Torben Felix Witte

Development of an Indoor Air Quality Monitoring System based on a Microcontroller

Helsinki Metropolia University of Applied Sciences

Bachelor of Science

Electonics

Thesis

3. June 2014

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Torben Felix Witte Development of an Indoor Air Quality Monitoring System based on a Microcontroller 43 pages + 43 appendices 3 June 2014
Degree	Bachelor of Science
Degree Programme	European Electrical Engineering
Specialisation option	
Instructor(s)	Janne Mäntykoski, Senior Lecturer

With the increasing time people spend indoor, the importance of the Indoor Air Quality increases. Especially in learning facilities, a high Indoor Air Quality is required to provide a good learning situation to students and teachers.
This project is targeted on improving the Indoor Air Quality in a learning environment by monitoring the Indoor Air Quality and to provide information and reasoned advice to the users

The operation principles of gas sensors are explained. A special focus lies on NDIR sensors and resistive MOX sensors, which are used to measure characteristic factors of the indoor air. The data are used to evaluate the Indoor Air Quality. Additionally data logging functionality and a radio link are intended, as well as a user interface to communicate the gained information.

During the development, the data logging feature had to be temporarily excluded, due to interference with other parts of the system. Even though this problem is not solved yet, a functional prototype without data logging functionality has been developed. First tests demonstrated the functionality of the implemented features, but there are parts left to be implemented in future.

Keywords	IAQ, Indoor, Air, Quality, CO2, VOC, MOX, NDIR, Gas, Sensor, mbed, LPC1768, microcontroller, MCU

Contents

Appendices

Abbreviation

Indoor Air Quality	IAQ
Non-dispersive Infrared	NDIR
Carbon dioxide	CO2
Space Charge Region	SCR
Volatile Organic Compounds	VOC
Microcontroller Unit	MCU
Real Time Clock	RTC
Software Development Kit	SDK
Automatic Baseline Correction	ABC
File Allocation Table	FAT

1 Introduction

Nowadays we are spending an increasing amount of time in closed environments like houses, schools and offices. We live, study and work in there and consequentially we breathe there. Hence we should be aware of the quality of the air we breathe and of its impact on us. Especially in study and working environments the air quality is of high importance, having a significant effect on the study and respectively working potential. Further, certain compounds in the air may cause health issues, especially when considering long-term exposure. Therefore sufficient ventilation is required to keep the air quality at an acceptable level, and prevent the possibility of health issues. Unfortunately many buildings are not and will not be equipped with modern ventilation systems due to the high cost of those devices or problematic building structures.

This is where this project is located. It is meant to improve the conditions in a learning facility by increasing the awareness and sensibility for this topic. The project has reached its goal, if the system is able to improve the learning conditions in the short term by advising the people in the room, when to ventilate and in the long term by increasing the sensibility concerning Indoor Air Quality (IAQ) and providing long term data to develop strategies to automatically maintain a suitable air quality.

2 Gas Sensor Theory

2.1 Introduction to Gas Sensors

Gas sensing is a wide field and contains several techniques and methods and even more sensor types. Therefore there will be a short introduction of the main sensing techniques used in gas sensing. Afterwards, the sensing methods used in this project will be described in more detail.

One common type of gas sensors are electrochemical sensors. They may appear as amperometric, potentiometric as well as conductiometric sensors, but their main structure stays the same. It consists of three electrodes in a common electrolyte, which can be either solid or liquid. The principle of operation is a chemical reaction of the target gas and one of the electrodes, resulting in a change of the electrodes potential. The remaining two electrodes are not reactive towards the target gas and are used as a ground and a reference potential. Connecting the three electrodes, the electrolyte enables a charge transfer in between the electrodes and an electronic circuit may detect the change.

Mass-sensitive sensors recognize mass change introduced by a present gas. A sensing layer reacts with the target gas, attaching its molecules to a sensing mass. The attached molecules introduce an additional mass to the sensing mass, which may be detected in different ways. The sensing mass may be used in a resonance circuit and the introduced mass change will differ the circuit's resonance frequency. The resulting frequency drift will be detected electronically. Measuring stress or bending due to the additional mass is another way of detection. This may be done by optical, piezoresistive, capacitive, and interferometric means.

Another technique available is magnetic sensing. This technique is used to detect gases which have strong inherent magnetic properties. It is mostly used to detect oxygen, which has a high magnetic susceptibility, hence obtaining paramagnetic behaviour. Paramagnetic molecules are attracted towards the strongest magnetic field, which is used to create pressure differences or alter the gas flow. These effects may be measured and transformed into electrical signals.

Like many other sensing techniques, thermoelectric sensors are based on chemical reaction. In this case the analyte reacts with a sensing material, creating a change in temperature. The increase or decrease of temperature results in a change of the sensors electrical properties, which can be transformed into an electrical signal.

2.2 Optical Sensor

Optical Sensors detect changes in electromagnetic radiation in the spectrum from Infrared to ultraviolet light, due to the presence or absence of target gas molecules in a sample. If the target gas has intrinsic optical properties, it may directly cause a detectable change in the radiation. Otherwise a sensing material is needed to create the change in the radiation by changing its optical properties, due to a chemical reaction with the target gas. Several different sensing techniques are used, observation of spectroscopic properties like refractive index, light scattering & reflectivity as well as fluorescence chemiluminescence and absorbance.

The sensor used in the project is a non-dispersive infrared (NDIR) sensor. This type of sensor uses a direct way of measurement. The gas concentration is measured by means of absorption. Some molecules absorb radiation of a certain wavelength, in the given case, infrared radiation.

2.2.1 Absorption of Infrared Light

In our case we can refer to a molecule as a system of joined masses, where the atoms represent the masses and the atom bonds represent the springs. Analogue to this model the natural frequency of a vibrating atom in a molecule depends on its mass and on the strength of the bonding. [1] The electric charge of an atomic bond is not equally distributed. The atom with the higher electro-negativity has a bigger effect on the bonding electrons and attracts them closer towards itself. This imbalance creates a dipole along the bond. The molecules dipole moment is the vector sum of all dipole moments within the molecule. An introduced radiation changes the magnitude of the molecules dipole moment which results in stretching or bending the bonds. When a bond is hit by electromagnetic radiation with a frequency, similar to its natural frequency the bond, the bond absorbs the radiation and turns it into molecular vibration. The bond has left its vibrational ground state and reached a higher vibrational state. Different atom bonds absorb radiation of different wavelength due to different atom mass and different bond strength. Therefore it is possible to identify certain molecules or molecule groups based on their absorption spectrum.

Any molecule has 3N degrees of freedom, where N is equal the number of atoms. If you reduce this by 3 degrees for translation and 3 degrees for rotation, respectively 2 degrees of rotation for linear molecules, you end up with the amount of different vibrational modes. Taking a linear carbon dioxide molecule as example you will see that there are 3N-5 = 4 degrees of freedom left and therefore there are 4 vibrational modes of carbon dioxide, displayed in figure 1

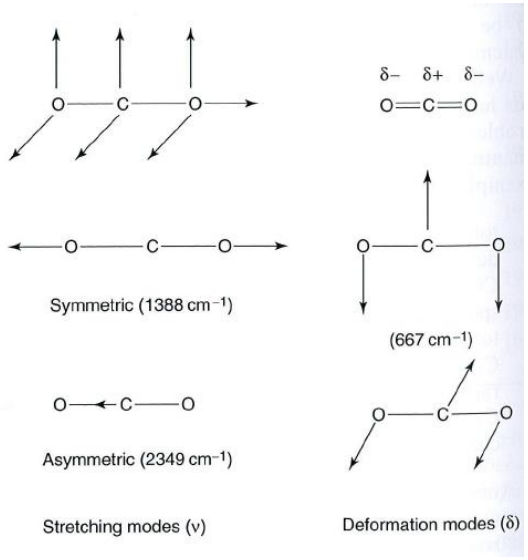


Figure 1. Vibrational modes of a CO2 molecule.Reprinted from[1,1916]

But not all of them may be excited by infrared radiation. The absorption of infrared radiation results in a change of the dipole moment of the molecule. This change is not given in the symmetric vibrational mode. As shown in Figure 1, both bonds are stretched in the opposite direction with the same amplitude. This results in an equal change of both dipole moment, but in opposite directions. Considering the total dipole moment of the molecule both cancel each other out and this vibrational mode is not infrared active. In asymmetric vibrational mode one dipole moment increases while the other decreases, accordingly there is a change in the molecules dipole moment in total. The bending of the bonds in deformation mode changes the directions of the bonds' dipole moments and therefore their vector sum, hence the molecules dipole moment changes as well. The deformation modes are IR-active but due to the fact that the vibration is perpendicular to the bond, it absorbs less energy than the other modes and has a natural frequency different from asymmetric mode. [2] However, absorption does not occur exclusively at the natural frequencies of each vibrational mode. The excitement from a lower vibrational state to a higher one is always accompanied by the excitement of rotational modes as well [1].

Furthermore the surrounding influences the natural frequencies as well. Neighbouring atoms may attract each other and create small non-covalent connections. These attractions may be considered as very weak additional springs, shifting the natural frequency a little bit. Taking all of this into account, it shows, that not only radiation of one frequency gets absorbed but radiation from a small region around this frequency, called fingerprint region[4]. These fingerprint regions are still unique for a certain type of covalent bonding, but the possibility of close fingerprint region overlapping has to be considered.

2.2.2 NDIR Sensor

There are several types of non-dispersive infrared sensors available. Even though they use different technical setups, the main idea stays the same. All of them measure the transmittance respectively the absorbance of a gas sample at a certain part of the infrared spectrum. The Lambert-Beer-Law relates the transmittance with the concentration of the target gas and allows us to estimate the concentration c based on the measured transmittance T, the distance L, the light traveled through the gas sample and the extinction coefficient a

$$T = e^{-a\,c\,L}$$

The basic structure of an NDIR senor, which most types have in common, consists of IR-source, a band pass filter, the sample cell which contains the gas sample and a photo detector. First the radiation, emitted by the IR-source, passes through the band pass filter, followed by the sample cell and finally hits the photo detector. The band filter limits the IR-spectrum to a certain region, which contains the fingerprint region of the target gas. In the sample cell the radiation interacts with present molecules of the target gas, which reduces the radiation's intensity. Afterwards the reduced intensity is measured by the photo detector and the signal processing compares the measured intensity with a known value for the incident intensity. The downside of this simple setup is that a drift of an aging IR-source is not taken into account. To prevent respectively slow down the aging process, two similar IR sources may be used alternately [4].

To compensate occurring changes of the IR-source and to improve stability a reference light path may be added. The radiation is alternately passed through the sample cell and the reference cell. The latter consists of a closed gas cell, containing no target gas. Hence no absorption occurs and the detector can measure the actual incident intensity of the emitted radiation and compare it with intensity passed through the sample cell.

Another way to compensate changes is to replace the band pass filter by two alternating filters. The first filter limits the radiation to a broad spectrum which includes the fingerprint region as well as region which will not be absorbed. The second filter limits the radiation to the same region, but excludes the fingerprint region of the target gas. In this setup the intensity of the radiation, excluding the fingerprint region, is considered as reference. [1] [3]

2.3 Electrical Gas Sensor

General overview

Electrical sensors are widely used. They are robust and simple and therefore relatively cheap. Further they are very sensitive to low concentration, but on the downside they have a lack of selectivity. Selectivity, as well as, stability may be increased by catalysts like noble metals, or sensor geometry [5].There are several materials used and even more target gases, detected by this sensor type. Possible materials are polymers, metals, metal oxides or semiconductors [6]. Further these materials are used with different sensing techniques like, conductiometric/resistive measurements or capacitive detection. But the main principle of operation of electrical sensors stays the same, utilizing a reversible chemical reaction of target gas and sensing layer to alter the sensing layer's electrical properties. In the latest times the development of FET/work-function based sensors became more popular[7].

2.3.1 Capacitive Sensing

Electrical sensors, using capacitive sensing methods, change their capacitance due to the presence or absence of a target gas. This change may be caused in different ways. Basically the target gas reacts with a sensing layer, which is used as dielectric material in a capacitor and the layer changes its properties. It can either change its dielectric properties, or change its volume, resulting in a change of the electrode's size or its distance. Both influence the sensors capacitance, and this change will be transferred into an electric signal.

2.3.2 Resistive Sensing

Another way of detection is the resistive measurement. The presence of a certain gas alters the electrical conductivity respectively the resistance of the sensor.
In most cases semiconducting metal oxides are used, but conductive polymers are as well a possible sensing material [6]. The earlier sensors used thick film technology with porous material, increasing the surface area. Thick film sensing layers were made of sintered powders, which made them hardly reproducible, a major drawback in terms of mass production. Introduction of thin film technology to sensor production improved the reproducibility and enabled reliable mass production. As in many other fields of electronics, the use of nanostructures is evaluated. So called nanobelts have interesting properties due to size effects and may be used in future for gas sensing applications [7].The sensor used in the project contains a metal oxide sensing layer, due to that and due to the dominance of metal oxide sensors this studies will focus on its principle of operation.
The basic principle of operation of resistive metal oxide gas sensors is based on two counteracting chemical reactions, the adsorption of oxygen at the surface of the sensing layer and a reduction of the surface by the target gas. A key requirement is that the surface is reactive towards oxygen. This is highly dependent on the sensing layers lattice structure at the surface. At the interface of solid and gas the lattice structure of the metal oxide is different from the bulk. The unity cells are not completed, making the surface reactive. Given an elevated temperatures, providing enough energy, oxygen becomes chemisorbed at the metal oxides surface. During chemisorptions, the oxygen collects electrons from the surface, creating an oxygen anion. This may be single charged or double charged, even though the single charged anion appears to be the dominant one[6]. The elevated temperature is required because chemisorption is an endothermic reaction and needs a certain amount of activation energy. But it has to be taken into consideration, that a strongly elevated temperature softens the solid's lattice structure, allowing vacancies to diffuse through the bulk, causing irreversible change of resistance. Further ion diffusion becomes part of the charge transfer, supporting the sensor’s drift. [7]

The electron used for the chemisorption of oxygen is trapped and cannot participate in charge transfer any more. To restore the thermal equilibrium, electrons from the bulk move towards the surface, creating a space charge region (SCR).As more oxygen atoms get chemisorbed at the surface, the more electrons from the bulk of the sensing layer move to towards the surface and the space charge region increases. Figure 2 shows a SCR reaching over the whole surface.

