University of Bern
Institute of Computer Science

Development and evaluation of a low-cost, LoRa-based measurement device for microclimatic assessments in urban environments

Bachelor Thesis by Joel Hari

Supervisors:
Prof. Dr. Torsten Braun
Prof. Dr. Stefan Brönnimann

Advisors:
Jakob Schärer
Moritz Gubler

March 7, 2022

Contents

List of Figures

Chapter 1

Abstract

With the ongoing climate change it is crucial to understand the intra-urban temperature and air humidity variations. The urban heat stress and related consequences are becoming increasingly relevant for urban planning. Therefore, a heat monitoring network of up to 85 low-cost temperature sensors was built and evaluated in a previous research project.

We have developed a new version of this low-cost temperature sensor providing two major improvements. The first major improvement is wireless communication via LoRaWAN. This decreases the maintenance effort since the previous sensor required to read out the measurement data manually on site. The second major improvement is active ventilation. With the previous sensor only relying on passive ventilation, we achieved to increase the measurement accuracy using active ventilation. We were able to quantify the increased measurement accuracy, as we did numerous experiments including extensive amount of measurements in our lab.

Chapter 2

Introduction

With the climate playing a crucial role nowadays it is urgent to work towards a more ecological and sustainable urban landscape. Cities all around the globe are growing rapidly which makes understanding how different areas in a city affect the intra-urban air temperatures is key for the future of urban planning. Large amounts of concrete result in increased heat stress on the population with negative effects on public health.
Urban heat monitoring can show how vegetation or irrigation can help to cool down urban environments. For this reason, the climatology group at the institute of geography at the University of Bern has built a heat monitoring network in the city of Bern consisting of custom made low-cost temperature sensors. However, this low-cost sensor has no active ventilation and therefore it measures too high temperatures during daytime when it is exposed to the sun. Besides that, the data has to be read out manually at each sensor, resulting in a high maintenance effort.

We are developing a new version of this low-cost temperature sensor featuring wireless communication and active ventilation. For the wireless communication, we integrate a LoRa module in our sensor to communicate via LoRaWAN, a low-power, wide-area network. With the wireless communication we achieve to decrease the maintenance effort, since the data will be accessible from any web browser. Besides that, we equip the sensor with a small fan to provide active ventilation. With active ventilation we can improve the measurement accuracy over the previous sensor. The development of our sensor involves programming the software, building a prototype and doing extensive measurements to evaluate the effect of the active ventilation. We will develop different experiment and build a test environment that allows us to simulate real world conditions as close as possible, while still meeting the condition of being a low-cost sensor.

In summary, this new sensor will be a low-cost IoT device featuring wireless communication via LoRaWAN, features active ventilation and will take temperature and humidity measurements, while being powered by solar power to make it autonomous.

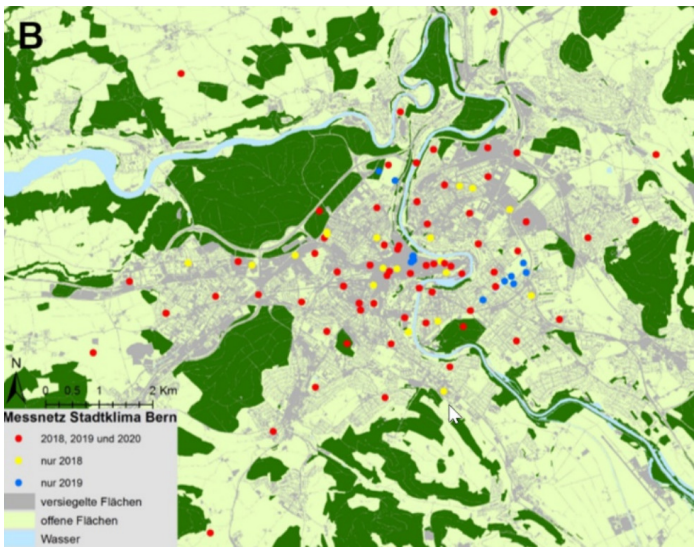
In the first chapter of this thesis we present the related work that is about the aforementioned heat monitoring network. Next, we explain which technologies we are using for our sensor and why they suit our requirements. In the following chapter, we then explain how we use these technologies for the architecture of our sensor network. After that, we have two chapters explaining the design and implementation of the sensor's hardware and software. In the following chapter we describe the measurements and evaluation we have done with our sensor. After the conclusion of our work we also provide a preview on future research following our work.

Chapter 3

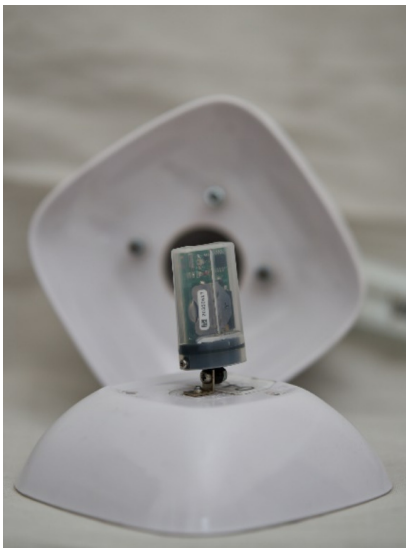
Related Work

3.1 Heat Monitoring Network

The climatology group of the institute of geography from the University of Bern started to build and maintain a heat monitoring network of up to 85 sensors in 2018 [1]. These sensors are placed within and around the city of Bern, as shown on the map in Figure 3.1a. The motivation for such a heat monitoring network is that the urban heat stress and related consequences are becoming increasingly relevant for urban planning, infrastructure and health services. With high-resolution air temperature measurements, we can understand how vegetation or irrigation helps to cool down urban environments.



(a) Heat monitoring network within the city of Bern (© Burger & Gubler 2020)



(b) Low cost temperature sensor (© P. Duschletta 2019)

Figure 3.1: Heat monitoring network consisting of custom low-cost temperature sensors

3.2 Low-cost Sensor

As off the shelf solutions are expensive and therefore not suited to get high spatial resolution air temperature measurements, what requires a large series of devices, a custom sensor is used, the one shown in Figure 3.1b. Therefore, the sensor should be low-cost. For this reason a small temperature logger is used, which is protected by a custom radiation shield allowing for natural ventilation.

Currently, this sensor has two main shortcomings. First, it has a relatively high maintenance effort since the measurements are only stored on the sensor itself. In fact, it is required to read out the data at each sensor manually. This results in a lot of work to gather all data from every sensor from the network, what is mostly done once per month. The second shortcoming is that the measurement accuracy varies quite a bit depending on the weather and time of day, because of heat accumulation inside the housing.

During the aforementioned research project [1] in summer 2018, these sensors were evaluated and tested on their potential for such a heat monitoring network. Three of the sensors were placed close to automated weather stations allowing to compare the temperature measurements. This showed that their sensors measured temperatures that were about 0.6 to $0.9K$ higher than the reference during daytime. 82.8% of this variance could be explained by radiative heating and insufficient ventilation because of low wind speed.

Taking these two limitations into consideration, we would like to suggest specific goals for a further development of this sensor. Our first suggestion is to reduce the maintenance efforts by sending the measurements via some network to a server. This would allow the user to easily access all data from every sensor in the network via a web browser. Second, we suggest to enhance the quality of the measurements by adding active ventilation for the housing in order to eliminate any heat accumulation.

Chapter 4

Background

For this project, some specific technologies are used, which are explained in this chapter.

4.1 LoRa and LoRaWAN

LoRaWAN [2] is a Low-Power, Wide-Area Network (LPWAN), which is designed to optimize for battery lifetime, capacity, range and cost. This makes LoRaWAN ideal for Internet of Things (IoT) devices like our sensor, which has a very limited amount of power and needs to be low-cost.
LoRa is the physical layer used to create the long-range communication link and is based on chirp spread spectrum modulation. This technology has already been used in military and space communication for decades, but LoRa is the first low-cost implementation for commercial usage.

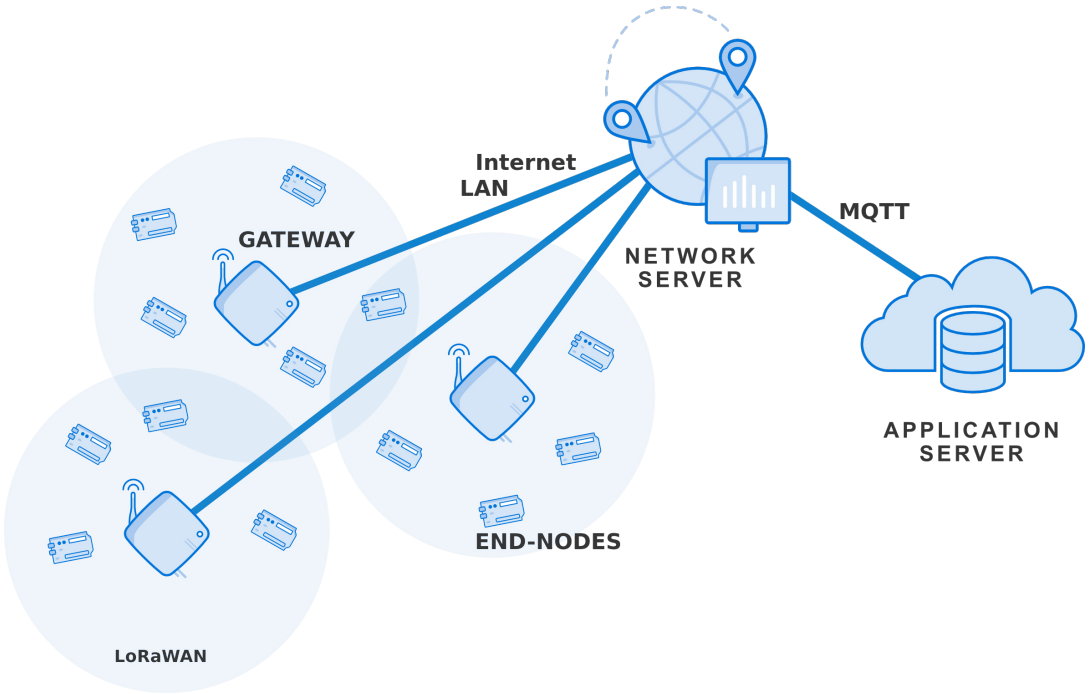
To establish a LoRaWAN network of some end-nodes (IoT devices), one or multiple gateways are required as LoRaWAN uses a star architecture. This means that end-nodes will transmit their data directly to a gateway, which is connected to a cloud-based server system for example via Ethernet or Wi-Fi. Therefore, the long range capability of LoRa is essential for a high communication range. In contrast to a mesh network, where high communication range is achieved by having the end-nodes forwarding messages from other end-nodes, with LoRaWAN the overall energy consumption is substantially lower.
The end-nodes do not send their data to a specific gateway moreover it is received by multiple gateways, and therefore the cloud server has to filter out redundant messages. However, this has the advantage that moving nodes do not need to perform a handover from one gateway to another as its the case for other network architectures.

Another aspect reducing the energy consumption of nodes in a LoRaWAN network is their asynchrony. These nodes can send data whenever they are ready to do so. In contrast, in synchronous networks, the nodes have to synchronize with the network from time to time. With our sensors, this would unnecessarily consume energy.

In summary, we use LoRa for this project as it is low-cost, which is one of the main requirements of the sensor itself. It has also a low power usage enabling the sensor to run only on solar energy. Finally, it has a long communication range so that we can expect every sensor placed in an urban environment to have network coverage.

4.2 The Things Network

The Things Network (TTN) [3] is a global collaborative Internet of Things network using LoRaWAN. It is built around a LoRaWAN network server and allows the user to manage devices, gateways and applications. Such applications expose a MQTT broker giving access to the data sent to it (see section 4.3). Before a device can send data via LoRaWAN, it needs to join an application specified by its unique app key. Therefore, we have created a application on The Things Network for our sensors.



Source: https://www.thethingsnetwork.org/docs/network/overview.png

Figure 4.1: Network architecture overview

The TTN network server is receiving all messages that are sent via LoRaWAN. This means, the network server needs to know, which TTN applications should process the messages from which end-nodes. Therefore, the end-nodes have to be registered as devices in the TTN application. An end-node is registered by entering its device EUI. The device EUI is a 64-bit extended globally-unique identifier, assigned by the manufacturer of the LoRa module. The EUI is stored on the LoRa module and can be read out by sending a specific command to the module.

4.3 MQTT

MQTT is a lightweight publish and subscribe messaging protocol [4]. It consists of the MQTT server, which is also called the broker, and numerous clients. A message consists of a topic and the message content. Clients can publish data on a specific topic to the broker, which then forwards the message to all subscribers of its topic. The broker does not store messages unless the retain flag was set to true by the publisher. Then the broker stores this message, and when a new client subscribes to its topic, he immediately receives the retained message. Nevertheless, the broker only stores one retained message per topic.
For our project, we are using The Things Network as its applications expose a MQTT server. All messages sent via LoRaWAN as well as information about newly joined devices get published to this broker. Therefore we will develop a server application that will subscribe to this broker in order to receive the measurements from the sensors. This is explained in more detail in Chapter 7.

