Technical Report

for

Assignment 4

Fontys University of Applied Sciences

ICT & Technology

Embedded Systems

Group 5

Serhiy Medvedyev & Peter Szilagyi

23.05.2023

# Abstract

This assignment features working with I2C communication. The communication was used to control BME280 sensor and second Arduino accessing their registers and changing them depending on the requirements. One of the largest parts of the assignment was creating a library for BME280 as it included researching datasheets, filtering and extracting needed data and performing essential calculations to receive human-readable values. The last part of the assignment required combining the first and third parts to work simultaneously, proving the robustness and conciseness of the code and communication.

# Table of contents

Contents

[Abstract 2](#_Toc135506818)

[Table of contents 2](#_Toc135506819)

[**1.** **Introduction** 3](#_Toc135506820)

[**2.** **Procedure** 3](#_Toc135506821)

[2.1. Assignment A 3](#_Toc135506822)

[2.2. Assignment B 4](#_Toc135506823)

[2.3. Assignment C 5](#_Toc135506824)

[2.4. Assignment D 5](#_Toc135506825)

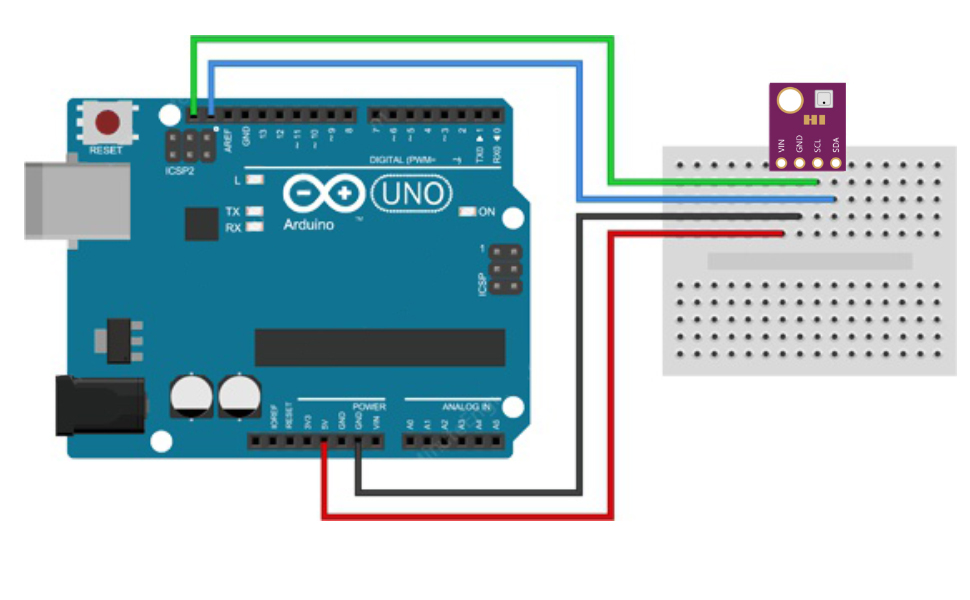
[**3. Conclusion** 6](#_Toc135506826)

# **Introduction**

The practicum consists of four parts which are logical continuations of each other. Assignment A featured writing a driver for BME280 sensor to get temperature, humidity and pressure as well as changing oversampling settings. Assignment B included researching OnReceive() and OnRequest() asynchronous events in I2C communication and their application in master <-> slave Arduino communications. Third assignment required creating an own model of master <-> slave communication. It was needed to simulate the register system where we have 2 Read-Write registers to store values which come from master and 2 Read-only registers which are calculated depending on first two registers values and then are sent on request from master Arduino. The last assignment required combing BME280 sensor with master <-> slave Arduino communication to work simultaneously using I2C communication.

