Beehive Health Monitor

B3M38SPD seminar project

Ondřej Hruška

Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Measurement

Abstract

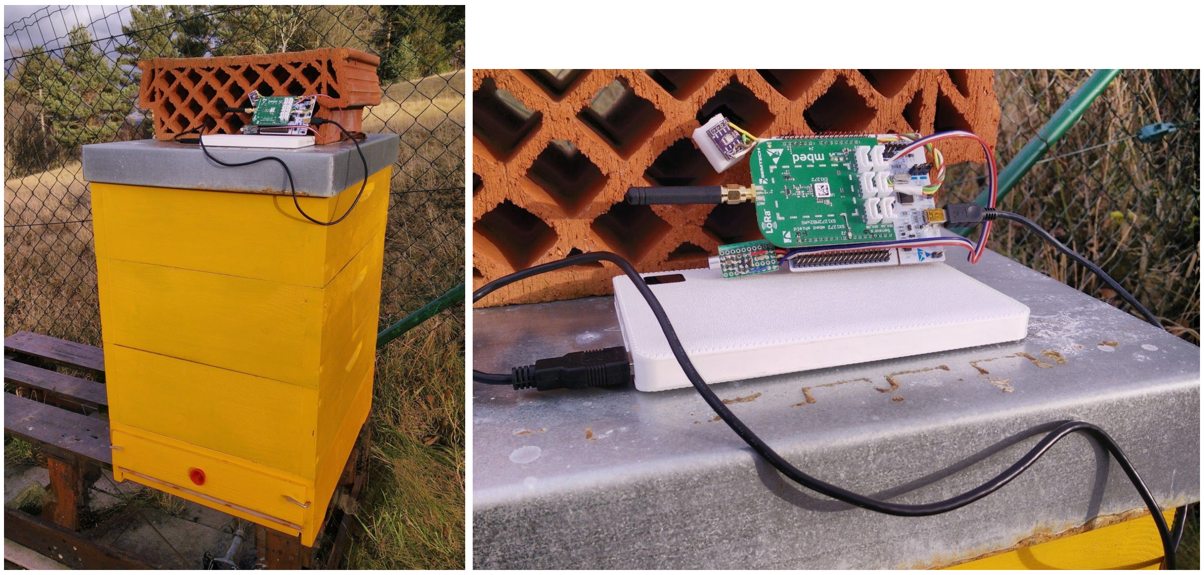
The beehive health monitor is a battery-operated IoT device using LoRa gateways provided by The Things Network (TTN) to autonomously report beehive health indicators. The device monitors relative air humidity, ambient temperature, and concentration of volatile organic compounds (VOC) inside the hive, as well as characteristics of captured audio spectrum for analysis of the bee buzzing patterns. A simple MQTT client using Node-RED was developed for capturing and storing received datagrams.

Due to lack of labeled data, limited experimental time, no The Things Network coverage in the place of the beehive and the unfortunate time of the year when the prototype was ready for testing, experimental data from real conditions could not be collected. The next stage of this research should, therefore, consist of collecting labeled training data for use with appropriate ML techniques and developing a local point-to-point LoRa connection to allow use without an existing coverage.

Key points

- Environmental sensing using BME680 on a custom breakout PCB
- Audio capture using an electret microphone, a transistor amplifier and ADC
- In situ spectral analysis and peak detection of the audio signal
- Periodic reporting to The Things Network
- MQTT client using Node-RED for datagram capture, parsing and storage
- Prototype built on Nucleo L073RZ and an SX1272MB2xAS mbed shield

Figure 1: The prototype kit. It couldn't be tested inside the hive within the scope of the project to avoid disturbing the wintering bees.



Functional overview

The LoRa portion of the beehive monitor firmware is based on the *End_Node* example project from the STM32CubeExpansion_LRWAN_V1.1.2 middleware package. The device spends most of the run time in deep sleep mode waiting for an alarm interrupt from the RTC peripheral which times the periodic measurements. The measurement interval is set to 15 minutes.

