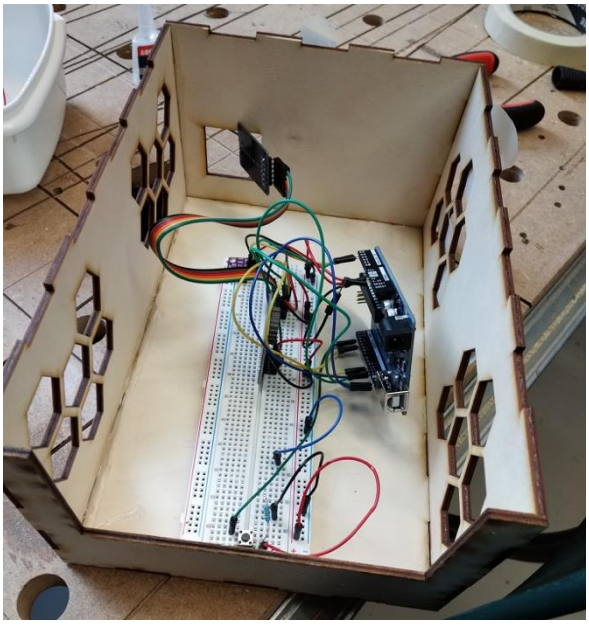
Projekt 1:Sensors and data

KRISTOFFER ZANCHETTA KLERCKE S183633
FREDERIK HOLM NIELSEN S173864
MARIUS HAVALESKA LYHNE S183655
VIKTOR HOLM S165273
GR8.A



41030 Mechatronics

DTU | ANKER ENGELUNDS VEJ 1, BYGNING 101A, 2800 KGS. LYNGBY

Table of Contents

Introduction

This project has the purpose of introducing Mechatronics through project work. This project specifically focuses on the interaction between electronics, software and prototyping. Namely, this Project 1 is about the creation of a weather station using basic external components and an enclosure of this. The weather station is to gather data of humidity, air pressure and temperature with variation in the logging frequency depending on the time of the day. Since this project is executed in relation with a university course, the ultimate purpose is, of course, education in this specific field of mechatronics. This project also serves the purpose of showcasing our acquired competency within the field of programming, data logging, rapid prototyping and use of sensors.

The project has been executed by Group 8 consisting of four students in association with the course 41030 Mechatronics Engineering Design. The project was effectuated between 19/02/2020 and 11/03/2020.

To see code and data, check appendix and the GitHub page for this project: https://github.com/Frehoni/Lads/tree/master/Projekt%201

Methods and setup

The Arduino build is made from the following list of components:

Component list

Arduino UNO

Arduino UNO is a basic microcontroller[1] and is typically used for smaller mechatronic projects. We chose to use it because it suits the complexity of the project.

Gy-BME280 sensor

The BME280 sensor is a combined temperature, humidity and pressure sensor[2]. The compact module is key for the project because it encompasses all the functions of a weather station.

DS1307 RTC

Real-Time-Clock is a clock and calendar module[3]. This is a standard component, and is powered by a battery, meaning the RTC stays up to date even when the Arduino is not powered.

[1]Arduino UNO specs: https://store.arduino.cc/arduino-uno-rev3
[2] BME280 specs: https://protosupplies.com/product/gy-bme280-pressure-humidity-temperature-sensor-module/
[3]Datasheet of RTC: https://datasheets.maximintegrated.com/en/ds/DS1307.pdf

MicroSD card reader and MicroSD card

The card reader module can read and write in a MicroSD card[4] and is therefore useful for logging data.

Breadboard, wires, resistors and button

These are all standard components in the avid Arduino builder's arsenal. The breadboard and wires connect the microcontroller to the external components. Resistors ensure that the voltage delivered to certain components is reduced to the required voltage of the component. We chose to use a button to switch between the two states of data collection.

Production methods

To create the casing, we use a laser cutter to cut 6 panels of a box. The design requirements are; an easy method of assembling the box and for air to flow through the box, so that the environment around the sensor is accurate to the room. We therefore found a pattern from MakerCase.com[5] which allows easy assembly. Thereafter we add the air-flow pattern in Illustrator. Finally, we use the laser cutter in SkyLab to cut out the patterns from a 3 mm slate of birch plywood.

Process

The hardware was assembled, inputs were noted and RTC was calibrated to run on current time. Using the examples provided by the component libraries, the functions of the components were tested. With the obtained knowledge, we then worked on the general architecture of our script. The architecture of the script consists of 4 documents, two functions, a main script, and a document of variables. Thereby dividing the program into manageable parts. This made it easy for us to write the code necessary for fulfilling the given assignments in a relatively short amount of time.

