the device it will send a packet to. This is done by port forwarding.

Port Forwarding

By default, you can connect remotely to the RPi by SSH using port 22 and TCP using port 1080. If more computers are on the same network (e.g., other RPi), you need to configure them to have non-default ports, often on the range between 1024 and 65535. To do so, follow these steps:

1. Access the router's main page at `192.168.1.1` or `ROUTER_IP`
   Ask one of the lab's researchers for the credentials.
2. Go to `Forwarding / Virtual Servers`
3. Add new entry by setting the port you want (Service port) and the static IP address you set for the corresponding device previously.