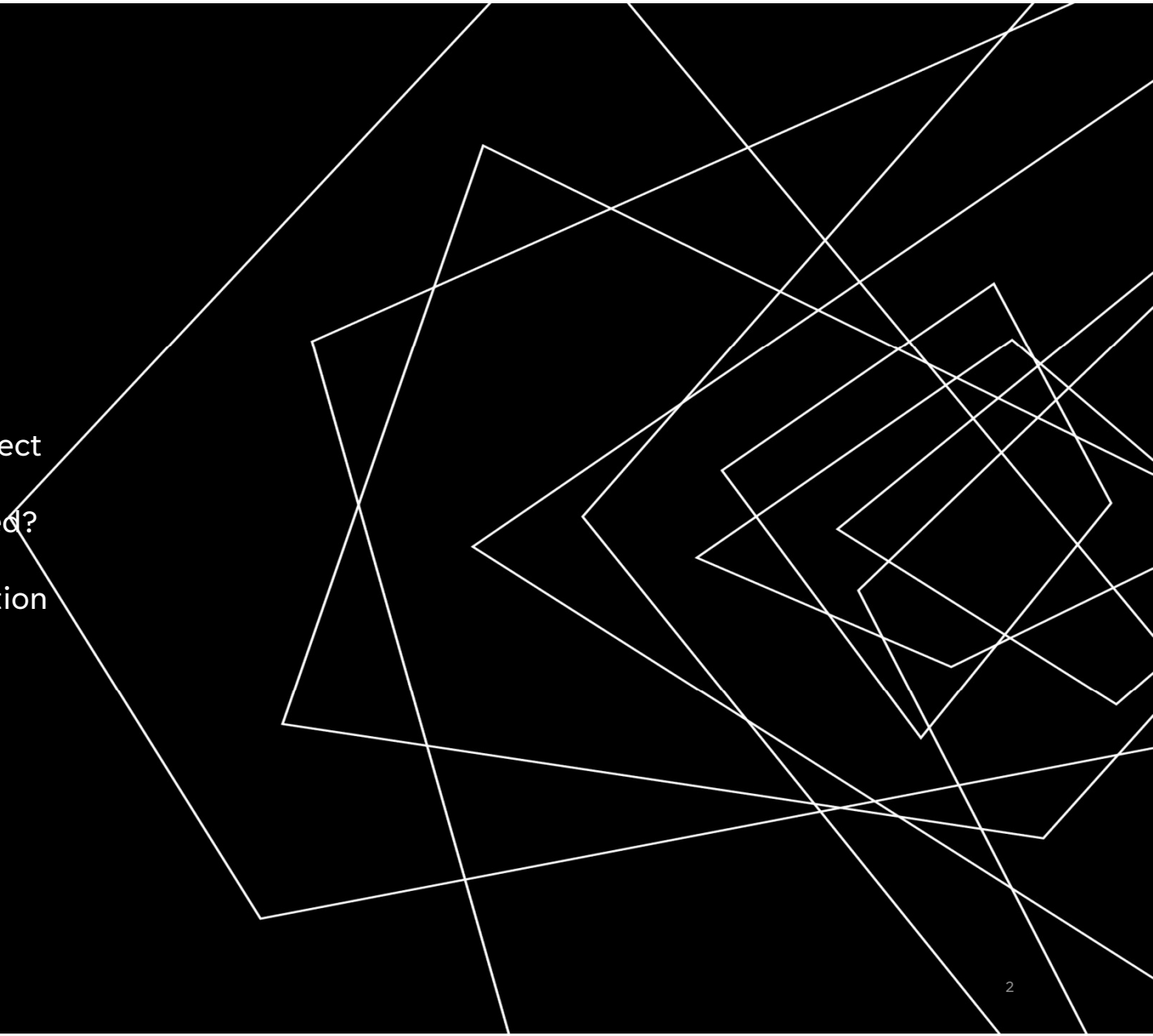
ENERGY & ENVIRONMENTAL MONITORING SYSTEM

CONTENT



PURPOSE AND DESIGN PRINCIPLES OF PROJECT

- Develop a good knowledgebase for a home to monitor energy usage and environmental conditions
- Allow the user to find energy leaks and optimise energy consumption
- Do not tie the design to existing home automation platforms
- Use open platforms and interfaces as much as possible (MQTT, mySQL, Grafana)
- Experiment and learn about relevant technologies
- Create a framework which is extensible and contains re-usable software components
- Combine the power of open source with programming components (in C, Python, shell scipt)

WHY WERE SHELLY COMPONENTS SELECTED?