Each time a measurement is due, the relative air humidity, temperature, VOC level and atmospheric pressure are measured followed by an audio sample that is analyzed using FFT and a peak detection algorithm (explained later). The obtained data is packed to a binary datagram and sent using the LoRa transceiver to The Things Network gateways. The datagram is then broadcast by the network servers via MQTT and can be received for storage and further analysis.

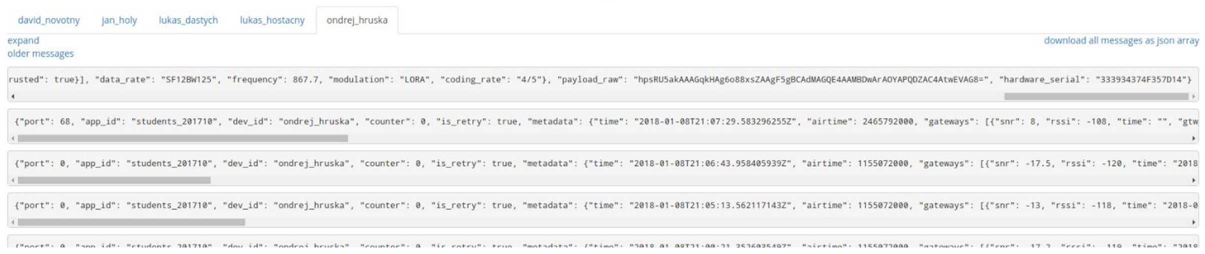
One limitation exists that lead to a few compromises, and that is the payload length, which appears to be capped at about 50 bytes. Longer messages are not received correctly by the network. For this reason some data are transmitted as *uint16_t* instead of *float*, first multiplying the number by 100 or 1000 to preserve some decimal places.

Datagram structure

- *int16_t* - °C x100 (two decimal places)
- *uint16_t* - rel. humidity % x100 (two decimal places)
- *uint16_t* - atm. pressure Pa - 85000 (85000 subtracted to fit within 2 bytes)
- *uint32_t* - VOX active layer resistance in Ohms
- *float* - audio total power (sum of all bins)
- *float* - noise level volume
- Detected peaks - 8x:
    - *uint16_t* - peak position in Hz
    - *uint16_t* - peak magnitude x1000 (3 decimal places)

The payload has total size of 50 bytes and all numbers are encoded as little-endian.

Figure 1.5: Datagrams as captured by *The Things Network* gateways. Sample payload: hpsRU5akAAAGqkHAg6o88xsZAAgF5gBCAdMAGQE4AAMBDwArAOYAPQDZAC4AtwEVAG8=



david_novotny jan_holy lukas_dastych lukas_hostacny ondrej_hruska

expand
older messages
download all messages as json array

```
rusted": true}], "data_rate": "SF12BW125", "frequency": 867.7, "modulation": "LORA", "coding_rate": "4/5"}, "payload_raw": "hpsRU5akAAAGqkHAg6o88xsZAAgF5gBCAdMAGQE4AAMBDwArAOYAPQDZAC4AtwEVAG8=", "hardware_serial": "333934374F357D14"}
{"port": 68, "app_id": "students_201710", "dev_id": "ondrej_hruska", "counter": 0, "is_retry": true, "metadata": {"time": "2018-01-08T21:07:29.583296255Z", "airtime": 2465792000, "gateways": [{"snr": 8, "rssi": -108, "time": "", "gtw
{"port": 0, "app_id": "students_201710", "dev_id": "ondrej_hruska", "counter": 0, "is_retry": true, "metadata": {"time": "2018-01-08T21:06:43.958405939Z", "airtime": 1155072000, "gateways": [{"snr": -17.5, "rssi": -120, "time": "2018
{"port": 0, "app_id": "students_201710", "dev_id": "ondrej_hruska", "counter": 0, "is_retry": true, "metadata": {"time": "2018-01-08T21:05:13.562117143Z", "airtime": 1155072000, "gateways": [{"snr": -13, "rssi": -118, "time": "2018-0
```

Hardware of the beehive monitor

Pin assignments

- PB2 - I2C SCL
- PB9 - I2C SDA
- PA1 - Audio IN (ADC channel 1)
- PC7 - indicator LED (soldered on top of the shield, indicates ongoing measurement)
- Other pins (SPI, assigned according to the shield pinout and example code.

