8. Wireless sensor network solutions for building microclimate and strutctures long term monitoring

8.1. The objective

The objective is to develop a cheap sensor network for heat flux, humidity and temperature measurements inside building walls, as wall as the means for long term data logging. Solution ideas are described in detail in the following subsections.

8.1.1. Data transfer with ESP8266

The initial idea was to use the $ESP8266$. It is a cheap microchip with built-in Wi-Fi and full internet protocol suite. In deep sleep mode it has low power consumption and $I2C$ communication opportunity.

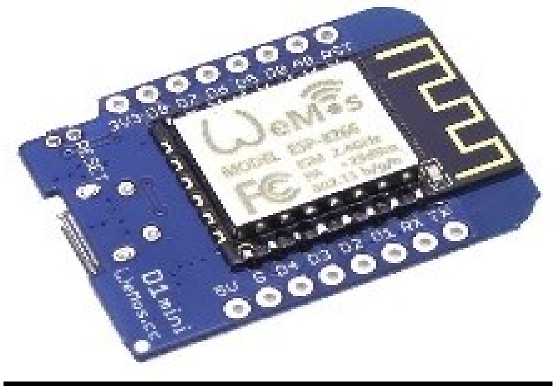


Fig. 8.1.1. *Microchip* $ESP8226$.

Since heat flux, humidity and temperature cannot change very rapidly inside within walls, for the application was set to perform one measurement per minute, for sensors to conserve energy.

Tests have shown that in deep sleep mode with USB communication, current is about $0.14\ mA$ and with no communication it can be even less than $15\ \mu A$. However, during data transfer, the current can reach up to $80\ mA$. The biggest problem with the $ESP8266$,was connecting time to Wi-fi, which took about 3 seconds and during which current is more than $80\ mA$, so it is not power efficient at all.

8.1.2. Testing Arduino Nano with a NRF24 radio module

$Arduino\ Nano$ (Fig. 8.1.2.) is a famous board based on the $ATmega328P$. It can be programmed via $Arduino\ IDE$, so after the code is written, it will be easy to reproduce the result.

According to its data sheet, $NRF\ 24$ has very low power consumption in operating mode, $11.3\ mA$ in TX mode, and it is cheap. Both can by found on eBay for less than $0.8\ \$$.

After some tests with a standard open source library $RF24$, I realized that it only supports up to 6 radio modules within the sensor network. The limitation comes from the fundamental capability of the $NRF24$, which supports only 6-channel reception.

Fig. 8.1.2. *Arduino Nano.*

To circumvent that, I found an interesting solution – the open source library $NRF24\ Network$ created by James Coliz.

The idea is to create a tree topology to handle more nodes with sensors.

On the first tree level one has a master node with the number 00, as shown in *Fig. 8.1.2.* The 2nd level can handle sensors or slave nodes for higher level. The largest node address is 05555 and with that the network can handle up to 3125 nodes.

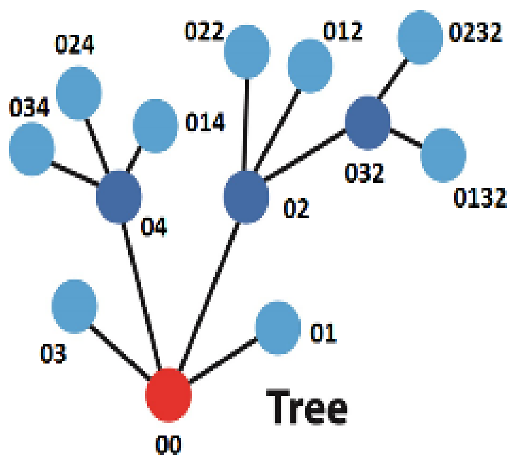


Fig. 8.1.3. *Node Tree. Sensors are designated by light blue circles, dark blue stands for support nodes.*

Having connected the Arduino and the radio as shown in Fig. 8.1.4., I got a much better result – during data transfer, the average current was $23\ mA$ and data trasfer was mach faster than with $ESP8266$: $0.02\ sec$. However, in sleep mode the current was $4.65\ mA$.