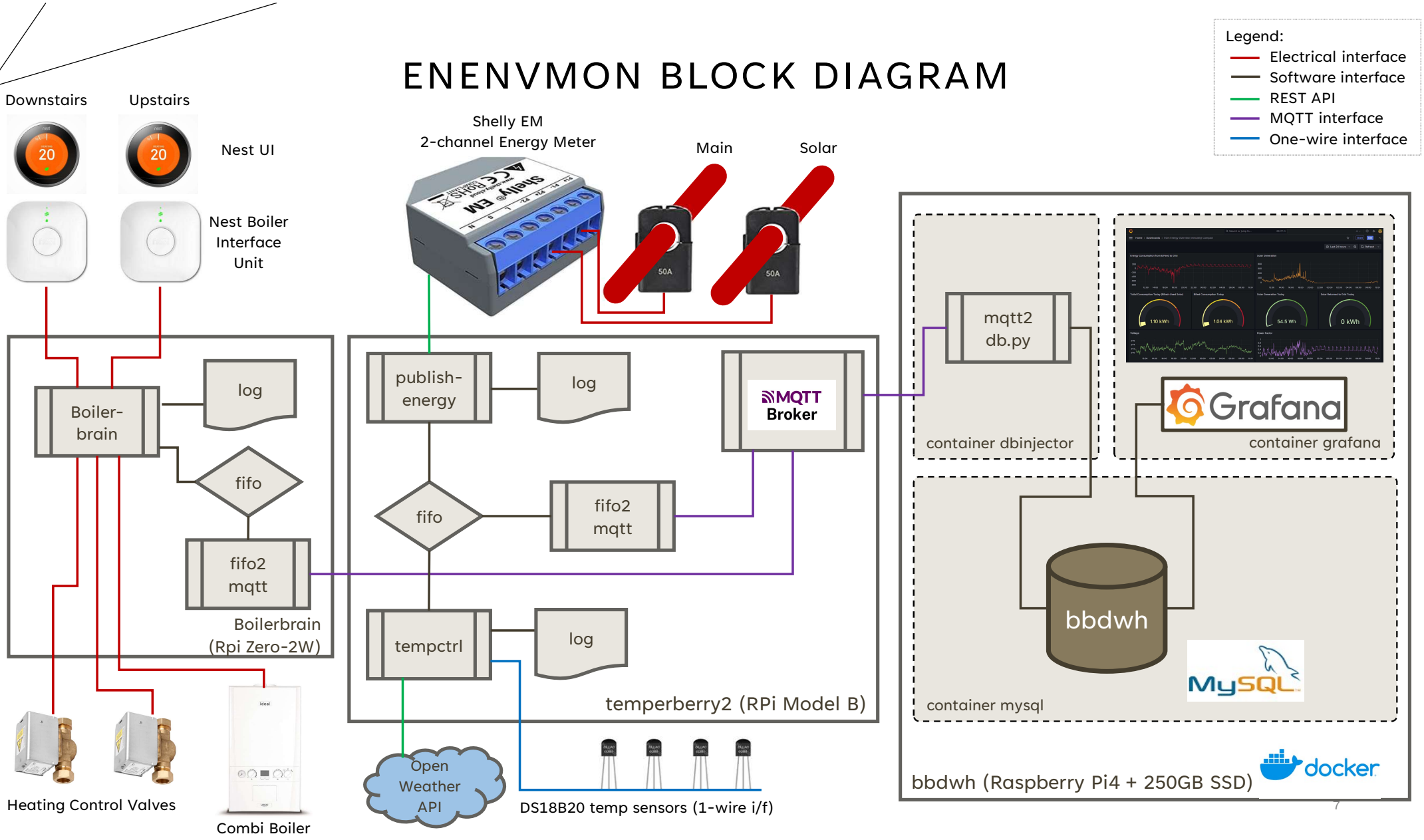
WHY WERE SHELLY COMPONENTS SELECTED

- Versatile devices
- Proven reliability
- Experience from previous projects
- Open interfaces, can operate outside of closed ecosystem
- The Shelly cloud infrastructure (though not utilised in the project) provides a secondary access channel
- Multiple interface options (REST API, MQTT*)
- Project backlog contains expansion plans with Shelly 1PM devices and/or refactoring boilerbrain component with Shelly 2PM/1/i4 devices.

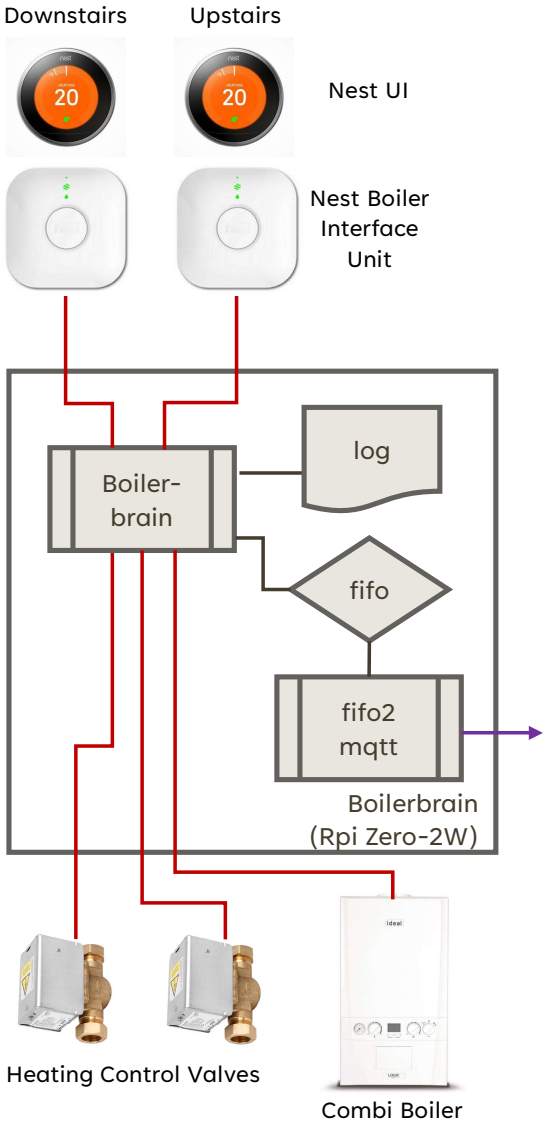
**Note: MQTT could not be utilised in this project, as it would have made the device inaccessible via Shelly Cloud*

BLOCK DIAGRAM AND OPERATION

ENENVMON BLOCK DIAGRAM



THE BOILERBRAIN BLOCK



Legend:
- Electrical interface
- Software interface
- REST API
- MQTT interface
- One-wire interface

The home has a single combi boiler, but the central heating system has two zones, one for downstairs and one for upstairs. Both zones are independently controlled by off-the-shelf Google Nest Heating Control units.

The Boilerbrain component is sitting between the Nest Boiler interfaces and the zone control valves. It reads the status of the Nest controllers and logs it and transparently controls the zone control valves.