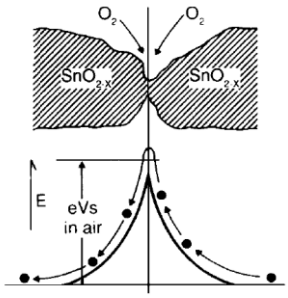


Figure 2. Space Charge layers and resulting Energy Barrier Reprinted from[8, 2]

In the band model, this space charge region is represented as an energy barrier shown in figure 2, and increase with the width of the space charge region. This barrier has to be overcome by each electron passing through the SCR. Looking at the whole sensing layer, the SCR creates a resistance for moving electrons. With an increasing number of SCR to pass through, the overall resistance for a current passing through the sensing layer rises. In metal oxides, those SCR arise at the material's grain boundaries. Accordingly, the number of SCRs is as well dependent on the grain size, and the resistance rises with a decrease of grain size.

If the grain size is small enough, the SCR covers the whole grain, totally depleting it from free charge carriers. A small change of the SCR recreates some of the charge carriers, causing the grain to change from highly not conductive to conductive, hence decreasing the resistance significantly and making the sensing layer highly sensitive to small concentrations of the target gas, one of the main characteristic of metal oxide sensors. [7] The outcome of this is that the sensors with high sensitivity are preferably made by polycrystalline material, with a small grain size. Furthermore a high porosity is required to expose the grain boundaries to the gaseous surrounding.

Introducing a reductive gas to the surface, it reacts with the oxygen, adsorbed to the surface. The reductive gas removes the oxygen from the surface, and bond with the oxygen. The removed oxygen releases the electron, trapped in the chemisorption bonding, donating it back to the bulk.

This decreases the SCR in the sensing materiel and the related energy barrier in the band model. A current flowing through the sensing layer, passing from one grain to another, faces smaller energy barriers at the grain boundaries which it has to overcome. Hence the total resistance of the sensing layer decreases.

The processes of surface oxidation and surface reduction counteract each other. Given a stable concentration of oxygen and the reductive gas, a state of equilibrium will be reached and a resistance dependent on the gas concentration will arise.

This mechanisms works with all reducing gases and therefore it has a very poor selectivity. To encounter this problem, membranes, filters or special sensor geometry may be used to prevent certain gases from reaching the sensing layer. In addition to this structural means, introducing catalysts to the sensing layer surface is used to further improve the selectivity towards certain reactants [8]. The catalysts improve the selectivity by supporting one or more reactions the target gas participates in.

3 Analysis

As described in the introduction, the main goal of the project is to improve the learning situation in a lecture room. According to this, the system to be developed has to rate the present air quality and communicate the evaluation to room occupants (user), recommending when to ventilate. Furthermore all measured data will be stored, allowing the user to evaluate measured data and show the trend of the monitored factors.

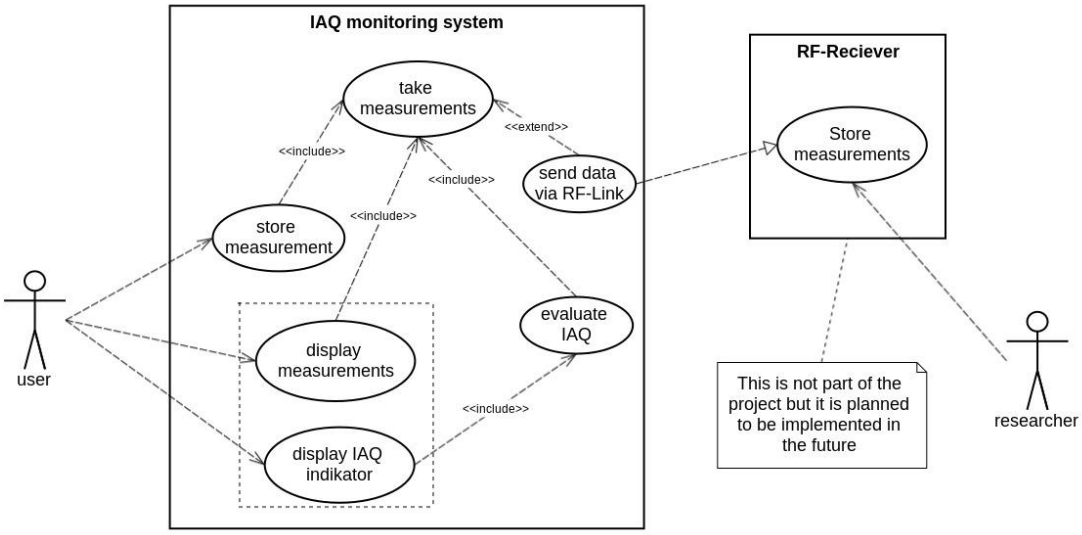


Figure 3. Use-Case diagram of the Indoor Air Quality Monitoring System

The outcomes of this are five main tasks the system has to accomplish, as shown in figure 3. First of all, important characteristics of the room air have to be monitored and based on these measurements an evaluation of the air quality has to be made. Further the rating has to be communicated to the people in the room, allowing them to react on the given situation. Finally the measured data will be stored and a possibility to access the data has to be provided. Additional to the internal data storage, a RF link is requested by the teacher.

The acquisition of the measurement data is the most important task, considering that all other tasks rely on it. The Indoor Air Quality may be affected by several factors. Monitoring all possible influences is a very extensive challenge concerning development, system complexity, costs and energy consumption. Therefore it is desirable to consider the future working situation and fit the system to it. Influences which occur with a low probability may be neglected and the system focuses on characteristics more likely to change. In the given case, the working situation will be premises used for learning and teaching. A contamination by toxic or combustible compounds like indus-

trial exhausts and chemicals is unlikely, because there are no sources of those substances. However, certain materials of furniture may vaporize and cause health issues, if exposure is long enough. The symptoms are well documented and known as “Sick Building Syndrome”. But even though it is well documented, its origin is not definitely determined, yet. Many studies on this topic suspect the group of volatile organic compounds (VOC) to cause the characteristic symptoms like headache, fatigue or qualm.[9]Due to the long time people spend in class room, the presence of VOCs may be a serious concern. Hence the VOC concentration will be monitored and taken in consideration for the Air quality rating.

Another very important factor is the concentration of Carbon dioxide. It is an inherent component of the air. The natural concentration in free air is 350- 450 ppm, depending on the chosen literature [10].Several studies show a correlation of long exposure to elevated concentrations of carbon dioxide and the appearance of various health symptoms. Possible effects of an increased concentration may be headache, fatigue, nasal, eye and respiratory symptoms [11]. Even though it has been believed, that these reactions are caused by other air pollutants, which occur together with the CO2 and the elevated concentration just indicates poor ventilation, newer studies encounter this assumption and indicate a direct impact on human cognitive abilities [12]. Irrespective of the actual influence of carbon dioxide on the human body, metabolism and the cognitive abilities, the concentration of CO2 is a reliable indicator for the air quality and ventilation of a room and is widely accepted by governmental institutions. The Finnish Ministry of Social Affairs and Health has issued a reference value of 1500ppm for indoor air and concentration above this reference is considered as hazardous to health, concentrations up to 1200ppm are assumed to be acceptable.[13]To assure proper conditions for studies and teaching it is important to keep the concentration as close to the free air value as possible. Hence the concentration of carbon dioxide has to be accurately measured and taken into account for the estimation of the Indoor Air Quality. Due to the fact that it changes constantly by means of respiration and its high effect on the learning ability, the carbon dioxide concentration will be the main factor for the estimation.

The prevention of health risks and the negative impact on the cognitive abilities are very important factors to improve the Indoor Air Quality in study areas. But the subjective well-being is as well an integral part of it. The subjective perception of the Air quality is highly based on the temperature and the relative humidity. Elevated temperatures as well as an elevated humidity may make room occupants feel uncomfortable and make it difficult to concentrate, same with to low temperatures and respectively with to

low humidity. Furthermore a low relative humidity may cause the mucosa and the eyes to dry. However, there is no absolute value for an ideal temperature or humidity, given that the perception of them is highly dependent on everybody's preferences, which makes it desirable to define certain range of acceptable temperature and humidity values. But due to the high dependence on personal perceptions, temperature, as well as relative humidity, will be considered as a weak indicator during the Air quality evaluation.

Another main task is the opportunity to access the measured data by hindsight to analyze the trend of the different factors over the time. The radio link shown in figure 1 is added by request of the teacher to provide connectivity to a certain receiver station. The receiver station, the system has to communicate with, is the ConnectPort X4 by Digi. The used model of the ConnectPort X4 uses the IEEE 802.15.4 protocol for wireless communication and Ethernet connection for network access. A custom application, which has to be developed, running at the receiver will receive the raw measured data via the radio link and forward this to a webserver. The receiver, as well as the web application, is not part of this project. Exclusively the transmitter will be implemented as part of the project.

To achieve a better redundancy and to maintain the system's functionality while the RF link is not implemented, an internal memory is needed. This memory will contain all measured data combined with a time and date information. Therefore a Real Time clock becomes necessary. To access the data, a removable memory device is added, which is filled up with the data from the internal memory when it is connected. Furthermore the system has to detect a connected external device automatically and initialize the transfer of the data. To enable a convenient examination, the data has to be stored in computer readable files, necessitating the implementation of a file system.

Even though the carbon dioxide concentration and a detected VOC contamination are strong indicators for the IAQ, it has to be assumed, that the room occupants are not aware of the meaning of the data and able to draw a conclusion from that. Hence it is the systems task to evaluate the data, grade the Indoor Air Quality and communicate the results to the room occupants in a way, which is easy to understand. An acoustic alarm similar to smoke detectors may draw the attention instantaneously, but as well it will disturb the teaching which is not desirable. A visual representation is the better choice. The visualisation has to be easy to understand and may not require any preknowledge. The well-known colour code of traffic lights fits both and is therefore the best choice

4 Architecture

In General, the system can be structured in several parts shown in figure 4, with the parts passing information to each other. The sensor block abstracts the sensing elements and its features, creating the measurement data. When the microcontroller (MCU) takes the measurement, the data are linked with a timestamp delivered by the Real Time Clock (RTC) and stored in a data container, protecting them against modification and keeping the data of one measurement together. For each measurement, an individual data container is created. The containers are stored in the data buffer with a fixed size in the microcontroller. Only if the data buffer is full, all stored measurements are moved to the internal memory at once, reducing the communication between the MCU and the memory device. The data transfer from the internal memory to the external is done stepwise. A part of the data stored in the internal memory is read and buffered by the MCU, and moved to the external memory as soon the buffer is filled.
Furthermore, the data buffer provides the latest measurement to the RF-Transmitter and the IAQ evaluation. After evaluation, selected data and the IAQ indicator are forwarded to the screen as well as the status of the RF Transmitter.

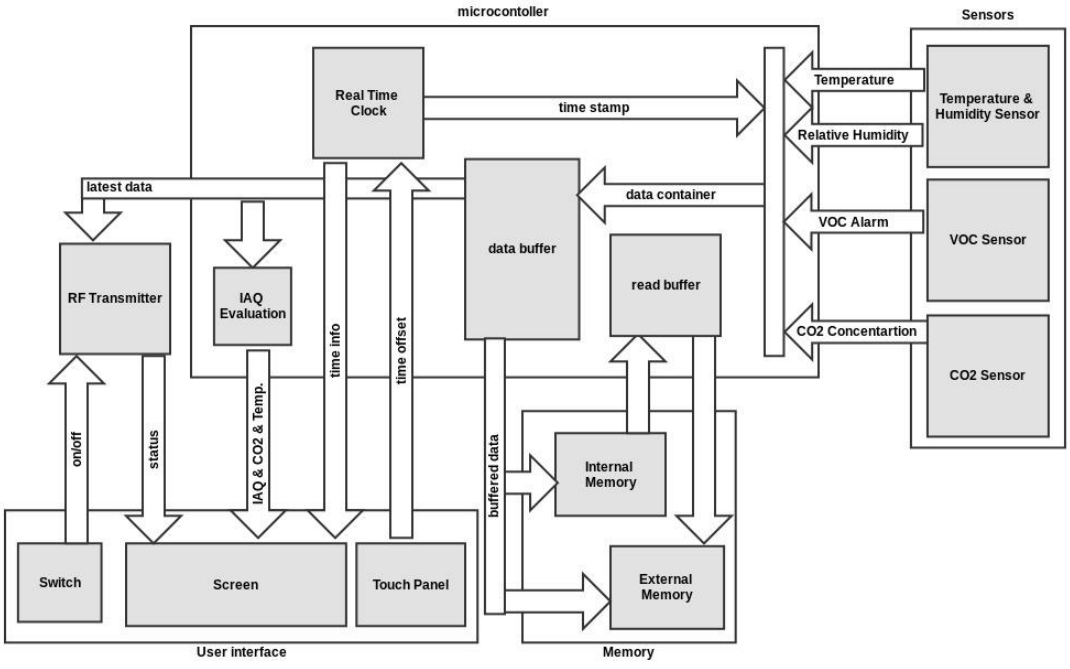


Figure 4. Block diagram of the IAQ Monitoring System

5 Hardware Design

5.1 Microcontroller Board

The microcontroller used in this project is the mbed LPC1768 prototyping board. It is a fast prototyping development board, containing a microcontroller (MCU) with the auxiliary circuits, a flash bootloader, a USB bridge and an Ethernet transceiver. The LPC1769, the MCU used on this board, contains a 32-bit ARM Cortex-M3 processor at 96MHz. Besides from 512KB FLASH and 32KB RAM it contains several peripherals modules USB-controller (Host & Device), CAN bus interface, SPI, I^2C, A/D-converters, D/A-converter, PWM and other I/O interfaces. The bootloader can be accessed through USB and will be recognised by any computer as a mass storage device. The program can be stored via drag&drop in the flash storage of the bootloader and is loaded into the MCUs Flash with the next reset.

5.2 Sensors

VOC Sensor

The Sensor selected to detect the presence of VOCs is the TGS 2602 by Figaro. It is a resistive gas sensor, consisting of a resistive heater and a metal oxide sensing film, which lowers its resistance due to present VOCs. As shown in figure 5 it has a high sensitivity to several gases from the group of VOC.

