TOI Project 1

Andrej Zaujec, Peter Hornak

April 14, 2022

Contents

1 Communication model and implementation details

1.1 ESP-ESP

Communicaton between EPSs is based on the ESP-Now communication protocol. The slave ESPs send master the values periodically. The values are encoded into simple C-struct where is brightness and temperature obtained from the sensors. ESP-Now is based on the Wifi protocol it espoxes to the user only application level while it encapsulates internet and transport layer from the TCP/IP stack.

1.2 ESP-RPI

The communication between ESP-RPI is based on the HTTP protocol where the payload is encoded into the JSON format. Each object under data attribute represents the ESP values obtained from the sensors. The example of the message payload can be seen below.

```
{
  "data": [
    {
      "name": "esp1",
      "temperature": 1.2,
      "brightness": 1.2
    },
    {
      "name": "esp1",
      "temperature": 1.2,
      "brightness": 1.2
    }
  ]
}
```

The ESP has to have correctly setup the IP address of the HTTP server, which is the gateway IP in the RPI created network based on WiFi. HTTP server is validating the payload against the JSON schema in order to prevent unwanted data.

1.3 RPI-Thingsboard

Each data sent from ESP master to RPi is pushed to the array that lives with the HTTP server so whenever server shutdown as well the unsent data does. When the data array obtains enough predefined length, for example 12 elements that should be 1 minute of data. The aggregation function as minimum, maximum, median, avarage are computed from the data in the array, labaled and send via MQTT to the thingsboard server.

The $\text{tb}_{\text{devicemqtt}}$ python library is used to send data from RPi via MQTT, this library is part of official SDK from the thingsboard. The send data are in key-pair format where, the key represents device, attribute and used aggregation function. Example of this can be seen below.

```
{
  "rpi_temperature_minimum": 25,
  "rpi_temperature_maximum": 25,
  "rpi_temperature_average": 25,
  "rpi_temperature_median": 25,
  "esp1_temperature_minimum": 23.88,
  "esp1_temperature_maximum": 23.88,
```

```
  "esp1_temperature_average": 23.88,
  "esp1_temperature_median": 23.88,
  "esp1_brightness_minimum": 743,
  "esp1_brightness_maximum": 760,
  "esp1_brightness_average": 751.5,
  "esp1_brightness_median": 752.5,
  "esp2_temperature_minimum": 0,
  "esp2_temperature_maximum": 42.69,
  "esp2_temperature_average": 34.152,
  "esp2_temperature_median": 42.69,
  "esp2_brightness_minimum": 0,
  "esp2_brightness_maximum": 420,
  "esp2_brightness_average": 336,
  "esp2_brightness_median": 420
}
```

2 Project outputs

2.1 ESP application

- Application is based on the esp-idf and uses freertos API to schedule tasks and esp-idf-lib to communicate with sensors
- To build, flash, monitor application use the commands below, ensure you have esp-idf correctly setup up before executing commands below

```
cd ./esp-master #or /esp-slave
idf.py -p /dev/ttyUSB0 flash monitor
```

2.2 RPi application

- Python application with Flask HTTP server that is containerized with docker
- Installation process is described in the rpi/README.md

2.3 Thingsboard

- Rules for raising alarm if the published telemetry is missing some data and transforming temperature into fahrenheit

- Dashboard where is overview of temperature/brightness averages over time and gauge for displaying max values