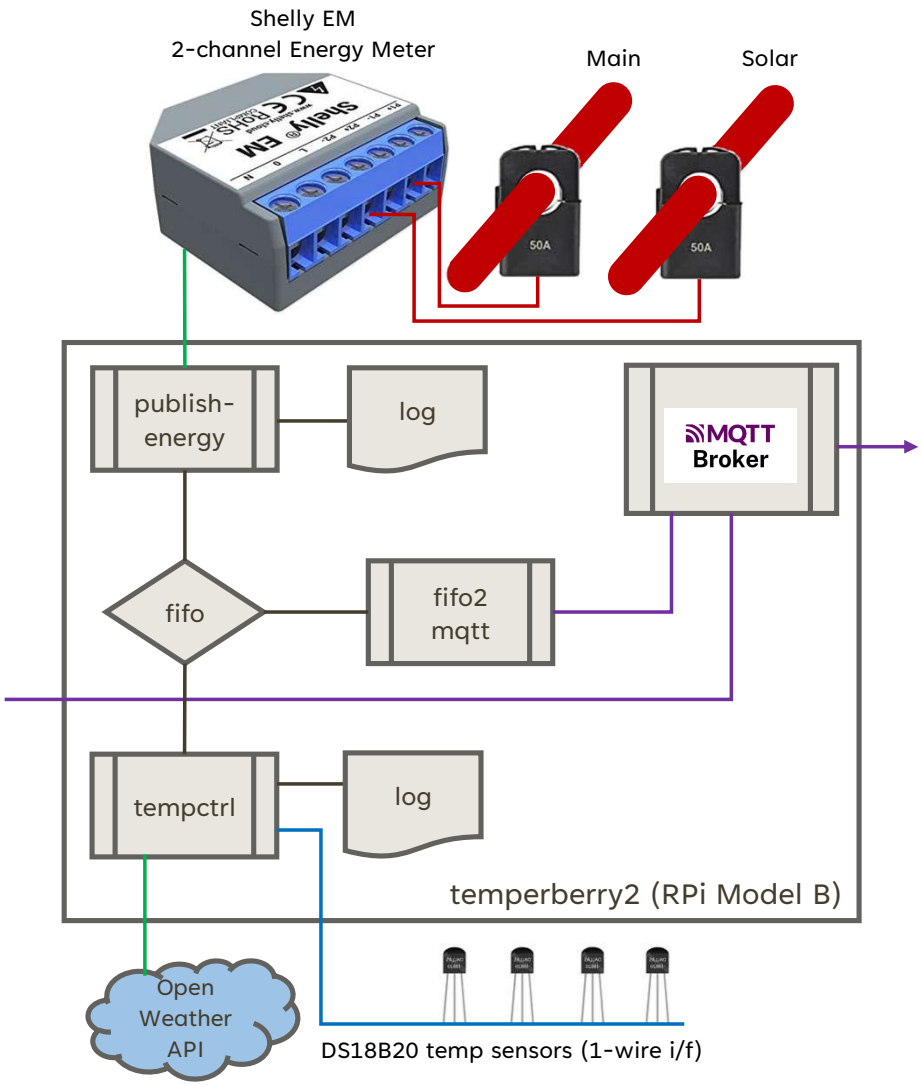
It also provides a combined ignition signal for the boiler in case there is heat demand from any of the zones. (As this simple OR logic was needed anyway for a 2-zone system and not provided by Nest, it was the initial trigger to develop a box sitting between the controllers, the zone valves and the boiler.

It is a Raspberry Pi Zero-2W hardware, running Raspbian (Linux) operating system. The controller, valve and boiler interfaces are relay-based ones*.

Boilerbrain component is a custom C program:
- Reads the status of Zone nest controllers
- Controls the zone valves
- Provides the ignition signal for the combi boiler
- Logs the heating duty cycle and the number of ignitions hourly
- Sends the above information to a FIFO (named pipe) for transferring to MQTT component

FIFO2MQTT component is a reusable C program (utilising paho-mqtt library)
- Takes MQTT topic and payload information from Linux FIFO and
- Publishes the topic to an MQTT broker

THE TEMPERBERRY BLOCK



Legend:
- Electrical interface
- Software interface
- REST API
- MQTT interface
- One-wire interface

The Temperberry block is based on a Raspberry Pi Model B and running Raspbian Linux.

Publishenergy component is a bash shell script, triggered from cron every minute:

- Reads both channels of the Shelly EM power meter: one for incoming/outgoing mains from/to the grid and one for generated solar power.
- Writes a log
- Also writes the data every minute to FIFO for MQTT publishing

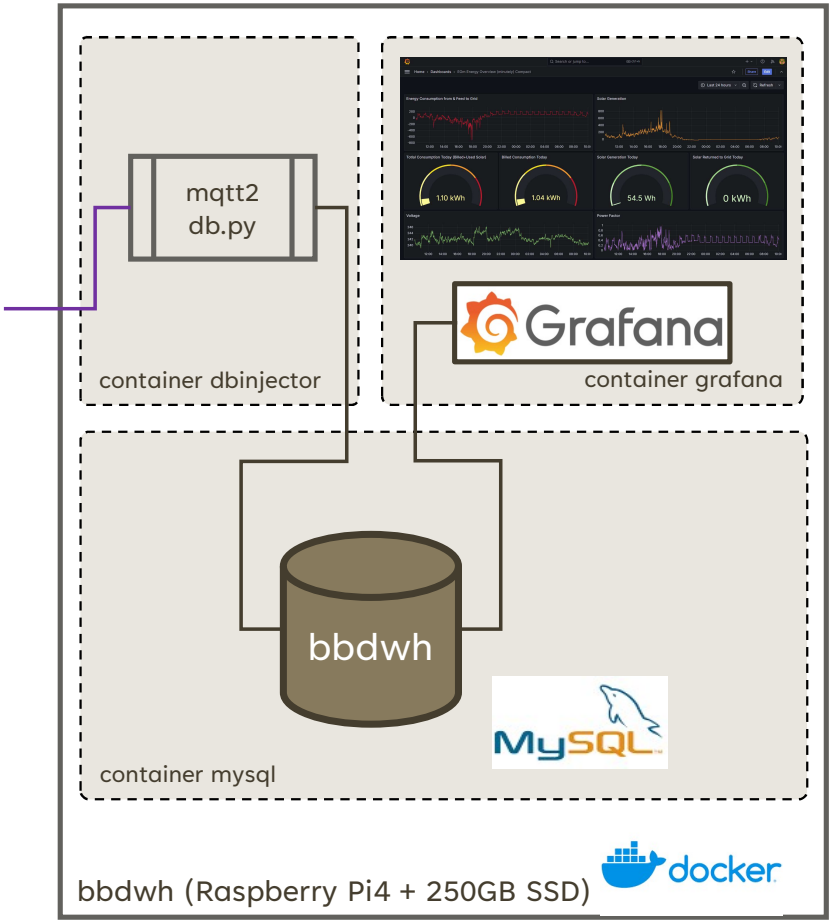
Tempctrl component is collecting environmental information. It is also triggered from cron, once every 15 minutes.