The I2C and ADC pins are available on the LoRa shield sockets, though the matching proprietary connectors. For this reason, the sensor modules were soldered to the connector pins on the backside instead.

BME680 on a custom breakout board

A MEMS sensor BME680 manufactured by Bosch was selected to measure ambient indicators of the bee activity. The sensor provides direct digital readout of relative air humidity, temperature, and a measurement of the concentration of volatile organic compounds (VOC). There is also a readout of atmospheric pressure, which, however, does not seem to be of much use in this application.

The VOC sensor is based on a hotplate covered with a gas-sensitive layer. The hotplate is momentarily heated to 200°C - 400°C during measurement which affects the temperature sensor and its own sensitivity if the starting temperature is not kept close to constant, as was experimentally tested. This is not a problem when safe delays are employed. In our case the measurements are performed only every 15 minutes, to save power and also to avoid this issue.

The breakout board designed for the BME680 supports both I2C and SPI connections. I2C was selected as SPI is already occupied by the LoRa shield and adding another device would require adjusting the LoRa driver.

The breakout board includes optional pull-up resistor positions for I2C (R1, R2), a solder jumper to select the I2C interface (JP1), an I2C address selection solder jumper (JP2) and blocking capacitors as mandated by the BME680 datasheet (C1, C2).

The PCB is available from OSH Park: https://oshpark.com/shared_projects/yRIhJcpg

Figure 2: Top and bottom view of the BME680 breakout PCB

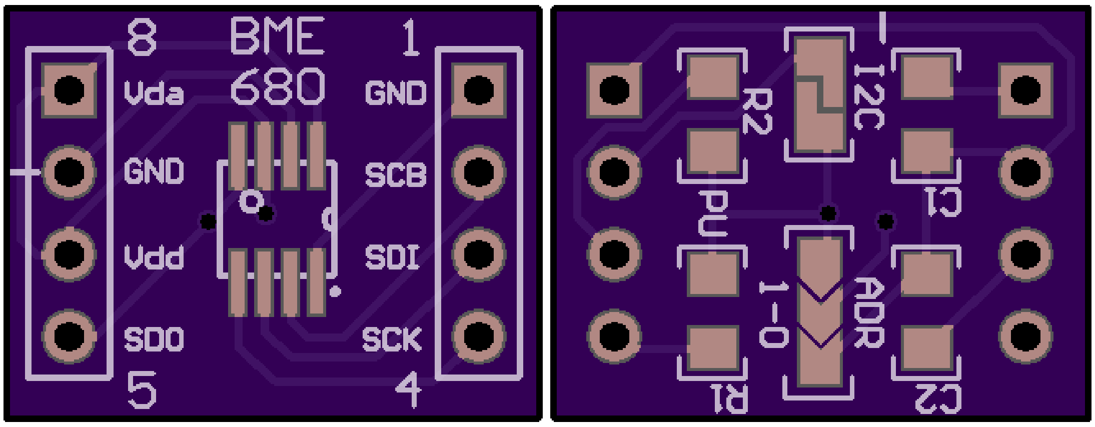


Figure 3: A BME680 soldered on the breakout board

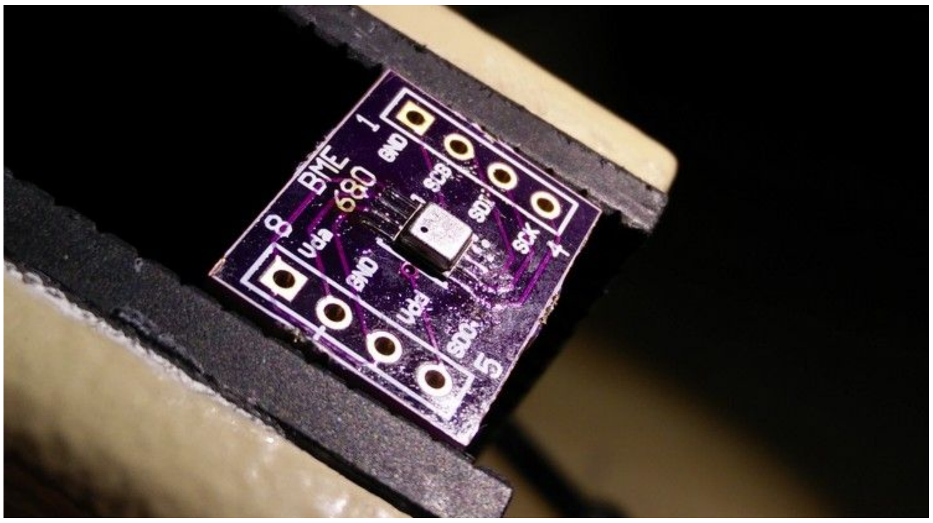
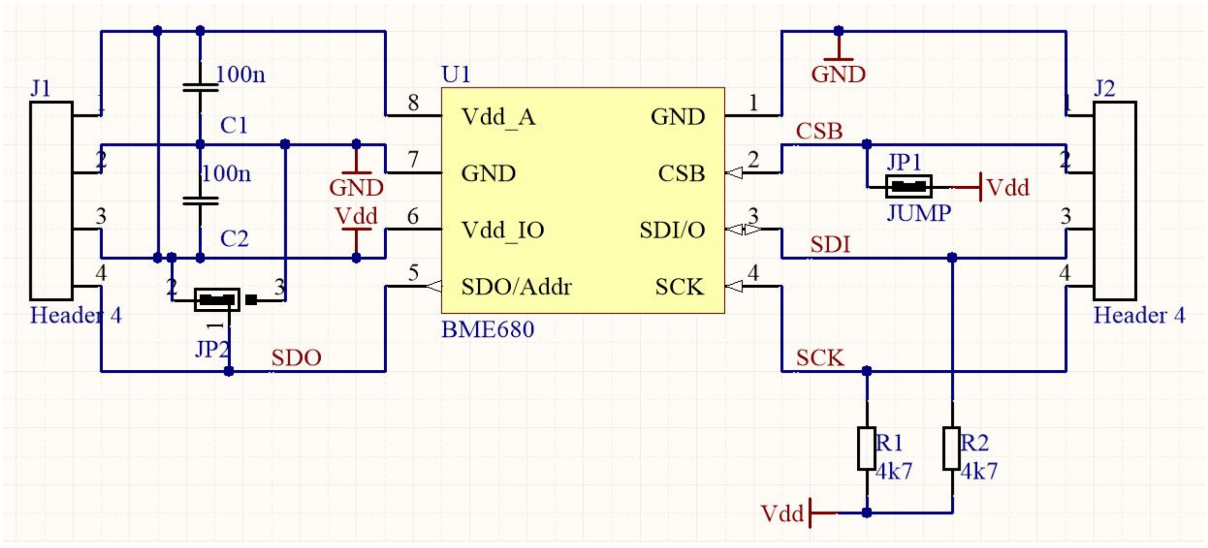


Figure 4: A schematic of the breakout board



Microphone amplifier

An electret microphone captures the bee buzzing sound. The microphone acts as an air-pressure-dependent variable resistor, however the variations are very small and not directly perceptible, necessitating an amplifier. Capacitor C1 is used for AC coupling.

The amplifier (pictured in Fig. 5) uses a two-stage common emitter design. Resistor R2 provides emitter degeneration for greater stability by the means of a negative feedback. This resistor is by-passed by C2 for useful audio frequencies to increase the amplifier gain.

The circuit was designed and tested to use most of the ADC input range while avoiding clipping for the expected input volumes.

