NTNU – Trondheim
Norwegian University of
Science and Technology

Handheld Spirometer

Ove-Joakim Istad

Master of Science in Cybernetics and Robotics
Submission date: February 2014
Supervisor: Amund Skavhaug, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU

Master Project

Handheld spirometer

Author:
Ove-Joakim Istad

Supervisor:
Assoc. Prof. Amund Skavhaug

Master of Science in Engineering Cybernetics

Department of Engineering Cybernetics

February 2014

Declaration of Authorship

I, Ove-Joakim Istad, declare that this thesis titled, 'Handheld spirometer' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

NTNU

Abstract

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Engineering Cybernetics

Engineering Cybernetics

Handheld spirometer

by Ove-Joakim Istad

It is estimated that between 200,000 - 250,000 Norwegians are subject to the incurable disease COPD (Chronic Obstructive Pulmonary Disease)[1], which counts for around 8.5% of the adult population. Only about one third of those are diagnosed, and need to go regularly to the doctor in order to monitor the progression.
The goal of this master project is to build a prototype circuit board that can be used to monitor those patients at home instead. Different methods for constructing such a device have been examined, and my solution consists of a spirometer that can measure the reduction in vital capacity and send the measurements to a computer. Using the device will make the doctor spot the progression of the decease, instead of only a one-time measurement. This project proves a possibility of constructing a user friendly, cheap and simple device, that is hopefully going to make the life of this patient group a whole lot easier.

Contents

List of Figures

List of Tables

Nomenclature

FVC	**F**orced **V**ital **C**apacity
FEV1	**F**orced **E**xpiratory **V**olume in **1** second
COPD	**C**hronic **O**bstructive **P**ulmonary **D**isease
GOLD	**G**lobal Initiative for Chronic **O**bstructive **L**ung **D**isease
UART	**U**niversal **A**synchronous **R**eceiver/**T**ransmitter
PIC	**P**eripheral **I**nterface **C**ontroller
ADC	**A**nalog to **D**igital **C**onverter
LCD	**L**iquid **C**rystal **D**isplay
CAD	**C**omputer **A**ided **D**esign
μC	(micro)**C**ontroller
USB	**U**niversal **S**erial **B**us
GSM	**G**lobal **S**ystem for **M**obile Communications
RTC	**R**eal **T**ime **C**lock
SD	**S**ecure **D**igital
SRAM	**S**tatic **R**andom **A**ccess **M**emory
PWM	**P**ulse-**W**idth **M**odulation
SPI	**S**erial **P**eripheral **I**nterface
I^2C	**I**nter-**I**ntegrated **C**ircuit

Preface

This thesis is the culmination of my master degree in Engineering Cybernetics. The reader is assumed to have some knowledge in embedded system theory, together with some insight in physics. Prior medical knowledge is not necessary as I will explain the medical theory used in this report.

The idea of building this hand-held spirometer was brought to me by my supervisor **Amund Skavhaug** as a part of Norsk Automatisering AS's commitment in health systems for elderly. Their aim is to produce a vast set of intelligent solutions that can expand the time elderly can live at home, rather than at a nursing home. Norsk Automatisering AS with **Amund Skavhaug** is therefore to be credited for giving me this opportunity in participating in this project.**Amund** is to be credited for giving me outstanding guidance during both the specialisation project and the master thesis. He has seemingly limitless knowledge of the things I ask him about, and was a true resource when writing this thesis.

The following deserve some extra credit:
John Olav Horrigmo and **Stefano Bertelli** for helping me at the lab with the milling machine, soldering and general technical questions.
Vegard Gulaker and **Stian Søvik** for priceless help when the code seemed stuck.
Olav Myrvang for being my lifelong friend and interlocutor, always up for some extreme science.
Andreas Kråkenes for being my study buddy, always inspiring me with the way you always does things perfect(-ish), and helping me with tips for the report.
My two parents, **Bente** and **Odd**, for giving me unconditional love, limitless help and all your time.

Odin, my first son. You are keeping me up all night but it's all worth it when you look at me, smiles the most adorable smile I have ever seen and melts my heart.

Last but not least, a huge credit to my fiancée, **Ingrid**. She has helped me when I need advice, whipped me when I did not work enough, and rewarded me when I did. Thank you for always being there for me.

Ove-Joakim Istad

February, 2014

Chapter 1

Introduction

Between 200 000 - 250 000 Norwegian citizens are subject to the progressive disease COPD[1], which counts for around 8.5% of the adult population. 20% of those have reached a critical stage of the disease. Only about one third of those are diagnosed, and need to get regularly to the doctor in order to monitor the progression.

In addition, a person with a well performing lung, might have a lung capacity above the average, say 130% instead of 100%. If his or her lung capacity decreases because of i.e. COPD, a score of 85% will not be flagged as dangerous because it is slightly above the limit of 80% which defines COPD. Nevertheless, it will be a loss of 1 - 85 / 130 = 35% the lung capacity he or she had before. This reduction can easily be felt by the patient, but not recognized by a normal spirometer.

To be able to diagnose such a patient more correctly, and also give the patient already diagnosed with COPD an option of being able to measure his or her change of lung function at home, this thesis will analyze the possibility of constructing a hand held personal device that can be brought home to the patient. This enables the patient to measure the progression, instead of just a one time measurement at the doctor.

1.1 The solution

A block diagram of the solution is displayed in figure 1.1.
The system is centered around a microcontroller. It receives pressure data from a differential pressure sensor that measure the pressure before and after a contraction in a pipe. When the user blows through the pipe, the speed of the air flow is, to a certain degree, proportional to the differential pressure measured. This data could either be saved to the SD-card, or it could be transferred to a computer using the RS-232 interface. An LCD is used to send information to the user, while a touch display, placed over the LCD, is used to send information from the user.

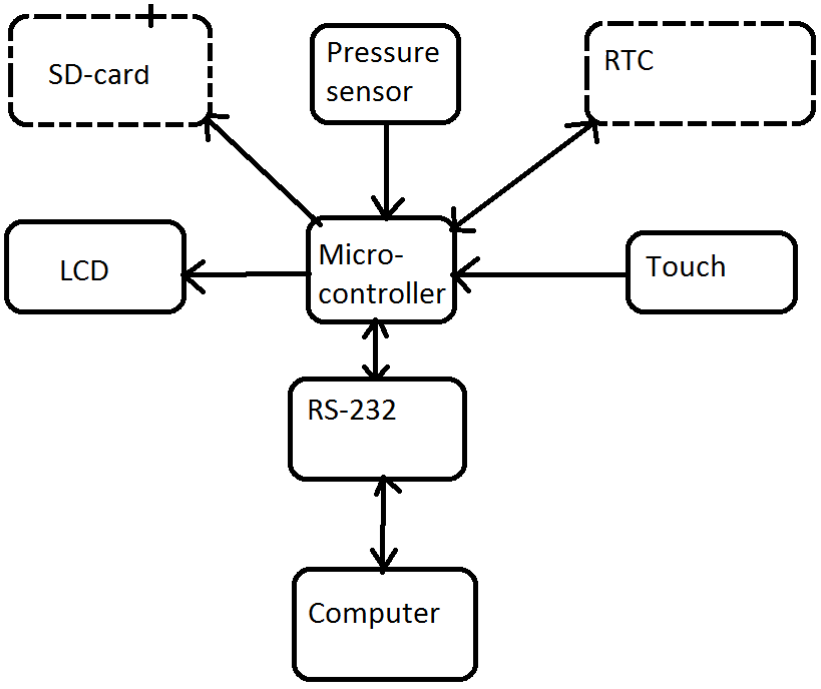


FIGURE 1.1: Block diagram of the solution

1.2 Scope

In order to avoid making the task to extensive, it has to be delineated to the following:

- Provide background material needed to perform the task.
- Construct a draft of how the instrument should work.
- Examine which main components are needed, and find these parts at a retailer.
- Create a test module for each module used, perform specific tests for each modules and enhance design based on the experience gained.
- Merge all sub modules into one circuit board, controlled by one micro-controller. Perform selected tests on this board and enhance the design if necessary.
- Develop a computer program, used to communicate with the circuit board. The communication performed involves setting up the device, receiving measure data and synchronizing device clock.
- Design a casing used to mount the circuit board onto, hide the components for the user and protect the components.
- Provide c-code for the device.