To sum it up, we use MQTT for the communication between The Things Network and our server application as it is provided by The Things Network and is easy to implement using Python.

Chapter 5

Architecture

With the technologies described in the previous chapter, we designed the following architecture, shown in Figure 5.1, for our sensor network. The blue box is highlighting the components in the system we developed for this project. These are the sensor and the server application. The other components required, namely the LoRa gateway and the TTN network server, are not developed by us.

In order for the system to work, a few setup and initialization steps are required. First, we create an application on The Things Network. In this application, we then register our sensors we want to connect. Afterwards, the server application and the sensor need to be initialized. At this initialization, the server application is setting up the web page, giving the user access to the data. Additionally, the server application creates an MQTT client, subscribing to the broker exposed by our application on The Things Network. When the sensor is initialized, it configures the LoRa module and connects to our TTN application. After the initialization, the sensor is ready to send data to our server.

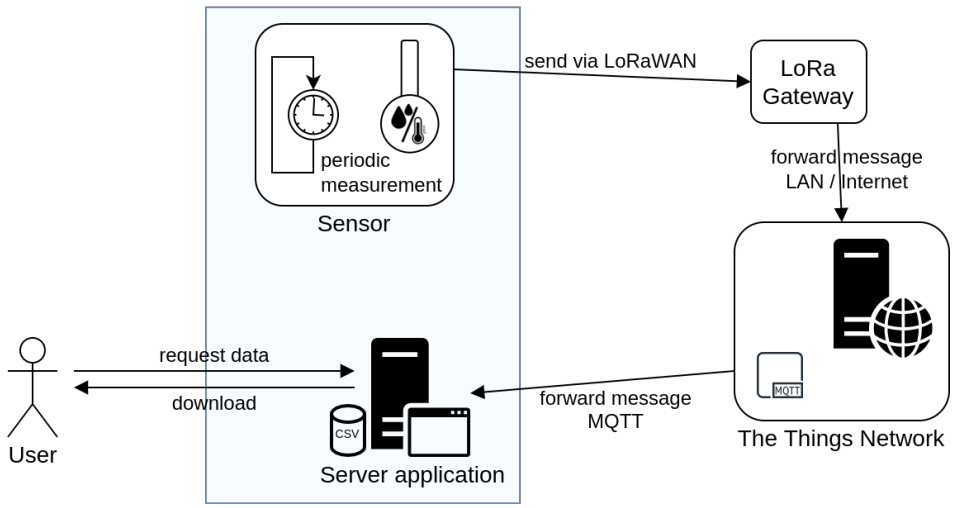


Figure 5.1: System architecture diagram

Once the all components are running, the sensor is periodically taking temperature and humidity measurements. Every time, immediately after the sensor has taken a measurement, it sends the data via LoRaWAN to a gateway. In order for the sensor to communicate via LoRaWAN, we equip the sensor with a LoRa module. The hardware of the sensor is explained in more detail in the following chapter. Once the gateway receives the message from our sensor, it forwards the message to the TTN network server. On the network server the message is then published to the MQTT broker, which is forwarding the message to the subscribers. In our case, the only subscriber is the MQTT client of our server application, that, as a result, receives the message. Both, the forwarding of the message from the LoRa gateway to the TTN network server and from the broker to our server, are done via LAN, Wi-Fi, and the internet. The server application is then storing the measurement data in a comma-separated values (CSV) file. The user can download this CSV file with all measurements through a web browser at any time.

In the diagram shown in Figure 5.1, we only included one sensor and one gateway. Nevertheless, when building a sensor network we will of course have multiple sensors, that send their measurements to multiple gateways. Since these sensors and gateways do not communicate with each other, we have not included more in the diagram. However, we will still have only one TTN application and one server to gather the data of all sensors.

Chapter 6

Hardware

To meet the specific requirements of our IoT sensor, specific hardware is needed. The design and selection of all the hardware and components has been made by Abilium.

6.1 Printed circuit board

For this project, we use a custom designed printed circuit board (PCB) we are having produced by Eurocircuits GmbH with the following key components:

- Silicon Labs EFR32MG21 System on a Chip (SoC)
- Microchip WLR089U0 LoRa module with antenna
- Sensirion SHT31A-DIS-B2.5kS Humidity and Temperature Sensor
- Pin headers to connect the fan, battery and solar panel
- USB port

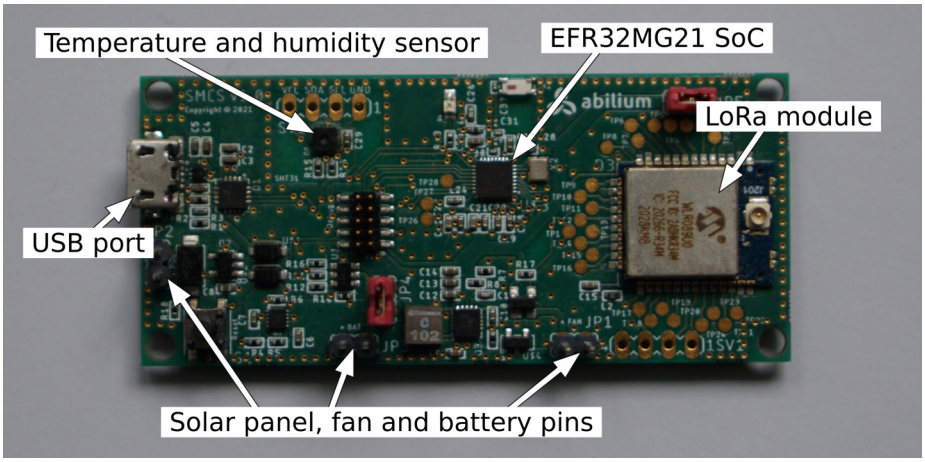


Figure 6.1: PCB of the new sensor

The EFR32MG21 [5] SoC includes a 32-bit ARM Cortex-M33 core with an 80 MHz frequency specially designed for IoT applications. This gives us more than enough computational power for our use case while featuring integrated security systems. Besides that, the SoC is equipped with 96 kB random-access memory (RAM) and 1024 kB flash storage.
The WLR089U0 LoRa module is manufactured by Microchip Technology Inc. and also includes a 32-bit ARM core, the Cortex-M0+. This makes the WLR089U0 a standalone module capable of communicating via LoRaWAN.
To get the air temperature measurements the Sensirion SHT31A-DIS-B2.5kS humidity and temperature sensor is included on the PCB. This sensor is, as its name suggests, also capable of measuring the relative humidity, what is also an upgrade over the previous sensor. Particularly interesting will be to observe how the humidity behaves during ventilation. According to the sensor's datasheet it measures with an accuracy of $\pm 0.3°C$ and $\pm 2\%$ relative humidity.

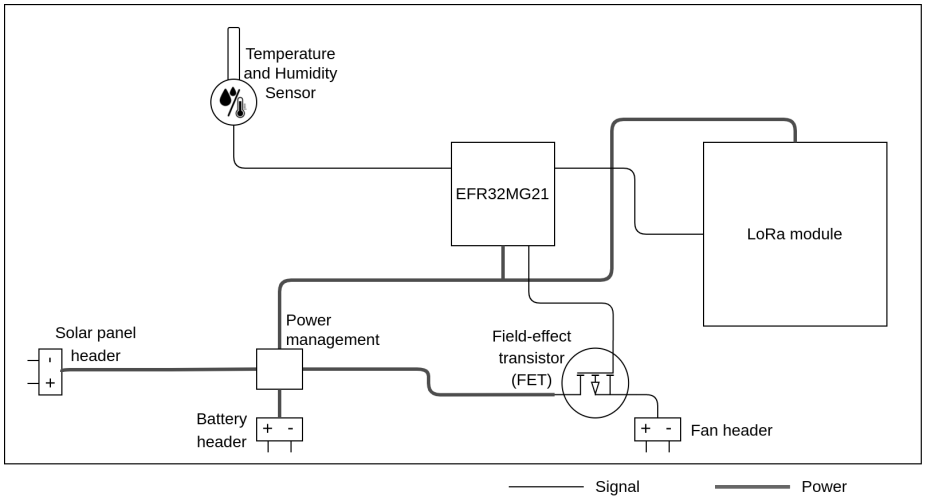


Figure 6.2: Simplified circuit diagram of the PCB

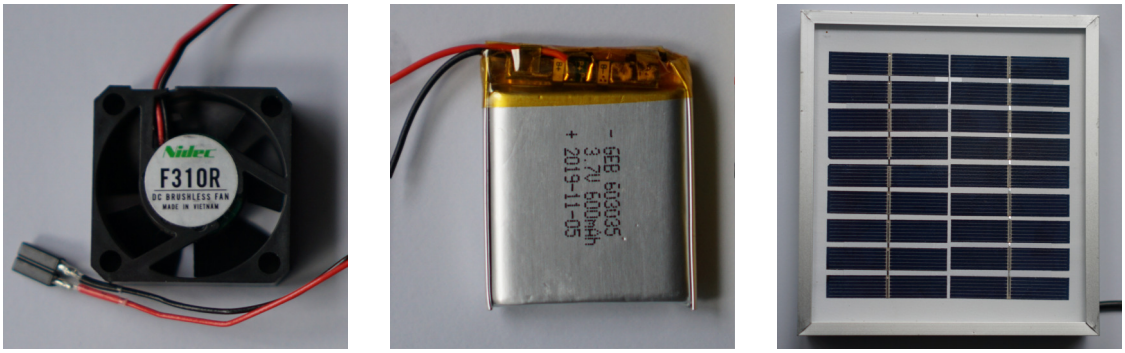
Figure 6.2 shows a simplified circuit diagram of the PCB. In the diagram we can see the most important components and how they are connected with each other. One interesting component, we have not mentioned before, is the field-effect transistor (FET). A FET has three terminals, the "source", that is connected to the battery, the "drain", connected to the fan, and the "gate", which is connected to the microcontroller. The FET uses an electric field to control the conductivity of a semiconductor, that is connecting the source to the drain. When a voltage is applied to the gate, current can flow between the source and the drain. Therefore, the FET acts as a switch for turning the fan on and off, which is controlled by the microcontroller. The fan has to be connected directly to the battery, since it draws a relatively large amount of power. Consequently, if the fan would be powered through the microcontroller, the large electric current would highly decrease the lifetime of the microcontroller.

In the diagram we can also see, that when designing the PCB, special attention was paid on placing the sensor with as much space around it as possible. Nevertheless, to improve the measurement accuracy we could consider having the temperature and humidity sensor separately, only connected with a cable to the main PCB. However, in order to keep the costs low it is placed directly on the main PCB. This way, when the device is drawing a lot of power, the PCB is most likely going to heat up also affecting the temperature measured by the sensor. If this is the case we will be able to observe it during our experiments and have to evaluate if the measurements are distorted in any significant way. Everything about the measurements and evaluation can be read in chapter 8.

Besides that, the PCB has pins to connect external components to it (see section 6.2). Furthermore, the PCB also has a USB port that is only used during development as it can serve as a power supply and is required to flash the software onto the chip.

6.2 External components

In addition to the PCB, some external components are required for the sensor to work properly. These are the following:



(a) fan (b) battery (c) solar panel

Figure 6.3: External components used for the new sensor

These are all standard off-the-shelf parts meaning they are reliable and inexpensive. Therefore they meet the condition of being low-cost.
For each of these components, the PCB has specific pins to connect them to. Which fan we are going to use for the final sensor, we will decide based on the results of the measurements and evaluation. Since the fan choice plays a major role on the effectiveness of the ventilation. For the battery, we will also do measurements in order to evaluate what capacity we will use. The capacity of the battery should allow the sensor to run for at least two weeks without recharging. The solar panel we use delivers a maximum electric power output of 15 watt.

6.3 Sensor Prototype

In order to run any ventilation experiments, we first had to design a prototype of the sensor with all its components. Low cost is again an important factor. Therefore, we are reusing the housing together with the mounting bracket from the previous sensor what helps to make the sensor cheaper. As by reusing the materials for the housing, we can reduce the time required to assemble the sensor.

Figure 6.4: 3D printed frames used to mount the PCB and fan inside the housing

The PCB is placed in a 3D printed frame that holds the fan in place. As shown in Figure 6.4, three different variations are required. One for each of the fans we are going to evaluate and compare with each other.

Figure 6.5: Upside down view of the prototype with the bottom bowl removed clearing the view onto the 3D printed frame with the medium fan installed

The housing is constructed from three breakfast bowls with a hole in the center that fits the 3D printed frame and one bowl each at the top and bottom to close it up. Figure 6.5 shows an upside-down view of the prototype with the bottom bowl removed. When fully assembled, there is about $5mm$ clearance to the fan, which is pushing the air up towards the sensor. We have chosen to place the fan this way since warmer air is rising up. This makes trying to push out the hot air towards the top very reasonable.