Figure 5: Amplifier schematic

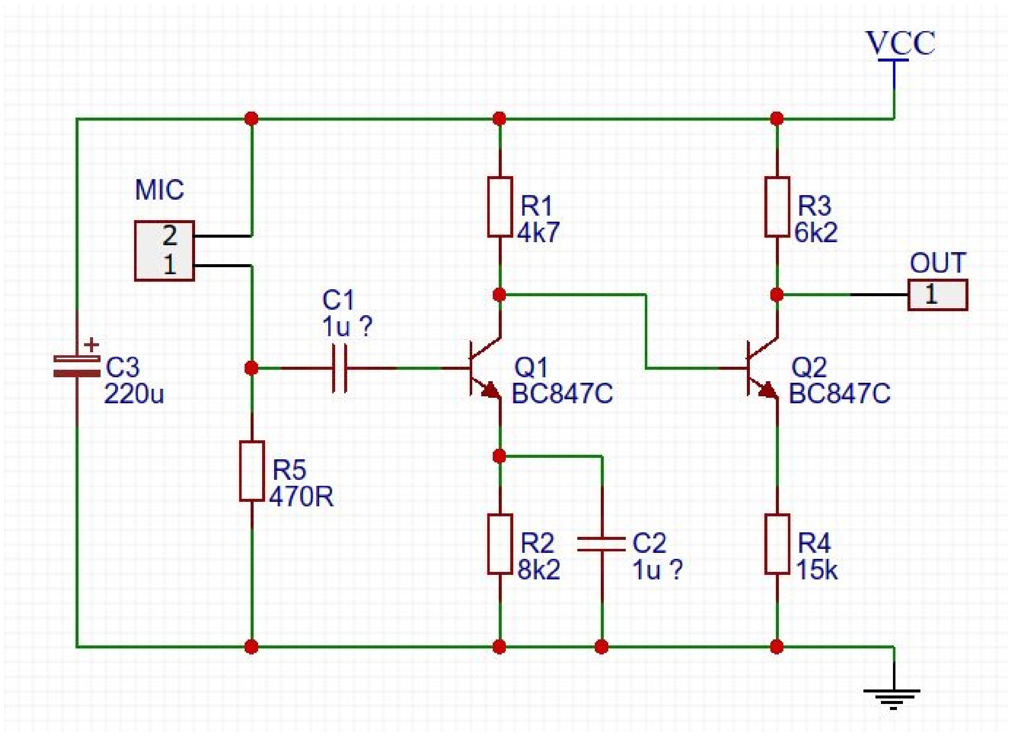
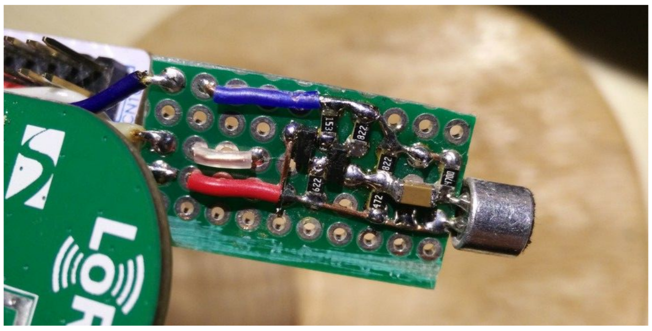


Figure 6: The amplifier board prototyped using SMD parts



Audio processing

The captured audio signal (1024 samples at 44.444 kHz) is processed using the following procedure to obtain the frequency spectrum and find important peaks.

Signal preprocessing and finding the spectrum

1. The signal is converted from `uint16_t` to `float`, and the range 0-4095 is scaled to ±1.0
2. A Hamming window is applied to the signal to avoid artifacts caused by discontinuity at the ends of the sample
3. The numbers are converted to complex with imaginary parts zeroed out
4. A complex frequency spectrum is obtained using a complex Fast Fourier Transform function provided by the ARM DSP library
5. The spectrum is converted to real values for easier processing by taking magnitudes of the frequency bins
6. The bin values are normalized by a factor based on the sample length for easier-to-use levels (typ. 1-50 for significant peaks)