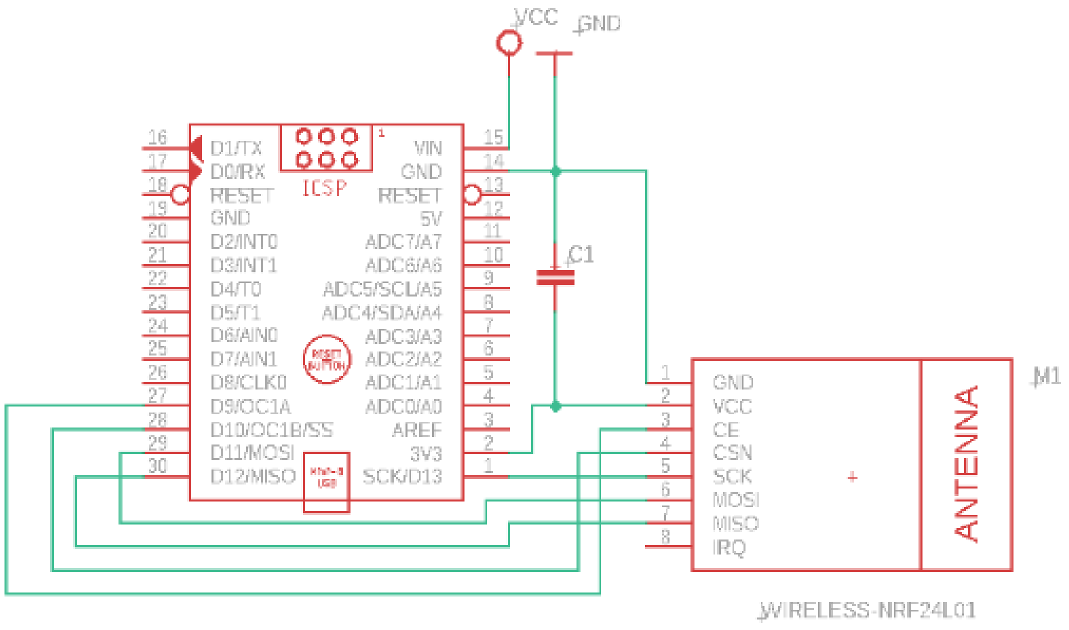


Fig. 8.1.4. *Arduino Nano connected to NRF24.*

8.2. Final PCB

Since there are many power consumers on boards, such as a stabilizer, $ch340$ for USB communication and LEDs, I decided to make my own board without unnecessary components.

I started with a microcontroller from Arduino Nano, $ATmega328P$. For more power efficient sleep mode, I decided to use the $Real\ Time\ Clock$ module to wake up the board from deep sleep.

RTC I have chosen $MCP7940M$, because of low power features, with $1.2\ \mu A$ operating current at $3.3\ V$. The first version of The PCB is shown in *Figure 8.2.1.*

