Frischen Documentation

Stefan Bethke

Mar 20, 2021

Contents:

The Frischen project contains components for building railway signal tower simulations. Components are implemented in Python and JavaScript, and use the MQTT protocol to interact with each other.

The Frischen project is open source software under the BSD license. You can find the code as well as the documentation at github.com/vvmev/frischen.

CHAPTER 1

Developing Frischen

1.1 Requirements

Frischen uses a decently large technology stack. For a complete development environment, you need:

- Docker and Docker Compose:

  To have an MQTT broker available locally, Frischen uses Docker and Docker Compose to run Mosquitto.

- Python 3.7 and pipenv

  Most of the tower control software as well as the utilities are written in Python. Make sure you install version 3.7 or newer, as Frischen uses a number of features not available in older versions.

  After installing Python 3.7, install pipenv.

- A code editor or IDE with support for Python and JavaScript

  While a standard text editor is sufficient, a code editor or an integrated development environment can be quite helpful. These work quite decently:

  - Atom (open source, free)
  - IntelliJ PyCharm and IntelliJ WebStorm (free for personal use community edition, or commercial)
  - Microsoft VisualStudio Code (free use)

1.2 Setting Up a Development Environment

1.2.1 Starting the MQTT Broker

In the main directory, run:

```
$ docker-compose up -d
Creating network "frischen_default" with the default driver
Pulling broker (eclipse-mosquitto:)...
latest: Pulling from library/eclipse-mosquitto
4fe2ade4980c: Pull complete
9ed5ccc7b14f: Pull complete
848281416363: Pull complete
Digest: sha256:b96c387e1ea3c7531f7fe45447462e60e75b76b9d3d9e26c50a46283353953e6
Status: Downloaded newer image for eclipse-mosquitto:latest
Creating frischen_broker_1 ... done
```

This will start the mosquitto docker container. The container listens on TCP port 1883 for MQTT and on port 9001 for HTTP/WebSocket.

The built-in web server serves files from the `web/` directory, so you can go to http://localhost:9001/ to see the available HTML/JavaScript apps.

1.2.2 Setting Up The Python Environment

Using a Python virtual environment ensures that you have the right versions of Python packages needed. First, make sure you have `pipenv` installed:

```
$ python3 -m pip install pipenv
```

Then, you can use `pipenv` to create a fresh environment and enter it:

```
$ pipenv install --dev
Pipfile.lock (3ea1da) out of date, updating to (4895ab)...
Locking [dev-packages] dependencies...
✓ Success!
Locking [packages] dependencies...
✓ Success!
Updated Pipfile.lock (3ea1da)!
Installing dependencies from Pipfile.lock (3ea1da)...
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
```

Finally, you can activate the freshly installed environment:

```
$ pipenv shell
Launching subshell in virtual environment...
 . /Users/me/.local/share/virtualenvs/frischen-0szepg5g/bin/activate
$  . /Users/me/.local/share/virtualenvs/frischen-0szepg5g/bin/activate
```

You can now run the Python scripts in the top level of the project directory.

1.3 Working With Python Code

1.3.1 Unit Tests And Test Coverage

The Python modules have unit tests. To run the tests, use the standard Python `unittest` package. To check test coverage, the `coverage` package is installed:

```
$ coverage run -m unittest && coverage report && coverage html
```

This will create coverage data and a report in `coverage/`.

1.4 Documentation

1.4.1 Building

The project uses sphinx to generate HTML pages, including API docs for the Python and JavaScript modules. The documentation sources are in `docs/`. To rebuild the HTML pages, run:

```
$ cd docs
$ make html
```

The output is available in `docs/build/html` and can be viewed in a browser.

CHAPTER 2

Frischens Use of MQTT

The Frischen project uses MQTT to couple various functions together. This document aims to define how topics and values are used between programs.

2.1 Topic Structure

In MQTT, the topic specifies what function a message addresses. Topics are strings (encoded in UTF-8). A topic consists of one or more topic levels, separated by a slash. Topics are case-sensitive.

All topics start with the `frischen` prefix. Programs might use other topics for functions not related to the Frischen system, for example, to control the room lights in a signal tower.

The second level determines the operations area for the topics. This is typically the name of the signal tower from which this area is controlled.

The third level determines the class of equipment: either controls and indicators inside the signal tower, or track-side and other outside plant equipment, like turnouts, signals, etc. Inside controls and indicators have a topic level of `panel`, track-side equipment uses `trackside`.

The fourth level determines the type of element addressed, for example `switch` or `turnout`.

The fifth level is the designation of the element, for example `W3` for turnout W3, or `N3` for the signal N3.

An optional sixth level allows distinguishing between commands from the signal tower to the element, and status reports from the element to the signal tower.

2.1.1 Examples

- `frischen/etal/panel/turnout/W13`: position, locked state and occupied state of turnout W13 as determined by the signal tower and displayed on the operating panel.
- `frischen/etal/panel/button/W13`: the button as part of the panel module. Pressing or releasing the button will send a message.

- `frischen/etal/trackside/turnout/W13/command`: Commands from the signal tower to the turnout: change position.
- `frischen/etal/trackside/turnout/W13/status`: Reports from the turnout to the signal tower: current position, end position reached. The turnout sends a status report every time the status changes.

CHAPTER 3

frischen

3.1 frischen package

3.1.1 Submodules

3.1.1.1 frischen.broker module

3.1.1.2 frischen.spdrl20 module

CHAPTER 4

Indices and tables

- genindex
- modindex
- search