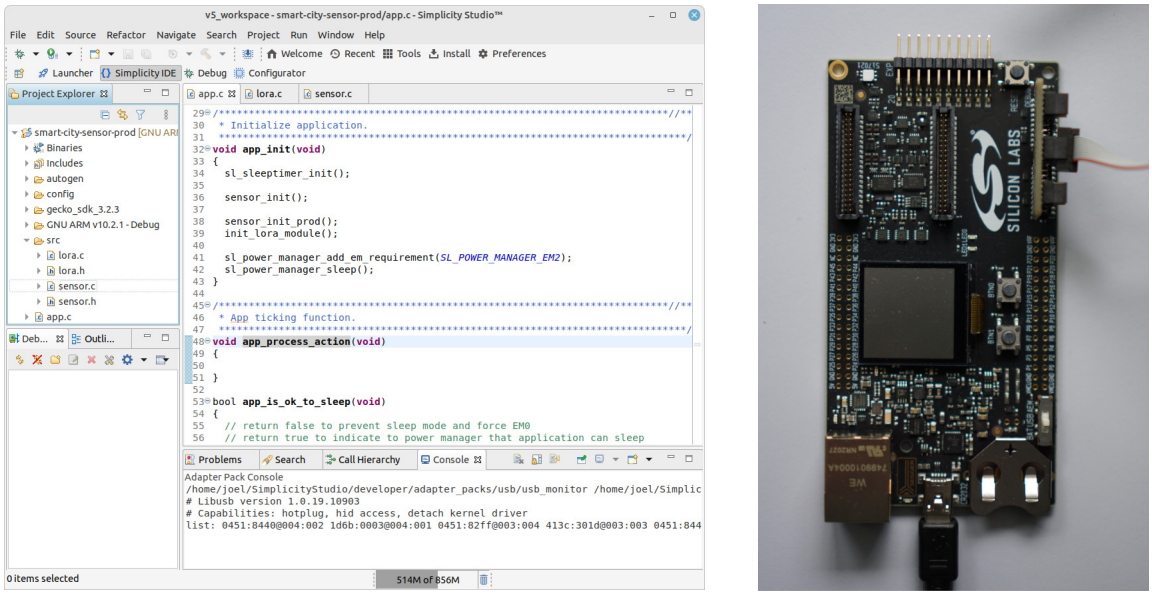
The battery is mounted in an additional enclosure, which will be attached to the mounting bracket as there is not enough space inside the housing. The solar panel will be mounted independently from the sensor. This has the advantage of aligning it to the sun based on the sensor's specific location. Besides that, it is probably best to mount the solar panel vertically. Preventing objects like leaves or snow to accumulate on it blocking all sunlight.

Chapter 7

Software

7.1 Development Environment

For the software development, we are using Simplicity Studio [6]. This is the integrated development environment (IDE) provided by Silicon Labs, specifically designed to program their microcontrollers. Simplicity Studio allows to easily manage software modules in order to minimize the overall size of the software. This module system has the advantage that no unnecessary processes are executed by the microcontroller, reducing the overall power consumption. The software itself is written with the C programming language.



(a) Simplicity Studio IDE

(b) Development kit

Figure 7.1: Development environment used to program the microcontroller

Besides the IDE, an additional development kit from Silicon Labs is required to flash the program onto the microcontroller. This PCB, shown in Figure 7.1b, gets connected to the sensor PCB with an 8-pin cable. This kit also allows debugging the software while it is running on the microcontroller, making the development process fairly simple.

During development, we also installed a gateway in our lab since sending data via LoRaWAN is only possible if the device can establish a connection to a gateway. Unfortunately, there is no network connection inside our laboratory, where we have developed the software. Besides that, LoRaWAN has a fair use policy limiting the amount of packages a node is allowed to send. Having our own gateway during development means we do not have to fully comply with this policy.

7.2 Software Requirements

The software for this newly developed low-cost sensor has the following requirements:

- Read the sensor data periodically
- Transmit the data via LoRaWAN
- Control the fan

With the sensors sending their data via LoRaWAN to The Things Network, an additional server application is required to receive those data. The requirements for this application are the following:

- Receive sensor data via MQTT
- Provide access to the data

7.3 Microcontroller Software

We have designed the sensor software in such a way that it is running in an infinite loop. It features two main components. The first component controls the fan and takes measurements, and the second component is responsible for the communication via LoRaWAN. For this first component, we are using the "sleeptimer" module, which has a function to start a periodic timer that executes a callback function after each time period. Therefore, three periodic timers, with the same period time, are initiated after each other with specific time intervals between. These intervals specify how long to ventilate and how long after ventilation stops the measurement will be taken. The `init()` function is called by the main program at the beginning setting all this up. In a simplified form, without being executable, this component looks the following:

```
1 uint32_t period_time = 3600;
2 uint32_t ventilation_time = 250;
3 uint32_t measure_delay_time = 0;
4
```

```
5  void _take_measurement(void) {
6      uint32_t sensor_temperature = 0;
7      uint32_t sensor_humidity = 0;
8
9      // take measurment with the sensor
10     _sensor_read_data(&sensor_temperature, &sensor_humidity);
11
12     // create send command with measurements as message payload
13     char *command;
14     sprintf(&command, "mac tx uncnf 1 ff%08x%08x", sensor_temperature,
       sensor_humidity);
15
16     add_command("mac resume");
17     add_command(&command);
18     add_command("mac pause");
19     add_command("sys sleep standby 105000");
20     send_to_lora();
21 }
22
23 void init(void) {
24     // start periodic timers with predefined intervalls between
25     wait(period_time - ventilation_time - measure_delay_time);
26     start_periodic_timer(period_time, _fan_on);
27
28     wait(ventilation_time);
29     start_periodic_timer(period_time, _fan_off);
30
31     wait(measure_delay_time);
32     start_periodic_timer(period_time, _take_measurement);
33 }
```

In the `_take_measurment()` function, the component responsible for the communication with the LoRa module is called. This second component holds a command buffer to which commands can be added. These commands then get sent to the LoRa module with a one second delay in between. However, after a `"mac tx ..."` command, telling the LoRa module to send data via LoRaWAN, a ten second delay is used. This is required as the LoRa module has to wait until it receives a confirmation before it can process the next command. Besides that, there is also the `init_lora_module()` function responsible for setting all parameters necessary to join our application on The Things Network. Again the following code is simplified a lot without being executable:

```
1  char *command_buffer[32];
2  const size_t command_buffer_size = 32;
3  uint8_t read_command_index = 0;
4  uint8_t write_command_index = 0;
5
6  void add_command(char *command) {
7      // add the command to the command buffer
8      sprintf(command_buffer[write_command_index], "%s", command);
9      increase_write_command_index();
10 }
11
12 void send_to_lora(void) {
```

```
13      // return if timer is running since on timeout send_to_lora is called
14      if (is_timer_running()) {
15          return;
16      }
17
18      // send command to lora module
19      size_t command_length = strlen(command_buffer[read_command_index]);
20      iostream_write(command_buffer[read_command_index], command_length);
21
22      if (read_command_index < write_command_index) {
23          increase_read_command_index();
24
25          // if buffer isn't empty repeat send_to_lora after a 1 second delay
26          // expect if the next command is "mac pause" wait for 10 second
27          if (read_command_index != write_command_index) {
28              uint32_t timeout = 1000;
29              if (match(command_buffer[read_command_index], "mac pause")) {
30                  timeout = 10000;
31              }
32              start_timer(timeout, send_to_lora);
33          }
34      }
35  }
36
37  void init_lora_module(void) {
38      // add commands to init the lora module and joining the network
39      add_command("mac reset 868");
40      add_command("mac set deveui 3413C9FEFF19276A");
41      add_command("mac set joineui 0000000000000000");
42      add_command("mac set appkey 6968A68996F1026B...");
43      add_command("mac join otaa");
44      send_to_lora();
45  }
```

With this implementation, the microcontroller is not listening for any response of the LoRa module after sending commands. In doing so, we could also add specific exception handling for messages that have not been transmitted correctly or if the network connection has been lost. However, this would also require a more advanced power mode management. With this implementation, the microcontroller is configured to always return to the lowest power mode. In this mode, not all messages sent from the LoRa module to the microcontroller are received correctly as some only trigger the controller to return to a higher power mode. Therefore it would require quite a lot of extra work to get it all working correctly. Besides, we do not expect a lot of these exceptions to happen. If a sensor loses connection it probably is in a place with bad network coverage. Therefore, it will lose connection repeatedly such that automatically reconnecting and resending would consume too much energy anyway. Additionally losing some measurements is not a critical error at all. After all, in a network of 80 sensors, we expect to lose connection to one sensor per month at most.

For the experiments and evaluation (see chapter 8), we had to develop an additional software in order to meet some specific requirements. For the experiments we will need to

start measurements without a USB cable connected. Additionally, instead of taking one measurement after a period of ventilating, the sensor should take multiple measurements in a series while the fan is running. Therefore we had to implement Bluetooth. With that, we can start a measurement series with a variable number of measurements and intervals between and with variable ventilation start and stop times.

7.4 Server Application

As for the server application, the requirements are that it has to get the data from the sensors and make them accessible to the end-user by downloading a comma-separated values (CSV) file.

The server application consists of two separate Python scripts, each with its own responsibilities. The first one is using the "paho-mqtt" [7] module to create an MQTT client. This client subscribes to the broker of our application on The Things Network. When the client receives a message, the payload is decoded and appended to a CSV file together with the device's id and the `received_at` timestamp. The timestamp is the time when the message sent from a sensor was received by a gateway. Highly simplified, this script looks the following:

```
1  import paho.mqtt.client as mqtt
2
3  client = mqtt.Client()
4  client.on_connect = on_connect
5  client.on_message = on_message
6
7  client.username_pw_set('testnet-bern@ttn', '*******')
8  client.connect('eu1.cloud.thethings.network', 1883, 60)
9  client.loop_forever()
10
11 def on_connect(client, ...):
12     client.subscribe('v3/testnet-bern@ttn/devices/#')
13
14 def on_message(client, ..., msg):
15     payload = json.loads(msg.payload)
16
17     received_at = payload.get('received_at')
18     device_id = payload.get(...).get('device_id')
19     measurements = payload.get(...).get('frm_payload')
20
21     temperature, humidity = decode_and_convert(measurements)
22     write_to_csv([received_at, device_id, temperature, humidity])
23
24 def decode_and_convert(msg):
25     # decode base64 msg, convert to hex,
26     # split temperature from humidity,
27     # calculate C and % rh values
28
29 def write_to_csv(data):
30     # write data on new line to csv file
```

To grant the user access to the stored data, a second script is used, which is running a web server using the Flask [8] web framework. On this web page, the user can directly see the measurements in a diagram. Using a very simple filter, the user can specify what data he wants to be shown on that diagram. However, the most important thing is the download button for the CSV file. With this button, the user can download the CSV file to which all measurements are stored.

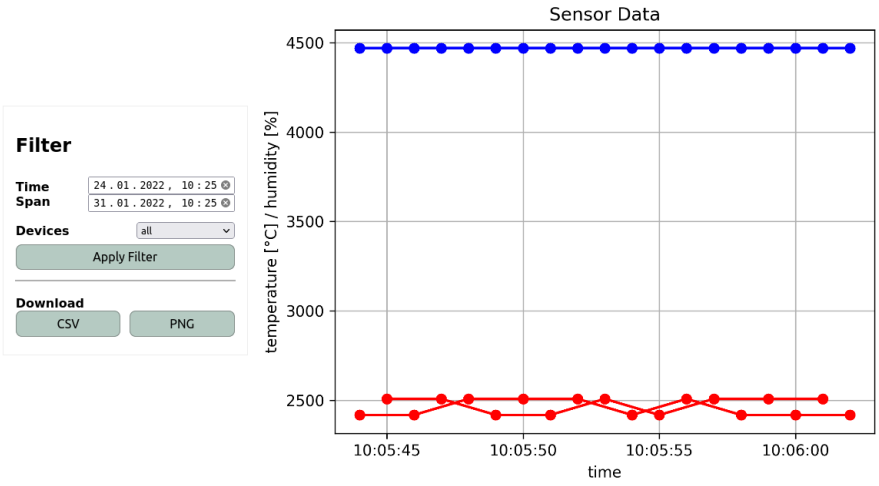


Figure 7.2: Web page built with Flask web framework

In the end, both scripts are running in a Docker [9] container that can be started through a Docker Compose script. This way, the server is easy to set up and rebuild if any changes were made to the code.

Chapter 8

Measurements and Evaluation

8.1 Research goals

The main goal of these experiments is to find an answer to the following question, that is:

> Does active ventilation improve the temperature measurements?

Besides that, we have also defined two secondary goals, which are finding the answers to the following questions:

> - Which fan provides the best performance?
> - How long should the fan be turned on?
> - What capacity should the battery have?

The first two questions will not only be evaluated based on the raw performance of the fans, but also based on the power consumption. This is important, since the sensor will be powered by a solar panel and, therefore, only has a limited amount of available energy. Finally, we will decide on which battery to use based on the required capacity. The capacity of the battery should allow the sensor to run for at least two weeks without recharging. We have three different batteries with different capacities to choose from.

8.2 Measurement Methods

8.2.1 Measurement Setup

For the ventilation experiments, the sensor is mounted indoors. This has the advantage over mounting it outdoors that the surrounding conditions only change a little even over a time span of several days. Therefore, different measurements can be compared easily. Furthermore, measurements can be repeated multiple times, allowing statistical analysis on the data.