At this point, we ran program A, and sampled data on a staircase. Afterwards, we switched to program B, for the data-collection marathon. Unfortunately we discovered that had been a switch from to program A to B while gathering data, rendering the gathered data of part B useless.

We suspected that this error was due to the potentiometer controlling the active program, with a growing suspicion of a loose connection between the breadboard and potentiometer.

[4] SD Card Board: https://www.trab.dk/da/breakout/31-micro-sd-board.html
[5] https://www.makercase.com/#/

Our solution was to use a more reliable button. This however brought the challenge of making our script switch between the two modes instantly. Luckily, we were able to resolve this problem by using the “millis” function, instead of the previously used “delay” function.

Results

For part A of the data collection, our device was powered up in Viktors apartment on the 3rd floor, and carried down the stairs, into the basement, and back up again. We sampled data every 2 seconds, to ensure that we would get plenty of data, on the brief climb down and up the stairs. The data was successfully collected and plotted in the diagram shown below:

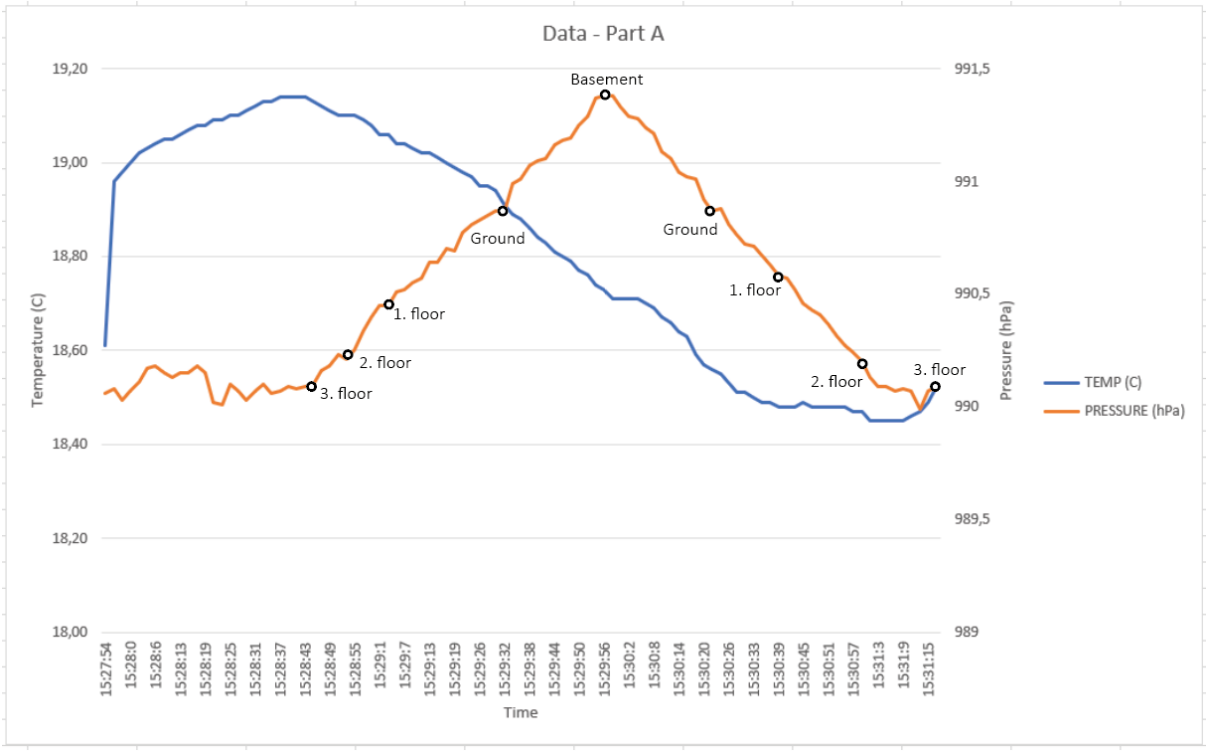


Figure 1

Due to the fact that we started the measurements on the 3rd floor, the measurements for the basement level are situated relatively close to the centre of the graph. The maximum pressure, which is expected to be experienced closest to the basement level, was 991.83 hPa measured at 15:29:58. The pressure on the top floor was measured at approx. 990.1. This gives a difference in pressure of 1.73 hPa. Using the ‘hypsometric formula[6]’ this gives Viktors apartment an altitude of around 14 meters, relative to the basement.
The graph also clearly shows the process of the data sampling. The device was switched on, and then picked up, which is visible due to the sudden rise of temperature in the beginning.

[6] https://www.mide.com/air-pressure-at-altitude-calculator

The trip up and down the stairs is clearly shown when looking at the pressure part of the graph. It is also noteworthy, that the door was opened at the bottom of the stairs, which could explain the more or less constantly falling temperature.