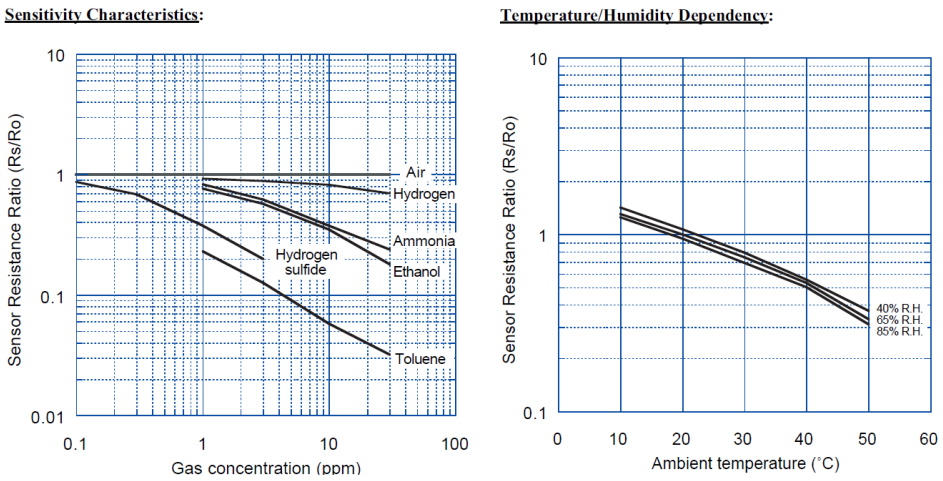
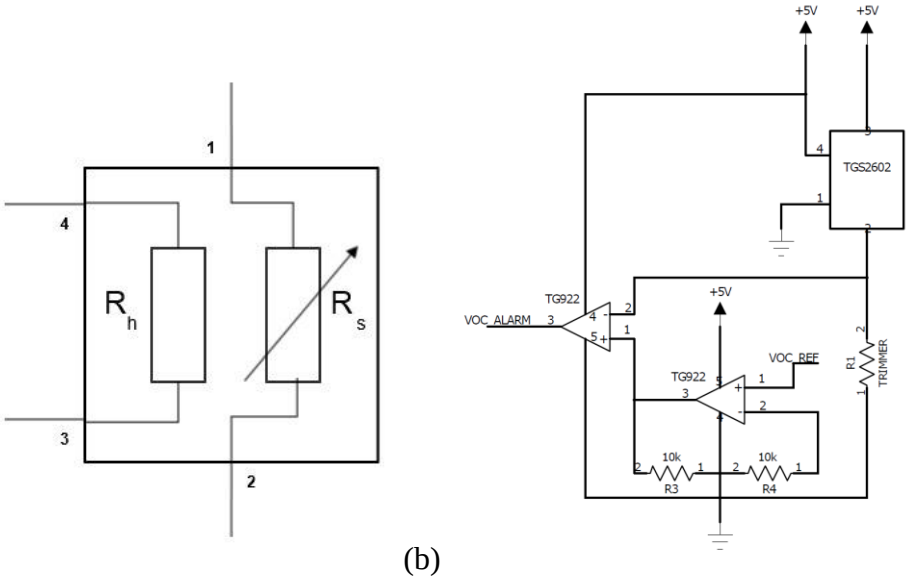


Figure 5. Sensor reaction to various gases & temperature & humidity V_C Reprinted from: [14, 6]

The sensitivity to multiple gases respectively the poor selectivity of the TGS2602 makes it impossible to determine the absolute concentration of present VOC. Therefore the sensing circuit will detect the presence of the supervised gases and create an alarm signal when VOCs are present.



(a) (b)

Figure 6. (a) TGS2602 equivalent circuit& (b) VOC sensing circuit

The sensing resistor R_S and the load resistor R_L create a voltage divider. The voltage over the load resistor increases, when the sensing resistor R_S decreases its resistance due to present VOCs. If the voltage over the load resistor R_S exceeds the reference voltage, the comparator creates the alarm signal.

Unfortunately the sensing resistance is highly temperature dependent, as figure 5 shows. To compensate this dependency and prevent false alarms, the reference voltage is controlled by the MCU. The amplifying circuit converts the 3.3V level of the microcontroller to the 5V level of the sensing circuit.

Figure 7 shows the dependency of the voltage divider voltage ration on the resistance ratio of the voltage divider. Given a resistance ratio of 1, the curve reaches its highest slope inflexion point. This point has the highest slope, which results in the biggest voltage change per resistance change. Hence, to achieve maximum sensitivity, the value of the load resistor has to be equal the value of the sensing resistor.

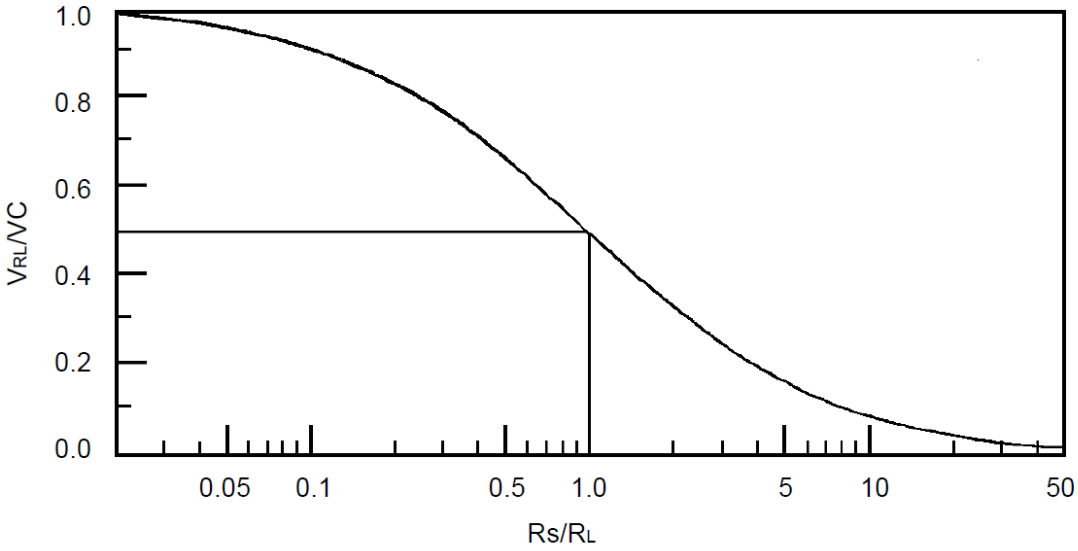


Figure 7. Relationship between R_S/R_L and V_{RL}/V_C Reprinted from: [8, 6]

For calibration purposes a trimmer is used as load resistance. To calibrate the sensor the load resistor is tuned under controlled conditions until the voltage over the load resistor reaches half of the sensors supply voltage.

CO2 Sensor

The CO2 concentration is measured with an optical sensor. The SenseAir S8 is a relatively small NDIR sensor. The advantage of this type of sensors is their higher accuracy and superior long-term stability, necessary for the long-term measurement. The sampling period of two second and hence the sampling frequency are high enough for our purpose. The measurement data are stored in the internal memory of the sensor. The sensor provides two ways to access the data. There is a PWM output and a UART interface. The duty cycle of the PWM represents the ratio of measured concentration and the normal measurement range from 0ppm to 2000 ppm. On the other hand, the stored measurement data can be requested via the SPI interface. The CO2 reading requested via the SPI is provided in the range from 0 to 3200ppm.

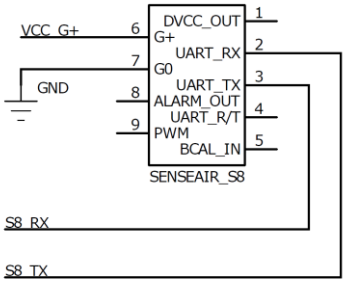


Figure 8. Senseair S8 pin assignment

Unfortunately none of the sensors pins shown in figure 8 are protected. To minimize the risk of damage, as few connections as possible are used. The SPI interface is used to initialize the calibration procedure as well as to read the sensors status and the measurement. Hence the bcal pin and the PWM stay unconnected and the internal 120k pullup resistors tie them to a fixed level and prevent unwanted activation.

Temperature & Humidity Sensor

The temperature and humidity are monitored by the DHT22. The temperature is measured by means of a thermistor, the relative humidity by capacitive means. Both values are accessible via a digital 1 wire communication line. A pull-up resistor R2 in figure 9ties the communication line to high when the line is silent which is required by the used communication protocol. Furthermore a decouple capacitor C5 is connected in between the Vcc and ground to stabilizes the power supply. The senor is capable to measure from -40°C to 80°C and from 0- 100% relative humidity, but the highest sampling rate is one measurement every two seconds. If this requirement is violated, the sensor may become unstable or deliver false values.

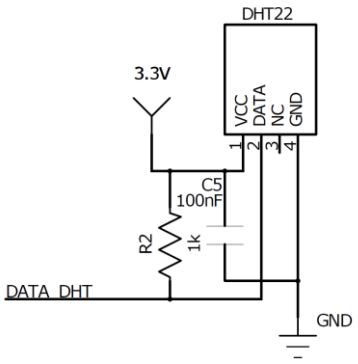


Figure 9. DHT22 pin assignment

5.3 Screen

To communicate the gained information to the users by visual means, a mikroTFT by Mikroelectronika is used. The mikroTFT is a breakout board for a 320x240 TFT touch screen consisting of a resistive touch panel and a MI0283QT-9A colour display, driven by a ILI9341 display controller. The screen can either be controlled via parallel connection with several different bit sizes or via serial connection. Because no big amounts of data are displayed and instant response is not required, the serial connection is used to save hardware resources. The unused parallel input pins are terminated by connecting them to +3.3V, respectively to Ground potential to prevent false inputs and malfunction of the screen. The serial connection uses the SPI protocol and is directly connected to the mbed's SPI interface.

A pre-resistor R13 protects the back light LEDs from overcurrent. The outputs of resistive touch panel X+, X- , Y+, Y- are directly connected to the MCUs A/D-converter inputs.

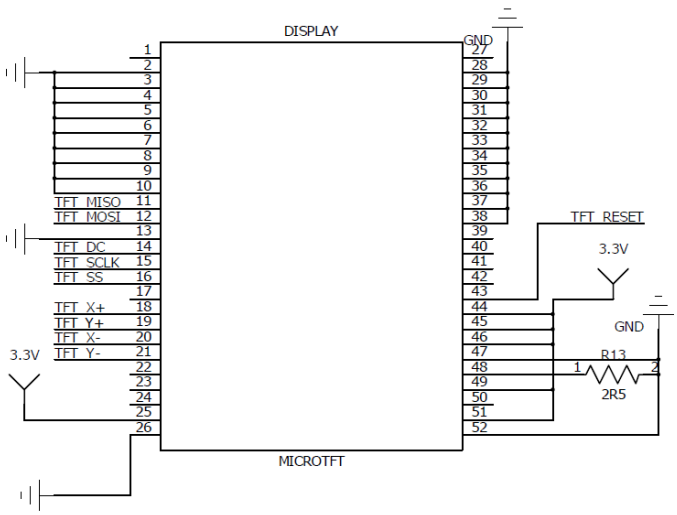


Figure 10. mikroTFT pin assignment

5.4 Memory

To store the measurement data, a non-volatile memory is needed, to keep the data even after a black out. The MCU is able to access the flash memory of the bootloader and store data, but the 2 Mb of memory is not sufficient. A test file filled with dummy data, for a whole day, reached a size of around 3 MB. Hence an additional memory device is needed.

A Secure Digital Memory Card (SD-Card) is a mass storage device consisting of flash memory and a control unit. The internal processor unit handles the communication with the host device, and takes care of the memory access and all write and read operations. Data and commands can be passed to the controller either via a 4bit parallel SD bus or via a serial SPI Interface. As the internal processor handles the write and read operations independently, the workload of the MCU is reduced, making the SD card a good choice, for both, the internal memory as well as external memory. Furthermore SD-Cards are widely available and small in size and power consumption. Because a SD-Card is used for the internal memory as well, it is easy to upgrade the system and increase the memory.

The chosen SD-Card holder contains a voltage regulator, converting a 5V input voltage to the required 3.3V power supply for the SD-Card and a switch, which connects to ground when a SD-Card is in place.

The MCU communicates with the used SD-Cards via the SPI interface. Figure 11 shows the implemented circuit. Because the two SD-Cards are never used at the same time, the SD-cards share the data lines as well as the clock signal to save hardware costs. The card used at the moment is activated with the corresponding Slave Select line.

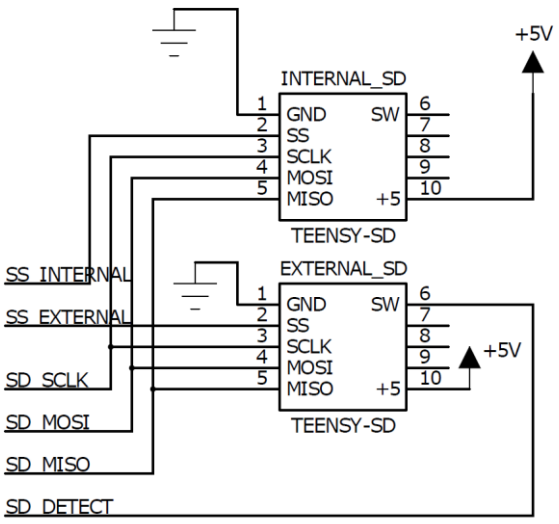


Figure 11. SD-Card pin assignment

5.5 Radio-Link

Xbee are intelligent RF modules, which are available in several different versions. The versions differ by their performance or by the used standard or the protocol used for the transmission. The used version is selected to communicate with the ConnectPort X4, hence the Xbee 802.15.4 Series 1 is picked. The implemented communication standard is the same like the standard implemented in the ConnetPort X4, the IEEE 802.15.4. Furthermore it contains digital input and output lines as well as A/D converters. The basic structure of a Xbee module can be divided into the processing unit and the Transceiver unit. The processing unit communicates with the host device and converts the received data into a message signal and vice versa. The transceiver unit handles the QPSK modulation and demodulation as well as the signal amplification.