Figure 8.1: Test setup with the sensor prototype mounted indoors

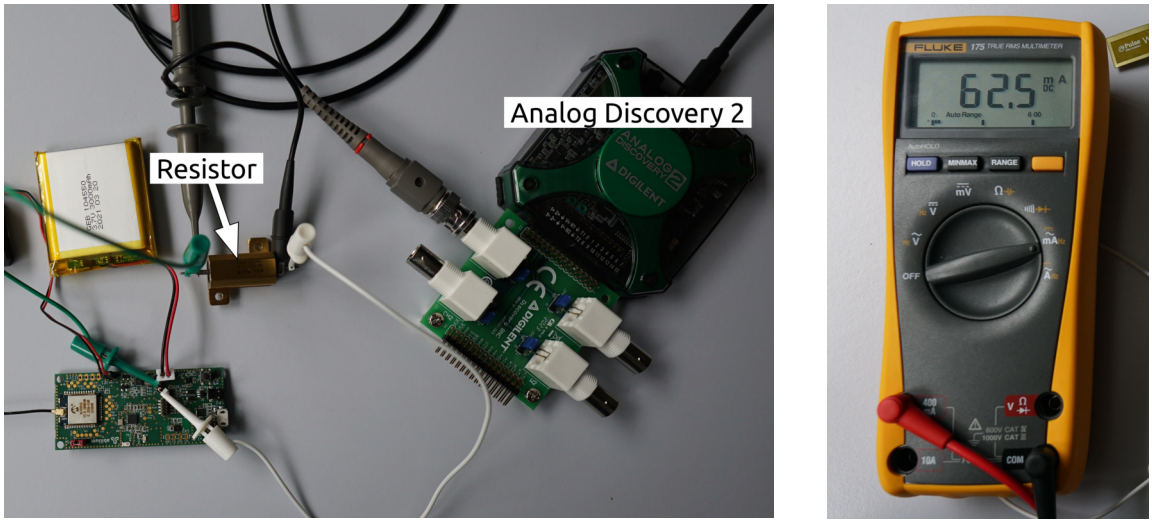
Additionally a Thunderboard from Silicon Labs is placed on the sensor. The Thunderboard is a microcontroller board with a temperature and humidity sensor. This sensor is used to measure the room temperature and humidity. These measurements will be used in the evaluation of the results and are also important during the heat accumulation simulation.

To measure the power consumption, we are using two measurement devices. The Digilent Analog Discovery 2 can measure voltages. To do so, a shunt resistor is interconnected at the positive terminal of the battery over which the voltage is measured. In the Digilent Waveforms software, the voltage value is converted to electric current with the equation of Ohm's law.

$$I = \frac{U}{R} \tag{8.1}$$

The Analog Discovery 2 is an oscilloscope, which is a device designed to measure the time domain accurately and not the amplitude. This allows us to create diagrams showing how

the power output changes over time, using the provided software. However, to get exact measurements of the electric current at the positive terminal of the battery we use the Fluke 175 True-RMS multimeter. The built-in current meter has accurate amplifiers to measure the amplitude.



(a) Analog Discovery 2 measuring the voltage on a resistor

(b) Fluke multimeter

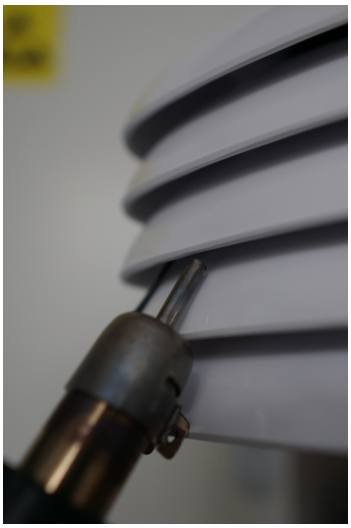
Figure 8.2: Setup used to visualize and measure the battery power output

For all power consumption measurements, the sensor board is running a production-ready software that is configured to take a measurement every two minutes. First, the fan is turned on for one minute while all other components are in sleep mode. Five seconds after the fan is turned off, the LoRa module is woken up from sleep mode. Then the sensor takes a measurement that gets transmitted via LoRaWAN after which the LoRa module is sent back to sleep mode. This process, from the LoRa module waking up to going back to sleep, lasts 15 seconds.

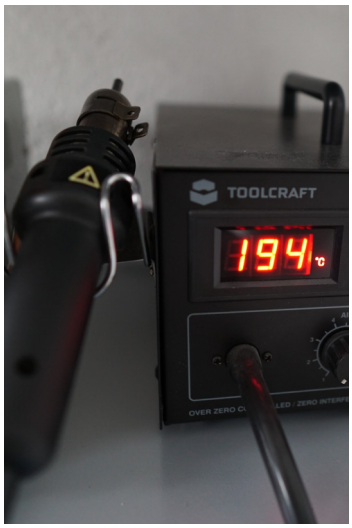
8.2.2 Heat Accumulation Simulation

As described in section 3.2, the previous sensors measured too high temperatures due to heat accumulation. This means, there was a temperature difference between the surrounding air and the air inside the sensor housing. Therefore, our goal is to simulate this heat accumulation as close as possible. As mentioned above, the room temperature is important here since the temperature surrounding the sensor housing needs to be lower than the temperature inside. Ideally, the surrounding air should keep its temperature during the the heat accumulation simulation. Therefore, a heat gun is used for the simulation. This allows to consistently heat up the inside air to the same temperature while the temperature of the surrounding air stays the same. In fact, throughout all the heat accumulation simulations we have done during our experiments, the room temperature consistently increased by only $0.1°C$. As described in section 8.2.1, we used a Thunderboard to measure the room tem-

perature. The temperature setting of the heat gun is set to 130° Celcius. The idea is that with a higher temperature setting, the air would heat up faster, making it more difficult to always turn off the heat gun at the same temperature. Contrary, using a lower temperature setting would increase the time required to reach the desired temperature.
In order to always heat up the inside temperature to the same level, an additional thermometer, shown in Figure 8.3c, is placed inside the housing. With that, we do not have to measure the temperature with the sensor during the heat-up process. This would require some sort of "live" temperature reading via Bluetooth. Implementing this would take much longer than just simply using an additional thermometer.



(a) Heat gun blowing air into the housing



(b) Temperature setting of the heat gun

(c) Additional thermometer

Figure 8.3: Heat accumulation simulation setup with a heat gun and thermometer

In order to evaluate if this method delivers the desired result, we will run a measurement series on the heat accumulation simulation. The series consists of 100 measurements, one every 5 seconds. Right from the beginning, we will heat up the inside temperature with the heat gun until the additional thermometer measures $35°C$. If this temperature is reached, we will turn off the heat gun and wait until the series is completed. In total, we are going to repeat this measurement series 5 times.

We have also considered different methods to simulate the heat accumulation. One option would be to use a climate chamber. This would have the benefit that the temperature and humidity could be controlled very precisely. However, it would be very difficult or even impossible to create a temperature difference from inside the sensor housing to outside. Another method would be to use a spotlight with a high heat output. Though, this would also increase the surrounding temperature, making it difficult to create a temperature difference.

8.2.3 Ventilation Experiments

We have designed two ventilation experiments, which are the following:

1. Experiment: Heat accumulation
   - Let the inside temperature level out with the room temperature as good as possible
   - Increase the inside temperature with the heat gun until the thermometer shows $35°C$
   - Start the measurement series consisting of
     - 200 measurements every 5 seconds
     - Start ventilation at 100 seconds

This first experiment is designed to measure how effectively each fan can push out hot air from inside the housing.

2. Experiment: Without heat accumulation
   - Let the inside temperature level out with the room temperature as good as possible
   - Start the measurement series consisting of
     - 200 measurements every 5 seconds
     - Start ventilation at 250 seconds
     - Stop ventilation at 750 seconds

This second experiment is designed to measure if ventilation has any negative effect if no heat accumulation is present, and therefore no ventilation would be required. Negative effects could be that ventilation is cooling down the air too much.

Both these experiments will be run 5 times with each fan and also 5 times without any ventilation. This allows to compare the fan's performances with each other while the measurements without ventilation show the effect of the ventilation in general. Besides that, repeating each measurement series 5 times allows to evaluate the data with a statistical approach. This way, when calculating the mean values, it will smooth out any outliers and show that one specific measurement series does not just show any very unlikely situation. In addition, the room temperature is measured and noted at the beginning of each measurement series.

8.3 Results

8.3.1 Power Measurements

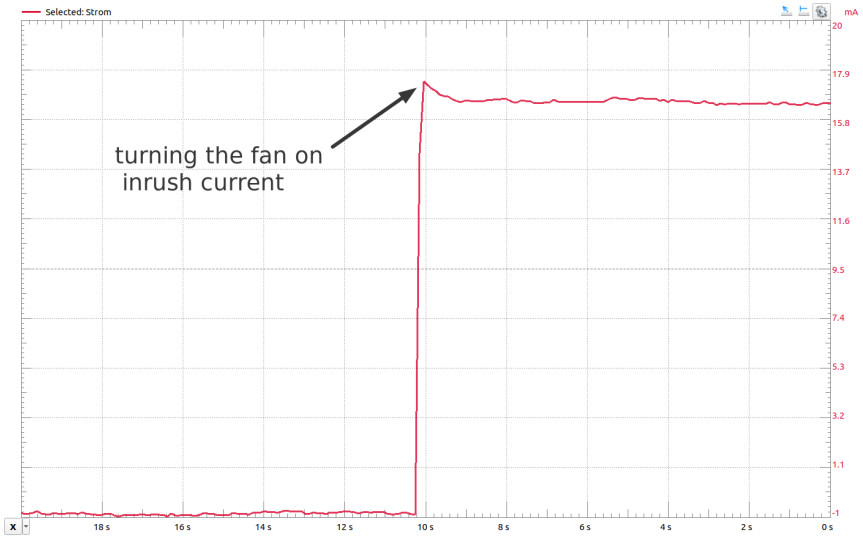
In this first subsection we present the results of the power measurements we did, using the methods described in section 8.2.1. The electric current output of the battery during the different phases are listed in table 8.1.

	electric current [mA]
small fan	16.50 ± 0.10
medium fan	61.60 ± 0.10
large fan	56.30 ± 0.10
sleep	0.06 ± 0.01
sensor measurement, data transmission	1.43 ± 0.01

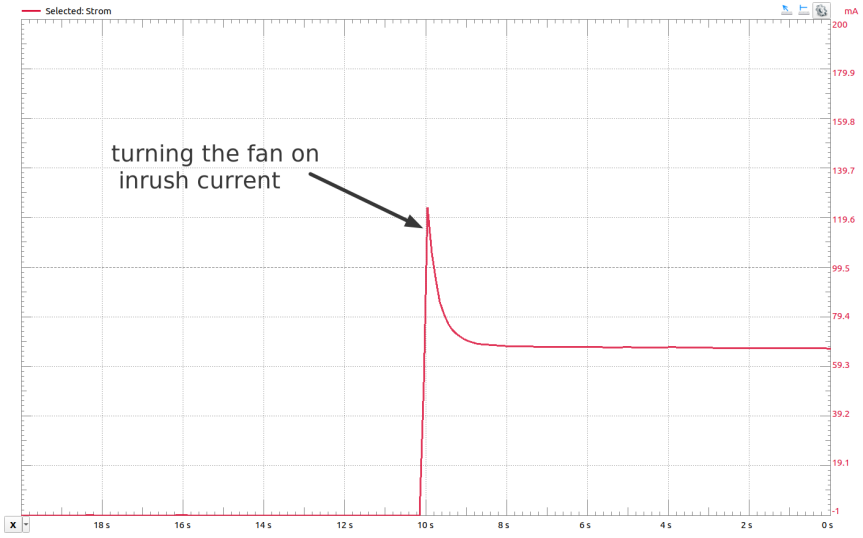
Table 8.1: Electric current output of the battery

These results show that taking a measurement and sending the data via LoRaWAN, during which the LoRa module is active, requires less than a tenth of the power required to run the small fan.

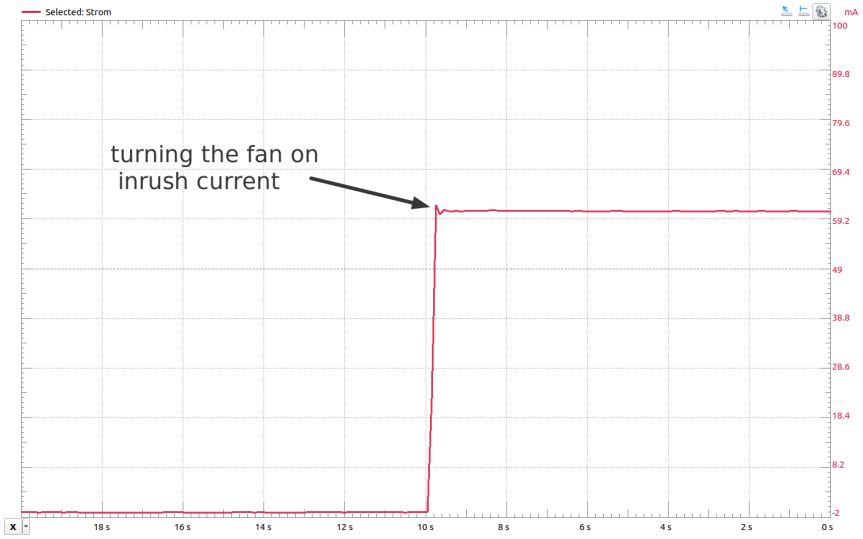
The diagrams in Figure 8.4 show how the power output of the battery changes over time when each fan is turned on. The power curve of the medium fan stands out as it shows a relative high peak at the beginning. However, regarding it lasts for less than one second it will only affect the overall power consumption very little. After that the curve stays level as it is also the case for the other fans.