- Reads a number of 1-wire DS18B20 temperature sensors across the home. (For this purpose, the prewired, but unused analogue telephone wiring is used)
- Reads outdoor weather information from OpenWeather's public API
- Writes the environmental information to log files
- Also writes that data to FIFO for MQTT publishing

FIFO2MQTT component is the same as described at the Boilerbrain block

THE DATABASE AND VISUALISATION BLOCK

Legend:
- Electrical interface
- Software interface
- REST API
- MQTT interface
- One-wire interface



The BBDWH block is based on a Raspberry Pi 4 extended with a 250 GB SSD, as it requires more memory and computing power. On top of Raspbian Linux it runs docker container environment, which allows easy configuration and maintenance of the components, without adversely affecting each other.

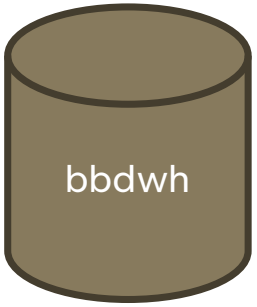
MySQL container is a standard MySQL 9.1.0 docker container:
- The necessary tables created for holding heating, environmental and energy data
- The database is written by Dbinjector and read by the Grafana module

Grafana analytics and visualisation platform is running in its own container:
- Provides dashboards for the user for short and long-term data visualisation

Dbinjector has received its own container. It is a Python scipt:
- Subscribes to the relevant environmental, heating and energy MQTT topics at the broker.
- Writes the relevant tables of MySQL database.

It is an excellent example of resolving compatibility issues. MySQL 9.1.0 does not allow username/password authentication for applications; one must use certificates. The python available on Raspbian-bullseye had only mysql connector v2.9.9, but a much later version was needed for certificate-based authentication. Therefore, the Dbinjector container is a minimal Debian container using a later version of Python3.

DATABASE TABLES



The followings are the tables created in BBDWH MySQL database:

```
show create table EnviroCond;
| EnviroCond | CREATE TABLE `EnviroCond` (
  `TimeStamp` timestamp NOT NULL,
  `TempLounge` float DEFAULT NULL,
  `TempHall` float DEFAULT NULL,
  `TempBedroom` float DEFAULT NULL,
  `TempDatactr` float DEFAULT NULL,
  `TempExt` float DEFAULT NULL,
  `Cloud` tinyint DEFAULT NULL,
  `Rain` float DEFAULT NULL,
  `Windspeed` float DEFAULT NULL,
  `Winddeg` smallint DEFAULT NULL,
  `Pressure` smallint DEFAULT NULL,
  `Humidity` tinyint DEFAULT NULL,
  `Visibility` smallint DEFAULT NULL,
  `Summary` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`TimeStamp`),
  UNIQUE KEY `TimeStamp` (`TimeStamp`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci |
```

```
show create table BoilerStats;
| BoilerStats | CREATE TABLE `BoilerStats` (
  `TimeStamp` timestamp NOT NULL,
  `Periodctr` smallint DEFAULT NULL,
  `DownOnperiod` smallint DEFAULT NULL,
  `DownOnctr` tinyint DEFAULT NULL,
  `DownOnpercentage` float DEFAULT NULL,
  `UpOnperiod` smallint DEFAULT NULL,
  `UpOnctr` tinyint DEFAULT NULL,
  `UpOnpercentage` float DEFAULT NULL,
  PRIMARY KEY (`TimeStamp`),
  UNIQUE KEY `TimeStamp` (`TimeStamp`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci |
```

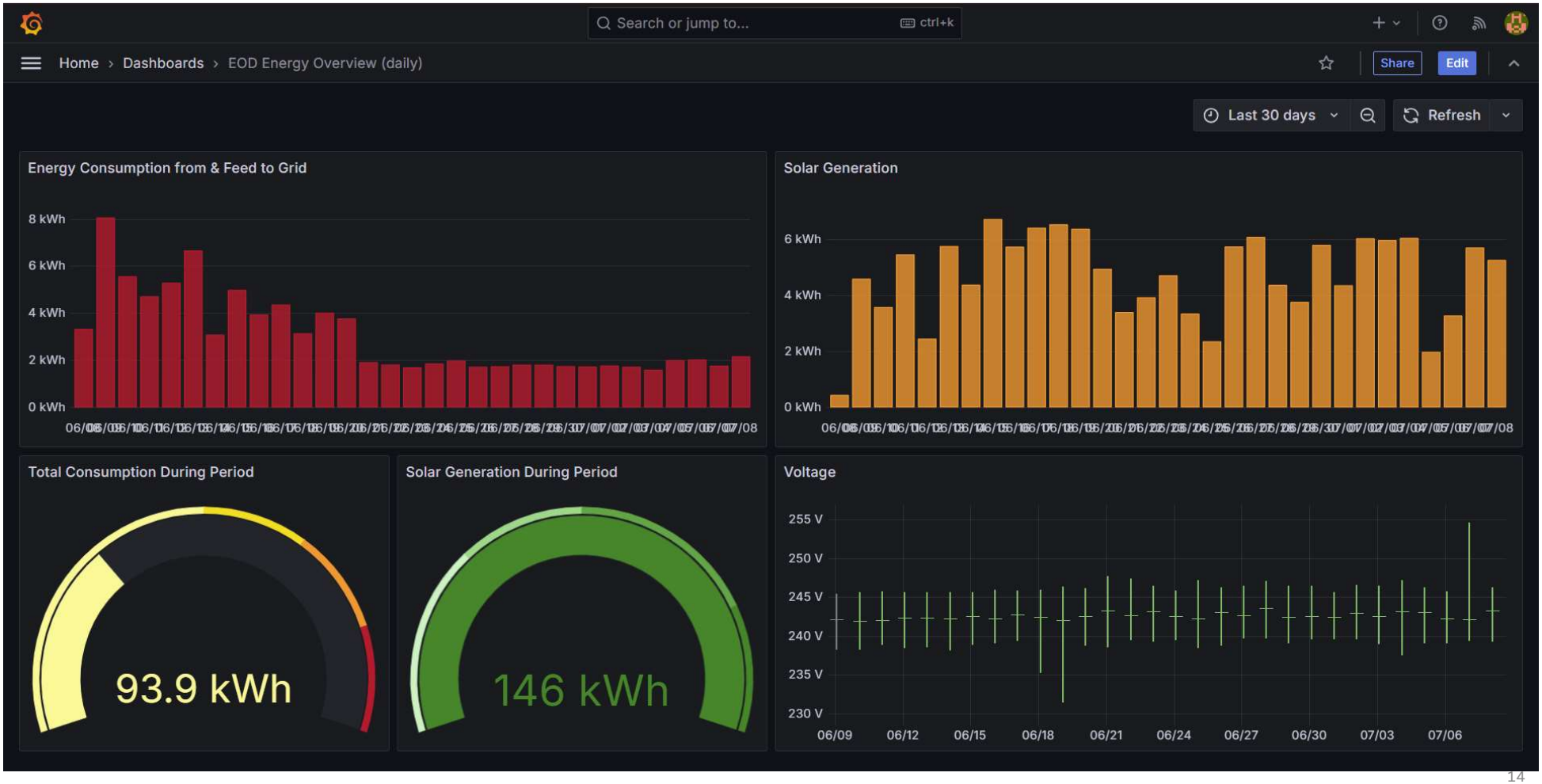
```
show create table EnergyMain;
| EnergyMain | CREATE TABLE `EnergyMain` (
  `TimeStamp` timestamp NOT NULL,
  `Power` float DEFAULT NULL,
  `Reactivepower` float DEFAULT NULL,
  `Powerfactor` float DEFAULT NULL,
  `Voltage` float DEFAULT NULL,
  `Total` double DEFAULT NULL,
  `Totalreturned` double DEFAULT NULL,
  `Isvalid` tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`TimeStamp`),
  UNIQUE KEY `TimeStamp` (`TimeStamp`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci |
```

```
show create table EnergySolar;
| EnergySolar | CREATE TABLE `EnergySolar` (
  `TimeStamp` timestamp NOT NULL,
  `Power` float DEFAULT NULL,
  `Reactivepower` float DEFAULT NULL,
  `Powerfactor` float DEFAULT NULL,
  `Voltage` float DEFAULT NULL,
  `Total` double DEFAULT NULL,
  `Totalreturned` double DEFAULT NULL,
  `Isvalid` tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`TimeStamp`),
  UNIQUE KEY `TimeStamp` (`TimeStamp`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci |
```