5.6 Power Supply

The system requires two different Voltage level. 5V as well as 3.3V are used in the system. The mbed LPC1768 has an internal voltage regulation circuit, shown in appendix 1. This includes two LD1117S33 voltage regulators, converting an input voltage from 9 to 4.5V into a 3.3V level. One of them is powering the mbed system, while the other one is directly connected to an output pin, providing a 3.3V. This voltage is used to drive the screen as well as the DHT22 sensor.
The remaining parts of the system require 5V power supply. They are powered by the voltage regulating circuit, shown in figure 12, which consists of two LM7805 voltage regulators. Additionally the LD1117S33 of the mbed are driven with the 5V as well.

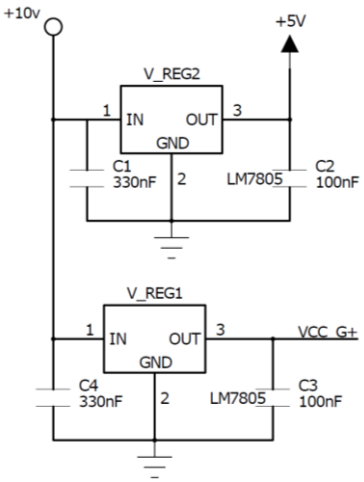


Figure 12. Voltage Regulator Circuit

The SenseAir S8 sensor requires special attention due to its non-protected inputs. Furthermore, the sensor draws instantaneously relatively high amounts of current, according to the specifications up to 300mA, causing noticeable voltage drops in the supply line. To protect other parts of the system from possible malfunction, due to the voltage drops and lower the risk of possible voltage spikes caused by other parts of the system damaging the sensitive SenseAir S8 sensor, the sensors supply voltage will be regulated with an independent LM7805.

6 Software Design

6.1 mbed SDK

The developed software is based on the mbed Software Development Kit (SDK). It is a C/C++ platform published under the open source Apache 2.0 and consists of an online programming environment, including an online compiler, test and debugging scripts and a set of official libraries and APIs. The APIs, based on the CMSIS APIs by ARM, are drivers for the MCU's peripheral and provide a tested interface to access the functionality of the peripheral blocks. Cortex Microcontroller Software Interface Standard (CMSIS) APIs abstracts the hardware of the processors of ARM's Cortex-M series, creating a common interface for all models of the series. Therefore and because of the abstraction the mbed APIs and code based on them, may be used with multiple platforms with different MCUs and processors.

Furthermore, official libraries are available, providing the network abilities, file system and Real Time Operation System.

Besides from the official libraries there is a big collection of libraries, created and provided by the developer community. Several of those libraries are featured in the mbed cookbook, a selection of published libraries and projects with explanation and documentation.

The mbed APIs used in this project are DigitalIn, AnalogIn, AnalogOut, Serial, SPI and Timer. They are implemented as classes and their functionality is represented by their member functions. The class DigitalIn represents an I/O pin used as digital input. The classes AnalogIn and AnalogOut represent the internal A/D-converters, respectively the D/A -converter. The classes Serial and SPI represent the internal communication blocks, where Serial represents the UART interface. The class Timer controls the internal timer and provides conversion from timer values to seconds and back.

6.2 Structure of Software

The structure of the project continues with the idea of the abstraction of hardware parts into software, introduced by the mbed library. The hardware is abstracted by a class and the functionality is represented by a member function. The VOC sensor is an exception, it does not have an own class due to it simple structure.

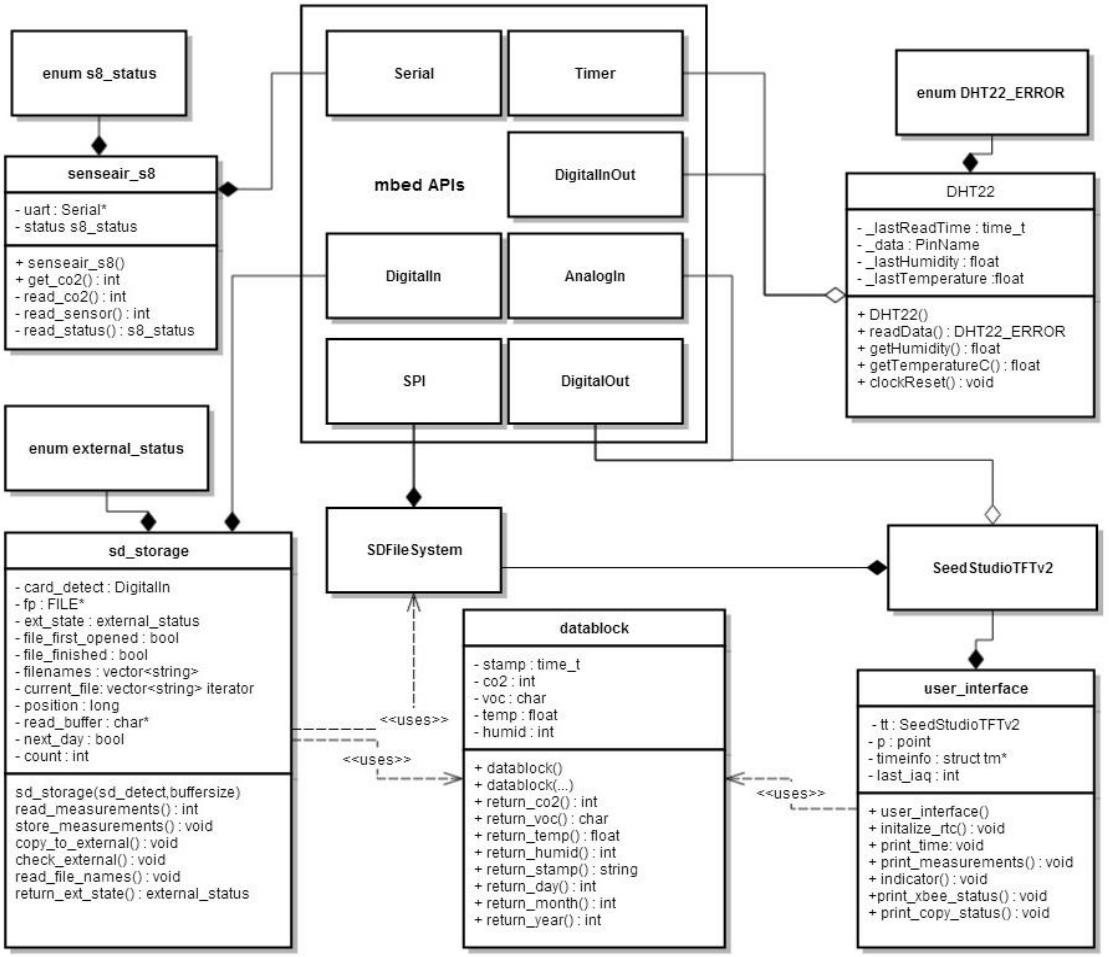


Figure 13. Class diagram of the system

Figure 13 is the class diagram, displaying the classes used in this project, their structure and their relations. In the following part the structure and the important functions of these classes will be explained

6.3 Class Datablock

One important class in this project is the *datablock* class. The class is built as a container for the measurement data. The purpose of this class is to protect the measurement data against unwanted modification and to keep the data of one measurement cycle together. Therefore the class has a private member variable for each measurement and one for the time information, which are only accessible through memberfunctions. The constructor is the only possibility to modify these variables. The measurement data are passed to the object as parameters and stored in the respective variable.

6.4 Sensors

DHT22

MaxDirect 1-wire-bus

The Sensor DHT22, also known as RHT03, uses a MaxDirect 1-wire bus for communication. The protocol determines the value of a transmitted bit by the duration of a high level on the data line. When unused, the bus is constantly pulled up to a high level. The MCU initializes the communication by pulling the line to low for 1 -10 ms before pulling the line back to high for 20-40µs and waiting for the sensor response. The sensor acknowledges the start signal by pulling the line first to low and then back to high, each for 80 µs. Afterwards the data transmission starts. The transmission of one bit starts with a low level for 50µs forced by the sensor, followed by a high level which determined the value of the bit. If the duration of the high level is around 26 – 28 µs, the bit is interpreted as a '0', respectively if the high level lasts for around 70 µs the bit is interpreted as a '1'. After the last bit of the data, the sensor pulls the line to a low level to determine the end of the latest bit's high period and releases the line afterwards. The MCU pulls the line back to a high level and the transmission is completed.[15]

Class DHT22

The sensor Dht22 is a simple and often used sensor and a library is available from the mbed cookbook. The library contains the class DHT22 which is used without further changes. To get the data from the sensor two main steps are needed. First the member function "`readData()`" has to be called. In the beginning the function checks if the time requirement has been violated.

If 2 seconds have passed since the last data request, the function follows requests a new set of data from the connected sensor and stores them in private member variables. To request the data, the function follows the MaxDirect protocol described above. Further the function returns a value of the `DHT22_ERROR` type, representing the error status of the sensor. Afterwards the temperature and the relative humidity values can be accessed with the member function "`getTemperatureC()`" respectively with the member function "`getHumidity()`"

SenseAir S8

ModBus Protocol

The serial communication interface of the SenseAir S8 is based on the ModBus protocol, an open protocol for Programmable Logic Controllers and Sensors. Next, the sensor's implementation of the protocol will be described.

The protocol is set up on top of the UART protocol, hence working with byte orientated transmission. The byte format for the UART interface is given by the implementation as well. The message starts with one startbit followed by 8 data bits. A parity bit will not be transmitted. Finally the sensor sends two stop bits but when receiving, the sensor accepts as well a single stop bit. As well the implementation sets the baud rate to 9600bps.The protocol divides between master devices and slave devices, while only master devices are allowed to initialize communication.

The first byte of a Modbus frame is the address field, determining the called slave. The address is followed by the function code. The function code determinates the operation, executed by the slave. When responding, the slave always repeats the address and the function code first. The data block is dependent on the function code, either parameters required by the function code or data requested by the master from the slave. The frame is terminated by a two bytes long CRC value.

Due to the various kinds of content of the data block the length of this block may vary as well. The maximum length of a frame defined by the ModBus specifications is 255 bytes including address and CRC but the sensors implementation limits the maximum length to 39 bytes. If the silent interval in between 2 bytes, measured from the latest stop-bit to the next start bit, exceeds 1.5 characters, the frame is considered finished.

Furthermore the specifications require a 3.5 character long silence interval before every new ModBus frame.If these requirements are not met, the sensor refuses to respond.

Several function codes are supported by the ModBus protocol, but not all of them are implemented in the sensor as shown in table 1. [16]

Table 1. Table of all available function codes in the ModBus protocol, with corresponding name and parameters

Functioncode	**Name of Fuction**	**Structure of Functioncode**
0x01	Read coil	Not implemented
0x02	Read Discrete Inputs	Not implemented
0x03	Read Holding Registers	Funct.Code+Start Address+Quantity of Registers
0c04	Read Input Register	Funct.Code+ Start Address+Quantity of Registers
0x05	Write single coil	Not implemented
0x06	Write single register	Func. Code &Address& Value
0x0E	Read Device ID	Not implemented, yet (same like 0x2B)
0x0F	Write Multiple coil	Not implemented
0x10	Write Multiple Registers	Not implemented
0x14	Read File record	Not implemented
0x15	Write File record	Not implemented
0x16	Mask Write Register	Not implemented
0x17	R/W Multiple Registers	Not implemented
0x2B	Read Device ID	Not implemented, yet (same like 0x2B)

ABC Algorithm

Under normal conditions the Senseair S8 works maintenance free, due to the build-in Automatic Baseline Correction (ABC) algorithm. The algorithm tracks the lowest values measured and slowly corrects the 400ppm reference value. To prevent maladjustment due to the ABC algorithm, the CO2 concentration has to settle down to the outside level once in a while. Considering that the learning facilities are unused during weekends and holidays and the intended goal of the system is to reach sufficient ventilation, this problem is negligible. But the outside concentration may vary between 350ppm and 450ppm depending on the region, and therefore the lowest value measured may differ from 400ppm which will cause a slight maladjustment as well.[17]

Class senseair_s8

The class senseair_s8 represent the sensor in software and uses mbed's Serial API to establish a communication in between mbed and the sensor's serial interface via UART. The member function represents the main functions required to run the sensor. Furthermore an enum type`s8_status`, representing the different errors which the sensor may return when reading its status register, was created for this project.

The Error flags are stored in the lowest 7 bits of the first Input Register IR1 and the latest CO2 reading is stored in the Input Register IR04 as a 16-bit value. In this case the "Read Input register" function code (0x04) is used to request the stored information from the sensor. First `the read_sensor()` functions sends the broadcast address (0xFE) using the Serial class of the mbed library. This is followed by the functions code (0x04). Afterwards the register, where the reading starts, is specified. The register IR1, has the address 0x0000. Following, the amount of registers to read is submitted. The amount is set to 4 (0x0004) to include both, the status register and the CO2 readings. The CRC value (0xC6E5) is terminating the ModBus frame and the lowest byte (0xE5) is sent first, followed by the last byte (0xC6). Resulting from this, the full message sent is:

"0xFE" "0x04" "0x00" "0x00" "0x00" "0x04" "0xE5" "0xC6"

Afterwards the function waits for the sensor to respond and buffers all incoming bytes. The response consist of the repeated address (0xFE) and the function code (0x04) followed by the amount of bytes read(0x08) and the data read. Again, the ModBus frame is terminated with the CRC value and again the lowest byte is send in advance of the higher byte.

"0xFE" "0x04" "0x08" "Status High" "Status Low" "0x00" "0x00" "0x00" "0x00"
"CO2 High""CO2 Low" "CRC Low" "CRC High"

After receiving all bytes, the function merges the 4tha 5th byte of the buffer by shifting the 4th byte 8 bits to the left and combining both with a bitwise OR operator. The result is stored with a cast operator in the status variable of the senseair_s8 class. The 10th and 11th byte, representing the CO2 concentration, are merged in the same way and returned as an integer value.

The `read_status()` function uses the same procedure as described above with the unchanged address(0xFE) and function code(0x04). The starting address (0x0000) is the same as well, but the amount of registers to read is reduced to 1(0x0001). Hence, the CRC(0xC525) alters as well. The sensor's answer is buffered as well and again the 4th and 5th byte of the buffer contain the wanted information. Both bytes are merged as

described above and converted into the `s8_status` type with a cast operator. The result is returned.

