https://www.model-railway-signalling.co.uk/

DCC Signalling Application Download Instructions

Version 1.0 – December 2024

The DCC Signalling application is free to download and use.

It will work on Linux Platforms and Windows platforms (I have tested both). In theory it should also work on MacOS as the software uses the standard cross-platform graphics libraries, but the last time I tried it, the signal and point buttons weren't rendering properly on the display.

Running on a non-Raspberry Pi platform without the SPROG-DCC programming controller, the application won't send out DCC commands and you won't be able to connect in external track sensors to drive the track occupancy, but you will be able to design, configure and test signalling schemes for your layout (or even simulate real-world signalling schemes as others have done).

This guide provides generic instructions on how to download and run-up the system. If you hit problems or need advice then please reach out to us at DCC Model Railway Signalling and we'll do our best to help. Feedback on these instructions is also welcome (so we can keep them up to date).

For information on how to use the signalling application, refer to the separate application quickstart guide, available for download from https://www.model-railway-signalling.co.uk/.

Table of Contents

Pre-requisites

Before installing the application you will need to install ‘Python’ on your computer, specifically Python 3 (any version from Python 3.8 onward should work)

For Linux-based platforms

The chances are that Python 3 is already installed. To test this, type at the command prompt:

```
$ python3 --version
Python 3.11.2
```

If Python3 is installed, this will report the version (as above). If not then python3 is easily installed by typing the following commands (note the required Python version is specified):

```
$ sudo apt-get update
$ sudo apt-get install python3.11
```

Alternatively you may be able to use the specific operating system’s package manager (similar to the Windows Add/Remove programs utility) to install Python 3.

For Windows Platforms

Python 3 can be downloaded and installed from the Windows app store (make sure you choose a version of python 3.8 or later).

- If it gives you the option to install 'pip' then select this.
- Also select 'add to python path' if it gives you the option

To check the installation, first open the 'command prompt' application (in the windows search bar start typing the above and the it should come up in the search). Then, in the window type:

```
> python3 --version
Python 3.11.2
```

This will report the installed version (similar to the Linux installation)

Installing and running the application

The instructions are broadly the same for both Windows and Linux platforms. All commands should be entered in the Terminal window (Linux) or Command Prompt window (Windows).

Installing the latest version of the application

If installing the application for the first time, first try:

```
$ pip3 install model-railway-signals
```

The above should work for most python installations, but if this errors (command-line version of 'pip' is probably not installed/enabled) then try:

```
$ python3 -m pip install model-railway-signals
```

When installing the application on Linux platforms with later versions of python, the installation may fail with a reported error of:

```
error: externally-managed-environment
```

To overcome this, add the '*--break-system-packages*' argument to the command :

```
$ pip3 install --break-system-packages model-railway-signals  OR

$ python3 -m pip install --break-system-packages model-railway-signals
```

Upgrading the application to the latest version

Upgrade of the application is achieved by adding the '*--upgrade*' argument to whichever of the above commands worked for the initial installation:

```
$ pip3 install --upgrade model-railway-signals    OR

$ python3 -m pip install --upgrade model-railway-signals
```

Removing the application:

Removing the application uses similar commands to install and upgrade. Again, use whichever command worked for the initial install:

```
$ pip3 uninstall model-railway-signals     OR

$ python3 -m pip uninstall model-railway-signals
```

Running the application:

The application can be started from the Terminal window / Command Prompt window:

```
$ python3 -m model_railway_signals
```

Note the use of underscores in the above command (rather than hyphens).

Additional notes and instructions

The application has minimum dependencies over and above the 'standard' python installation.

The two mandatory packages, '*pyserial*' and '*paho-mqtt*', should both be automatically installed when the application is installed. If for some reason this doesn't happen (I've been made aware of one instance on a Windows platform where it did) then these packages can be installed separately (prior to installing the model-railway-signals package):

```
$ pip3 install paho-mqtt   OR  python3 -m pip install paho-mqtt

$ pip3 install pyserial    OR  python3 -m pip install  pyserial
```

If you want to use Block Instruments with full sound enabled (bell rings and telegraph key sounds) then you will also need to install the '*simpleaudio*' package. If '*simpleaudio*' is not installed then the application will still function correctly (just without sound).

```
$ pip install simpleaudio   OR   python3 -m pip install paho-mqtt
```

On linux platforms, if you are running on a later version of Python you may also need to install '*libasound2*' before the '*simpleaudio*' installation will work.

```
$ sudo apt-get install libasound2-dev
```

On Windows Platforms, the '*simpleaudio*' package has a dependency on Microsoft Visual C++ 14.0 or greater (so you will need to ensure Visual Studio 2015 is installed first).

Resolving Wayland performance issues

When running the application on Linux installations that use 'Wayland' as the graphics back-end rather than X11 (e.g. the recent Debian Bookworm release for the Raspberry Pi), you may well find the user-interface performance is terrible. From what I can gather, this appears to be a wider issue with Tkinter (the 'standard' Python graphics library used by many applications including this one).

There are a number of active discussions on the internet (some of which are quite heated) as to whether Wayland is currently fit for purpose, but no specific solutions for Tkinter as yet.

In the meantime, the only workaround I've found is to switch the graphics back-end from Wayland back to X11 - performance of the Signalling application is immediately restored.

To change the back-end on a Raspberry Pi:

- Run "sudo raspi-config" from a Terminal window
- Select 'Advanced Options' then 'Wayland'
- Select X11 and apply.