# **Procedure**

## Assignment A



*Figure 1*

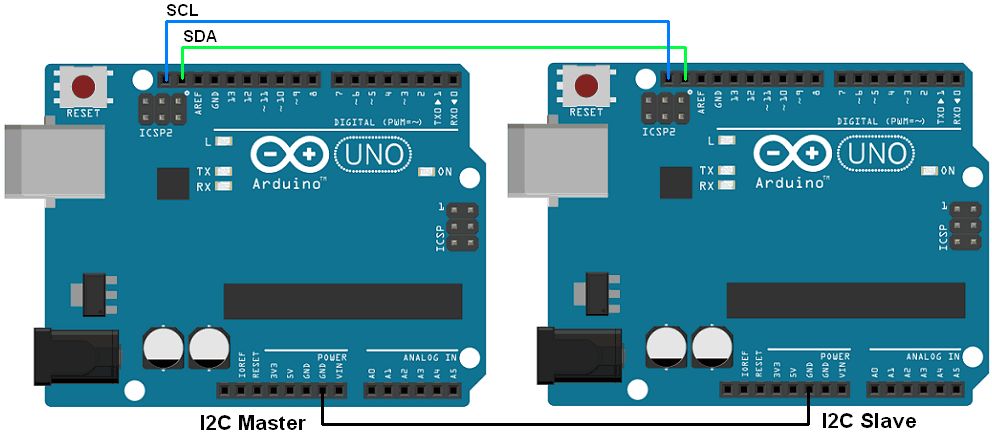
This assignment required creating a driver for BME280 sensor. Communication with the master device (Arduino) and BME280 sensor is handled by I2C bus which makes use of *Wire.h* library. The main amount of work was dedicated to reading datasheets and adapting algorithms and concepts described there for the team’s needs. The list of functions needed to be implemented:

uint8\_t BME280\_GetID();  
void BME280\_Reset();  
uint8\_t BME280\_CtrlHum();  
void BME280\_CtrlHum(uint8\_t bitpattern);  
uint8\_t BME280\_CtrlMeas ();  
void BME280\_CtrlMeas(uint8\_t bitpattern);  
long BME280\_ReadTemperature ();  
int BME280\_ReadHumidity();  
long BME280\_ReadPressure ();

The first two functions are general control functions which access relative registers and either get or set values to perform needed functionality: getting the device’s ID and resetting the sensor. Next four functions control data oversampling for humidity, temperature and pressure. It means the frequency with which data is sent to Arduino via I2C. It can be set to 0 (the data sending is skipped, so no data is acquired), or to set values starting from one to specify the needed frequency. The last three functions provide human-readable data from sensor on humidity, temperature and humidity and require the most manipulation to execute. At the beginning, the raw data extracted from these registers at once by recommendation from datasheets. This data is not human readable as it is just a set of meaningless numbers. To convert them to actual values it was needed to follow the protocol described in the datasheets. For each type of data (temperature, humidity, pressure) the manufacturer provided so-called compensators – constant values which can be found on specific register inside the sensor to perform needed calculations. The last step to get actual measurements was to calculate them using formulas found in the datasheets using raw data, bit manipulation and compensators.

## Assignment B

Assignment B featured using an Arduino as a slave in I2C bus communication. It was required to research different communication methods and events using provided materials, especially OnReceive() and OnRequest events. After that, a simple application was built to handle communication between two Arduinos. The whole application loop is represented by the idea that slave device is a listener and runs only when there is some request from master device. Master device would send an incrementing byte. Whenever slave receives this byte, it should calculate the response by the condition set. If the input byte is less than one hundred it sends 4, otherwise – 2. However, slave does not perform the transmission right after receive event. On the contrary, it waits (listens) for the request from master and only then transmits the response.