The `read_co2()` function uses the same procedure like the two functions previously described. But in this case, the only the Input Register, containing the CO2 reading, is read. Therefore the Starting address is 0x0003 and the amount of registers to read is 0x0001. Again the CRC has changed. In this case the CRC value is 0xC5D5.

The CO2 values are contained in the 4th and 5th byte of the buffer. Like in the two other functions, both bytes are merged via shifting and bitwise OR-operation. The result is returned as an integer value.

TGS2602

The reading of the TGS2602 is very simple, determine whether the input is either high or low by using mbed's DigitalIn API. So the VOC sensor is not represented by an own class.

The figure 5 (chapter 5.2) shows the temperature dependence of the sensing resistance R_S. Consequentially the voltage over the load resistor R_L changes with the temperature, as well. To prevent a false alarm, the reference voltage has to be adjusted in the same way, the voltage over the load resistor changes with the temperature. The voltage over the load resistor is given by equation (1)

$$V_{Rl} = \frac{R_L}{R_L + R_S} * V_{cc} \qquad (1)$$

Linear interpolation is used to approximate the resistance values over the temperature range. Therefore several nodes are taken from the temperature dependence curve in figure 5. These are shown in Table 2.

Table 2. Resistance ratio and corresponding temperature values, taken from figure 5

$Temperature/^{\circ}C$	$\frac{R_S}{R_0}$
10	1.4
20	1
30	0.75
40	0.525
50	0.225

The resistance ratio at a temperature in the temperature interval from T_0 to T_1 is defined as

$$\frac{R_S}{R_0}(T) = \frac{R_S}{R_0}(T_0) + \left(\frac{\left(\Delta\frac{R_S}{R_0}\right)}{(\Delta T)}\right)*(T - T_0) \quad (2)$$

$$\frac{R_S}{R_0} + \frac{\left(\Delta\frac{R_S}{R_0}\right)}{(\Delta T)}*(T - T_0) = k(T) \quad (3$$

$$R_S(T) = k(T)*R_0 \quad (4)$$

With this term for the temperature dependent resistor (4), the term for the voltage over the load resistor (1) becomes temperature dependent as well.

$$V_{Rl}(T) = \frac{R_L}{R_L + R_S(T)}*V_{cc} \quad (5)$$

$$V_{Rl}(T) = \frac{R_L}{R_L + k(T)*R_0}*V_{cc} \quad (6)$$

Considering a sensor calibrated for maximal sensitivity as described in chapter 5.2.1 the value of R_0 can be assumed to be equal to R_L, resulting in an approximation for the behaviour of the voltage over the load resistor.

$$V_{Rl}(T) = \frac{R_L}{R_L + k(T)*R_L}*V_{cc} = \frac{1}{1 + k(T)}*V_{cc} \quad (7)$$

The reference voltage for the VOC sensing circuit is defined as the temperature dependent voltage over the load resistor plus a voltage offset representing the possible voltage drop caused by present VOCs.

$$V_{ref}(T) = \frac{1}{1 + k*(T - T_0)}*V_{cc} + V_{alarm} \quad (8)$$

To prevent false VOC alarm, the `temp_compensate(float temp)` function implemented in the program, selects the temperature interval based on the temperature information passed through as a parameter. The resistance ratios of the nodes are stored in a look up table in the flash. Depending on the temperature range, the values of the next higher and the next lower node are used to calculate the compensated reference voltage. The build in D/A converter of the MCU is represented by the AnalogOut API and may create an output voltage in the range from 0V to 3.3V. To expand the voltage range to maximum 5V, an amplifying circuit with an amplifying factor of two has been implemented. This amplifying factor has to be taken in account during the D/A conversion.

6.5 Memory

FAT File System

To save the measured data in a computer readable format, a file system is required. The File Allocation Table (FAT) File system is widely used due to its simple structure.
The memory is divided into sectors and a fixed structure is implemented. The First sector is a bootsector, containing general information on the system and the bootloader code. The next sectors are reserved sectors, followed by the File allocation table. The remaining sectors are grouped to clusters and contain the actual data and directories. The first cluster after the FAT is normally the root directory. Every file or subdirectory has an entry in the directory it is stored in. A file occupies at least one cluster, even if its size is smaller than the cluster. The number of this first cluster is stored in the files directory entry. If the file's size exceeds the size of one cluster, the remaining data are stored in additional clusters, which do not need to consecutive. The File Allocation Table (FAT) contains the information which cluster belongs to a file. Every entry in the File Allocation Table represents one cluster in the accessible memory. These entries contain a value, either representing another cluster in the memory or the highest possible value. If the value represents a cluster, this cluster is the next cluster after the present one. If the value is the highest possible value, the current cluster is the last one of the file. To access a file, the number of the first cluster is stored in the file's entry in its directory. The following clusters are specified in the FAT.

Classsd_storage

The class `sd_storage` is used to store the data at the internal SD-card and to handle the transfer of data from the internal SD card to the external SD card.
To be able to use file operation, the class `sd_storage` requires a file system for each SD-Card to be implemented. Therefore, the SDFileSystem library, provided in the mbed handbook, is used to set a file system for both SD cards. With the file system set up, standard C-language file operation commands can be used.
Further the enum–type "`external_status`" is created, representing the status of the external memory device, indicating if it is present, and if the content has been updated or the copy process is still in progress.

The `store_measurements()` function stores the measurement data to the internal memory. Provided an external memory device is present and all previous measurement data have been transferred from the internal memory to the external device, the function stores the latest measurement data to the external memory, as well. Figure 14 shows the structure of function.

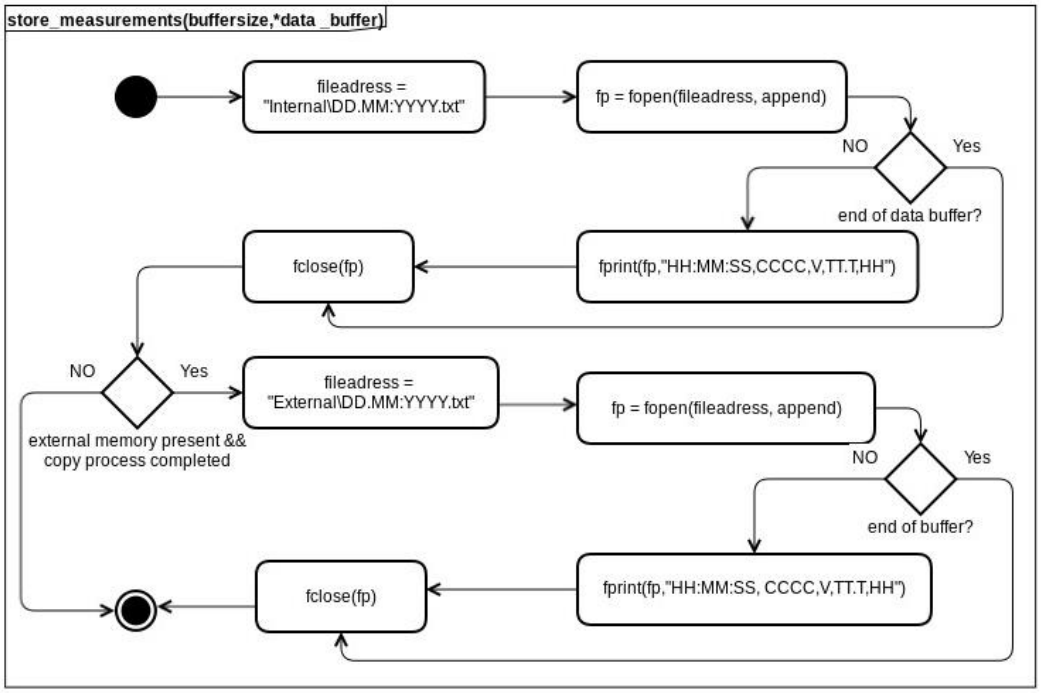


Figure 14. Activity Diagram of store_measurments() function

The measurements are stored in a buffer outside of the class. A pointer to the data buffer is passed to the store measurement function as a parameter. To store the data sorted by days, the filename is generated based on the time stamp of the data in the buffer. The date information stored in the first measurement are used to specify the filename. Together with the name of the internal file system, the name of the file is written into a string. The `fopen()` command is used to open a file and create a file pointer for the following file operations. The name and the location of the file is specified by the string containing the name of the internal file system and the filename created based on the date information. The measurement data and the time stamp, returned from the data container, are written in the file as a formatted string with a fixed size. First the time stamp in the HH:MM:SS format, 4 digits of CO2 reading, 1 Character for VOC alarm, temperature with 3 digits and two digits for the relative humidity. Unused digits are filled up with zeros. A For-loop is used to iterate through the data buffer. If an in-

crementing counter reaches the value of the buffer_size value, the For-loop is terminated and the file is closed.

Provided, that the external_status variable `ext_state` is set to "`copy_finished`", indicating, that all data stored in the internal memory have been copied to the external memory device, the latest measurement data are stored as well in the external memory. The data are written to the external memory in the same way, the data have been written to the internal memory, except that the name of the internal file system is substituted by the name of the external file system. After the data have been written in the file and the file is closed again the return command is reached and the function is finished.

This function `read_file_names(char* dir)` reads the directory, specified by the string parameter, and stores he names of all files in a vector of strings.

First the vector is cleared to erase the filenames of previous use. Afterwards the `opendir()` command is used to open the wanted directory. The structure `dirent` represent a directory entry which contains several information, including the file name. The directory entry is read with the `readdir()` command. The file's name is added to the filenames vector and the next directory entry is read.

The function `read_measurements(int size, string filename)` reads a specific number of bits from a file. Both number of bits to read and the name of the file are passed to the function as a parameter. Figure15 shows the structure of function.

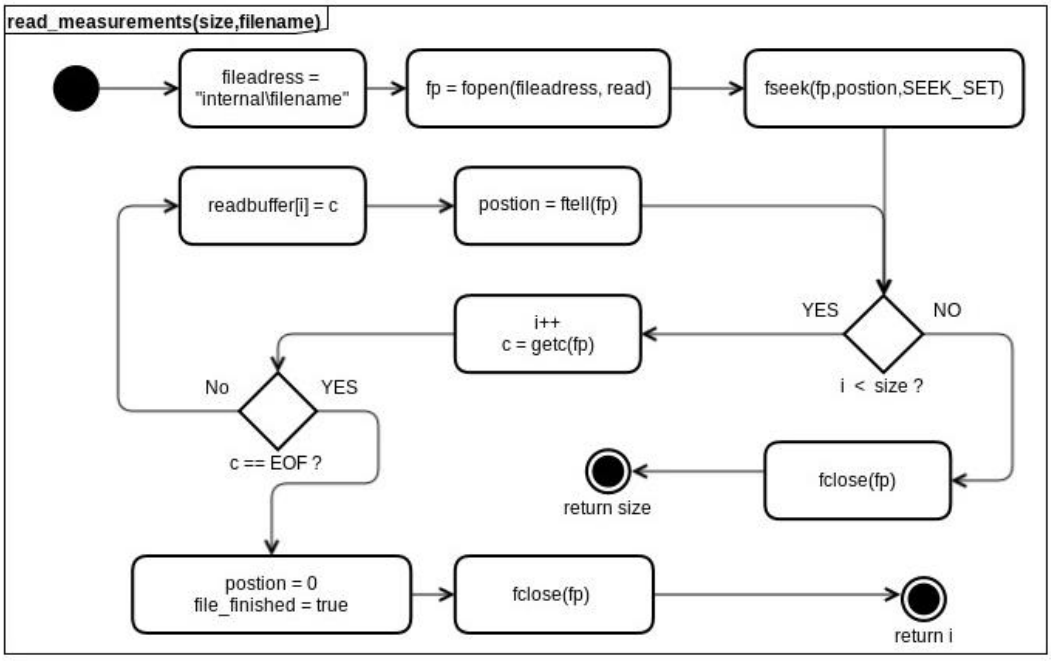


Figure 15. Activity Diagram of read_measurments() function

The file name passed over as a parameter is copied in a string with the name of the internal file system, specifying the location of the file to read. The `fopen()` command opens the file with read-only access. After the file is opened, the readpointer within the file is repositioned to the last character read the previous time. The offset of the read-pointer from the start of the file is stored in the variable “position”.
After repositioning a FOR-Loop is executed until a counter reaches the value of the passed parameter size, the counter is incremented each time the statement of the FOR-loop is executed.Figure16 shows the structure of function.