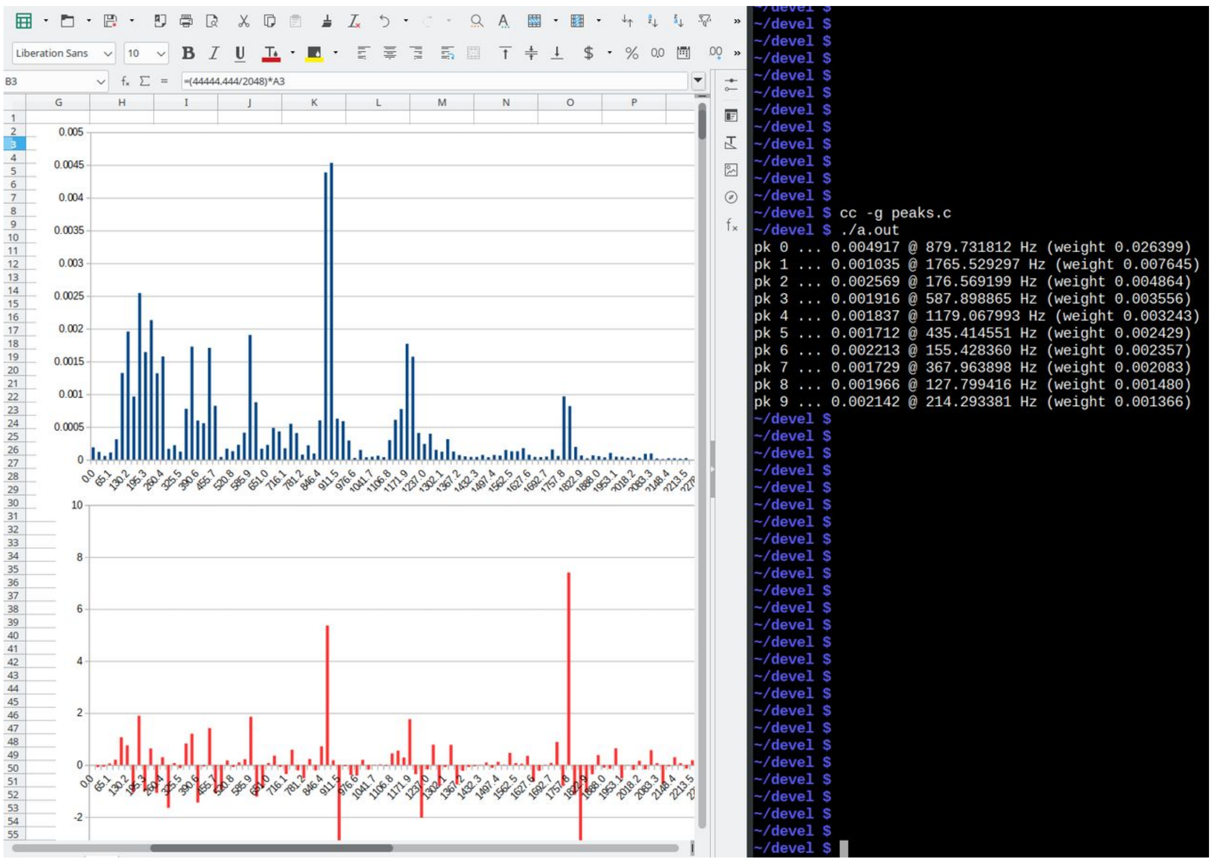
Finding peaks in the spectrum

The spectrum needs to be partly analyzed in situ to accommodate the limited LoRa payload length and required low transmit duty cycle. For this purpose, an algorithm to find the frequency peaks was developed.

1. Each bin is divided by an average of its neighborhood (±32 bins, excluding the bin itself) which serves as an estimate of the local noise level. This makes the algorithm work well for different and varying noise levels and gives prominence to smaller peaks.
2. The modified spectrum is differentiated and clamped to positive values.
3. The weight of each bin describing its importance in the spectrum is calculated by multiplying its difference value and magnitude.
4. The N bins with the highest weights are selected as the primary peaks. Peaks found within a certain number of bins from another peak are discarded, or replace the other peak if their weight is higher. This deduplication ensures that peaks spanning multiple bins are registered only once.
5. The precise peak position and magnitude is calculated using quadratic interpolation.