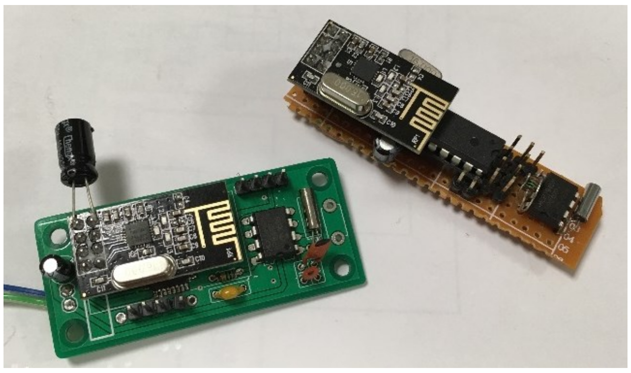
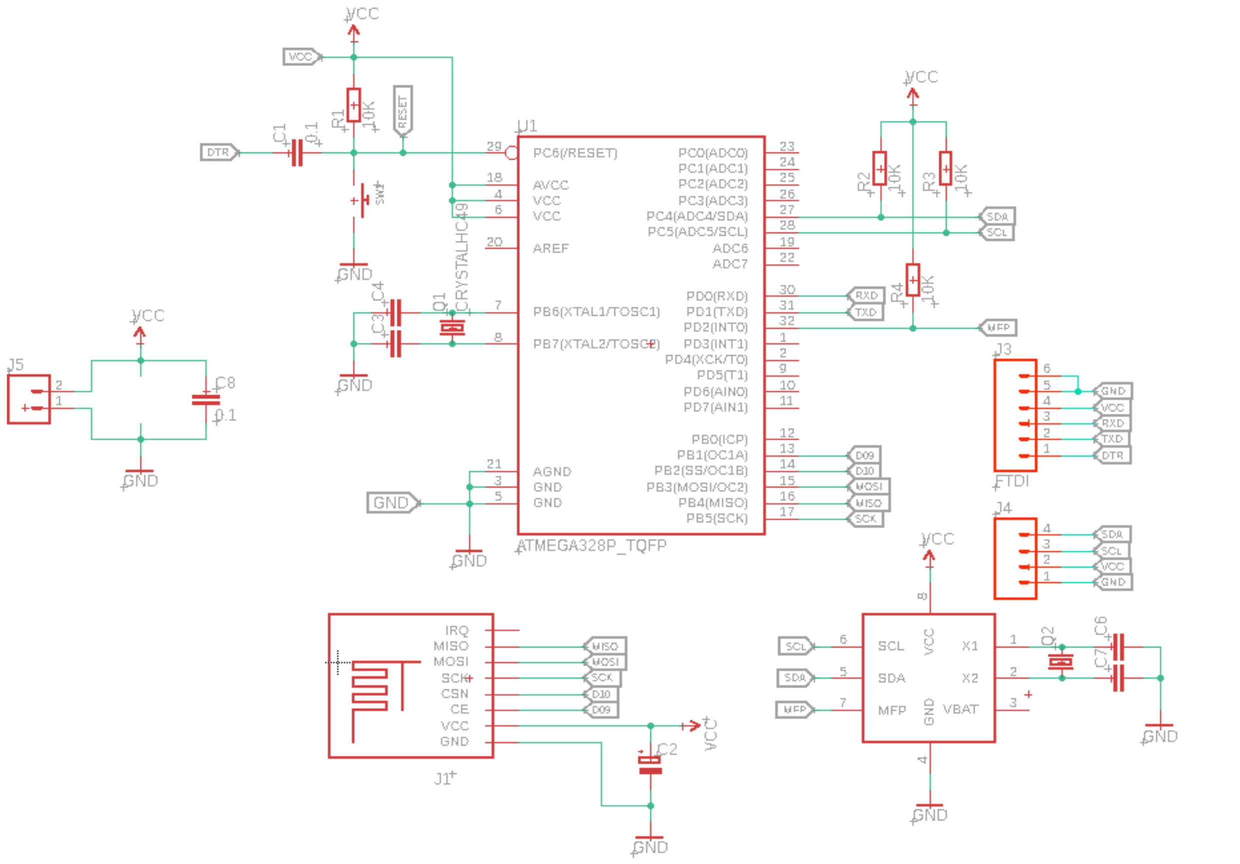