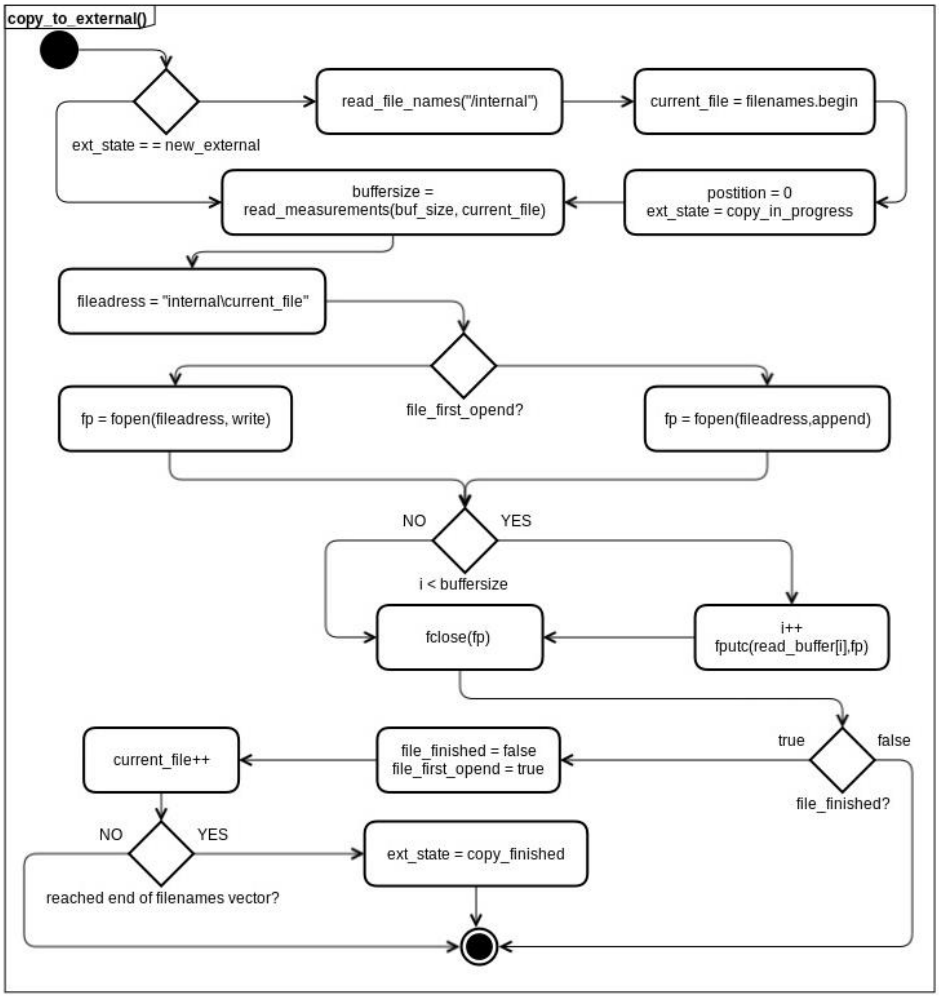


Figure 16. Activity Diagram of copy_to_external() function

Within the loop, one character is read from the file. This character is compared with EOF, a value representing the End of File. If the character is not equals EOF, the character is stored in the read_buffer and the current offset of the readpointer is stored in

the variable position. If the counter reaches the value of size, the FOR-loop is terminated. Afterwards the file is closed and the function returns the amount of characters read which is equal to the value of size.

If the file reaches its end before the counter reaches the value of size, and the value EOF has been read, position is set to 0. Further, the bool-variable file_finished is set to true, indicating that the data copied in the read_buffer have been the last of this file. Afterwards the file is closed and the amount of characters read is returned, which is equal to the value of the counter of the FOR-loop.

When an external memory device is detected, the system automatically copies the data stored in the internal memory to the external device. The *copy*_to_external() handles this copy process. The function reads a small part of a file, writes the data in a buffer and stores the position of the last character read. The data, stored in the buffer is written to a file in the external device. Further the function detects when one file is finished and automatically starts with the next one, or indicates that the copy process is completed.

If the ext_state variable indicates, that there is a new SD card in the slot, the function calls read_file_names(), a memberfunction described above, loading the names of the files stored in the internal memory in a vector. The iterator current_file is directed to the first entry of the vector and position is set to 0. Further, ext_state is changed to “copy_in_progress”.

Next, the read_measurements(size,filename) is called to read the data from the internal memory into the read_buffer. The size of read_buffer and the name of current file are passed over as parameters. The return value of the function is stored in the local variable buffer_size and used to just write as much characters from read_buffer as have been read in. A string is created, specifying the address of the file, in which the data is copied. The string is made up of the name of the file system of the external memory device and the element of the vector filenames the iterator current_file is pointing at. The string is handed over to the fopen() command to specifying the address of the file. The Access mode depends on if the has file been opened for the first time within the current copy process, indicated by the boolean variable file_first_opened. If file_first_opened is true the file is opened in write mode, overwriting every file with the same name in the external memory, to prevent problems with files from unknown origin. Otherwise the file is opened in append mode.

The function writes a certain amount of data in the file. When the amount of date written in the file reaches the buffer_size, the file is closed.

If the `read_measurement()` function has set `file_finished` to true, indicating that it reached the end of file when reading the data, the interator of the filenames vector is incremented and the indicators`file_finished` and `file_first_opened`are reset. If the iterator has reached the end of the vector after the incrementation, the copy process is completed and according to this `ext_state` is modified.

6.6 User Interface

The class `user_interface` handles the communication in between the user and the system. On one hand, it controls the information output to the screen and the screen layout. On the other hand, it provides a set up dialogue for the Real Time Clock. The class utilizes an object of the class *`SeeedStudioTFTv2`* to communicate with the display controller and modify the screen output. The class *`SeeedStudioTFTv2`* is part of a library provided in the mbed cookbook.
All member functions containing “print_” in their name are used to create the layout shown in Figure 17 and modify the displayed values during operation.

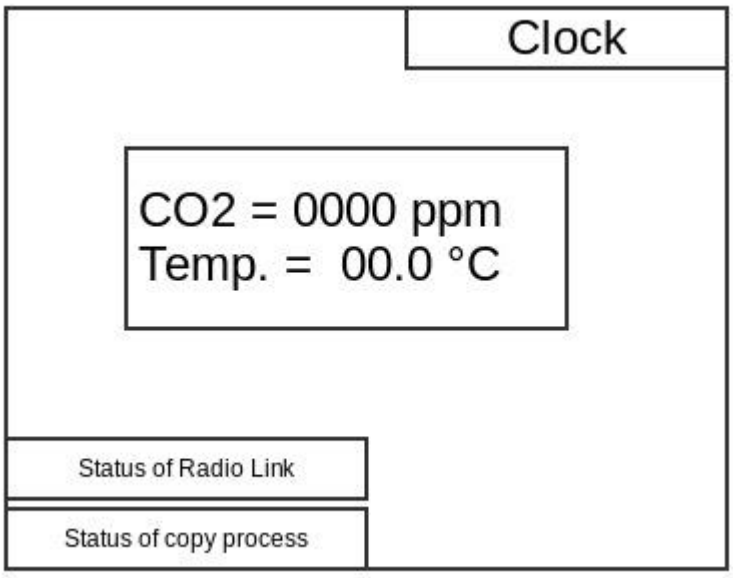


Figure 17. Screen layout of user interface

`print_measurements()` reads the CO2 concentration and the temperature from the datablock object, which is passed to the function as a parameter, and prints the values to the screen. `print_time()`directly accesses the RTC and reads the current time

information, before printing the current time in the HH:MM:SS format to the screen. `print_xbee_status()` and `print_copy_status()` print the present status of the radio link and the copy procedure.

The function `initalize_rtc()` is the used to set up the internal Real time clock of the mbed, which is used to generate the timestamp for each measurement taken. After power reset, the RTC is reset to the January 1st 1970. To get correct time information, an offset needs to be added. Therefore this is the first function to call. But before actually setting up the RTC, the touch screen is calibrated, by calling the `calibrate()` function of the used `SeeedStudioTFTv2` class. The function asks the user to touch two crosses displayed in opposite corners on the touch screen. When the user touches the crosses, the conversion coefficient from voltage to pixel is determined. The touch panel calibration is followed by a welcome screen, which remains until the screen recognizes a touch. Afterwards the RTC set up dialogue starts. Figure 18 shows the layout of the two steps of the dialogue

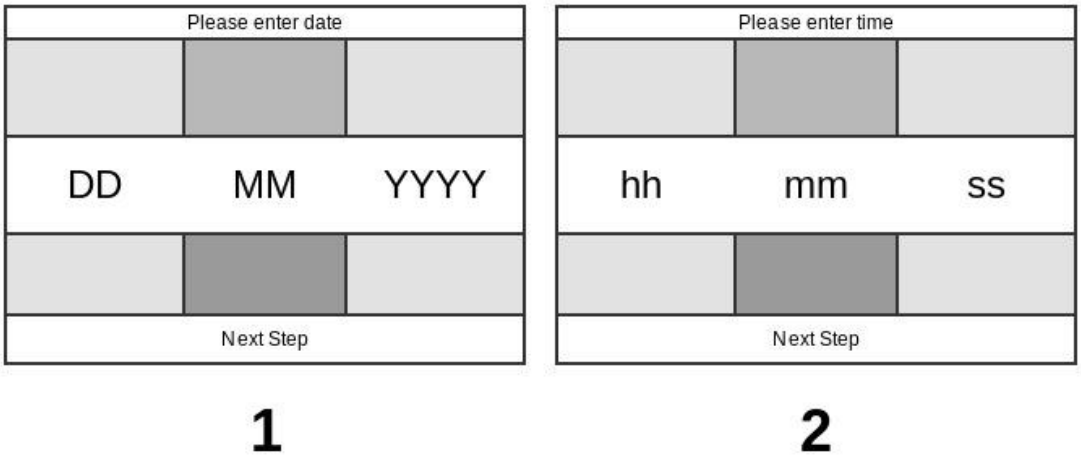


Figure 18. Screen layout during RTC set up procedure

The first step is meant to set up the date, the second step is meant to set up the time.

The function waits for the screen to be touched. When touched, the reading of the touch panel is converted into pixel values by means of the conversion coefficient gained earlier. If the touching point lies within one of the gray field the current date is modified. The value to be modified is selected by the column and depending on the row, the value is either increased or decreased by one. The current values are displayed in the middle row. If the touching point lies within the area at the bottom of the field, the current step is considered to be completed and the current values are stored.

After both steps are completed, the offset for the RTC is calculated and the RTC is set up.

6.7 Main program

The Main program is the central part of code, controlling the behaviour of all parts of the system by controlling objects of the classes representing each part. Appendix X shows the structure of the main program as an activity diagram. The time, passed since the last measurement, is checked. If it exceeds 2 seconds, the reference voltage for the VOC measurement is adjusted, a new set of measurements is taken, stored in an object of datablock and the object is stored in a buffer. Furthermore, if the program recognises that the RF module is switched on, the data from the latest measurement are printed into a string and transmitted via UART to the Xbee module. Afterwards, the latest data are examined and the Indoor air quality is rated. Next, all new information is handed over to the `user_interface` object and printed to the screen, including the new co2 and temperature values and the IAQ indicator. Independent from the time since the last measurement, the current time is printed to the screen. If the buffer, which stores the measurement data, reached its end, the storing function of the `sd_storage` class is called and the data are transferred to the internal memory. The variable indicating the current position in the buffer is reset to 0. Finally a package of data is transferred from the internal memory card to the external, given the fact, that the external card is present. Otherwise this part is skipped.

Unfortunately this set up does not work. It got stuck when using `the copy_to_external()` function or the `store_measurement()` function of the `sd_storage` class, even though the class and the functions were tested before and worked when used independently. During troubleshooting it emerged, that the problem starts to occur when the program reaches a certain size, respectively if a certain amount of objects have been initialized. Furthermore file operation commands like `fopen()` have been identified as origin of the problem. Resulting from these findings, it is suspected that an undersized RAM causes the problem.

The Random Access Memory (RAM) in an MCU is used to store all non-constant information of a program. Figure 19 shows a model of the RAM in the MCU.

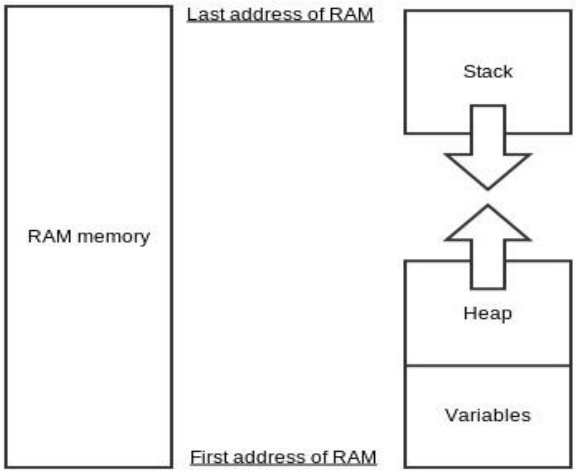


Figure 19. Memory Model of the mbed LPC1768

The RAM may be divided into three areas. The first area starts at the first address of the RAM and contains the all static and global variables which are defined in the program. The size of this area is dependent on the amount of static or global variables defined and will stay constant during the whole runtime of the program. The second area begins directly after the end of the first area and is called the heap. The heap contains all data which have been dynamically allocated. Therefore the heap varies its size during the program. New data are added at the end of the heap, hence it grows from towards the last address of the RAM.

The third area is called Stack and starts at the last address of the RAM. It is used to store variables with a limited lifetime, like local variables or data passed as parameters. The size of the stack is variable as well and changes during the runtime. The Heap and the stack grow towards each other and there are no protection mechanisms, preventing them to interfere. Hence, the stack may overwrite parts of the heap and vice versa if one of them grows too big.

When calling a function, the files parameters as well as all local variables are stored in the stack. The SDFileSystem library, which provides he functionality for the file operation commands, is highly structured and divided into several different classes. When calling the file operation commands many boxed functions are called, flushing the stack with parameters and variables and causing it to grow until the last function of the function tree is reached. Afterwards when returning back up the function tree the stack deceases. [18]

If there is little space occupied by other objects and variables, this has no effect on the program, because even though the stack grows very big, it decreases directly after-

wards. On the other hand, if many registers are already occupied by global variables, the stack grows and interferers with the heap or even the global variables. This leads to intense malfunction and causes the program to get stuck.

Even though the problem and its origin are known, it could not be solved, due to lack of time and recourses. There are possibilities to fix this problem. The used MCU contains 32kB of RAM, but only 16kB are accessible for the program, the other half is reserved for the Ethernet stack etc. Because the Ethernet is not used, this memory could be used to solve the problem and prevent a stack overflow. But the mbed SDK does not offer this possibility, yet. Another option to increase the accessible RAM, is to substitute the MCU. Due to the interchangebility of the mbed platforms, this would be an easy fix. If increasing the RAM is not possible, the structure of the system has to be changed and the amount of occupied registers during the file operations has to be decreased. Or the structure of the `SdFileSystem` has to be changed.