Total power is calculated by integrating the spectrum. A noise power estimate is obtained by subtracting the power of the area immediately surrounding the detected peaks from the total power. This value is divided by the number of contributing bins to estimate the noise floor level.

Figure 7: An example spectrum (left top), its noise-compensated and differentiated form (left bottom) and the results of the peak detection algorithm (right)

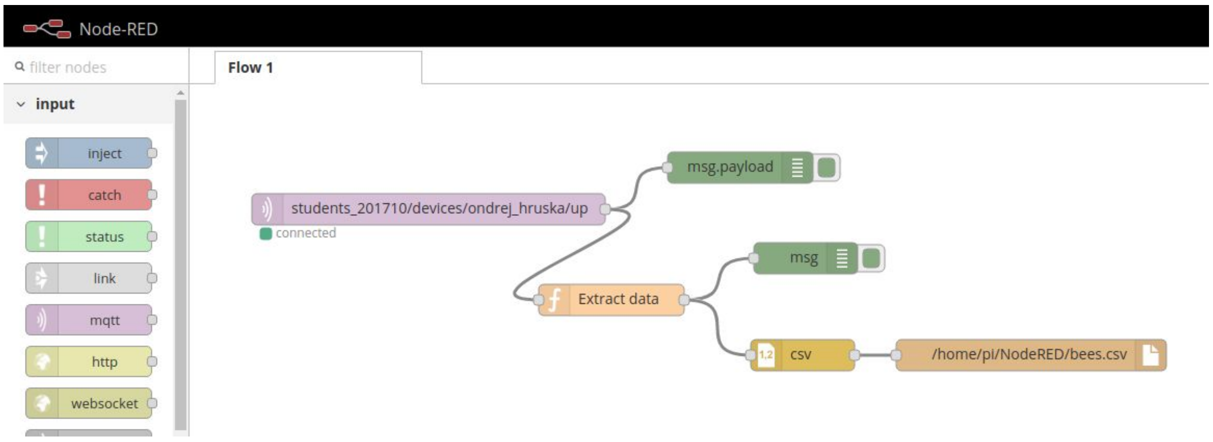


```
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $ cc -g peaks.c
~/devel $ ./a.out
pk 0 ... 0.004917 @ 879.731812 Hz (weight 0.026399)
pk 1 ... 0.001035 @ 1765.529297 Hz (weight 0.007645)
pk 2 ... 0.002569 @ 176.569199 Hz (weight 0.004864)
pk 3 ... 0.001916 @ 587.898865 Hz (weight 0.003556)
pk 4 ... 0.001837 @ 1179.067993 Hz (weight 0.003243)
pk 5 ... 0.001712 @ 435.414551 Hz (weight 0.002429)
pk 6 ... 0.002213 @ 155.428360 Hz (weight 0.002357)
pk 7 ... 0.001729 @ 367.963898 Hz (weight 0.002083)
pk 8 ... 0.001966 @ 127.799416 Hz (weight 0.001480)
pk 9 ... 0.002142 @ 214.293381 Hz (weight 0.001366)
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
~/devel $
```

Node-RED MQTT client

A simple MQTT client was designed for receiving and storing messages sent by the beehive monitor. The client connects to a TTN server and subscribes to a topic matching the assigned device ID. The messages are received as base64-encoded strings inside JSON objects.

Figure 8: A screenshot of the Node-RED flow



The MQTT input node must be configured as follows:

- **Server:** eu.thethings.network
- **Port:** 8883
- **Username:** (as assigned)
- **Password:** (as assigned)
- **Topic:** *username*/devices/*student_name*/up