(a) small fan



(b) medium fan



(c) large fan

Figure 8.4: Diagrams showing the electric current output of the battery over time while the three fans are turned on

The power curve for the measuring and data transmission phase, shown in Figure 8.5, is a bit more interesting. At the beginning of this time window (left side of the diagram) the LoRa module is in sleep mode. During this state, the power output of the battery is at $I = (0.06 \pm 0.01)\ mA$. After that, the LoRa module is woken up, which increases the power output to $I = (1.43 \pm 0.01)\ mA$. Next, a small peak is visible where the sensor is taking a measurement. After that, there are three larger peaks corresponding to the LoRa module sending the data and receiving confirmation. In the end, where the power output drops again, the LoRa module is going back to the sleep mode.

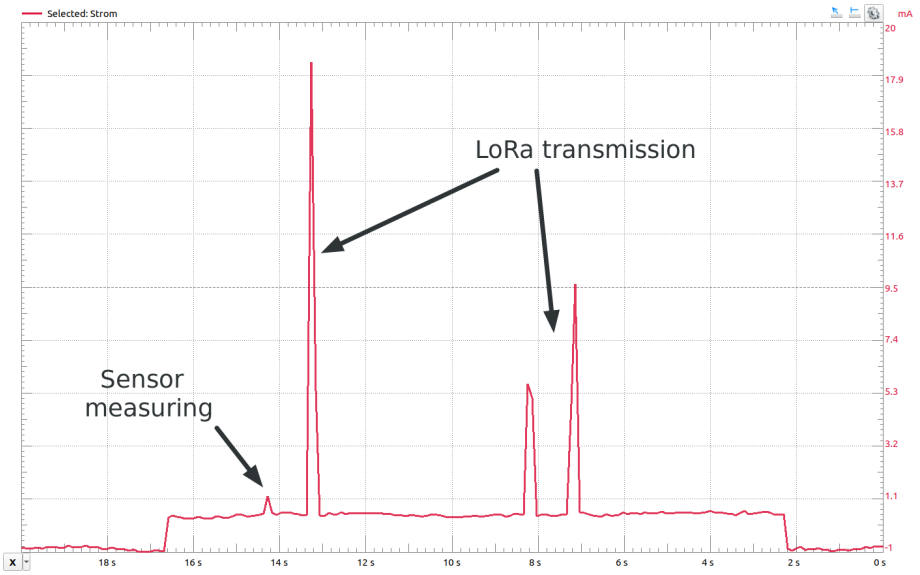


Figure 8.5: Diagram showing the electric current output of the battery over time during sensor reading and data transmission

As described in the previous section, these diagrams were created using the device from Analog Discovery, which is less accurate but makes for a good visualization. The time labels on the x-axis might be a bit confusing as new electric current values are drawn on the right side of the diagram. This means that the value measured ten seconds ago is at $t = 10s$.

8.3.2 Heat Accumulation Simulation

As mentioned before we repeated each measurement series 5 times. Therefore, all of the following diagrams show the mean value, calculated with equation 8.2, and the standard deviation, calculated with equation 8.3. For each experiment, these calculations are done with the 5 values measured at each $t = \{1, 2, 3, ...\}$, therefore is $n = 5$.

$$\overline{T} = \frac{1}{n} \sum_{i=1}^{n} T_i \tag{8.2}$$

$$\Delta T = \frac{1}{\sqrt{n-1}} \cdot \left(\sum_{i=1}^{n} (T_i - \overline{T})^2 \right)^{\frac{1}{2}} \tag{8.3}$$

Diagrams with all unprocessed measurements are included in Appendix A.

Figure 8.6 shows the temperature measurements during the heating process. Here the measurement series have been aligned such that the maximum temperature value of each series lies at $t = 95s$, before calculating the mean and standard deviation. Also the humidity measurements have been aligned using the same parameters. This means, that if the temperature measurements of one series are moved $10s$ to the right the humidity values are also shifted by $10s$ to the right.

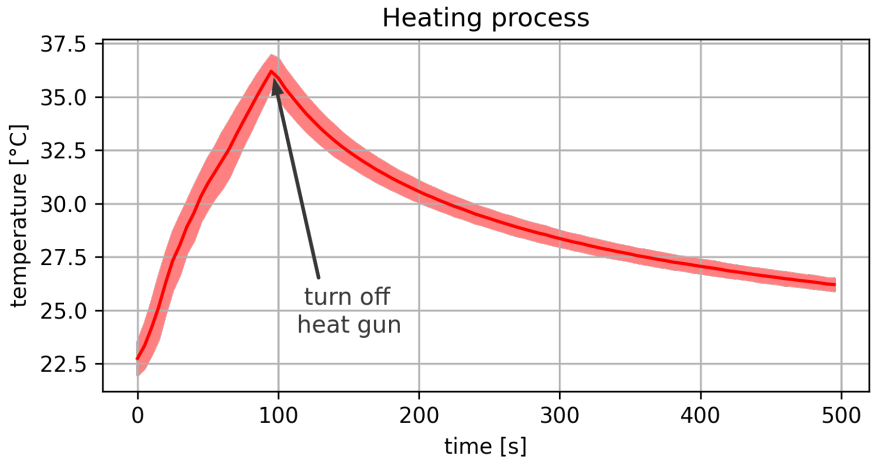


Figure 8.6: Temperature measurements during heat accumulation simulation

The diagram shows that while hot air is blown into the housing with the heat gun the temperature rises rather fast and starts to decrease after the heat gun is turned off. We can also see that the standard deviation is getting smaller from $t = 100s$ on until the end.

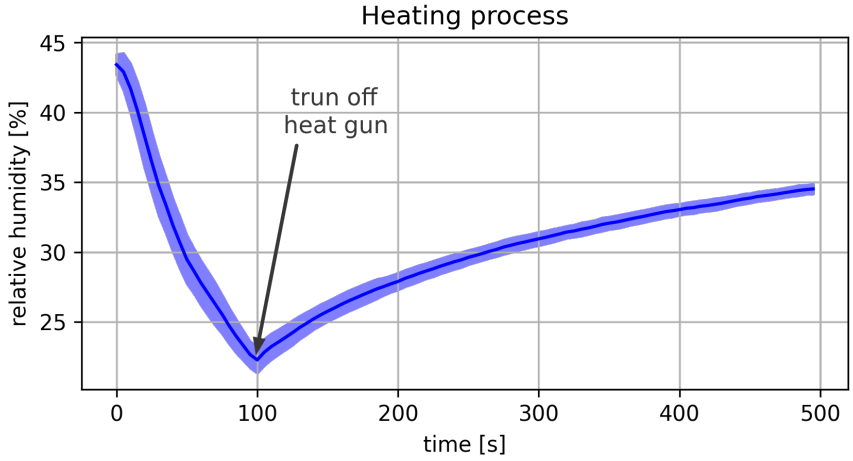


Figure 8.7: Humidity measurements during heat accumulation simulation

In Figure 8.7 we can see how the relative humidity changes during the heating process. It behaves analog but opposite to the temperature. The relative humidity is decreasing during the heating process and starts to increase after the heat gun is turned off. We can also see that the minimum lies at $t = 100s$ what is approximately $5s$ later than the temperature maximum.

8.3.3 Ventilation Experiments

Each of the following figures show the results from all the fans and without ventilation. The measurements in each diagram have been aligned according to the mean room temperature or the mean relative humidity. This has been done by calculating the mean of the room temperatures at which the experiments were run. For example, for experiment 1 with the medium fan we have measured 5 room temperatures. By using equation 8.2, we get the mean room temperature corresponding to the mean sensor temperature. As mentioned in section 8.2.3, the room temperature was measured at the beginning of each measurement series.

Figure 8.8 shows the temperature measurements of experiment 1, where a heat accumulation is simulated and ventilation starts at $t = 100s$. Without ventilation the temperature drop is decreasing over time as we have already seen in Figure 8.6. With the small fan, the temperature only decreases slightly faster at the start of ventilation. For the medium and large fans, the curve shows a much faster decrease in temperature.

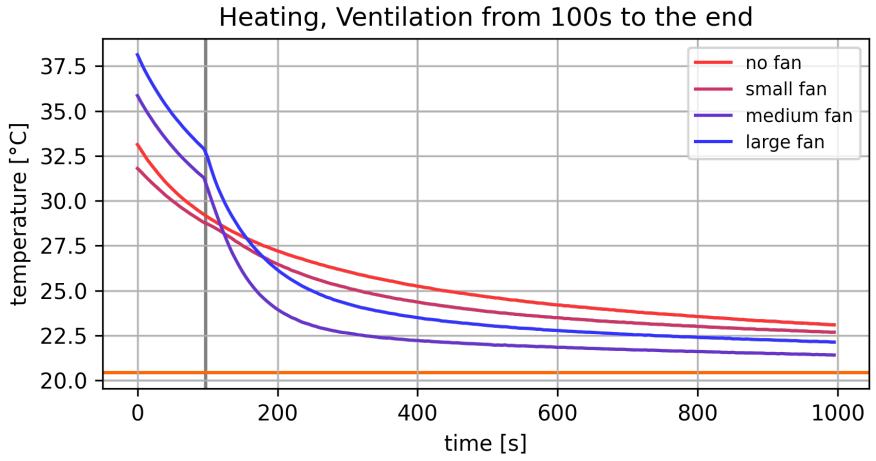


Figure 8.8: Temperature measurements of ventilation experiment 1

In Figure 8.9 we can see the humidity measurements of the different setups in experiment 1. As already seen with the heating process, the relative humidity behaves analog but opposite to the temperature. The small fan only has a small impact while the medium and large fans manage to increase the relative humidity quite fast.

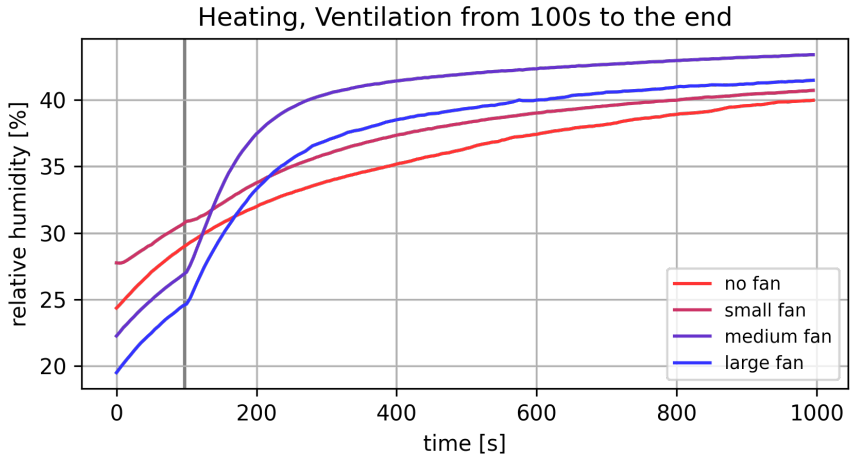


Figure 8.9: Relative humidity measurements of ventilation experiment 1

These two diagrams, with the results from experiment 1, show quite big differences in the starting temperatures and humidities. This is due to the fact that we did not quite manage to always start the series at the same time. After we were done with heating up, we had to turn off the heat gun and put it aside while also starting the series. And, as seen in Figure 8.6 and 8.7, the values change rather fast in the first few seconds after the heat gun is removed. Therefore starting one or two seconds earlier or later already makes a notable difference.

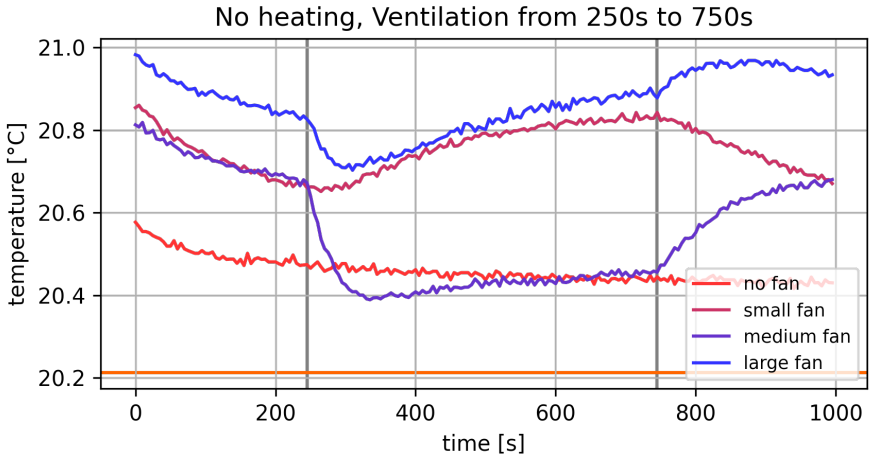


Figure 8.10: Temperature measurements of ventilation experiment 2

Figure 8.10 shows the temperature measurements of experiment 2, where we did not heat up the inside temperature prior to starting the measurement series. On this diagram, there are two interesting time points. The first is at $t = 250s$ where the fan is turned on. With

the small fan the temperature starts to increase almost immediately while with the other two fans it first decreases but then also starts to increase. The second interesting time point is at $t = 750s$ where the fan is turned off again. With the small fan the temperature starts to decrease while with the others it starts to increase.