To restore the functionality of the system, the `sd_storage` class is removed and the data logging feature is excluded temporary from the project. The structure of the slimed program is shown in figure 20

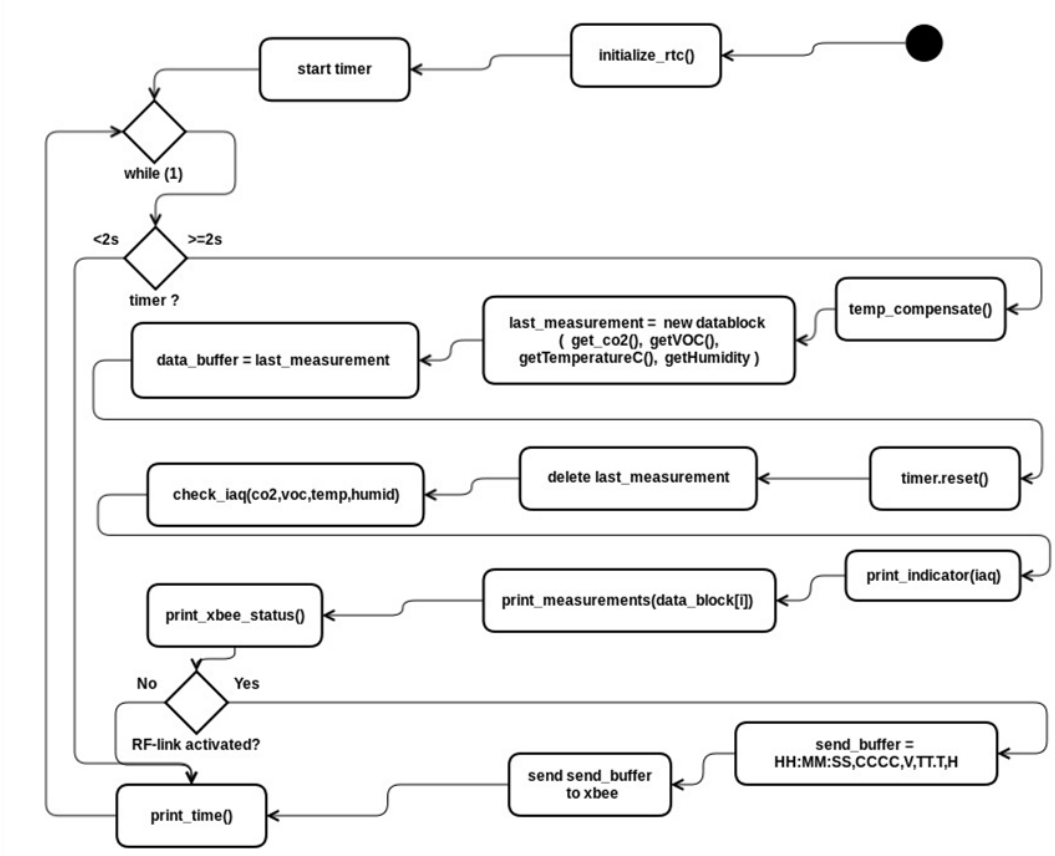


Figure 20. Activity Diagram of the current Implementation

The first step is to initialize the RTC, to provide correct timing for the measurements. Afterwards the timer, measuring the time since the last measurement, is started and the pull up resistor for `xbee_switch`, a digital input, is activated. Following, the continuous WHILE-loop starts. First the reference voltage for the VOC sensing circuit is adjusted by calling the `temp_compensate()` function. The next step is the acquisition of the measurement data by creating a new object of the `datablock` class. The parameters of the constructor are occupied by the functions of the sensor classes, returning the measurement values. The new datablock object is copied to the `data_buffer` which size is reduced to one. Afterwards the timer is reset and the new databock object is destroyed again. The function `check_iaq()` rates the IAQ based on the data, passed to the function as a parameter. Depending on the passed data, the function returns an Integer value from 0 to 2, representing either a good indoor air quality (0), an acceptable IAQ(1) or a bad IAQ(2). The returned value is passed to the `print_indicator()`function, which modifies the background colour for the screen depending on the passed value.

The next step is the transmission of the measurement data via the radio link. First the program checks if the radio link has been activated. The voltage level at `xbee_switch` is read and depended on the returned value, the function `print_xbee_status`() modifies the radio link's status information on the screen. If the switch is close and the voltage level is pulled to ground, a '0' is returned and the transmission procedure starts. Otherwise it is skipped. The transmission procedure consists of two steps. First the latest measurement data are printed into the string `send_buffer` in the format HH:MM:SS,CCCC,V,TT.T,R. Afterwards the string is transmitter to `the` Xbee module via the UART interface, using mbed's Serial API.

At the end of the WHILE-loop the `print_time()` function is used to update the time information on the screen.

7 Testing

7.1 Temperature compensation

To assure that the VOC circuit will not give any false alarm due to elevated temperature and to prove the functionality of the temperature compensation circuit, the reference voltage and the voltage over the load resistor are monitored over the expected range of temperature. The test setting is drafted in figure.

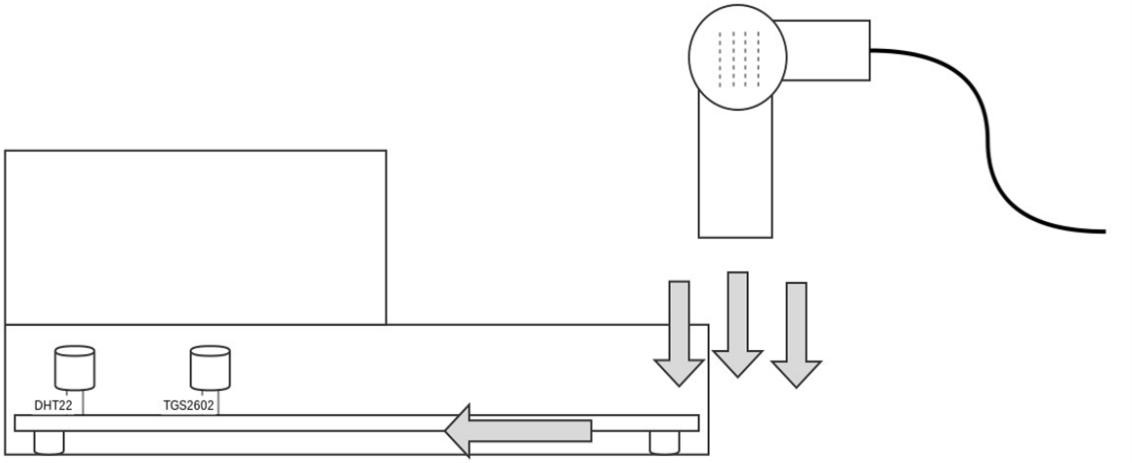


Figure 21. Test setting for temperature compensation test

The system is placed in a box and area above the temperature sensor and the VOC sensing circuit is covered as figure 21 shows. Through the uncovered opening of the box, warm air is inserted inside the box using a conventional hairdryer. The slope of the temperature raise is controlled by the amount of Airflow directed into the box. To create a slow but steady raise of temperature just a small part of the airflow is directed into the box. During the test, a temperature raise of 0.1°C every 2 seconds was achieved. The starting temperature was 23.0°C and at the end of the temperature cycle a final temperature of 40.0°C has been reached. The voltage curves in figure 22 have been taken with the oscilloscope.

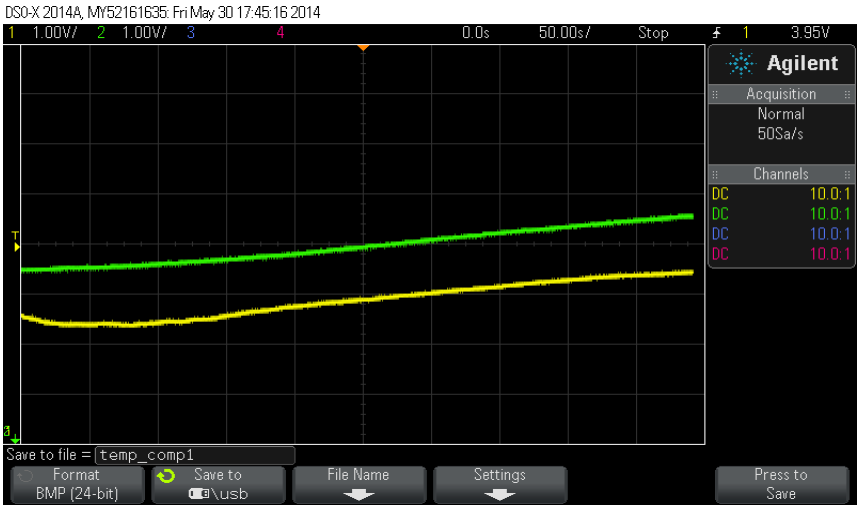


Figure 22. Oscillogram of V_{REF} and V_{RL}

The voltage over the load resistor V_{RL}, represented by the yellow curve in Figure X, is 2.6V at the beginning of the test. During the first minute, the voltage drops slightly to around 2.4V, even though it is expected to rise. Afterwards the voltages continuously raises as expected and reaches final value of 3.5V at 40°C.The green curve representing the reference voltage V_{REF}, created by the MCU, starts at 3.5V and continuously rises during the whole temperature cycle until it reaches ca. 4.6V at the end of the test. The voltage drop of the load resistor voltage in the first minute is due to incipient airflow. With a bigger airflow, a bigger voltage drop may be observed.

Both curves progress almost parallel and the offset between both voltages stays around 1.0V over the whole temperature cycle. Hence, the test can be considered as successful.

7.2 Operation Test

The system is set up in a class room to measure the CO2 concentration and temperature during an entrance exam for the university. The IAQ monitoring system is located at the teacher's desk in the front of the class room. At the beginning of the exam, 27 students and two teachers are equally distributed in the room. The measurement data are transcribed from the screen by the teacher .Figure 23 shows the trend of the CO2 concentration as well as the trend of temperature before, during and after the exam. The transcription of the measurement protocol can be found in the appendix 3

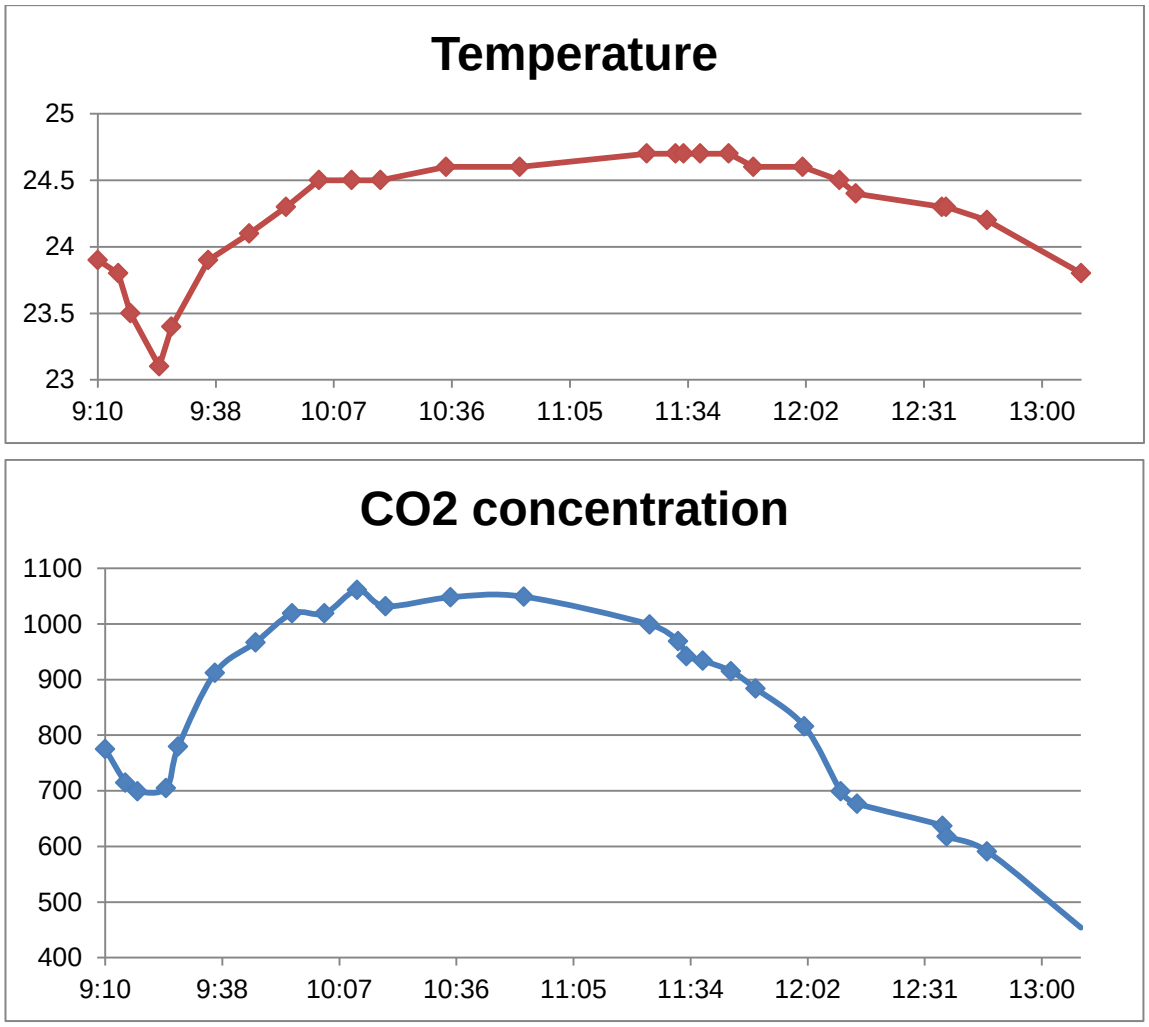


Figure 23. CO2 concentration and temperature during Entrance Exam

In the beginning from 09:10 until 9:25 the CO2 concentration decreases due to an open window. The exam started at 9:30 and before the door of the class room was open, as well enabling the air to ventilate and the CO2 concentration to decrease.

When the exam started and all doors and windows have been closed, the CO2 concentration, as well as the temperature increased. This development was expected, due to respiration of the room occupants and missing ventilation. The CO2 concentration rises until 10:53. The next measurement was taken at 11:24 and the CO2 concentration had dropped by 50ppm. From this point, the CO2 concentration continually decreased until the end of the measurement. At 11:20 the first student finished his exam and left the room and afterwards the amount of people in the room continuously decreased until the exam finished at 12:40. The decreasing concentration of CO2 results from the decreasing amount of room occupants. A picture of the prototype used in this test, can be found in appendix 4

8 Conclusion

The aim of this project was to improve the circumstances in a learning environment by monitoring the Indoor Air Quality. To accomplish this task, the system has to monitor characteristic values of the indoor air, rate the Indoor Air based on this measurements and communicate the rating to the room occupants. Furthermore a long-term measurement has to be established by storing the measurement data internally as well as transmitting them to an external receiver.