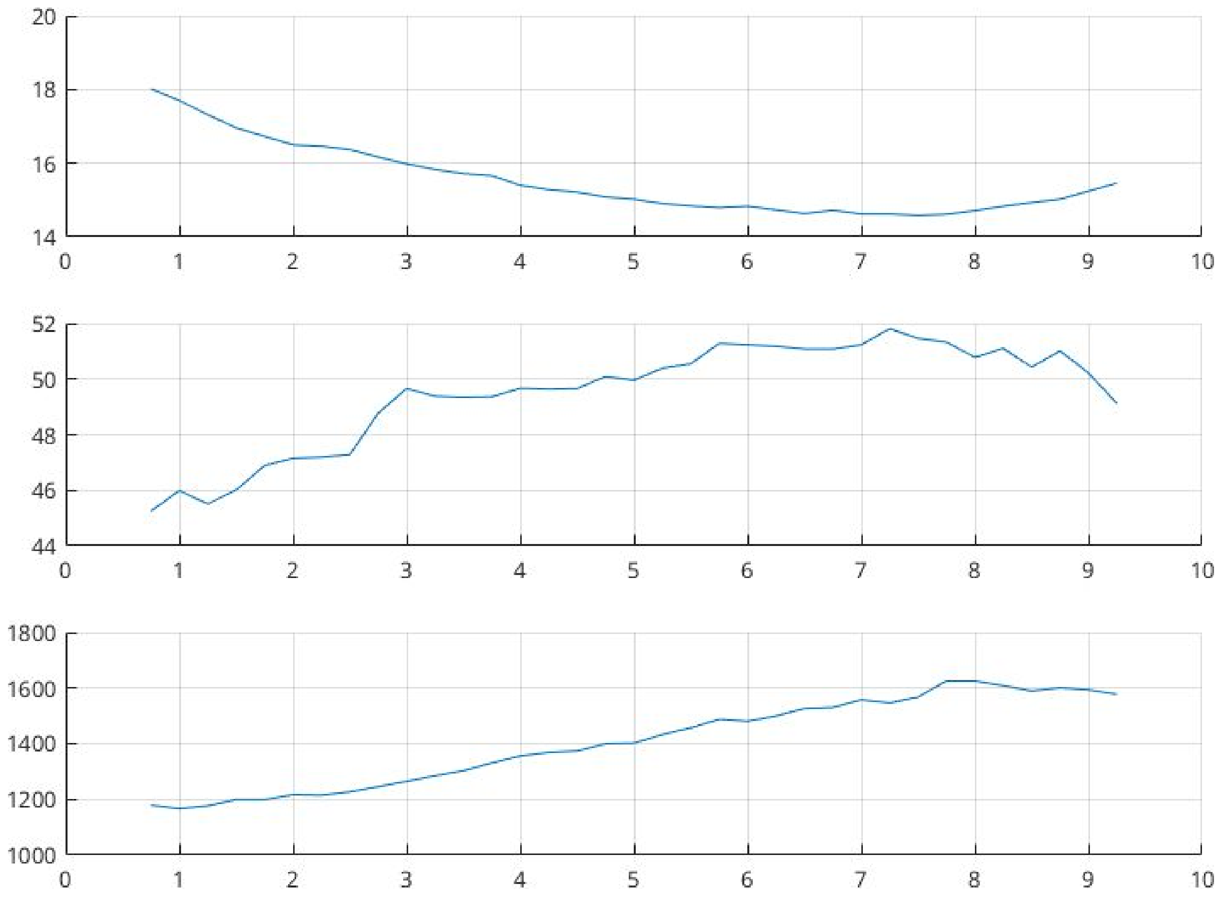
The message from The Things Network is, when received, available as a string payload (`msg.payload`) inside a connected function node. The string needs to be parsed to a JavaScript object and the useful payload is then available as a Base64-encoded binary buffer in the field `obj.payload_raw`.

The following procedure can be used to obtain the byte array and a unix timestamp of the message for further processing:

```
var obj = JSON.parse(msg.payload);

var bytes = Buffer.from(obj.payload_raw, 'base64');
var ts    = Math.round((+(new Date(obj.metadata.time)))/1e3);
```

Figure 9: Example data collected using the shown Node-RED flow while testing the BME680 in a bedroom, overnight. The horizontal axis is marked in hours since midnight. Subplots: (1) temperature °C, (2) relative humidity %, (3) VOC concentrarion (active layer resistance) Ω


20
18
16
14
0
1
2
3
4
5
6
7
8
9
10
52
50
48
46
44
1800
1600
1400
1200
1000