Wintergarten Docs

Docs for the https://wintergarten.mitkov-systems.de/ project.

- Overview

Overview

The essence of the project is two-way communication over MQTT for the control of a winter garden device, which controls things like temperature, fans, and others. Note that all communication over MQTT is encrypted with AES-256-ctr. In this doc, messages are shown decrypted.

The communication is actually done through an HMI (human-machine interface) device, which relays the commands to the actual device.

HMI

The HMI's interface is replicated in this web app as well, so you can control it from here too. Also, if someone is using the actual HMI and has the web app opened, both UI's are synchronized.

In this project the replicated HMI has the following options (which the user can change):

- **min. temp** - when mode is auto, you can change the min. temp through the plus and minus buttons
- **fan** - this is the setting of the main fan, and you can change it through the plus and minus buttons if the mode is set to manual
- **mode** - you can choose between '**auto**' and '**manual**'
- **aktive zuluft** - I believe this controls the secondary fan with options from 0 to 6 (need to confirm this with Marto)
- **on/off** - when off, the user can't change the other options of the HMI

Aktive Zuluft Timeout

Aktive zuluft has a timeout (in seconds) specified by **zuluftRT** in the payload. When this number is bigger than 0, it starts to countdown the seconds after which aktive zuluft is switched off.

Mode Timeout

There is also a timeout when the mode is **manual,** after which it switches to **auto**. Specified by **stateRT** in the payload. However, this is not implemented in the app atm.

Except for the **Dashboard,** there are two other pages: **Settings** and **Current**

Values

Settings

Allows you to update the following settings:

Max temp
Min pwm
Max pwm
Unique ID - this updates in unique ID of the device associated with that user in the web app. Used to subscribe to the device specific topics on MQTT
zulutfT
Periphery

Current Values

Allows you to change the following:

Temp in
Temp out
Fan 1 pwm
Fan 2 PWM

Full range of options along with some helpful comments you can see in the 'state.js' file in the project's files - look at the beginning.

MQTT topics

The app listens on two MQTT topics: **/v3/rt/<device-unique-id>/settings** and **/v3/rt/<device-unique-id>/payload**

On the payload topic, we expect messages similar to this one (note this is requested by the **fetchPayload** command):

[{"unique_id":"D8132A859750","hw_ver":"1.2.3","fw_ver":"2.3.4","bl_ver":"5.6.7","hmi_hw_ver":"8.9.10","hmi_fw_ver":"11.12.13"},{"channel":0,"type":"temp_in","unit":"°C","data":[{"value":28.2}]},{"channel":1,"type":"temp_out","unit":"°C","data":[{"value":27.9}]},{"channel":2,"type":"fan","unit":"%","data":[{"value":50}]},{"channel":3,"type":"fan","unit":"%","data":[{"value":0}]},{"channel":4,"type":"valve","unit":"","data":[{"value":1}]},{"channel":5,"type":"valve","unit":"","data":[{"value":1}]},{"channel":6,"type":"valve","unit":"","data":[{"value":0}]},{"channel":7,"type":"mode","unit":"","data":[{"value":1}]},{"channel":8,"type":"stateRT","unit":"sec","data":[{"value":101}]},{"channel":9,"type":"zuluftRT","unit":"sec","data":[{"value":1}]},{"channel":10,"type":"errors","unit":"","data":[{"value":0}]}]

On the settings topic, we expect messages similar to this one (note this is requested by the **fetchSettings** command):

```
{"periphery":1,"minTemp":28,"maxTemp":33,"minPWM":50,"maxPWM":89,"zuluftT":9,"noise":4,"aktivezuluft":0}
```

There is another topic: **/v3/rt/<device-unique-id>/commands** - here we send commands from the web app to the HMI. These look like this:

```
{"token":"bBm9FMLG","status":"pending","settings":{"minTemp":29},"fetchPayload":{"values":1},"fetchSettings":{}}
```

On this topic, we also receive responses that look like this:

```
{"token":"bBm9FMLG","status":"done"}
```

Note that multiple commands are combined into one, e.g, here we have '**settings**', '**fetchSettings**', and '**fetchPayload**' commands.

Commands, also have "**token**" and "**status**" fields, the token is used to recognize responses from the device when commands have finished executing successfully, so we know these commands have been successful, and we note that in the database. For successful commands, the status must be "**done**".

There is a **NodeJS** mqtt forwarder which listens for messages on the commands topic and forwards them to the backend so their status can be updated in the db. This mqtt forwarder project is located at: **/var/www/mqtt_forwarder2http-wintergarten** on the server.