1.3 Existing solutions

There are multiple solutions on the marked, however they are all either only for use at the doctor or a hospital, or they are to expensive to be used at home. The cheapest spirometers the author could find, came at around 2000 NOK, and was more complex than necessary.

Chapter 2

Background

The background chapter is divided in two parts; medical background and technical background. This is done because both parts are quite different from each other, but both are relevant for understanding *why* the solution was designed the way it was. The medical background explains the theory of why the solution used the different techniques, while the technical background explains to a broader sense *how* the different techniques was implied.

2.1 Medical background

This section covers the basic principles of COPD, how it works, how it is diagnosed, and how it is measured.

2.1.1 COPD

COPD (Chronic Obstructive Pulmonary Disease) is, as the name indicates, a chronic disease in the airway system[2].

In a healthy lung, an underpressure in the lung will cause a flow of air into the bronchial tubes of the airways. Within these bronchial tubes, the air is forced

into smaller and thinner bronchioles, ending in a vast amount of tiny round sacs of air, called alveoli. The exchange of oxygen from the lung into tiny blood vessels, capillaries, is done through the walls of these alveoli. The blood inside the capillaries circulate around these walls, exchanging oxygen from the alveoli and into the blood, and vice versa for the carbon dioxide. This is illustrated in figure 2.1 and part A of figure 2.2.

In a person with COPD, the amount of air flowing in and out of the airways is severely reduced because of one of the following[2]:

- The airways and air sacs lose their elastic quality.
- The walls between many of the air sacs are destroyed.
- The walls of the airways become thick and inflamed.
- The airways make more mucus than usual, which can clog them.
- There is an increased occurrence of inflammatory cells and exudate in the airway lumen[3].

This is illustrated in part B of figure 2.4

The three latter points can be reduced by removing the cause of the symptom, i.e. by quitting smoking or avoiding hazardous environments. The two first, on the other hand, are symptoms of lung emphysema, and can not be cured.

2.1.1.1 Causes of COPD

The leading cause of COPD is smoking, however, a study from 2009 reveals that as many as 25-45% of people with COPD have never smoked[3]. These people may have been subject to COPD by passive smoking, working in an environment with industrial dust or simply just inherited the disease.

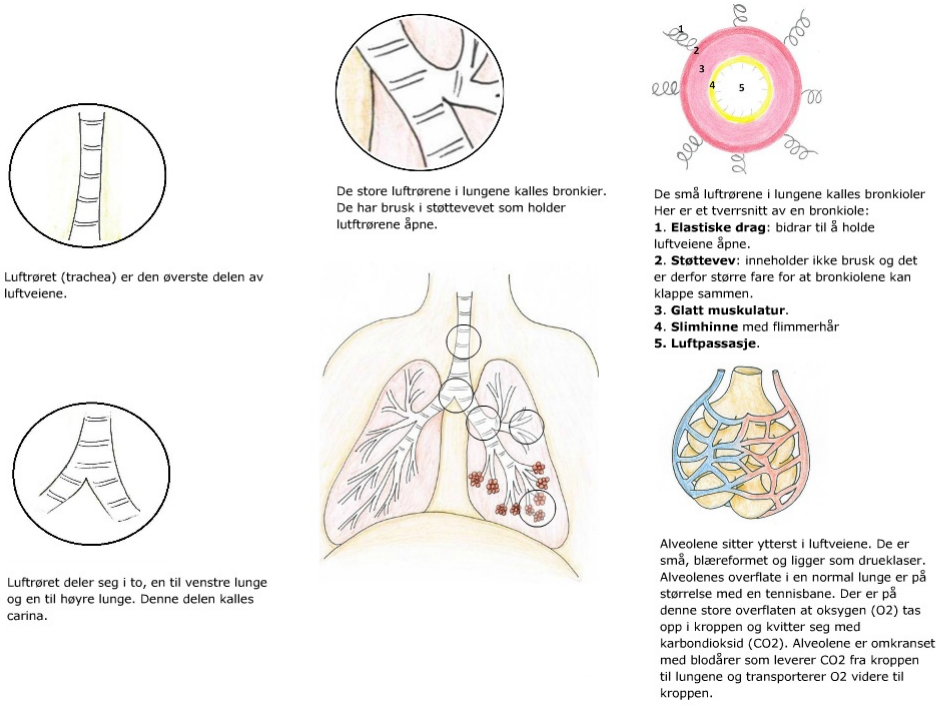


FIGURE 2.1: Airway system (Picture taken and modified from [1])

2.1.1.2 Diagnosis

The most common way to diagnose COPD is using spirometry, which is performed by blowing as hard and as long as possible into a spirometer (see section 2.1.2). This way, one can measure the amount and speed of the airflow from the lungs.

Figure 2.3 show the volume-time graph of a person with COPD.

2.1.1.3 Treatment

As of today, there are no cures to completely cure the disease, however some cures can reduce, or even halt, the progress. Giving up smoking is obviously the first, and best way to prevent getting worse. In addition, there are a lot of medicine on the market which can alleviate the symptoms and reduce the risk of complications due to pneumonia. One type of medicine is called Bronchodilators, which relax the muscles around the airways and make it easier to breathe.