Again the humidity curves, shown in Figure 8.11, behave analog but opposite to the temperatures.

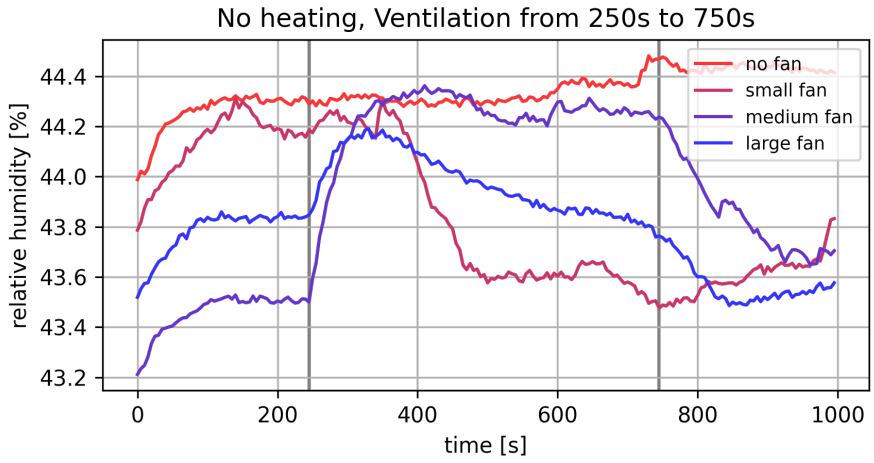


Figure 8.11: Relative humidity measurements of ventilation experiment 2

These two diagrams with the results form experiment 2 also show quite big differences in the starting temperatures and humidities. The reason for this is, that it takes very long for the temperature and humidity to level out to room conditions. Therefore, in order to always have the same starting conditions, we would have to wait probably for more than 30 minutes every time before starting a new measurement series. Nevertheless, the results still show some interesting behaviors, which are discussed in the next section.

As already mentioned previously, all unprocessed temperature and humidity measurements used to create the preceding diagrams, are included in Appendix A.

8.4 Discussion

As seen in Figure 8.6, using the heat gun delivers a consistent and reproducible method for the heat accumulation simulation. On the one hand this is shown at the maximum temperature, which is $T_{max} = (36.2 \pm 0.8)$ °C. This means that the difference between the maximum temperatures in every experiment we run is expected to be 1.6°C at most. On the other hand, as the air is cooling down, this expected difference is only getting smaller and smaller. Nevertheless, the chosen method has some limitations, which are mentioned when discussing the results from the ventilation experiments.

First, we will discuss the temperature measurements from both of the experiments. We start with experiment 1, where we did heat up the air at the beginning (see section 8.2.3). The diagram in Figure 8.8 shows how good the airflow is, these fans create. The small fan has nearly no effect as it only manages to create a weak airflow. This airflow does not speed up the cooling process a lot compared to without ventilation. The medium and large fans manage to push out the warmer air quite well as the significantly faster drop in temperature shows. Moreover, the medium fan performs the best, given that its curve is steeper for a longer period of time. Furthermore, its curve reaches a lower temperature as the large fans does. And already after 200 seconds of ventilation, at $t = 300s$, the temperature with the medium fan is substantially lower than it is without ventilation all the way at the end, although the starting temperature was higher.

However, none of the curves gets as low as the measured room temperature. One reason for this probably is, that we used the heat gun set to $130°C$. This means that during the heating process some spots of the housing potentially got quite a lot warmer than the air the sensor measured. Therefore it will take a lot longer until all this heat dissipates. This shows that after all, this method is not fully capable of simulating the real-world heat accumulation.

The diagram in Figure 8.10 shows the effect of the electric current, that is running from the battery pins through the circuit board to the fan. In case of the small fan we have seen in experiment 1 that it has almost no impact on the temperature measurements. With the results of experiment 2 we can see that this is also due to the fact that the circuit board is heating up. The temperature curve of the small fan starts to increase because the electricity causes the PCB to heat up. Hereby the sensor heats up more than the weak airflow can cool it down. From these two experiments, we can clearly say that the small fan is not suitable for this project. Comparing the temperature curves in Figure 8.10 of the medium and large fan. We can see that the medium fan not only manages to decrease the temperature more at the beginning of ventilation but it also manages to cancel out the heating effect of the electric current. Moreover, the medium fan decreases the temperature more than twice as much as the large fan although it only consumes 10% more power. The medium fan is also the best considering its curve is the one getting closest to the curve without ventilation. Also worth mentioning is that with ventilation the sensor does not

measure any temperature, which is below the room temperature. This means that with active ventilation we are most likely not going to distort the measurements in cases where there is no heat accumulation.

The fact that the large fan is not performing as well as the medium fan is most likely also due to its diameter being larger than the diameter of the 3D printed frame, as shown in Figure 6.4. Therefore, the print has a funnel shaped narrowing leading from the fan to the part holding the PCB. This does not seem to benefit the airflow at all.

One thing that stands out with the temperature diagram in Figure 8.10 is, that the measured room temperature is approximately $0.3°C$ lower than the temperature measured without ventilation. We suspect that this difference is due to the fact that inside a building there is always a small temperature gradient. Since the air is very static and with the design of the housing preventing warmer air from rising up, this results in a small heat accumulation. We do not expect this to be seen in an outside environment.

The main goal of the active ventilation for this sensor is to improve the temperature measurements. Since we also do not have any measurements from the previous version sensor on how the humidity behaves under heat accumulation, the humidity is only secondary in this evaluation. However, regarding that the humidity curves all show an analog behavior to the temperature curves, we expect that all the things said about the temperatures can be said analog about the humidities. This means that also the humidity measurements see an improvement with active ventilation and will not get distorted by it.

Lastly, we wanted to determine which battery to use. Comparing the measurements listed in table 8.4 with each other, we can see that the process of taking a measurement and sending the data, that lasts for 15 seconds, requires less power than even powering the small fan for only 2 seconds. This means that the biggest power consumer will be the fan and not the LoRa module. This justifies the approach of sending the data after every measurement.

To specify how long the fan can be turned on we also need to get an understanding on how much power will be consumed and produced. We have done this with the following example calculation.
With the sensor configured to take one measurement per hour and ventilating for five minutes in advance the electric charge required for one cycle is:

$$\begin{aligned} Q &= \int_{t_0}^{t_1} I \cdot dt = \int_0^1 I_v \cdot I_l \cdot I_i \cdot dt \\ &\approx \frac{1}{3600} \cdot (300s \cdot 61.6mA + 15s \cdot 1.43mA + 3285s \cdot 0.06mA) \\ &= 5.19\ mAh \end{aligned} \tag{8.4}$$

With I_v being the electric current output of the battery during ventilation, I_l during the LoRa module being active and I_i during the idle state.

For one day of measuring this totals to:

$$E = (124.6 \pm 0.1)\ mAh \tag{8.5}$$

This means that we have to use a battery with a capacity of at least $14 \cdot 125 mAh = 1750\ mAh$ in order to have enough power for 14 days without recharging. Out of the three batteries we have in our lab, only one has a higher capacity with 3000 mAh. With this battery the sensor can even take measurements for 24 days without requiring to recharge. Nevertheless, we will also connect a solar panel with a maximum power output of 1.5 watts to the sensor in order to recharge. When looking at the climate standard values for Bern, published by Meteo Schweiz [10], one can see that December is the darkest month of the year with an average of 1.7 hours of direct sunlight per day. With the battery charging at 9 volts this gives the following amount of produced power:

$$\begin{aligned} E &= 1.5 watt \cdot 1.7 h \cdot \frac{1}{9V} \cdot 1000 \\ &= (283.3 \pm 0.1)\ mAh \end{aligned} \tag{8.6}$$

Even this is enough electricity for two days of measuring. In total, the sensor should have enough power to run the whole year round. If for some reason the electricity produced turns out to be too little we could always use a bigger solar panel.

Concluding, with the measurements we have done, we could show that the medium fan delivers the best performance. It manages to effectively eliminate heat accumulation from inside the sensor housing, and the extra power it requires to run over the others is well justified by its superior performance. Based on the experiments we suggest that the fan should be turned on for 250 to 300 seconds before a measurement is taken. This depends on how many measurements should be taken per day, regarding the overall power consumption. However, 250 to 300 seconds is long enough to eliminate the heat effectively while ventilating for any longer would only have a very minimal effect.
We could also determine that we will use a battery with a capacity of 3000 mAh. This is enough power for the sensor to run for 24 days with out recharging.
Ultimately, with the measurements we could show that active ventilation improves the temperature measurements of this low-cost sensor.

Chapter 9

Conclusion

We have designed the architecture for a new low-cost sensor with active ventilation, capable of wireless communication. Subsequently, we designed a prototype sensor with a radiation shield, a solar panel, active ventilation and LoRaWAN networking, while meeting the concept of being low-cost. We have implemented the software for the sensor and developed a server application. This server application collects the data of all sensors and makes them accessible to the end user. Finally, we run several experiments in our laboratory, measuring and evaluating the effect of the active ventilation.

In this evaluation we could show that active ventilation helps to improve the measurement accuracy of the sensor under lab conditions. We determined which fan provides the best performance out of three fans tested. Additionally, we measured the power consumption of the sensor in order to determine what capacity the battery needs to have. With the power measurements we could also show that the used solar panel can produce enough power for our sensor. We were also able to show that the wireless communication via LoRaWAN works as intended.

Nevertheless, our experiments had some limitations in terms of parameters and conditions we could simulate. In our lab we could only simulate positive temperatures and did not have the ability to change the surrounding conditions.

However, this sensor is an improvement over the previous version as the results of our measurements in our lab have shown.

Chapter 10

Future Work

In our lab, our experiments had some limitations in terms of conditions. Therefore, in future work we will run experiments with the sensors placed outside. Specifically, we want to place five of our sensors next to high quality sensors operated by institutions like Meteo Schweiz. We can then compare the measurements of our low-cost sensor with the very precise measurements of these high quality sensors. This will show how the active ventilation behaves in a much broader range of different conditions. For example, how active ventilation affects the measurements at negative or very hot temperatures. Moreover, we will also be able to see the effects of active ventilation with different relative air humidity.

With the prototype we have built and used for our experiments, the fan is placed in such a way that it blows air up towards the sensor. As described in section 6.3, this is conceivably the best option since warmer air is rising up. Nevertheless, in future work it would also be interesting to evaluate the other configurations (pull down, pull up, push down). With these results we could then justify more specific why to use which configuration and potentially even gain an improvement in ventilation performance.

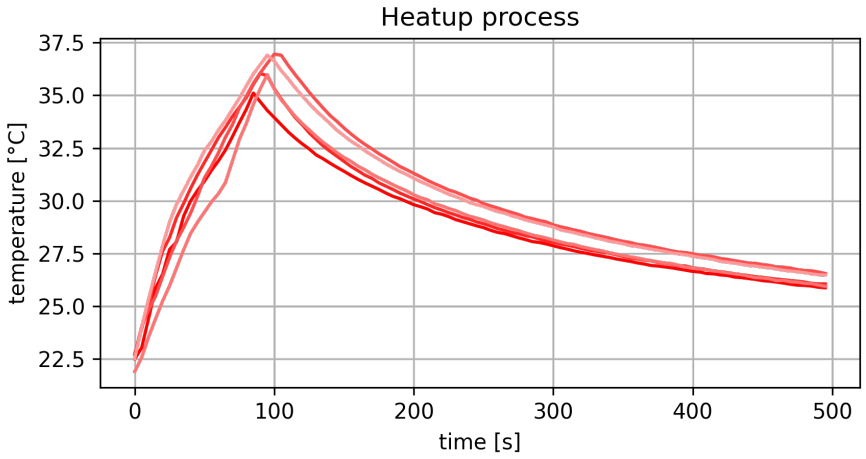
While this sensor is an improvement over the previous version we still see some possibilities on how to improve it even further with adaptive ventilation. One way would be to measure the temperature during the ventilation. Then, if the absolute temperature difference between two measurements falls below a certain value, the fan could be turned off. As seen with the results of both ventilation experiments, Figures 8.8 and 8.10, some circumstances require less ventilation than others or even none at all.
Another option can be derived from the research described in section 3.2 since heat accumulation was only observed during daytime. This means that active ventilation is only really required during daytime and at night the fan could be kept turned off. Hereby we could gain a huge reduction in power requirements and reduce the risk of distorting measurements. However, this requires the time of the microcontroller to be synchronized. Besides that, the controller would also have to know the times of sunrise and sunset based on the current date regarding the fact that throughout a full year the times of sunrise and sunset vary quite strong.