Fig. 8.2.1. *PCB prototype (left?) and manufactured version (right?).*



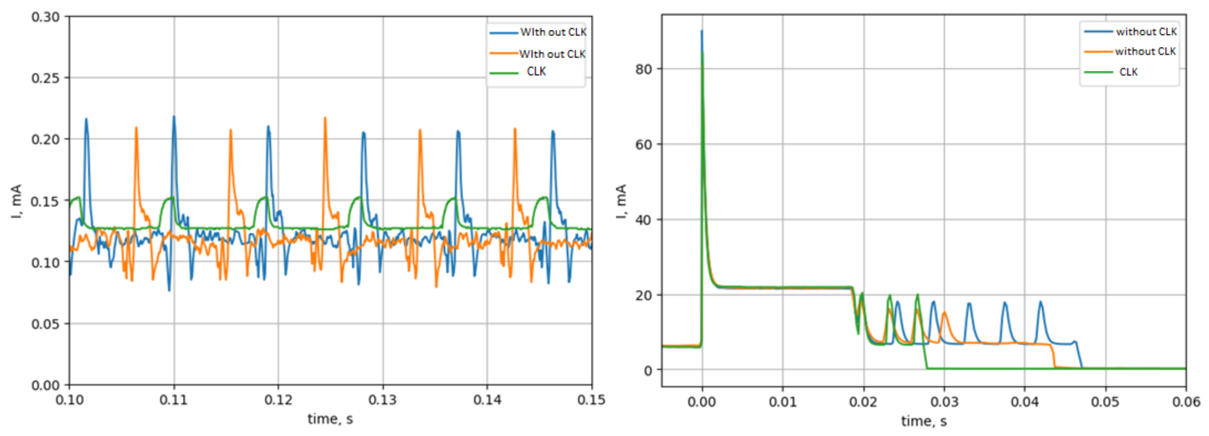


Fig. 8.2.2. *ATmega328P connected with NRF24.*

Fig. 8.2.3a. *ATmega328P in deep sleep power down mode.*

Fig. 8.2.3b. *ATmega328P in operational mode.*

With this configuration in *deep sleep power down mode* I got $\sim 0.14\ mA$ and tests have shown that RTC does not help to conserve energy, as seen in *Fig. 8.2.3*. Therefore, it was removed in future versions. Instead of it the system uses a timer watchdog and it also prevents node from getting stuck in the middle of code execution – should that happen, the node is reset.

Looking at the operational mode graph one can see that the maximum current is $\sim 90\ mA$, but, as it was mentioned above, data transfer time is very short and the peak current lasts less than $1\ ms$.

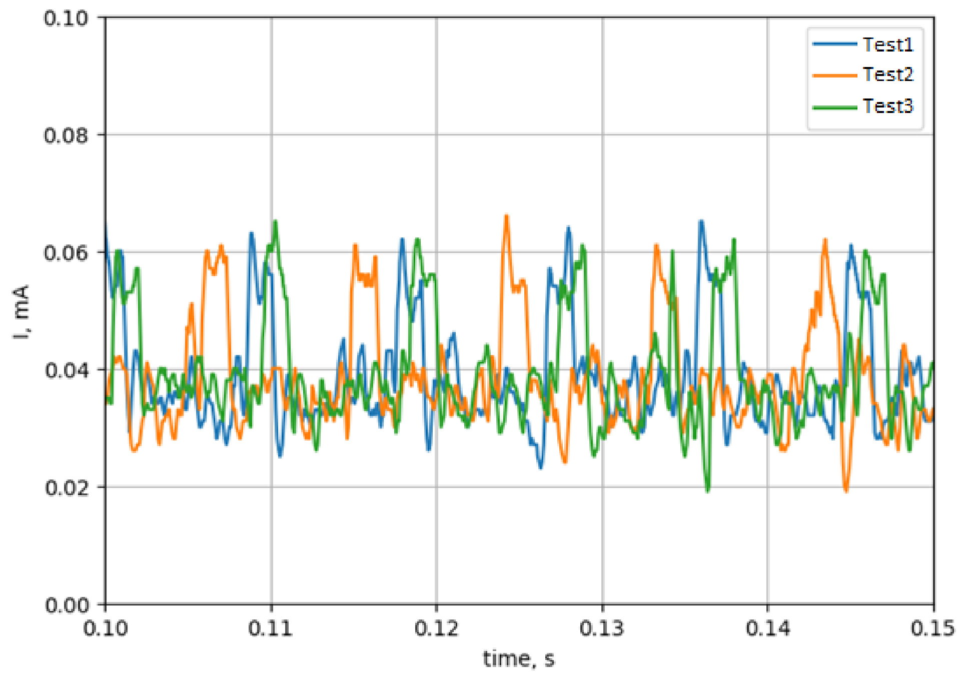


Fig. 8.2.4. *ATmega328P in deep sleep power down mode 2.*

Looking at the $ATmega328P$ datasheet, these values are far from possible power consumption. I did a little research and found that there are some modules within $ATmega328P$ that can consume power in *deep sleep power down* mode. Switching off ADC and brown-out detector and putting all unnecessary pins as inputs, the situation changed. At that point the average current in sleep mode was $\sim 36.5\ \mu A$, as seen in *Fig. 8.2.4*.

Therefore, the average power consumption was $\sim 0.042\ mA$ and at this point the theoretical operational time is ~ 8 years, without taking into account battery time discharge and calculating for $3000\ mAh$ battery.