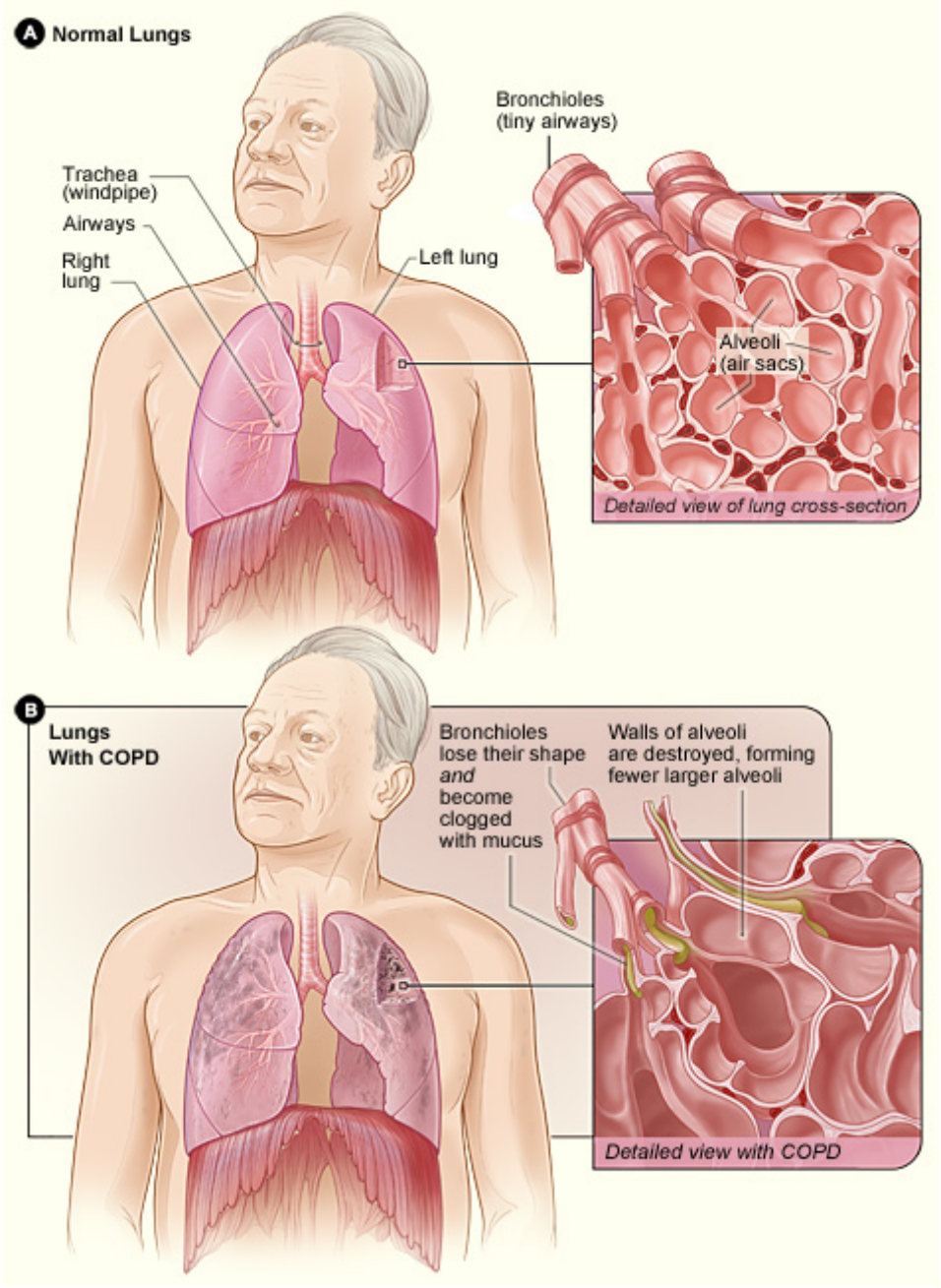


FIGURE 2.2: Lung without and with COPD (picture taken from [2])

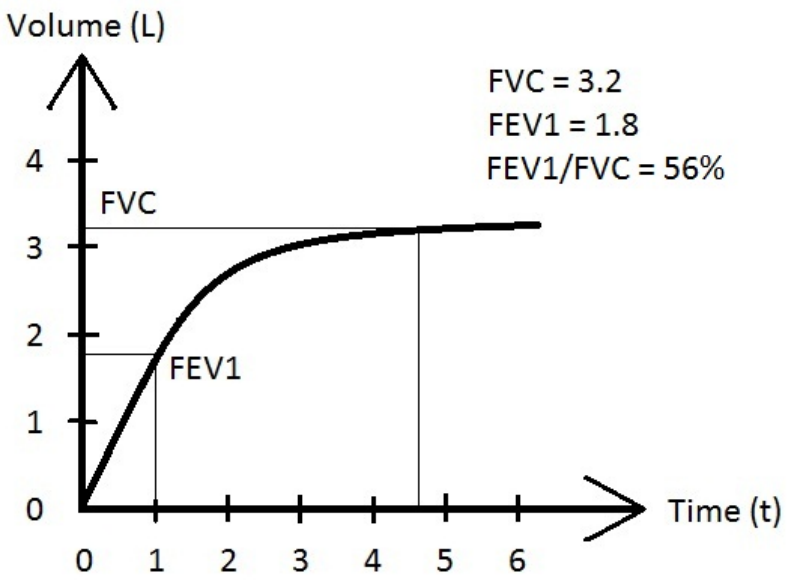


FIGURE 2.3: FEV1 and FVC for a person with COPD

If the symptoms worsen, inhaled Glucocorticosteroids can help reduce airway inflammation. A number of vaccines could also reduce or remove the risk of getting seriously ill during different disease outbreaks [4][5][6].

2.1.2 Spirometry

Spirometry is the basic measurement of lung function. The test is designed to measure the maximum volume of air that the patient can blow, at the fastest rate possible. When measuring the volume and flow rate, one can calculate two important factors, FVC (Forced Vital Capacity) and FEV1 (Forced Expiratory Volume after 1 second)[7].

FVC is the total vital capacity from a maximally forced expiratory effort and is measured in litres.

FEV1 is the volume of air exhaled within the first second of an maximally forced expiratory effort, and is also measured in litres.

These two factors can be used to diagnose a patient whether or not he has COPD and how severe the case is.

In order to be diagnosed with COPD, the FEV1/FVC needs to be less than 70 percent. This means that less than 70 percent of the total amount of exhausted air needs to be exhaled within the first second.

In addition, using the GOLD (Global Initiative for Chronic Obstructive Lung Disease) scale (Table 2.1.2), together with the expected value of a person at the same age, sex, height, weight and ethnicity, one can divide the disease into four stages[1]:

The GOLD scale	
Mild COPD:	FEV1 > 80% of expected value.
Moderate COPD:	FEV1 = 50% - 80% of expected value.
Serious COPD:	FEV1 = 30% - 50% of expected value.
Very serious COPD:	FEV1 < 30% of expected value.

TABLE 2.1: The GOLD scale[8]

As one can see from figure 2.4, there is a difference in the airflow speed and total air volume of a healthy person, and a person diagnosed with COPD.

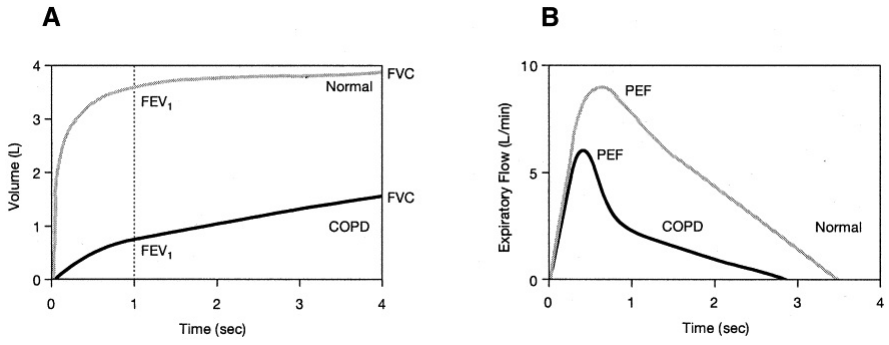


FIGURE 2.4: Volume/Flow graphs showing the difference between a healthy person and a person with COPD. (Picture taken from [9])

2.2 Technical background

This section explains the different interfaces used to communicate between the microcontroller and other peripherals on the circuit board. The last subsection also explains the Bernoulli principle, a century old method used for calculating air flow.

2.2.1 Two Wire Interface (TWI)

TWI is a multi-master bus with two lines. The first is called SCL and it transfers the clock pulses. It is simply an oscillating square pulse, with a frequency given by the TWI master. The other line is called SDA and transfers the address, data and control signals, such as ACK, NACK, START and STOP.

Both lines needs to be pulled high using two pull-up resistors, which values depends on the transfer speed (>100kHz or <100kHz), V_{CC} and the bus capacitance. The calculation for the pull-up resistor values are shown in formula (2.1) and (2.2):

Frequency < 100 kHz:

$$R_{min} = (V_{cc} - 0.4V)/3mA \tag{2.1a}$$

$$R_{max} = 1000ns/C_{bus} \tag{2.1b}$$

Frequency > 100 kHz:

$$R_{min} = (V_{cc} - 0.4V)/3mA \tag{2.2a}$$

$$R_{max} = 300ns/C_{bus} \tag{2.2b}$$

where R_{min} is the minimum resistor value, R_{max} is the maximum resistor value, V_{cc} is the voltage the system is running on and C_{bus} is the total capacity of each bus lines. The capacitance each IC is adding to the bus is found in the datasheet.

To find the total capacitance for the bus, simply add together all the different capacitance values for every IC on each line.