During this project, several problems were encountered and most of them were solved, but not all of them. Therefore the system is still a prototype and more work has to be done, to turn it into a finished product. But even though not all intended features are implemented yet, the system is functional at this point. As the tests have shown, the system is able to gather the needed data and give a reasoned Indoor Air Quality rating. Furthermore, the requested RF-Link is functional as well, providing access to the raw measurement data. These data are enough to enable the room occupants to react reasonably on changing air conditions in the room. Therefore, the main goal, the improvement of the IAQ, can be considered as accomplished.

However, there are three major steps in this project coming up next. First some more effort has to be put into the data logging feature. The problem concerning the file operations has to be solved and a function limiting the maximum amount of files stored in the internal memory has to be implemented. Furthermore a function to initialize the calibration procedure of the SenseAir S8 sensor is missing and needs to be implemented. Afterwards the system has to be tested under controlled conditions, concerning temperature, humidity and gas concentration, to confirm the validity of the measurement data. The last step has to be the design of a circuit board and the final assembly. Additionally, a power management system might be implemented, which powers down unused parts of the system.

References

1 Hirschberger, R. Automotive Emission Analysis with Spectroscopic Techniques In: Meyers R.A. Editor-in-Chief Encyclopedia of Analytical Chemistry.John Wiley & Sons

2 Stuart B, George WO, McIntyre PS. Modern Infrared Spectroscopy. Chichster, U: John Wiley &Sons; 1996.

3 Van Ewyk R, Willatt BM. Infrared Gas Detection. In: Mosely PT, Norris JO Williams DE, editor. Techniques and mechanisms in Gas Sensing. Bristol, UK: IOP-Publishing; 1991. p.234 - 259

4 Frodl R, Tille T, A High-Precision NDIR CO2 Gas Sensor for Automotive Applications [online]. IEEE; December 2006
URL:http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=4014157&queryText%3DHigh-precision+NDIR+CO%3Csub%3E2%3C%2Fsub%3E+Gas+Sensor
Accesssed 2 Juni 2014

5 Mosely PT, Stoneman AM, Williams DE. Oxide Semiconductors: Patterns of Gas Response Behaviour according to Material Type. In: Mosely PT, Norris JO Williams DE, editor. Techniques and mechanisms in Gas Sensing. Bristol, UK: IOP Publishing; 1991.

6 Korotcenkov G. Handbook of Gas SensorMaterials Volume 1. Springer; 2013

7 Comini E, Faglia G, Sberveglieri G. Electrical-Based Gas Sensing. In: Comini E, Faglia G, Sberveglieri G. Solid State Gas Sensing. New York, NY: Springer; 2009

8 Unitronic AG. General Information for TGS Sensors Düsseldorf, De: Unitronic AG: March 2005

9 'EPA. Indoor Air Facts No. 4 Sick Building Syndrome [online]. EPA, Washington, United States: EPA; February 1991.
URL: http://www.epa.gov/iaq/pdfs/sick_building_factsheet.pdf
Accesssed 2 Juni 2014

10 co2meter.com. What is carbon dioxide?[online]. Flint,MI ,United States: co2meter.com;
URL: http://cdn.shopify.com/s/files/1/0019/5952/files/what-is-carbon-dioxide.pdf?1280877879
Accesssed 2 Juni 2014

11 ErdmannCA, SteinerKC,Apte MG.INDOOR CARBON DIOXIDE CONCENTRATIONS AND SICK BUILDING SYNDROME SYMPTOMS IN THE BASE STUDY REVISITED [online]. EPA, Washington, United States: EPA; 2002
URL:http://www.epa.gov/iaq/base/pdfs/base_3c2o2.pdf
Accesssed 2 Juni 2014

12 Satish U, Mendell MJ, Shekhar K, Hotchi T, Sullivan D, Streufert S, Fisk WJ. Is CO2 an Indoor Pollutant?[serial online].December 2012; p.1671-1677

URL: http://ehp.niehs.nih.gov/wp-content/uploads/120/12/ehp.1104789.pdf
Accesssed 2 Juni 2014

13 Ministry of Social Affairs and Health. Asumisterveysohje[online].
Helsinki, Finland: Ministry of Social Affairs and Health; 2003.
URL: www.finlex.fi/pdf/normit/14951-asumisterveysohje_pdf.pdf p.63
Accesssed 2 Juni 2014

14 Unitronic AG. Product Information TGS2602 Düsseldorf, De: Unitronic AG: August 2003

15 Liu T. Datasheet RHT03[online].Boulder, CO, United States: SparkFun Electronics;
URL:http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Weather/RHT03.pdf Accesssed 2 Juni 2014

16 Senseair AB. ModBus on Senseair S8[online]. Flint, MI ,United States: co2meter.com;February 2010
URL: http://co2meters.com/Documentation/Datasheets/DS-S8-Modbus-rev_P01_1_00.pdf
Accesssed 2 Juni 2014

17 Senseair AB. ABC Key to Maintenance-free Gas Sensors.Delsbo, Sweden: Senseair AB;
URL: http://www.senseair.se/wp-content/uploads/2012/03/ABC20120326.pdf
Accesssed 2 Juni 2014

18 mbed handbook. Memory Model[online]. Cambride, UK: ARM;
URL: https://mbed.org/handbook/Memory-Model
Accesssed 2 Juni 2014

Mbed LPC 1768 Power Supply Circuit

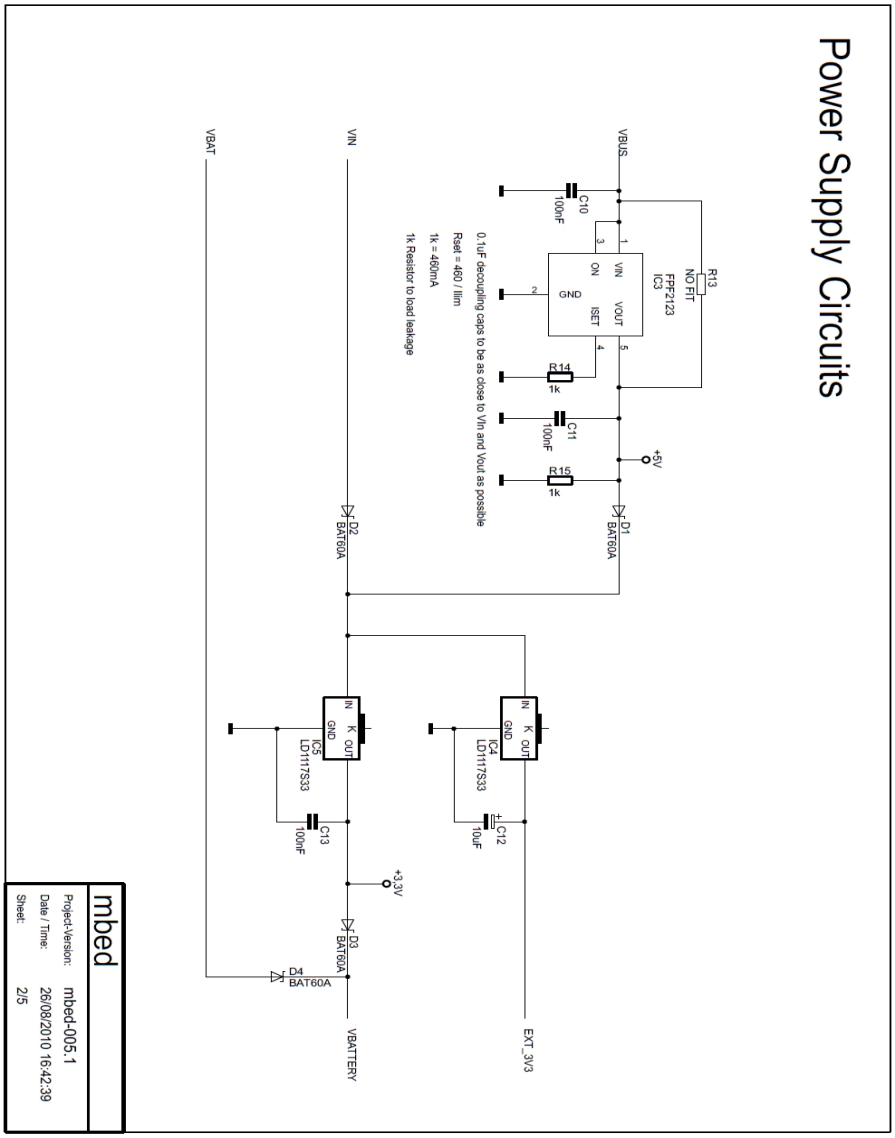
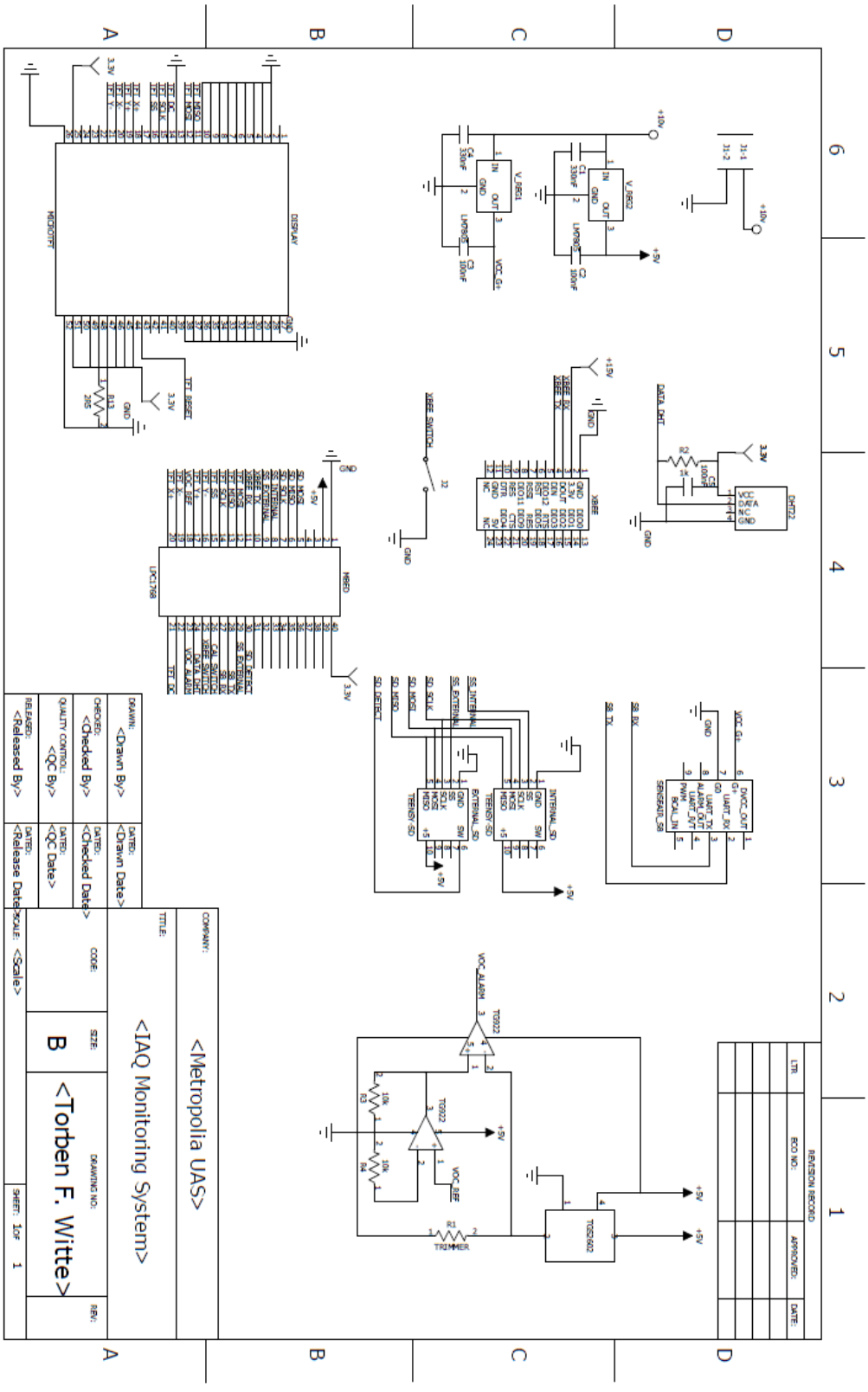


Figure 24. Voltage regulating circuit of the mbed LPC 1768 [datasheet]

Schematic


A
B
C
D
1
2
3
4
5
6
REVISION RECORD
LTR
ECO NO:
APPROVED:
DATE:
COMPANY:
<Metropolia UAS>
TITLE:
<IAQ Monitoring System>
DRAWN:
<Drawn By>
DATE:
<Drawn Date>
CHECKED:
<Checked By>
DATE:
<Checked Date>
QUALITY CONTROL:
<QC By>
DATE:
<QC Date>
RELEASED:
<Released By>
DATE:
<Release Date>
CODE:
SIZE:
B
DRAWING NO:
<Torben F. Witte>
REV:
SCALE: <Scale>
SHEET: 1 OF 1

Transcript of Measurement Protocol

Time	CO2 [ppm]	Temperature[°C]	Notes
9:10:00	775	23,9	Windows open
9:15:00	715	23,8	
9:18:00	699	23,5	Windows closed
9:25:00	705	23,1	
9:28:00	780	23,4	27 students & 2 teachers
9:37:00	912	23,9	
9:47:00	967	24,1	
9:56:00	1019	24,3	
10:04:00	1019	24,5	
10:12:00	1061	24,5	
10:19:00	1032	24,5	
10:35:00	1048	24,6	
10:53:00	1049	24,6	
11:20:00			first student left
11:24:00	999	24,7	
11:31:00	969	24,7	6 students left
11:33:00	942	24,7	
11:37:00	934	24,7	
11:44:00	915	24,7	15 students remaining
11:50:00	884	24,6	
12:02:00	816	24,6	
12:11:00	699	24,5	5 students remaining
12:15:00	677	24,4	
12:36:00	637	24,3	
12:37:00	618	24,3	
12:40:00			Exam ends
12:47:00	591	24,2	
13:10:00	454	23,8	

Picture of Circuit used in Operation Test