After some tests, it was found that some packets lost between modules. At first, the idea was that there was a mistake during PCB circuit assebmly, because the prototype was working normally. After double checking all connections, I realized that the problem lies elsewhere. Experiments lead me to conclude that many factors can influence signal quality: sensor geometry, the fact that NRF24 modules are not all perfectly identical, the choice of a correct frequency. There is even a difference in performance between a sensor resting on a table and one that is being held in hand. From experience indicated, I can say that shielding the radio with aluminum foil and connecting ceramic capacitor to antenna and ground highly improves the signal and can lead to high signal quality when combined with a correct radio frequency.

8.3. Verification of the radio system in real conditions

For 2nd and higher-level slave nodes, Arduino Nano was initially chosen, and slave nodes can be connected to a power supply, as they can be situated outside the walls. 10 sensors were installed in 3 different test buildings seen in *Fig. 8.3.1.* The distance was up to $9\ m$ and all the nodes were inside the construction.

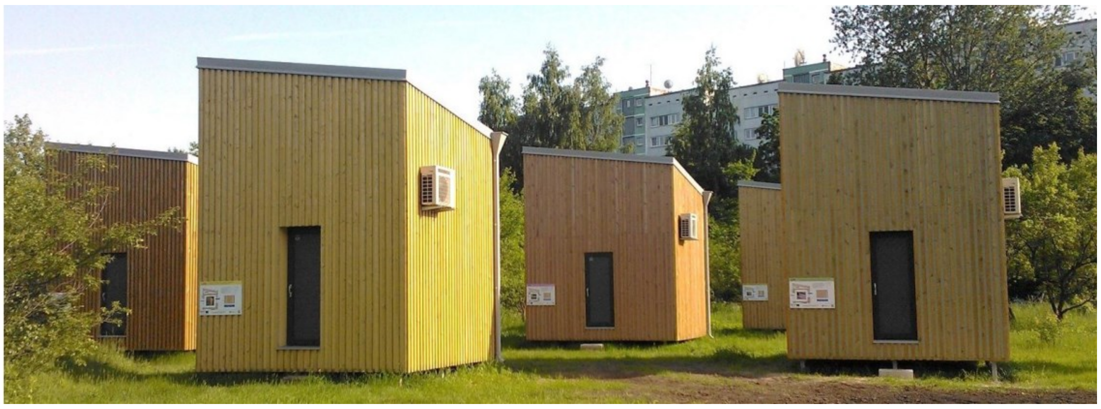


Fig. 8.3.1. *Experimental test buildings.*

After installation, initial inspection indicated the data was being transferred properly, but after some days it was found that one of the buildings, which was farther away from the master node, started losing packages. To solve this problem, it was decided to use the nrf module with an SMA antenna and a power amplifier, as seen in Fig. 8.3.2.. The standard Arduino Nano or Uno are not suitable for this application, because of new nrf24 operating current(120mA). The $ESP\ 8266$ has the better $3.3\ V$ stabilizer and works fine with nrf24+sma+pa radio module.

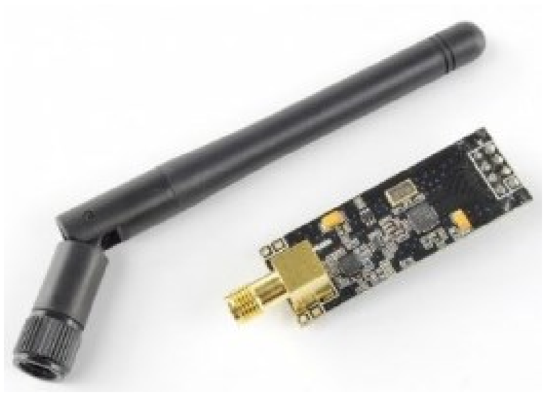


Fig. 8.3.2. *NRF24 with SMA antenna and power amplifier.*

After changing the slave nodes to ESP8266 with nrf24+pa+SMA, the data sampling rate improved, but after one week of testing a new pattern of bugs appeared. Approximately every 22 hours during different periods of day, $esp8266$ “freezes” for 20-30 minutes. This board, just like Arduino, has a built-in timer that resets the board if events like this one occur, but these functions did not appear to be working properly. After adding a different voltage stabilizer, situation had slightly improved, but the problem wasn’t resolved. After adding a proper 3.3V voltage stabilator LF33CV to Arduino Nano, it was noticed, that all the 2nd level nodes work without “freezing”, but