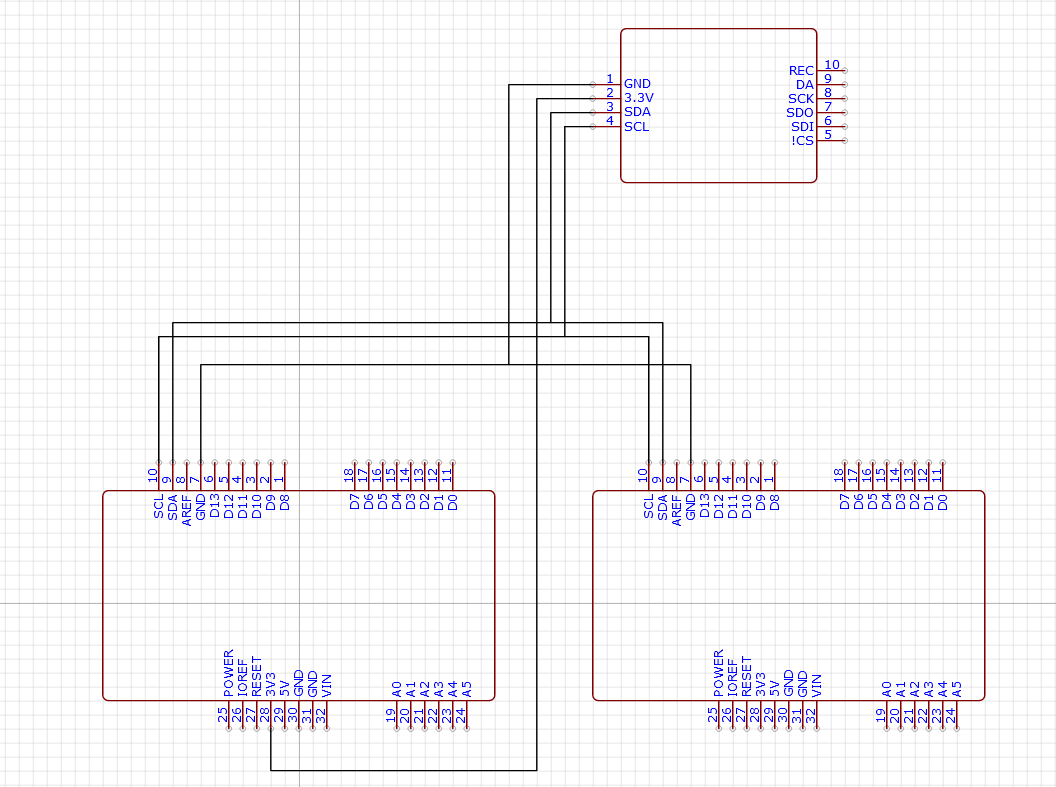
*Figure 2*

## Assignment C

Assignment C is a continuation of Assignment B. Two Arduinos were connected via I2C bus as pictured in Figure 2. The slave Arduino connected to the bus with 0x50 address. Its main responsibility was to handle made up registers which are either Read – Write or Read-Only. These registers are changed depending either on values received on OnReceive() method which changes the values on Read-Write registers or sends data on request from master. In general, the whole idea is that Read-Write registers represent two variables, which can be set or change from master, and two Read-Only registers, which calculate minimum and maximum value out of provided variables. After that master receives which value was maximum or minimum on request event.

## Assignment D

The last assignment is a combination of Assignment A and C implying that both BME280 and slave Arduino are connected to the same I2C bus and communicate with master Arduino. This assignment the usage concept of the driver created for BME 280 as the only thing done was including header and implementation file to the new project. The hardware is connected as pictured in figure 3.



*Figure 3*

As soon as the addresses of the devices are predefined and different it is possible to use functionality of two libraries at once: get the atmospheric data from BME280 and communicate with slave by accessing premade registers.

# **3. Conclusion**

This assignment's main aim was to continue exploiting device registers on a more advanced level and establishing I2C connections between two or more devices. The first assignment included working with datasheets, detecting and extracting vital information from it and applying it to the software development process to use the device in the intended way. Furthermore, access to the sensor is established by I2C bus. The work with datasheets helped to extract needed data from the sensor, sort it by measurements topic and calculating the actual values, which are then shown to the user. Communication was developed using *Wire.h* which allows communication with all connected devices by their addresses. The next three assignments featured more research on *Wire.h* libraryand I2C communication in general, learning other techniques and applying them in connection between two Arduinos. Overall, these assignments required implementing the event techniques for I2C bus when slave devices act only when they receive information from master or receive the request for information from master. Last assignment had the concluding role of combining all the concepts and knowledge acquired throughout implementation. BME280 sensor (representing Assignment 3) and slave Arduino (representing Assignment B & C) were connected to one I2C bus and communicated with master as intended by their set behavior.