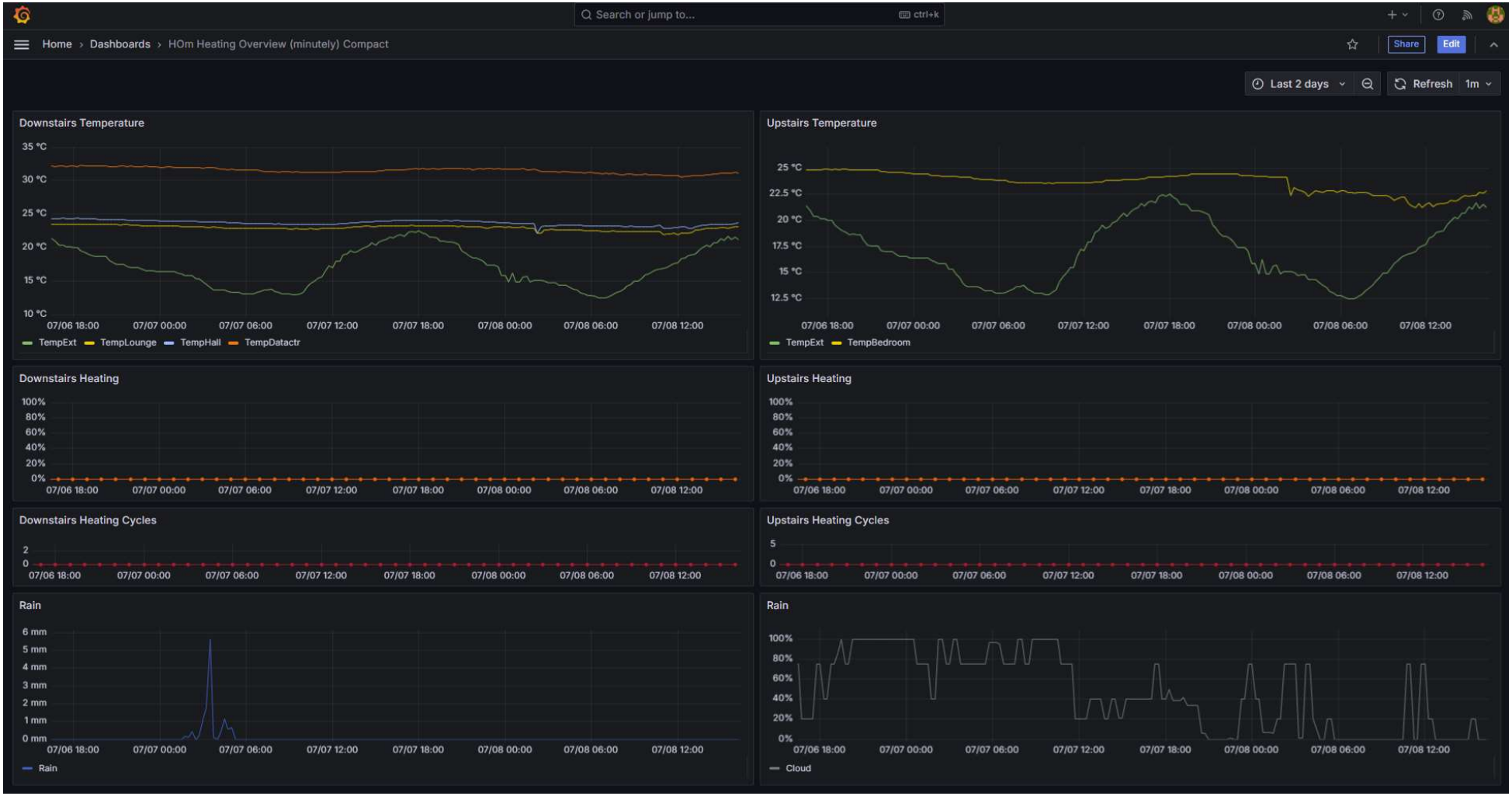
GRAFANA WEB-UI SCREENSHOTS

ENERGY DASHBOARD (CURRENT - 1 MINUTE RESOLUTION)