NATIONAL DEVELOPMENT PLAN 2020
EUROPEAN UNION
European Regional Development Fund
INVESTING IN YOUR FUTURE

some nodes were missing signals from sensors. After some experimentation with different positions of antenna on 2nd level nodes, it was noticed that the angle of antenna, as well as shielding the radio with aluminum foil, can significantly affect the quality of the signal. Then I found the new nrf24 radio module on eBay E01-ML01DP5 (Fig. 8.3.3.). With these modules, data transmission rate is almost 100%. After 6 months of testing, the average packet lost is less than 1%.

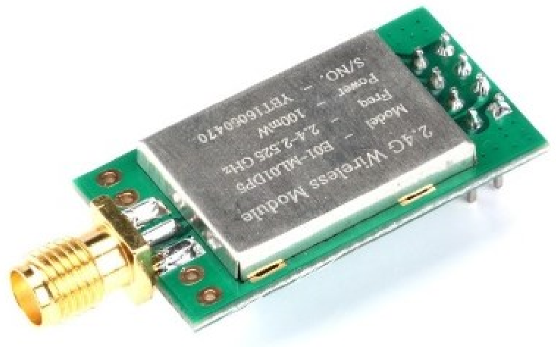


Fig. 8.3.3. *NRF24 E01-ML01DP5.*

8.4. Network master node

Raspberry Pi was chosen as the network master node. These are essentially small PC that are equipped with wired and Wi-Fi Ethernet hardware, and an SPI interface for connection with NRF24L01. Since Raspberry Pi is based on Linux OS, there is a lot of already tested open source software. In our case the system uses an open source FTP client to transfer data from test buildings to the university server once per day, as well as VNC remote access software to monitor data collection in real time. In addition to that, Raspberry Pi also has many build-in tools for data analysis out of the box. For stable work with RPi one needs only to have a good power supply and an SD card with class 10 or higher.

8.5. Sensor choose and measurements

One of the most important parameters for sensors is power efficiency and speed of measurements, which is why famous sensors such DH11 are not suitable for this application – measurement time can exceed 2 seconds, and, in this moment, the microchip must be in operational mode and consumes a lot of energy. For temperature and humidity measurements, HDC1080 was chosen. Firstly, these sensors have a small measurement time, about 6.5 ms for humidity measurement and 6.35ms for temperature. Secondly, they have small operating current 1.3μA and as little as100 nA in sleep mode. As the sensor produces a digital signal which is sent via the I2S protocol, information within the signal will not be distorted on its way to the board (Fig. 8.5.1.). To protect sensors from external physical influence, a special casing was designed and printed (Fig. 8.5.2).

For heat flux measurements, 3 different sensors (Ahlborn, Hukseflux and Almemo) were used. Sensors output analog signals, so the system must convert these to digital. For this purpose, 16Bit ADC 1115 was chosen. This module can measure voltage from -0.256 to +0.256 (V) with resolution 0.0078125 mV in x16 gain differential mode, and one measurement can be done in 9 ms.