Due to the pull up resistors, both lines are pulled high when no device are using them. In order for the master device to to initialize data transfer, it has to drive the SDA low when SCL remains high. It then starts the TWI clock at a frequency defined in the software. The slave can then read from the data line every time the clock is pulled high. When the clock is pulled low, the data line is in an undefined state and can not be read. After the initial start condition, a unique 7-bit slave address is fed onto the bus, followed by a data direction bit. The latter is used for telling the slave whether it should be read or if it should put data on the SDA line. When the data direction bit is received by the slave, it acknowledges to the master by holding the data line low for one clock cycle. Thereafter, the data is put onto the data line by either the master or slave. The device receiving the data acknowledges every byte received. This is repeated until the master issues a stop condition, by a low to high transition on the data line while the clock is high.

More information on how the TWI bus works is given in appendix on the CD called ”Using the TWI module as I2C master”.

2.2.2 Serial Peripheral Interface (SPI)

The Serial Peripheral Interface, or SPI, uses four wires to make up a bus. These four wires include the clock(SCLK), slave select (SS), master-out-slave-in (MOSI) and master-in-slave-out (MISO). An arbitrary number of devices can be connected to the SPI bus, only limited by the amount of input output lines that each device require to be activated. This is illustrated in figure 2.5.

SPI operates in full duplex mode, meaning that it can transfer data both to and from the device, at the same time.

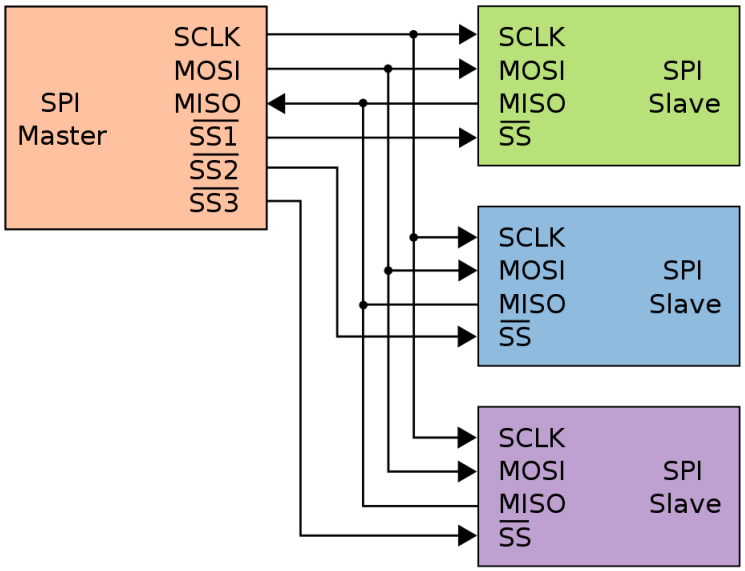


FIGURE 2.5: SPI bus

To start transferring, the master configures the clock to use a frequency less than or equal to the maximum frequency the slaves support. Thereafter it pulls the chip select line low for the device receiving the transmission. When this has been done, the MOSI and MISO lines will transfer its data simultaneously each clock cycle. See figure 2.6.

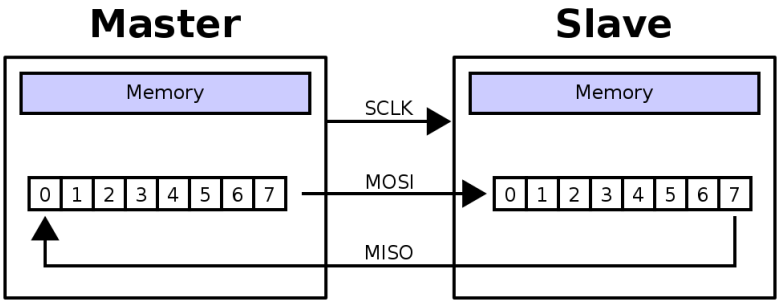


FIGURE 2.6: SPI transfer

To end the transmission, the master stops toggling its clock and pull the slave select wire high.

2.2.3 FatFs

FatFs is a generic FAT file system module for small embedded systems, which makes it easy to access low level disk I/O, like an SD card [10]. It is written by Frank Zhao in 2013 and is free software that can be used for personal, non-profit or commercial products. The software provides several functions enabling the user to open files, write files, create directories, changing timestamps, erasing files and much more. A full list of application interface is found in [10] or on the CD that follows this report.

2.2.4 Bernoulli's principle

Bernoulli's principle is one of several methods of measuring air flow. Figure 2.7 shows a venturi tube. When air, with a density of ρ, flows through a venturi tube with a diameter d_1 that is decreased to a diameter of d_2, the pressure will increase from p_1 to p_2 [11].

If the two nozzles in a differential pressure sensor can be connected to the venturi tube before and after the contraction, the differential pressure will be directly proportional with the airflow through the pipe.

$$p_1 - p_2 = \rho/2 * (v_2^2 - v_1^2) \tag{2.3a}$$

An exact equation for the fluid flow is found in 2.3, but due to turbulent flow, various viscosity of the air flowing through the pipe, and inexact sensor calibration, the device needs to be calibrated experimentally before use.

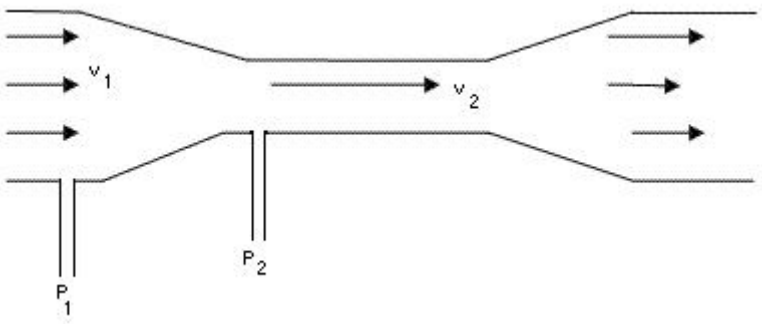


FIGURE 2.7: Venturi flow

Chapter 3

Method

This chapter describes what method was used in order to produce the result in this thesis. The progress is not exact or linear, the writer has jumped back to the different stages, when applicable.

Some resemblance can be found to the V-model, although it is not exactly equal. The differences has been made in order to adapt the workflow into the scope of this thesis.

3.1 Receiving problem and discussing possible solutions

The start of this master project consisted of receiving a problem statement, and discussing different solutions for this. A few different methods was chosen to be further examined.

Some of the methods was later discarded, either because they were found to be abundant, because they turned out to be too difficult to implement, or because better solutions were found when further investigating the task.

3.2 Background study

A background study was performed in order to map the already existing solutions, and what was the preferred method of determining airflow.

The different patents was found using google scholar, a useful tool to do patent searches and to find better solutions to similar problems. The medical background study was found using among other pubmed, Norwegian health pages and general searching around the web using Google. Some of the technical solutions was, to a certain amount, already known to the author. This includes the different interfaces, the Bernoulli principle and general knowledge of the AVR microcontrollers. Where lack of knowledge occurred, forums on the web together with some schoolbooks was commonly used. The supervisor was also generous with input on how the different problems could be solved. This included questioning the different approaches, giving tips of already existing methods and help of how the report could be built up.

Where Internet or library-sources was used, this has been explicitly cited in the ”References”-chapter of this report.

3.3 Creating and testing modules

When it was clear how the device should behave, and what specifications it should have, the different modules was examined. They was build using a breadboard in combination with an STK600 card and the ATMEGA2560 microcontroller. To debug the code, a UART interface was created that communicated with the computer using ”Termite”, a RS-232 terminal program.

After the module was set up correctly and source code was written, a test bench was made to check that both the software and hardware was working correctly. When the module passed the testing, the schematic and source code was backed up and work on the next module could begin.

3.4 Combining the modules into one PCB and writing program code

The resulting PCB can be created when all the modules has been designed and tested. To make the schematic, the modules needs to merge and the previous software is composed into C and header files that are included in the main c-code.