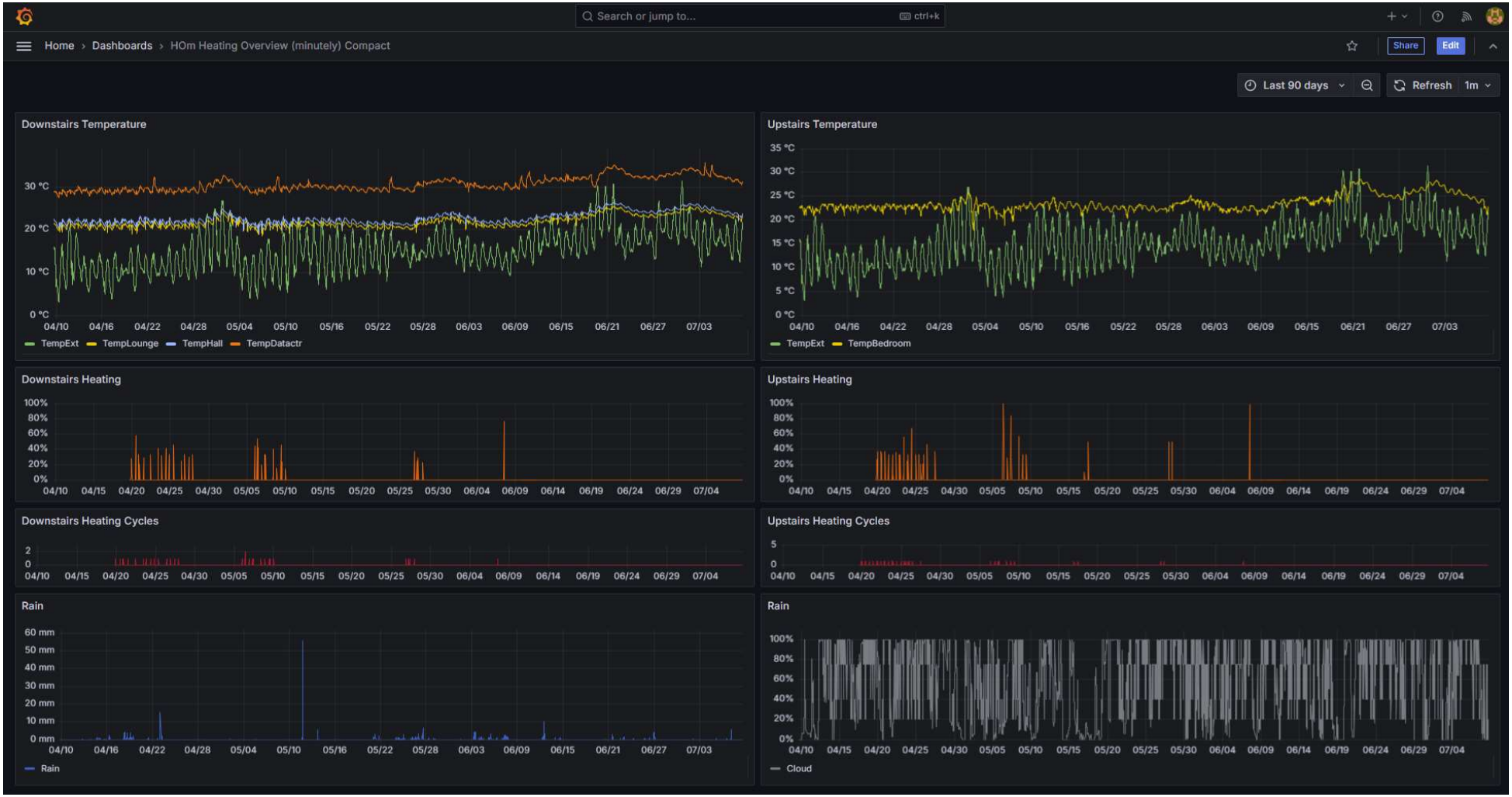
ENERGY DASHBOARD (DAILY PERFORMANCE)



ENVIRONMENTAL CONDITIONS (CURRENT)