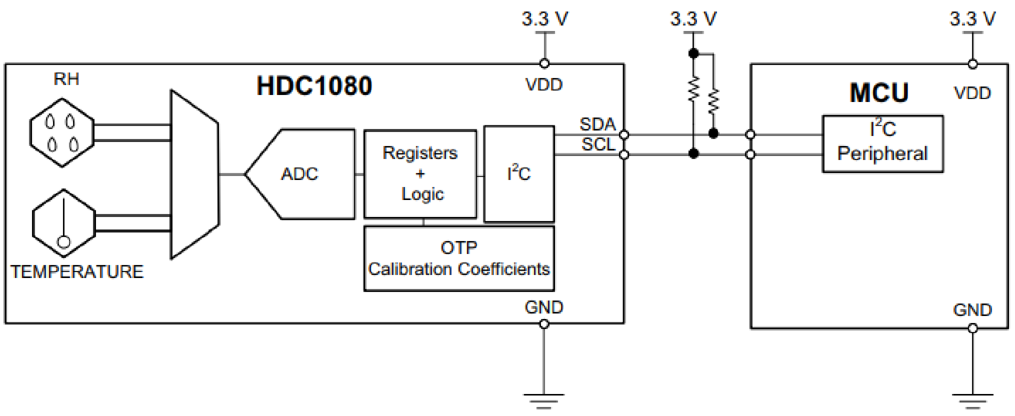


Figure 8.5.1. *HDC1080.*

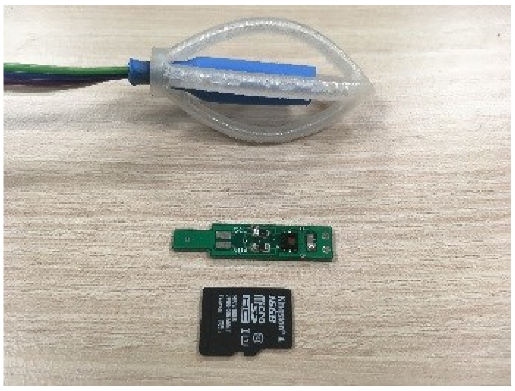


Figure 8.5.2. *HDC1080 with and without the casing.*

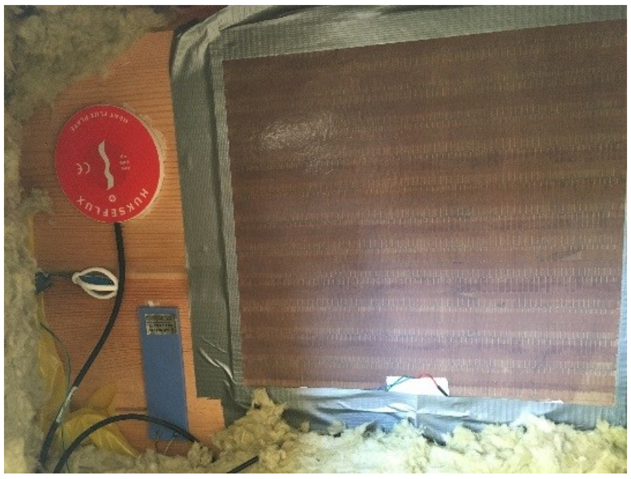


Figure 8.5.3 *Heat flux sensors inside of the wall.*

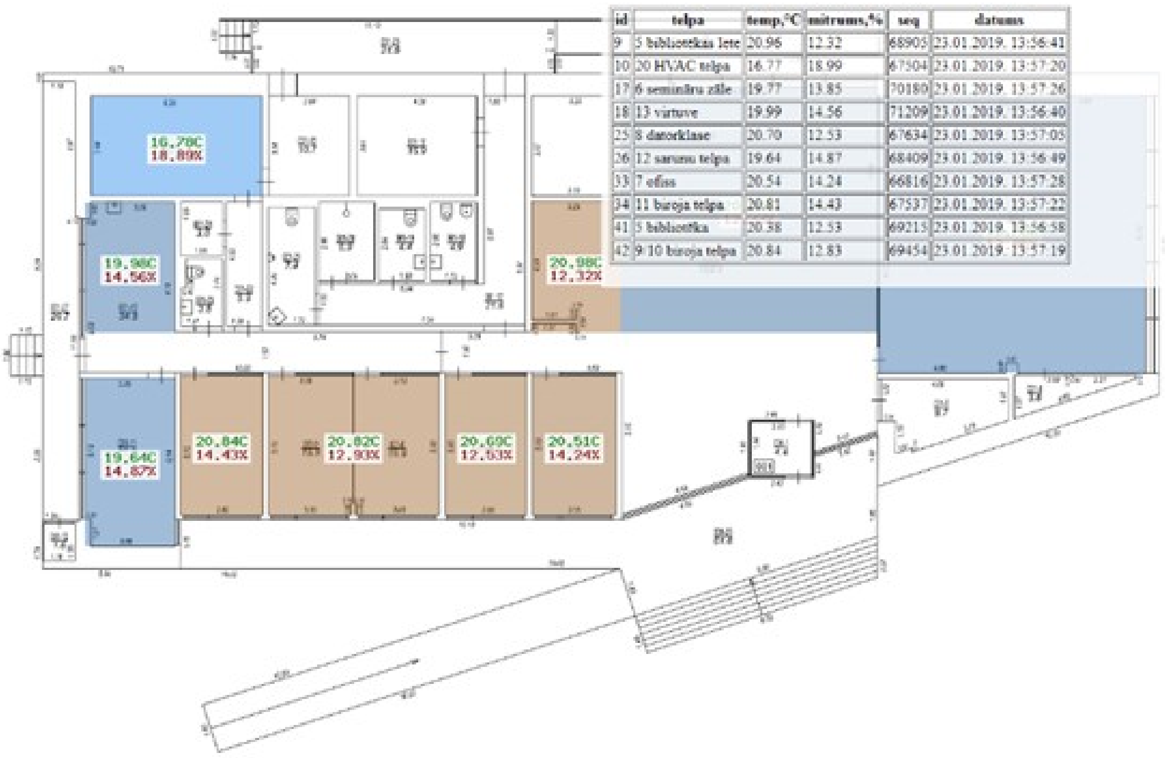
After first test it was noticed that a lot of noise present in data. To improve the signal, all wires for sensor connection to ADC were replaced with cables with good shielding connected to GND on ADC. For a more precise result, a 4.4 nF ceramic capacitor was connected to ADC input to average the result. In addition, the software also takes 10 measurements and sends the average value. Then, 3 different sensors were installed inside of the wall close to each other, as shown in Fig. 8.5.3. After one day, when temperature was stable, a significant difference was noticeable between sensors measurements, but measurements were much more stable compared to unprotected sensor error (500%) – now the difference was ~ 3% depending on the rate of change of heat flux. Two sensors returned similar values (small ones, Fig. 8.5.3), but the bigger one produced values that were to times less than the others. The small sensors were glued to the wood, but big one was attached using tape. After replacing tape with glue, the difference in output was greatly reduced and was no more than 20%.

8.6. Application of the system in real conditions

Fig. 8.6.1 *Alojas business centre.*