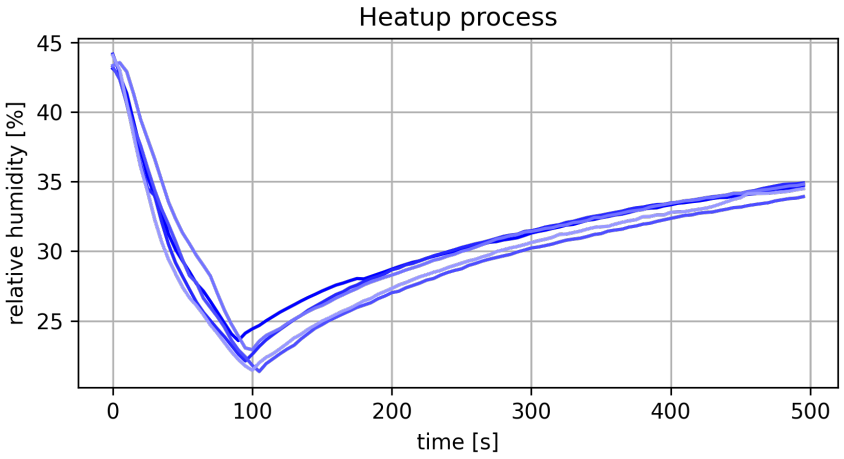
A third option would be by ventilating only if the solar panel is producing a certain amount of power. The solar panel producing power suggests that the sun is shining with the effect of heat accumulation. However, this would require small adjustments to the PCB in order for the microcontroller to be able to read the power output of the solar panel.

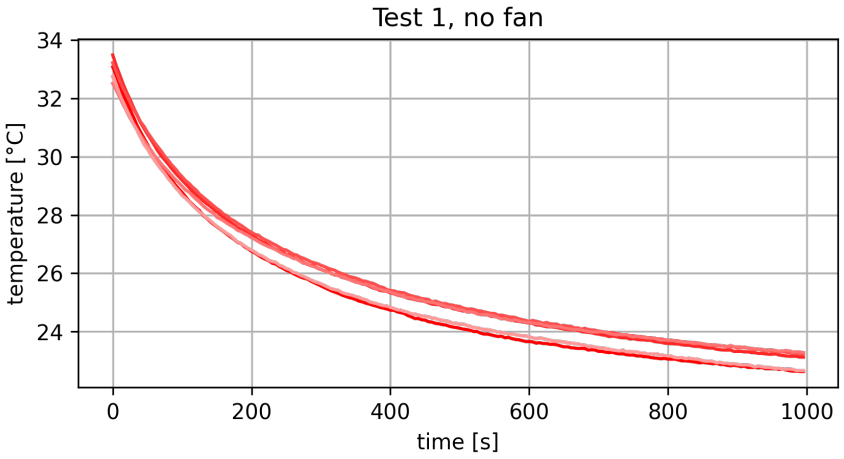
Appendix A

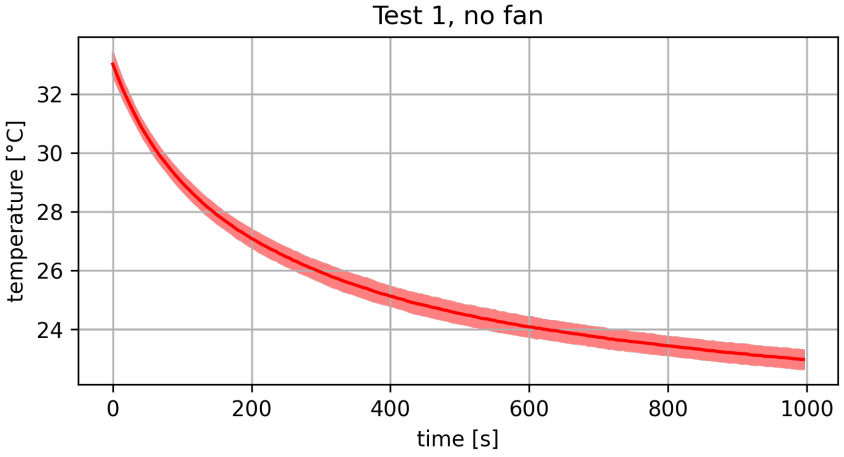
Measurements

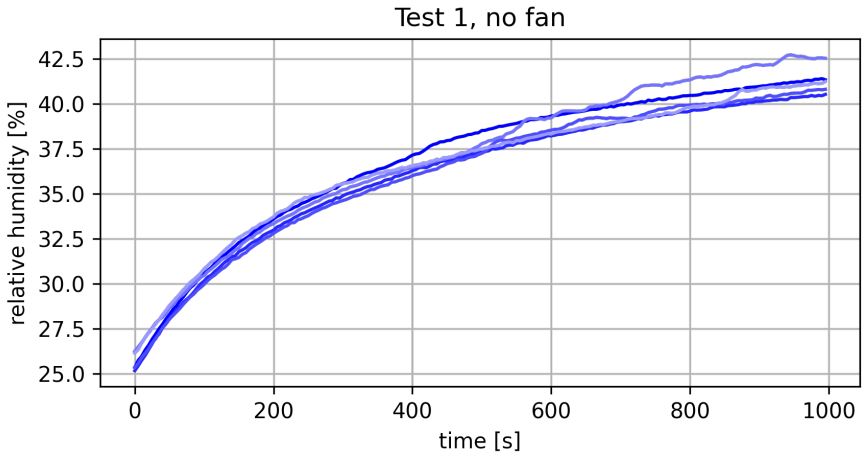
On the following pages are diagrams with all results from the experiments. The first two diagrams show the unprocessed results of the heat accumulation simulation. All diagrams afterward show the results from the ventilation experiments. There are always two corresponding diagrams. The first show the unprocessed measurements, and the second shows the mean value with the standard deviation. The mean values are those shown in the diagrams of section 8.3.

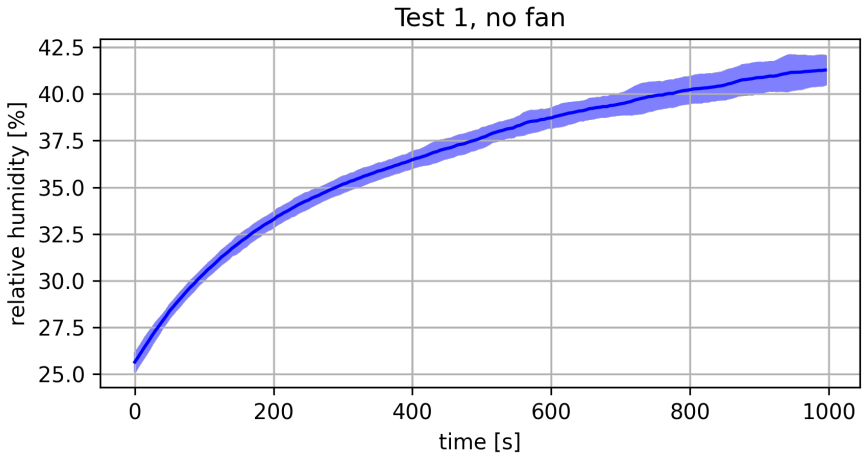


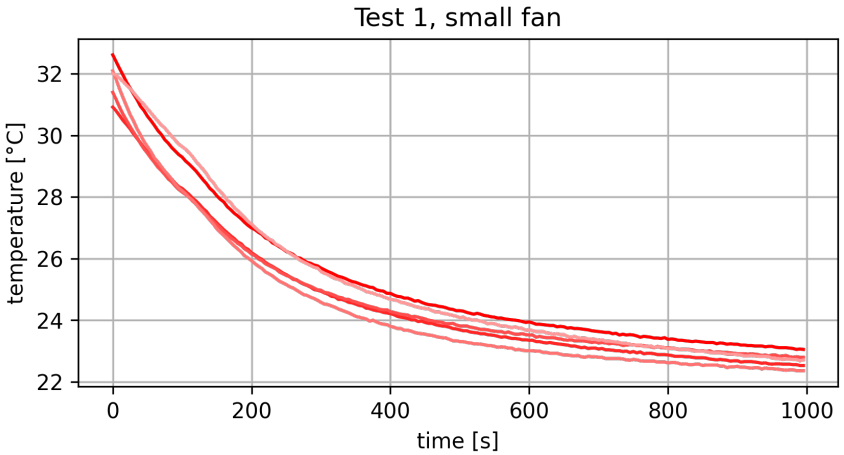

Heatup process
relative humidity [%]
45
40
35
30
25
0
100
200
300
400
500
time [s]

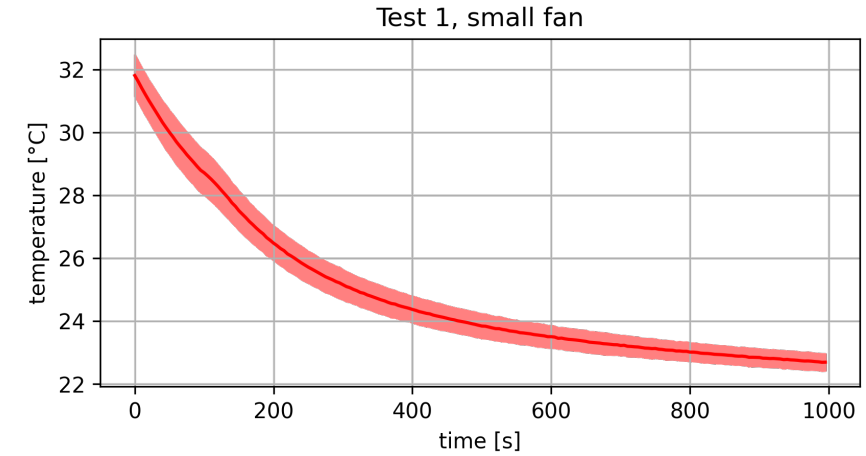

Test 1, no fan
temperature [°C]
34
32
30
28
26
24
0
200
400
600
800
1000
time [s]

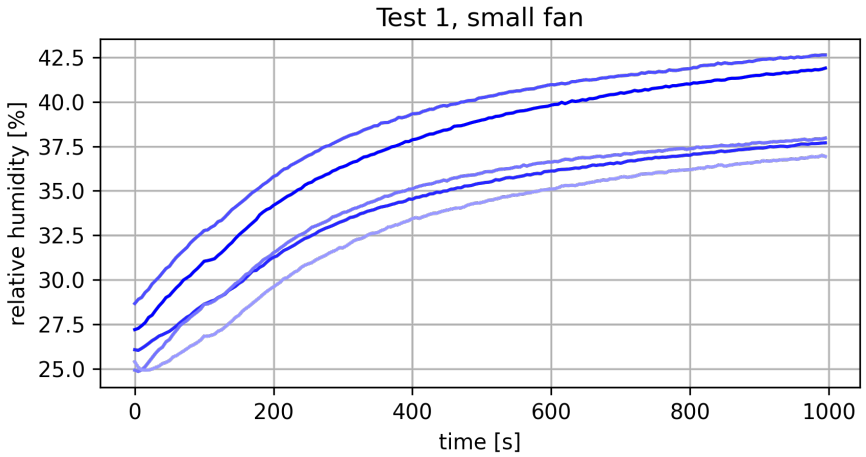

Test 1, no fan
temperature [°C]
32
30
28
26
24
0
200
400
600
800
1000
time [s]

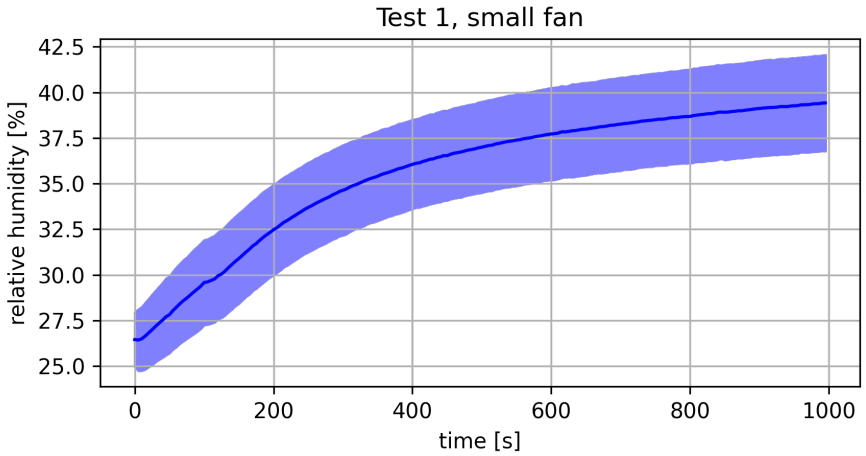

Test 1, no fan
relative humidity [%]
42.5
40.0
37.5
35.0
32.5
30.0
27.5
25.0
0
200
400
600
800
1000
time [s]