3.5 Testing PCB

When the PCB had been created and all the components had been soldered onto it, the card had to be tested. This was done using both a multimeter, for current and voltage inspections, in addition to using the RS-232 interface to send debug messages to the computer.

Chapter 4

Modules

4.1 microcontroller

The microcontroller (μC) chosen for this task was the 8-bit AVR ATMEGA2560 16AU. It was chosen for its many applications and versatility.

The different features are as followed:

- Up to 8/16 MHz clock frequency (2560V/2560 version).
- Can run on 2.7 - 5.5V (only 2560V version).
- Low power consumption, 500 μA @1MHz in active mode, and 0.1μA in power-down mode.
- 86 programmable I/O lines
- 10-bit ADC, SPI and I^2C interface, and multiple PWM channels
- Several interrupt pins.
- 256KB Self-programmable flash, 4KB EEPROM and 8KB internal SRAM.

For more information about the microcontroller, the data sheet is appended on the CD.

Programming the μC is easiest done using the SDK600 (Software Development Kit). This allows the programmer to mount the different components onto a socket. The components can then be easily added or removed, without the need for soldering. Another way to program the μC is to connect the STK600 to a JTAG-header on the circuit board containing the chip. This is called in-circuit programming, and is a convenient way to program chips already mounted on a circuit board, given that the board already has a JTAG-header connected to the chip.

Both of the methods above was used in this project; the socket connection for the test phase, and the in-circuit programming mode later on, when the circuit board had been made.

The JTAG connection set-up can be viewed in the result chapter.

4.2 Pressure Sensor

Several different pressure sensors from different suppliers was examined for this task. The following features was favourable when choosing pressure sensor:

- Pressure range, from -10 to 10 mbar.
- Large resolution, full scale span of at least 1024 counts.
- Large repeatability and high accuracy (< 2.5%).
- Working on a 3V system.
- Using an interface that exist on the microcontroller, either analogue, TWI(I^2C), SPI or parallel data transfer.
- Relatively low price, < 700 NOK for large quanta.
- Sampling frequency of at least 100 Hz.
- Low power consumption and start-up time.

Three pressure sensors was selected for further work, the ASDX-series from Honeywell and the LDE and HDI series from Sensortechnics. The sensor from Honeywell was discarded because they could not deliver one with the correct pressure range within reasonable time. After testing the LDE series it too was discarded because their maximum pressure range only went to ±5 mbar, slightly lower than what was considered as the lower limit of pressure span. The final sensor, HDI, met all specifications. It could run on either 3V or 5V, had both digital (I^2C) and analogue interface, a pressure range from 10 mbar up to 1000 mbar, accuracy of 0.5% (prime grade) or 1.5% (high grade). The ATMEGA2560 provide both I^2C (albeit they have adopted their own standard, called Two Wire Interface) and ADC (Analogue to Digital Converter) interface. Since the pressure sensor could output a full scale span of 21,845 counts when using the digital interface, but the ATMEGA2560 had only 10 bit ADC resolution, resulting in 1024 counts, the digital interface was chosen.

4.2.1 Schematic of the pressure sensor circuit

The schematic of the pressure sensor circuit is shown in figure 4.1. It is a very simple connection, where the ground is connected directly to pin 3. Vcc is connected to pin 1 in parallel with a 0.1uF capacitor connected to ground to remove ripple that may occur when heavy loads are drawn from the µC. The two wires of the I^2C interface, SDA and SCL are connected to pin 4 and 6, respectively. Not shown in this circuit are the two pull-up resistors that has to be connected from the signal lines to Vcc in order to pull the signals high, according to the way the I^2C interface operate. See chapter 2.2.1 for information on how to choose the correct values for these two resistors.

Note that Vcc is not connected directly to the battery, but via the microcontroller. This enables the system to power on and off the pressure sensor when required, conserving battery power.

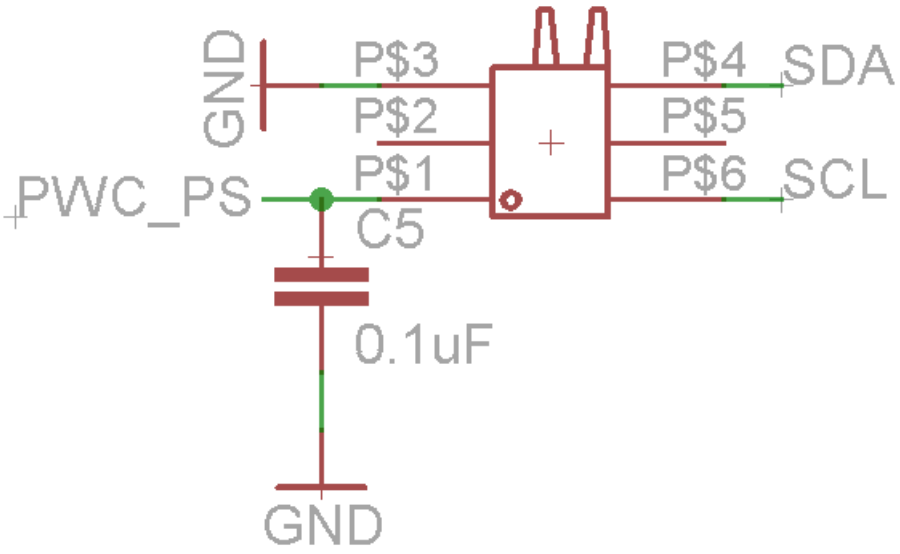


FIGURE 4.1: Pressure sensor circuit

4.2.2 Testing the Pressure Sensor

Two main features are crucial in using a differential pressure sensor to measure the airflow.

First and foremost the sensor needs to be able to transfer some sort of pressure to the microcontroller. This is performed in the first test.

Secondly, airflows of different magnitude must be measured correctly. The flow speed should be proportional to the pressure difference and equal flow speed should yield equal pressure measurements. This can be tested and verified in the second test.

4.3 Real Time Clock/Calendar

The entire system depends on being able to give a correct time stamp to each measurement, because it is the patients change of lung capacity in time that

Reading values from pressure sensor	
What is tested:	If the pressure sensor can transfer pressure data to the computer.
How is it tested:	Connect the pressure sensor to the microcontroller using the TWI bus, power it up and write data to it to initialize a reading. The microcontroller should be connected to a computer using the UART interface, and the termite console program. The test begins with blowing into one of the two nozzles of the pressure sensor.
Desired result:	The Termite console program should output some pressure data depending on how much pressure is applied to blowing into the nozzles.
Actual result:	The console displayed the pressure data as desired.
Comments:	The pressure sensor will reach a saturation point if blowing directly into the nozzles. This is however not a problem since only a fraction of that pressure is applied when using the Bernoulli principle of measuring airflow.

TABLE 4.1: Reading values from pressure sensor

is interesting. Using the microcontroller built in RTC is possible, but during a system restart the clock needs to be set up again. This could be avoided using a dedicated RTC, that are connected to its own power supply, i.e. a coin cell battery. In this manner, the time only needs to be set during production of the spirometer, and possibly synchronized every time the spirometer is connected to a computer.

The RTC chosen for this project is called PCF8563T by NXP Semiconductors. It is based on a 32.768 kHz quartz crystal, is interfaced using the I^2C bus, can operate from 1.0V - 5.5V, consumes only 0.25 µA at stand by, have alarm and timer functions, and and a programmable clock output of 32 768 Hz, 1024 Hz, 32 Hz and 1 Hz.

As discussed in subsection 2.2.1, the TWI requires one pull up resistor each signal wire. The upper limit of the resistor value depends on the capacitance of the bus and the bus clock speed. The lower limit does, however, only depend

on V_{cc} and can be calculated using the equation

$$R_{min} = (V_{cc} - 0.4V)/3mA \tag{4.1a}$$

This yield 967Ω for a 3.3V system.