First use in real conditions was in Alojas business centre (SALA). The building was equipped with smart system and program for building inner climate control, but unfortunately the installation was not finished in time and only a half of system was working. So, to collect the data and then analyse, the building inner climate behaviour, to give recommendations, how to program building control system, was used developed wireless system.

For additional value of measurement was created web visualization of measured data, see Fig. 8.6.2.



id	telpa	temp,°C	mitrums,%	seq	datums
9	5 bibliotekas leте	20.96	12.32	68905	23.01.2019. 13:56:41
10	20 HVAC telpa	16.77	18.99	67504	23.01.2019. 13:57:20
17	6 semināru zāle	19.77	13.85	70180	23.01.2019. 13:57:26
18	13 virtuve	19.99	14.56	71209	23.01.2019. 13:56:40
25	8 datorklase	20.70	12.53	67634	23.01.2019. 13:57:05
26	12 sarunu telpa	19.64	14.87	68409	23.01.2019. 13:56:49
33	7 ofiss	20.54	14.24	66816	23.01.2019. 13:57:28
34	11 biroja telpa	20.81	14.43	67537	23.01.2019. 13:57:22
41	5 bibliotēka	20.38	12.53	69215	23.01.2019. 13:56:58
42	9/10 biroja telpa	20.84	12.83	69454	23.01.2019. 13:57:19

Fig. 8.6.2. *WEB data visualization in Alojas business support centre Sala.*

Literature

[1] https://www.cnx-software.com/2016/02/19/4-wemos-d1-mini-esp8266-board-supports-shields-with-a-temperature-sensor-a-button-a-relay-or-a-micro-sd-slot/
[2] https://store.arduino.cc/usa/arduino-nano
[3] https://bigdanzblog.wordpress.com/2015/02/21/the-very-frustrating-nrf24l01/
[4] http://tmrh20.github.io/RF24Network/
[5] http://www.modlab.lv/lv/publikacijas-autori.php?Autors=jakovics
[6] http://www.ti.com/lit/ds/symlink/hdc1080.pdf