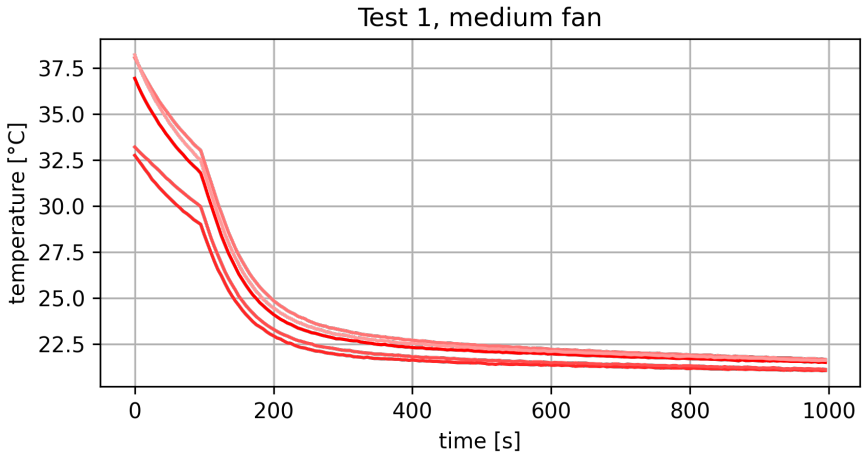

Test 1, no fan
relative humidity [%]
42.5
40.0
37.5
35.0
32.5
30.0
27.5
25.0
0
200
400
600
800
1000
time [s]

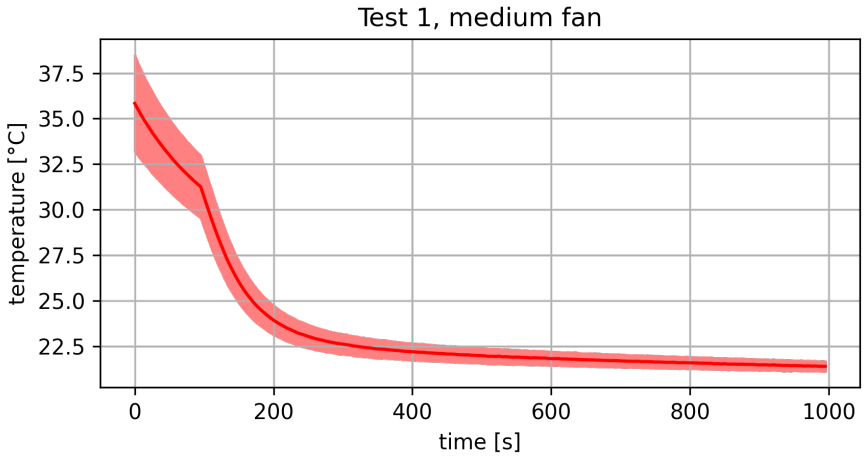

Test 1, small fan
temperature [°C]
32
30
28
26
24
22
0
200
400
600
800
1000
time [s]

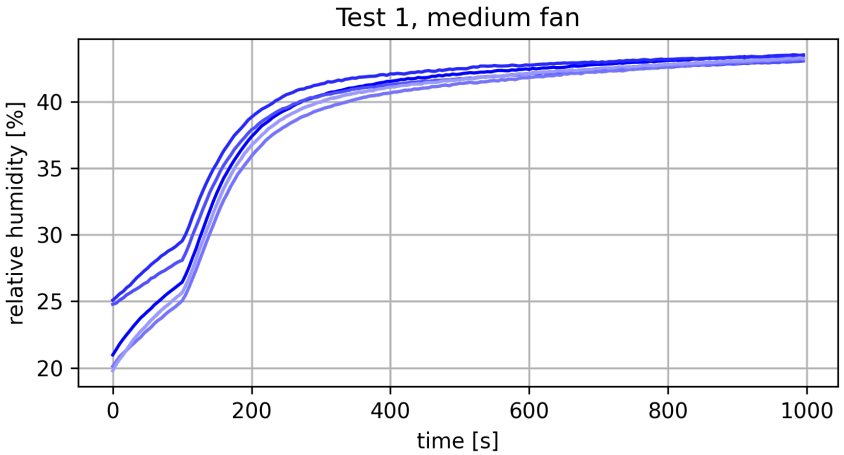

Test 1, small fan
temperature [°C]
32
30
28
26
24
22
0
200
400
600
800
1000
time [s]

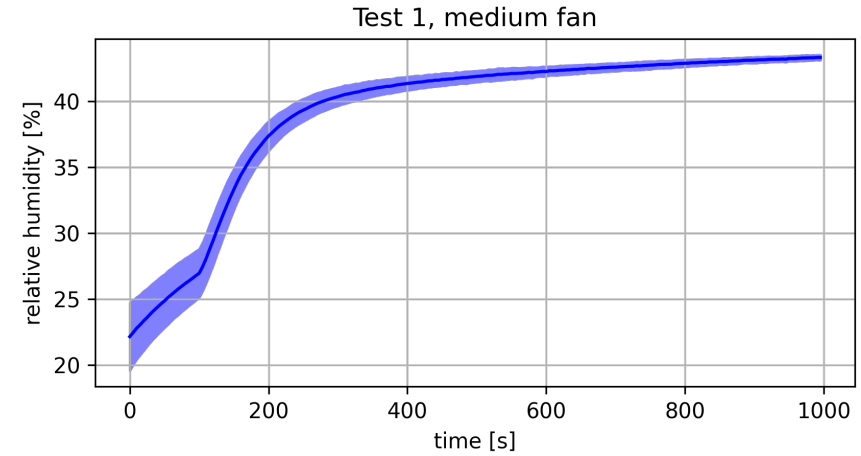

Test 1, small fan
relative humidity [%]
42.5
40.0
37.5
35.0
32.5
30.0
27.5
25.0
0
200
400
600
800
1000
time [s]

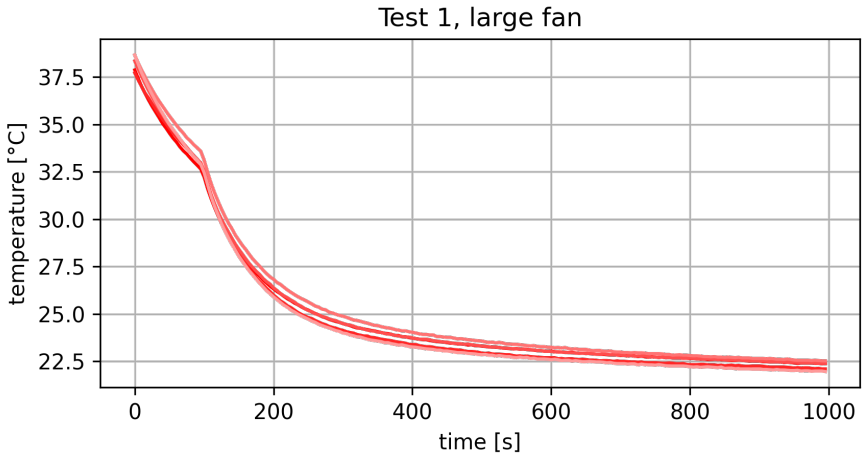

Test 1, small fan
relative humidity [%]
42.5
40.0
37.5
35.0
32.5
30.0
27.5
25.0
0
200
400
600
800
1000
time [s]

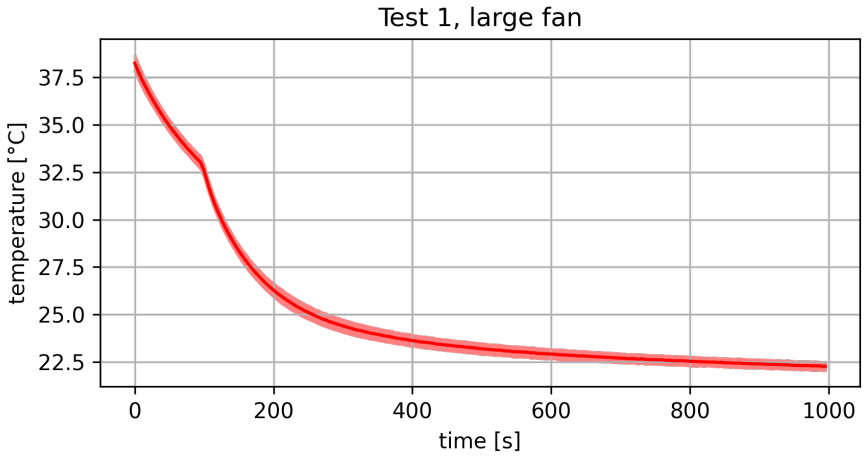

Test 1, medium fan
temperature [°C]
37.5
35.0
32.5
30.0
27.5
25.0
22.5
0
200
400
600
800
1000
time [s]

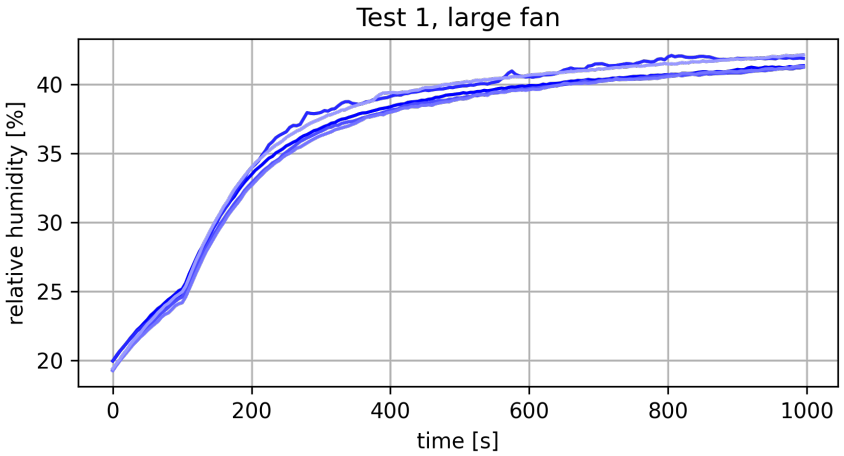

Test 1, medium fan
temperature [°C]
37.5
35.0
32.5
30.0
27.5
25.0
22.5
0
200
400
600
800
1000
time [s]

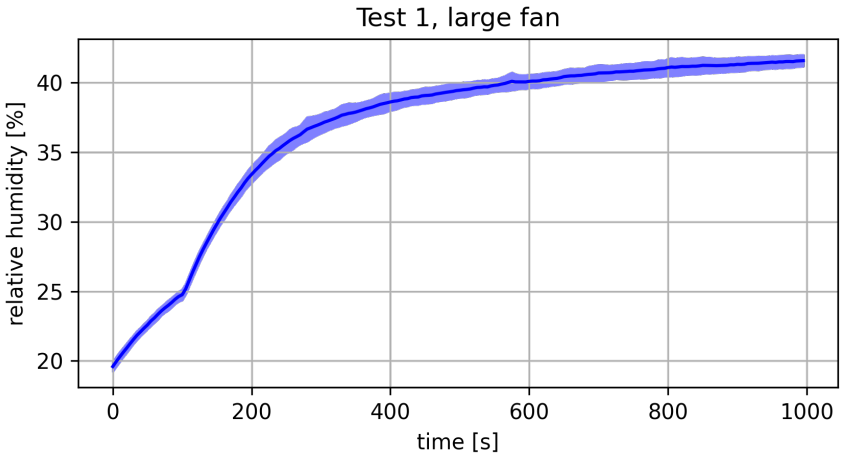

Test 1, medium fan
relative humidity [%]
40
35
30
25
20
0
200
400
600
800
1000
time [s]


Test 1, medium fan
relative humidity [%]
20
25
30
35
40
0
200
400
600
800
1000
time [s]


Test 1, large fan
temperature [°C]
22.5
25.0
27.5
30.0
32.5
35.0
37.5
0
200
400
600
800
1000
time [s]


Test 1, large fan
temperature [°C]
22.5
25.0
27.5
30.0
32.5
35.0
37.5
0
200
400
600
800
1000
time [s]


Test 1, large fan
relative humidity [%]
20
25
30
35
40
0
200
400
600
800
1000
time [s]


Test 1, large fan
relative humidity [%]
20
25
30
35
40
0
200
400
600
800
1000
time [s]

Bibliography

[1] M. Gubler, A. Christen, J. Remund, and S. Brönnimann, "Evaluation and application of a low-cost measurement network to study intra-urban temperature differences during summer 2018 in bern, switzerland," *Urban Climate*, vol. 37, p. 100817, 2021.

[2] A. Yegin, T. Kramp, P. Dufour, R. Gupta, R. Soss, O. Hersent, D. Hunt, and N. Sornin, "3 - lorawan protocol: specifications, security, and capabilities," in *LPWAN Technologies for IoT and M2M Applications* (B. S. Chaudhari and M. Zennaro, eds.), pp. 37–63, Academic Press, 2020.

[3] "The things network." `https://thethingsnetwork.org`.

[4] "Mqtt." `https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html`.

[5] "Silicon labs efr32mg21b020f1024im32." `https://www.silabs.com/wireless/zigbee/efr32mg21-series-2-socs/device.efr32mg21b020f1024im32`.

[6] "Simplicity studio." `https://www.silabs.com/developers/simplicity-studio`.

[7] "Paho mqtt." `https://pypi.org/project/paho-mqtt`.

[8] "Flask." `https://flask.palletsprojects.com`.

[9] "Docker." `https://www.docker.com`.

[10] "Bundesamt für meteorologie und klimatologie meteoschweiz." `https://www.meteoschweiz.admin.ch/home/klima/schweizer-klima-im-detail/klima-normwerte/normwerte-pro-messgroesse-und-station.html`.