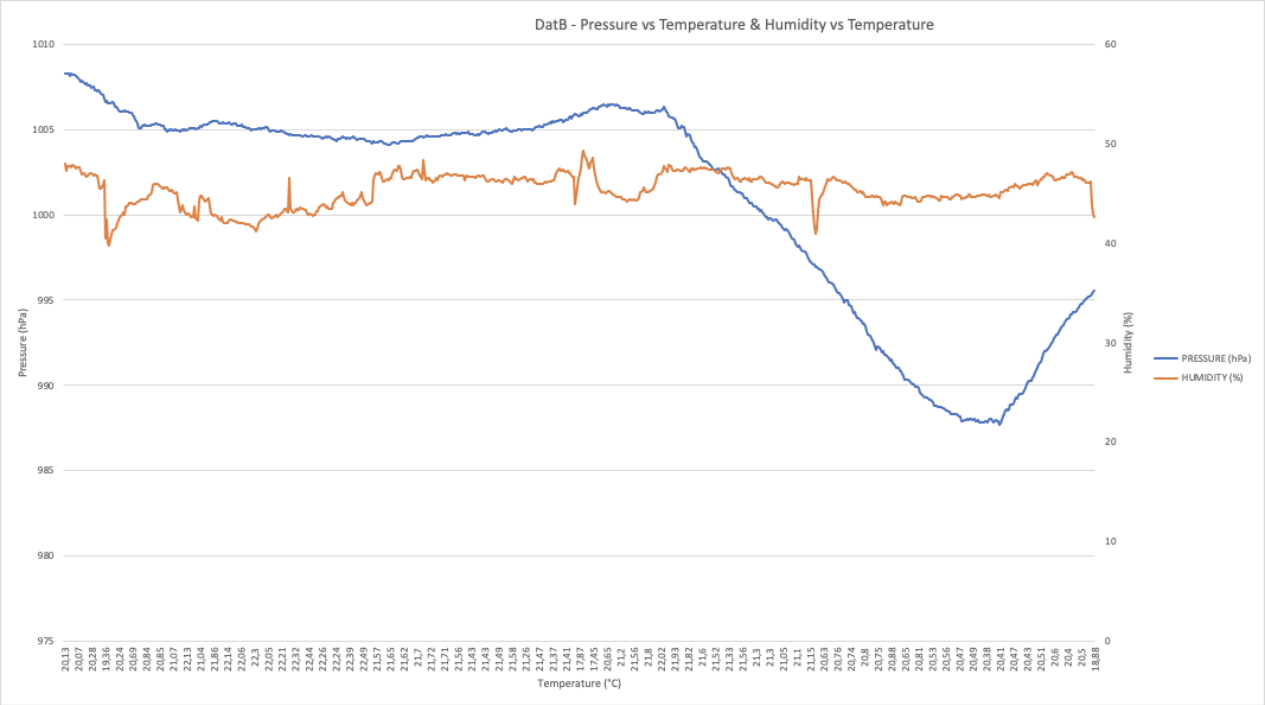


Figure 2

For part B of the data collection, the device was once again the setup in Viktor's apartment on the 3rd floor. This time it was placed in Viktor's bedroom for a duration of three days, where the humidity was expected to vary as a function of time. The logged data is graphed underneath. The first graph shows the air pressure and humidity as a function of temperature, whereas the second graph shows the air pressure, humidity and temperature as a function of time:

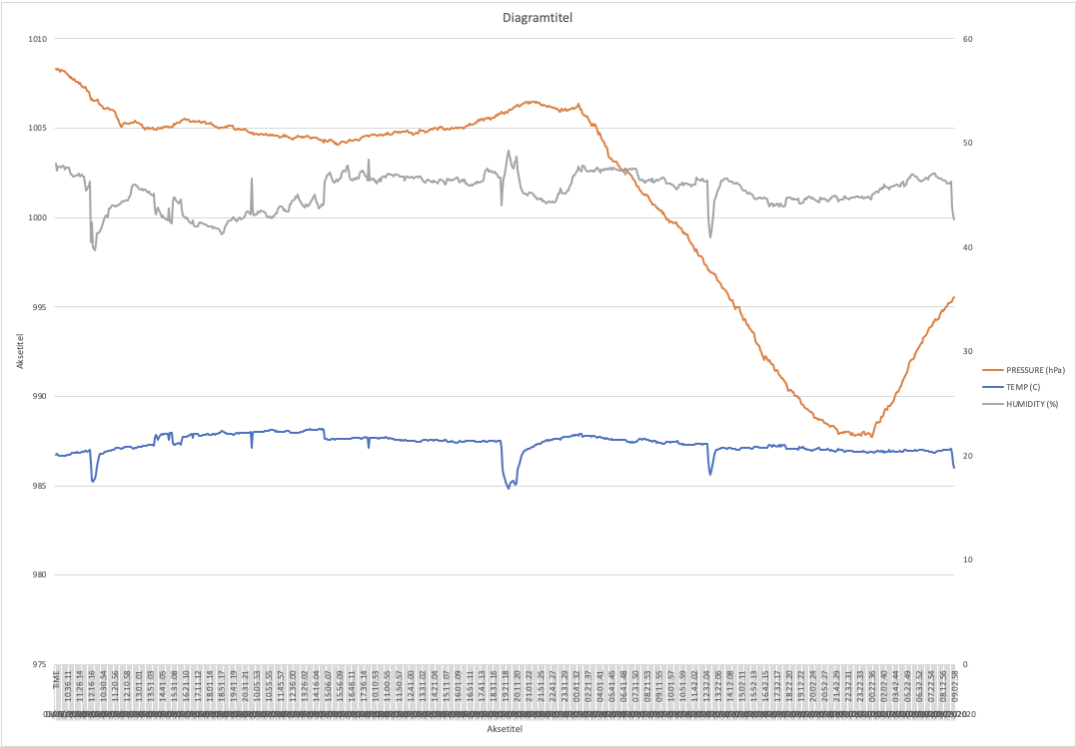


Figure 3

During daytime, the setup recorded data every five minutes and during nighttime (24:00-06:00), the setup only sampled data every ten minutes. The graphs of temperature and humidity clearly shows a connection between these two entities - when the temperature drops, so does the humidity. This has to do with the *specific humidity* of saturated air in a state of equilibrium in Viktor's apartment. The graph of the air pressure depicts changes in pressure relative to a change in altitude of about 750 meters even though the device remained stationary in one place, which equates to approximately three of the Great Belt Bridge's pylons stacked on top of each other.

Discussion

The results presented in part A are a realistic depiction of the changes in pressure experienced when walking up and down the stairs. The drop in temperature is solely caused by the interruption of the room temperature inside from the outside air. These results also relatively clearly depicts the drop in air pressure connected to the descent in each set of stairs.

The results presented in part B are more of a mystery. Although our interpretation of the relation between temperature and humidity is depicted in the graphs, it still contains spikes in both of these quantities. The drop in air pressure might have something to do with atmospheric pressure connected with changes in weather. Weather reports suggest that our geographic situation experienced changes in atmospheric pressure of about 1040 to 970; equal to

a "normal" change in pressure associated with a change from a low pressure area to a high pressure area[7].

Hardware

Regarding the final results, getting the appropriate amount and correct data, turned out to be challenging. Starting with the hardware, both the RTC and BM280 sensor have given us multiple complications throughout the data-gathering process. In gathering data for program B, we found that the RTC didn't store the correct time after unplugging the arduino, since when the arduino again was powered on, the time would be set to the time of the first initial data was collected. This error didn't corrupt the relevant data, but made it hard to track, at what time of the day the data was recorded, which therefore also made the "night-part" of the program function unreliably. The error could probably have been fixed by using a new battery for the RTC, but it could also be a weird software function, that makes the RTC react this way. With the BM280 sensor, it's hard to pinpoint the origin of the errors, since it could both be hardware restrictions, as well as a software problem.

Switch issues

Originally we wrote both Part A and Part B in one code, using if-functions and a potentiometer to make the switch between the two parts. This worked as a solution, but with instability of the potentiometer and our idea that the program should be able to switch between the parts at any given point in time, we switched to using a button and the function "millis()" instead of delays to avoid pausing the program. What this resulted in, was an error no-one in the group, nor TA-team could identify, as when the function for the part that was NOT currently running, the BM280 would still "record" and store the data internally for that part of the program. When the switch was made, all that data was then dumped into the serial monitor and on the SD-card. This error could be the result of using millis instead of delays, as the millis function doesn't stop the program. Therefore the sensor never stops recording, but only when the function is later called in the switch, does the program know what to do with the recorded data.

An unverified solution to this problem, is to increase the logging-frequency of the inactive function to an extreme number, so that the sensor never gets to record any data before the switch is made. Then when the new function is called, the logging-frequency is then again set to it's required number.

[7] Diagram of barometer, shows drop of pressure [https://da.wikipedia.org/wiki/H%C3%B8jtryk_og_lavtryk]

Conclusion

Being reflective and mindful of what could be smart to use in future projects, it's safe to say that solving one function at a time could be smart. What we should have done after switching from delays to millis, was to make and test each part of the program individually, then after verifying the data, combine the two programs into one, which is controlled by a switch.