4.3.1 Schematic of the RTC circuit

Connecting the RTC to the microcontroller is simple. See figure 4.2.

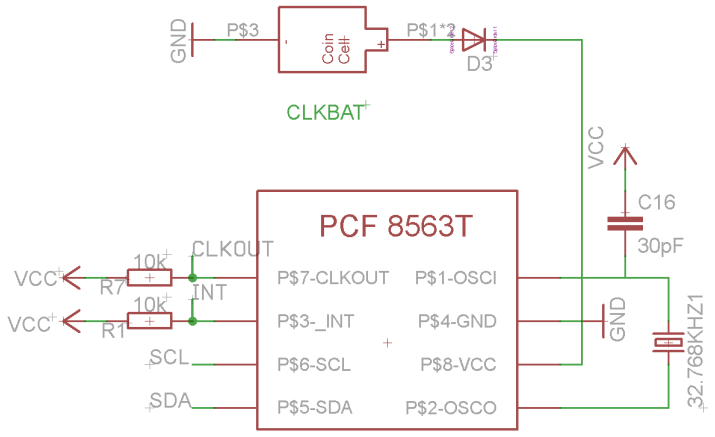


FIGURE 4.2: RTC circuit

4.3.2 Testing the Real Time Clock/Calendar

Testing accuracy of RTC	
What is tested:	Accuracy of RTC
How is it tested:	Connect the RTC to the microcontroller/STK600 card using the RTC circuit displayed at figure 4.2. Set the time. Read the time afterwards to check that it is correct. Leave it on battery power for a number of days. Read the time again. Calculate the deviation in time.
Desired result:	Time has not deviated more than 0.1%.
Actual result:	Success. The RTC had an error of 20 seconds in five days (432,000 seconds), resulting in an error of 20/432,000 = 0.0046%, far beyond the acceptable limit.
Comments:	If desirable, the clock can be trimmed using a trimming capacitor connected between pin one at the RTC and ground.

TABLE 4.2: Testing accuracy of RTC

Testing RTC alarms	
What is tested:	That RTC can signal an alert due to a preset alarm.
How is it tested:	Connect the RTC to the microcontroller/STK600 card using the RTC circuit displayed at figure 4.2. The alarm/timer port should be connected to the cathode part of a LED while the anode part of the LED should be connected to V_{cc}. Set the time. Set the alarm one minute forth in time. Afterwards, set a timer to one minute.
Desired result:	When the alarm and timer goes off, it should light a LED.
Actual result:	Both the alarm and timer lit the LED as predicted.
Comments:	No comment.

TABLE 4.3: Testing RTC alarms

4.4 Memory Card

A memory card is preferred in order to store the data collected by the device, without having to connect it to the computer every time. There are multiple types of memory cards on the marked, including the SD, SF or MMC. In addition, there are multiple versions of each card. The SD card, for example, comes in a small (Micro SD), medium (Mini SD) and large (SD) size.

The following criteria was weighted when choosing what card to use:

Measuring power usage	
What is tested:	Power usage of the RTC during normal operation.
How is it tested:	Connect the RTC to the microcontroller/STK600 card using the RTC circuit displayed at figure 4.2. Connect a multimeter/ampere-meter between the clock battery and the V_{cc} input pin of the RTC. Measure the current during normal operation.
Desired result:	A current usage low enough to make the clock battery last for five years.
Actual result:	The current could not be measured because the multi meter could only measure down to 1 µA. Nevertheless, the RTCs data sheet states that the IC only uses 0.25 µA in backup current. Since the battery has a capacity of 48 mAh, it can last for $48/0.00025 = 192,000$ hours or about 21 years.
Comments:	A battery with larger capacity can be used instead, increasing the capacity, although that does not seem necessary in this case.

TABLE 4.4: Testing RTC power usage

- Cost.
- Size.
- Complexity.
- No royalty.
- Can operate on a 3-5V system.
- Can be read in most computers

The above mentioned memory cards was examined, and the regular size SD card was chosen. Although it is the largest of the three different versions of the Secure Digital cards, it was among the cheapest and most common card on the marked. It used the SPI-interface, which made it compatible to the chosen ATMEGA2550 microcontroller. Using the card required no royalty to be paid, in contrast to the MMC. A downside was that the voltage supply had to be between 2.7 and 3.6V in order for it to operate. Nevertheless, the card was chosen and proved useful in the test-module phase of the project.

Software for writing and reading to the card was already excising. The best is called FatFs and is explained in chapter 2.2.3.

4.4.1 Schematic of the SD circuit

Figure 4.3 shows a schematic of the circuit used to connect the SD card to the microcontroller. The SD card connector is connected to the SPI bus on pin 1 (SS), 2 (MOSI), 5 (SCK) and 7 (MISO). Pin 3 and 6 is connected to ground, whilst pin 4 is connected to the microcontroller power control. The latter is also connected to ground via a 0.1µF decoupling capacitor, to reduce ripple when power is switched on and off.

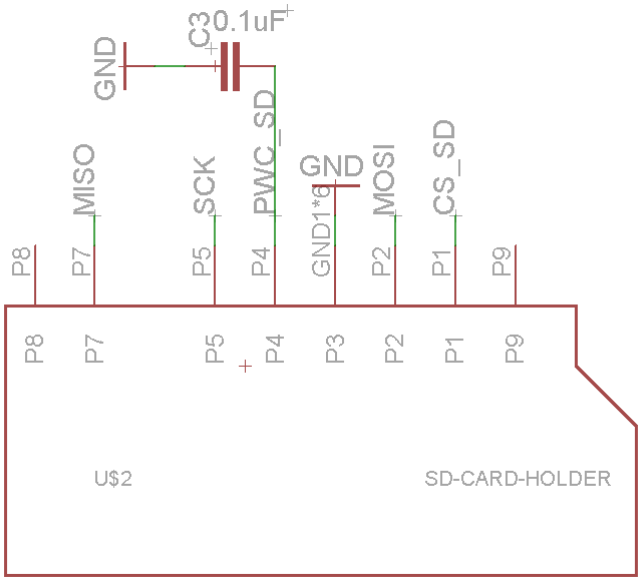


FIGURE 4.3: SD circuit

4.4.2 Testing the Memory Card

Does it always read the correct data. Does it always write the correct data. Should perform a write/read cycle 10.000 times, and check result. Can ACKing be used to ensure correct result? What is the transfer speed.

Read from and write to SD card	
What is tested:	That the microcontroller successfully can read and write to/from the SD card without corrupting data.
How is it tested:	Connect the SD card connector to the microcontroller using the circuit at figure 4.3. Write to 1,000 files the number 1 - 1000 to each of them. Restart the system. Read the files. Sum up the numbers from each file.
Desired result:	The numbers should sum up to 500,500.
Actual result:	Success, the numbers summed up to 500,500.
Comments:	No comment.

TABLE 4.5: SD test 1 - Read/write from/to SD card

Erasing from SD card	
What is tested:	That μC can erase files from the SD card .
How is it tested:	Connect the SD card connector to the microcontroller using the circuit at figure 4.3. Create 1000 files of different name and varying content. Check that all files exist on the SD card. Erase them afterwards.
Desired result:	All files are erased.
Actual result:	Success, all files got erased.
Comments:	No comment.

TABLE 4.6: SD test 2 - Erasing files from SD card

Setting time stamp on files	
What is tested:	That files can get a correct timestamp, received from RTC.
How is it tested:	The two circuits for SD card and RTC is connected to the μC as displayed in figure 4.3 and 4.2. Read correct time from RTC. Create a file and set this time as time stamp.
Desired result:	Correct timestamp is set on all files.
Actual result:	Success, all files got correct timestamp.
Comments:	No comment.