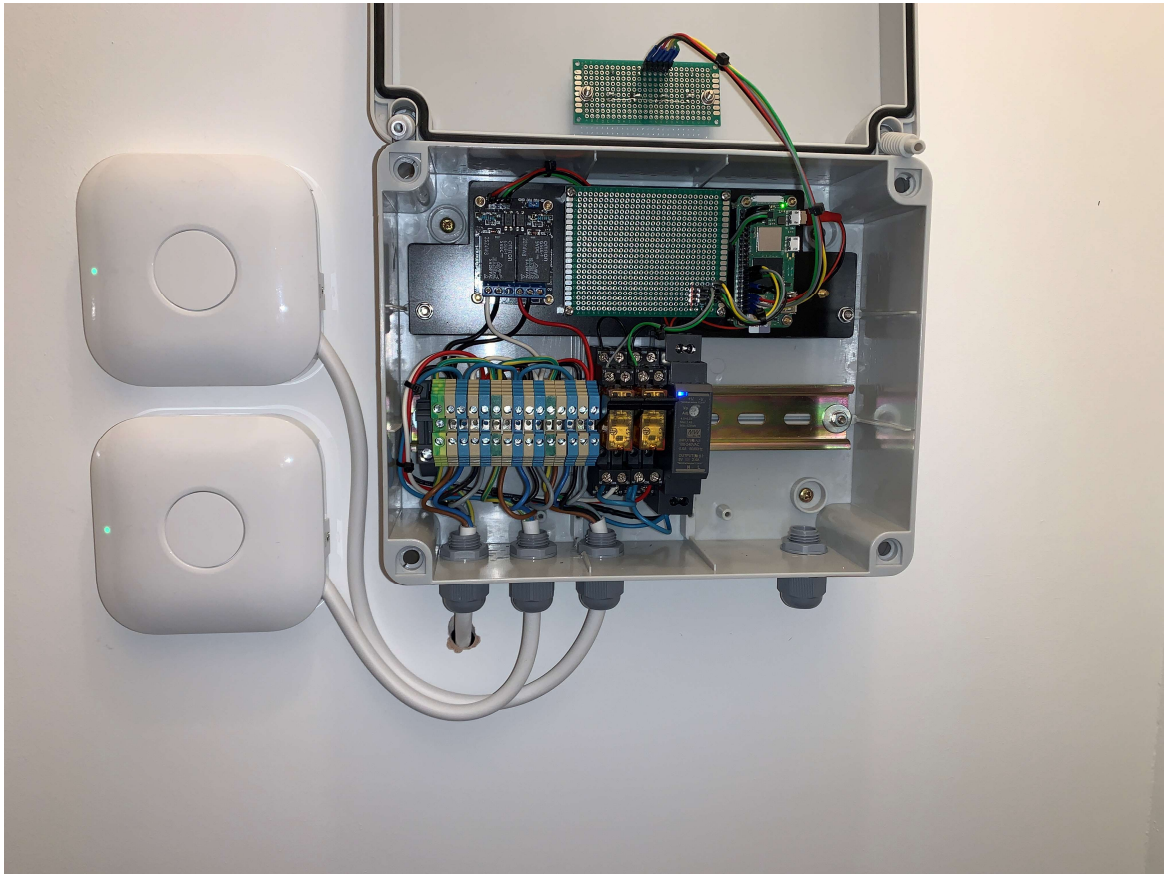


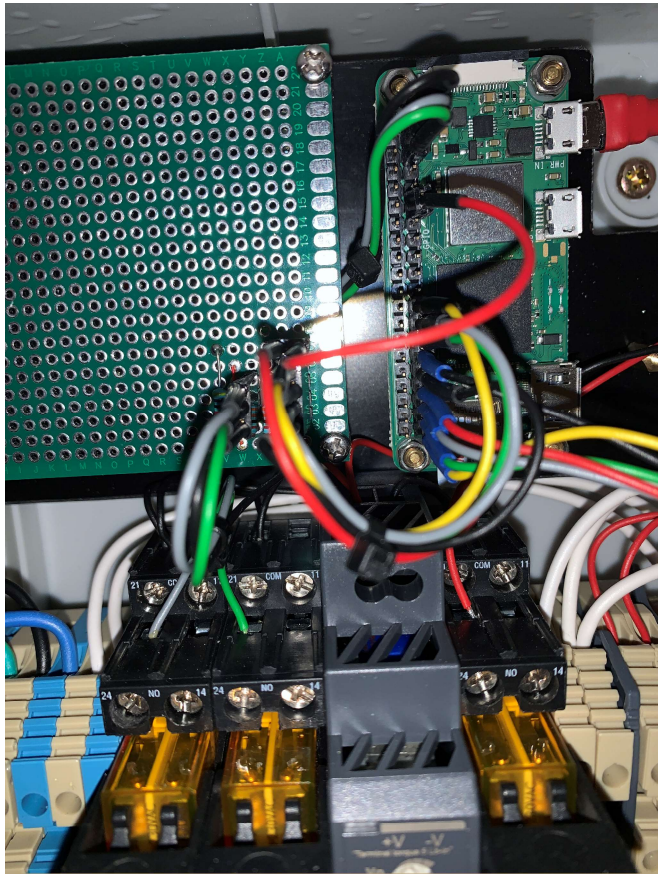
ENVIRONMENTAL CONDITIONS (LONG-TERM OVERVIEW)