TABLE 4.7: SD test 3 - Setting timestamp

4.5 LCD

In order to communicate information between the user and the device, a LCD with touch and background LED's was used.
The DOGS102W-6 from Electronic Assembly is a 102 x 64 pixel, high-contrast liquid crystal display, for use in small application projects. It is interfaced using SPI, runs on 3.3V, and uses typically 250µA, which result in a power usage of about 0.825 mW. Using the back light source at full capacity, an extra 50-60 mA, or 0.2 W, could be added to the power usage.

Programming the LCD is performed by pulling the Chip Select (CS) pin low, before a transfer can begin on the MOSI wire. If commands are to be sent to the LCD controller, the Command(CD)-pin need to be pulled low. If pixel-data are to be sent, the CD-pin needs to be pulled high.

The actual set of commands can be found in the data sheet of the UC1701x controller, which is put in the appended CD.

When the initial set-up procedure has been transferred to the LCD, pixel-data can be transferred. The display is arranged so that the 64 rows are divided into eight pages, each eight bit. For example, setting both the page and column address to zero, and writing 255 (FF in hexadecimal) result in a 1 x 8 black column up to the left of the display. If increasing the column address by one, and writing another 255 into the first page, result in a 2x8 black column up to the left. If instead increasing the page with one, and doing the same, the black column will have dimensions 1 x 16.

Obviously, this leads to a rather difficult way of drawing images into the LCD. To deal with this, EA has made a program called KitEditor, where one can start a bitmap editor and draw an image of arbitrary size. This can be exported into a .c header file, containing an array of this picture, ready to be transferred into the LCD.

4.5.1 Schematic of the LCD circuit

Figure 4.4 displays the schematic used to connect the LCD to the computer. The four pins at the bottom, pin 1-2 and 13-14 are used to control the background lights. Pin 1 and 14 are connected to the microcontroller and pulling these lines high result in a current running from the anode (pin 1 and 14) part of the back light to the cathode (pin 2 and 13) part, and forth to ground.

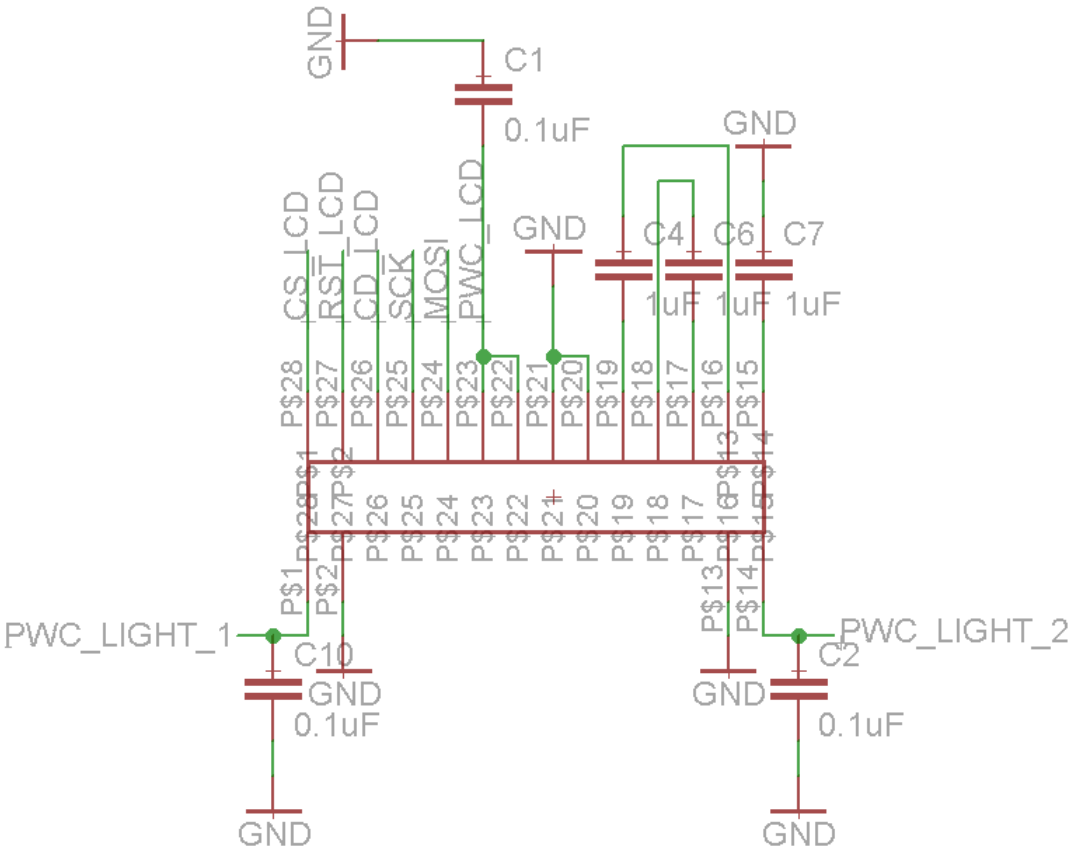


FIGURE 4.4: LCD circuit

The three 1µF capacitors connected between pin-pair 17-18 and 16-19, and 15 and ground, are required for 3.3V operation.

Pin 20-21 are bound to ground and pin 22-23 are connected to the µCs I/O ports and decoupled to ground using one 0.1 µF capacitor each. In the latter case, a hard-wired connection to V_{cc} could also be used, but this connection will give the µC control of powering the LCD.

Pin 24 and 25 are connected to the SPI bus' MOSI and SCK lines. To differ between command and data transmissions, pin 26 can be put high or low. Pulling pin 27 low for a short period of time, will reset the LCD. At last, the LCD is selected using the slave select pin number 28.

4.5.2 Testing the LCD and back light

The testing of the LCD is performed to check if the display can be turned on and off using software, if it can draw pictures onto the display and if they can be removed again. Due to electromagnetic noise, the display could not be tested while mounted on the breadboard. Rather, it had to be mounted on a dedicated PCB. This also reduced the need for wires, making it easier to grasp.

Testing the back light is performed to check if it can be turned on and off.

LCD test 1	
What is tested:	Display a picture
How is it tested:	Connect the LCD to the microcontroller/STK600 card using the LCD circuit displayed at figure 4.4. Load into the LCD controller the picture to the left in figure 4.5.
Desired result:	A clear and viewable image is displayed on the LCD.
Actual result:	Success, pictures of the LCD is displayed to the left in figure 4.5.
Comments:	No comment.

TABLE 4.8: LCD test 1

4.6 Touch Screen

In order to provide a method for the user to navigate to the menu and control the device, a touch panel was added to the LCD. The analogue touch panel

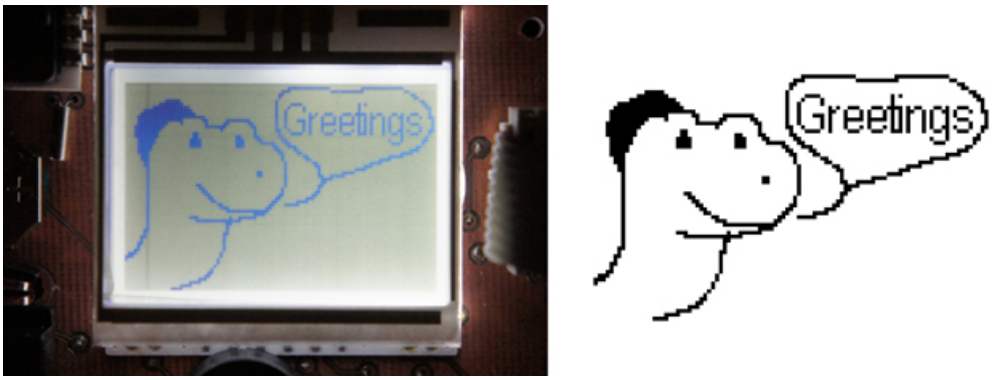


FIGURE 4.5: LCD test image. Input to the right, result to the left.