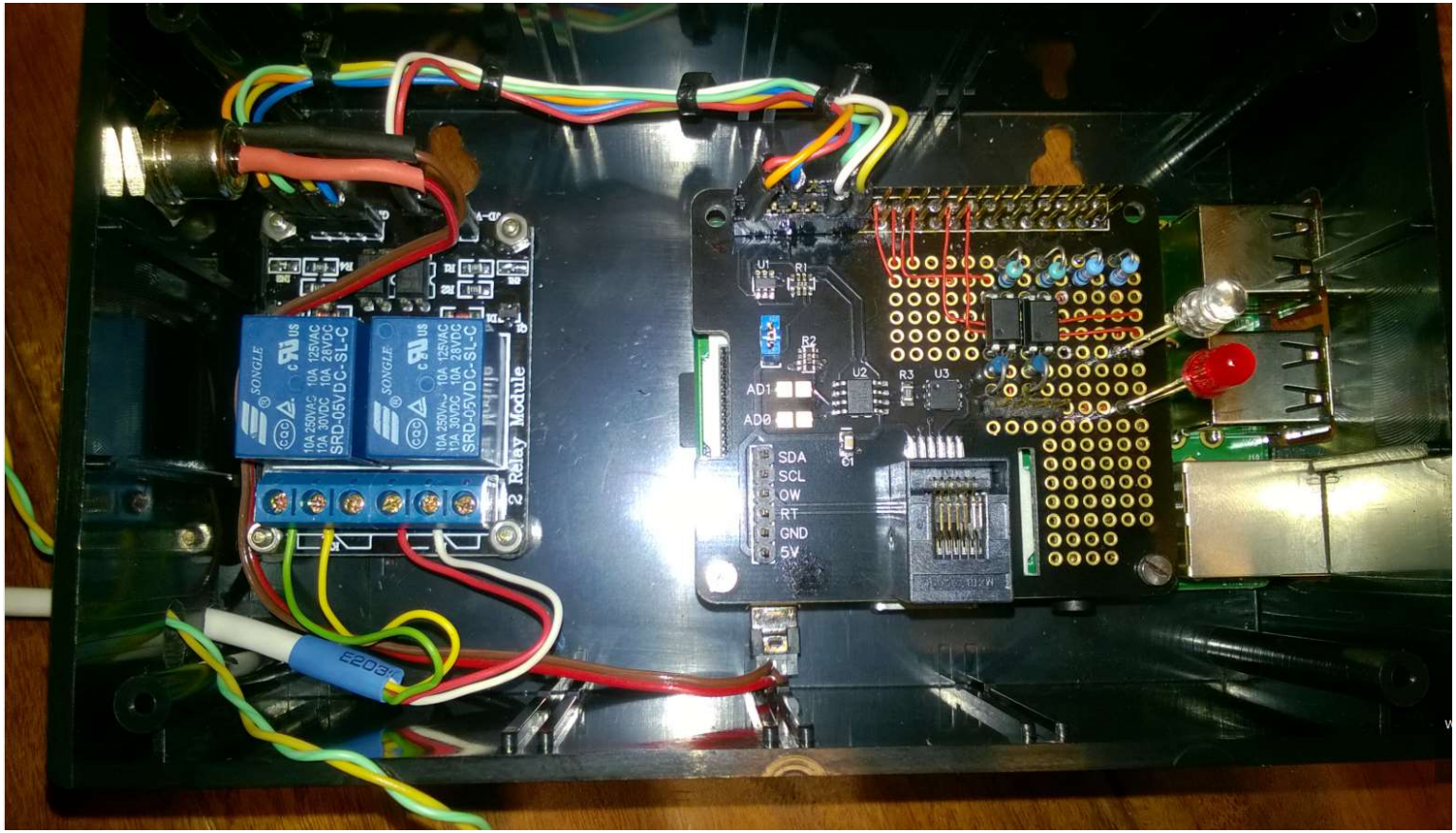
PROJECT PHOTO AND CODE LIBRARY

BOILERBRAIN MODULE





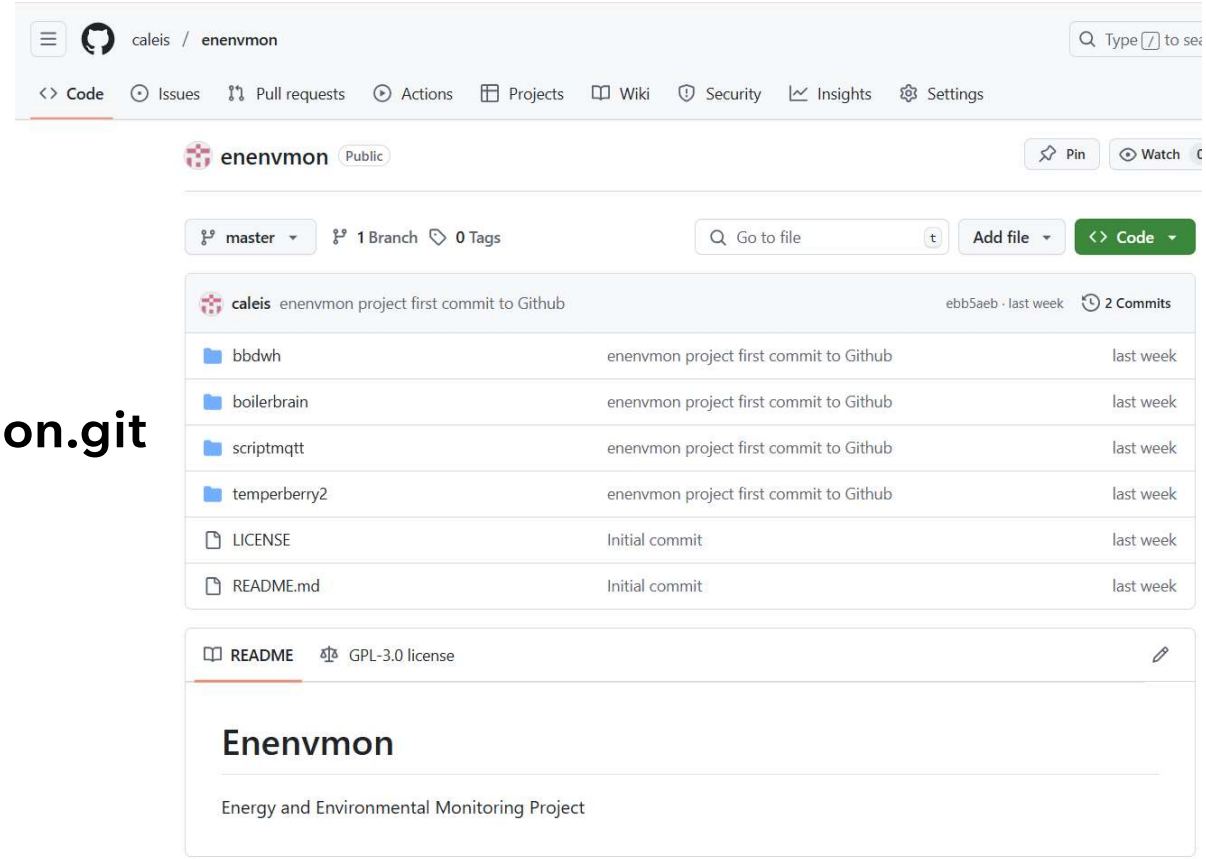
TEMPERBERRY MODULE



PROJECT CODE REPOSITORY

Get it from GitHub:

https://github.com/caleis/enenvmon.git



DEVELOPMENT BACKLOG

DEVELOPMENT BACKLOG

Current backlog contains the following items

- **Solar control (Energy saving feature)**

  EnEnvMon would allow the user to start energy-hungry appliances, when solar power is available and the weather forecast is favourable for the operating time of the appliance (e.g. starting a washing cycle).

  As exported energy is much cheaper than purchased one, it can result in monetary savings.

- **Boilerbrain to use Shelly input and relay devices**

  Boilerbrain was implemented with legacy components (relays and drivers). If someone is trying to re-create the project it would be much simpler using:

  - Shelly 2PM Gen3/4 for zone-control,
  - Shelly 1 Gen3/4 for ignition control and
  - Shelly i4 Gen3 for input state sensing.

  Boilerbrain C-program may be replaced with a simpler shell script.

- **Opentherm interface**

  Since both the boiler and the Nest thermostat are OpenTherm compatible control and status would be much more accurate. Needs the implementation of an OpenTherm summarising gateway

  OpenTherm interface (OpenTherm to serial bi-directional converter) is already implemented

THANK YOU

Zoltan Fekete

GitHub: caleis