Backlight test	
What is tested:	If the backlight can be turned on and off.
How is it tested:	A PCB is made to mount the LCD and backlight. The card is connected to the microcontroller/STK600 using a ribbon cable. Light is then switched on and off, with a frequency of 2Hz. Afterwards it is left on for ten seconds to measure the current consumption in order to calculate the power usage.
Desired result:	The backlight should successfully be switched on and off with a frequency of 2Hz. Correct current should be measured.
Actual result:	The backlight was successfully turned on and off in the given frequency. Current consumption was measured to 25mA per anode/cathode pair, total 50mA.
Comments:	No comment.

TABLE 4.9: Backlight test

EA TOUCH102-1 is an accessory to the above mentioned LCD. It can simply be stuck onto the display and a 4-pin flexible cable is then connected to a ZIF connector.

By connecting two of the four pins to ground (pin 3 and 4 in figure 4.6) and sourcing either pin 1 or 2 to V_{cc}, pin 2 or 1 output an analogue signal that is linear to either the x- or y-coordinate of the pressed point. This analogue signal can be read by the microcontroller by connecting it to one of the ADC ports.

4.6.1 Schematic of the touch circuit

Figure 4.6 displays the four pin ZIF connector and how it is connected to the microcontroller. Pin three and four is connected to ground. Pin one are connected to ADC1 at the microcontroller, and pin two are connected to ATD2. When pin 1 is to be read, ADC1 is set as an input and ADC0 is set as an output. Writing a logical one to ADC0 will make it possible to read the x-direction of the panel. Vice versa, setting ADC1 as an output, ADC0 as an input, and writing a logical one to ADC1, will make it possible to read the y-direction of the panel.

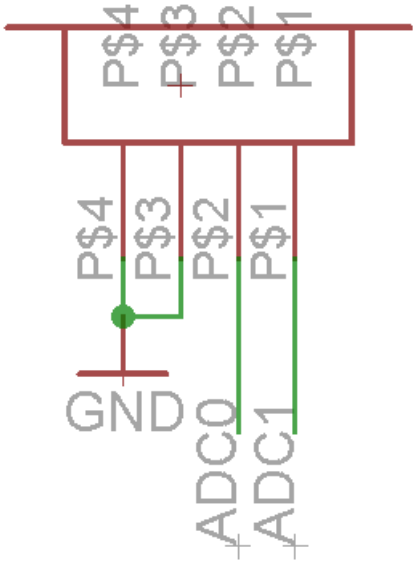


FIGURE 4.6: Touch circuit

4.6.2 Testing the touch panel

Since the analogue output from the touch panel output $V_{cc}/2$ when no pressure is applied, the touch panel can not properly signal when a pressure is applied in the center of the panel. This does, however, not influence the design significantly if the options are selected from the rim of the panel.

Testing touch panel	
What is tested:	That pressure applied on the panel is registered by the microcontroller.
How is it tested:	The touch panel should be connected to the microcontroller as displayed in figure 4.6. A picture with nine boxes, as displayed in figure 4.7, is displayed by the LCD. Each box should be pressed once, and the microcontroller should register what box was pressed.
Desired result:	Every push should be registered and displayed through the terminal.
Actual result:	Not working properly. The touch panel did not produce a linear value of where it was pushed.
Comments:	It does appear that the panel is not working properly. This might either be due to a faulty/bad designed component, or it might be the set up that is incorrect. Regarding the latter, the data sheet claims that "Connecting a voltage of e.g. 3.3V to the pins Top-Bottom makes it possible to read out a voltage on pin Left or Right which is linear to the Y-coordinate of the pressed point. The X coordinate will result when the voltage will be supplied to Left-Right and measurement is done at Top or Bottom". As depicted in figure 4.6, two of the wires, Top and Right was hard-wired to ground while the two other, Bottom and Left, was connected to two of the microcontrollers ADC-ports. The idea behind hard-wiring the two wires to ground was to avoid running wires all the way to the microcontroller. Since the Bottom and Left pins could be respectively connected to V_{cc} and ADC when the left-right axis should be read, vice versa when the top-bottom should be read.

TABLE 4.10: Testing touch panel

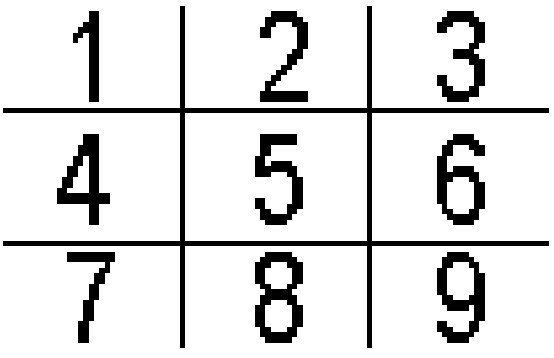


FIGURE 4.7: Touch test image

4.7 Computer Interface

Interfacing the computer from the microcontroller is easiest done using the UART protocol, and transferring over the RS-232 serial port. The RS-232 port are, however, not common in use by most personal computers, therefore it was decided to rather implement a USB interface on the device.

The CP2102 is an USB to UART bridge, able to convert between UART and USB, in a very easy manner. Connecting the CP2102 between the microcontroller and a micro USB receptacle will then transfer UART signals over the USB port and be received by the computer as if it was a regular serial interface, like RS-232.

The bridge is USB 2.0 compliant, capable of up to 12 Mbps transfer/receive speed. Using the bridge with a computer does require a driver. This is put on the CD that follows this report.

In hindsight, the CP2102 turned out to be rather difficult to solder and test. This was because it only existed in a QFN-28 package, meaning that it has altogether 28 pins, divided on four sides on the bottom, i.e. seven pins per side. Each pin had a width of 0.23 mm, and the spacing between each pin was 0.5 mm. This, in addition to that the pins was on the bottom, not on the sides, resulted that it was close to impossible for the author to solder this pin. Also, it turned out to be impossible to test the connection with a multimeter, since the pins was at the bottom and thus covered by the chip.

After a mere month with repeatedly trial and error, the USB to computer interface was given up.

Substituting the USB interface, was the RS-232 serial port originally used in the specialization project. This was not an ideal solution, due to the above mentioned fact that few personal computers are equipped with a serial port.
It could nevertheless be justified because this device was just a prototype, not directly intended for home use. The USB interface is not a crucial part of the

task, and if required, it could be tested and added to a later version in another project.

4.7.1 Schematic of the USB circuit

Figure 4.8 display the schematic of the original USB circuit. This is added to ease an eventual continuation of this project, by another student. It is not the circuit used on the device.
Pin 7 and 8 is connected to the VBUS pin at the micro USB receptacle. This supplies the bridge with the 5V from the USB cable. The VBUS is also connected to the battery with two diodes in series, but this is explained in detail in section 4.8.

Pin 6 output a 3.3 V regulated power supply that could be used to power other ICs in the circuit. This is not done due to the battery that supplies the circuit with the power it needs. Both pin 6 and 7/8 is connected to ground with a decoupling capacitor of respectively 0.1 µF and 1 µF.

Pin 9, reset, is connected to V_{cc} via a series resistor of 4.7 kΩ to avoid accidentally reset.

Pin 4 and 5 is connected to D+ and D- at the receptacle.

At last, pin 26 and 25 is, respectively, connected to the RXD and TXD pin at the microcontroller.

Figure 4.9 display the alternative RS-232 circuit as used on the device. It uses a MAX233CPP charge pump to ”pump up” the signal from about 3 voltages as used on the device, to around 12 voltages as used by the computer.

The MAX233CPP have two internal capacitors that are used for the ”pumping”, thus eliminating the need for external capacitors. The receive and transfer signals from the micro contoller is routed into pin three and two, respectively, and routed from the MAX to the DB9 RS-232 connector from pin four and five.

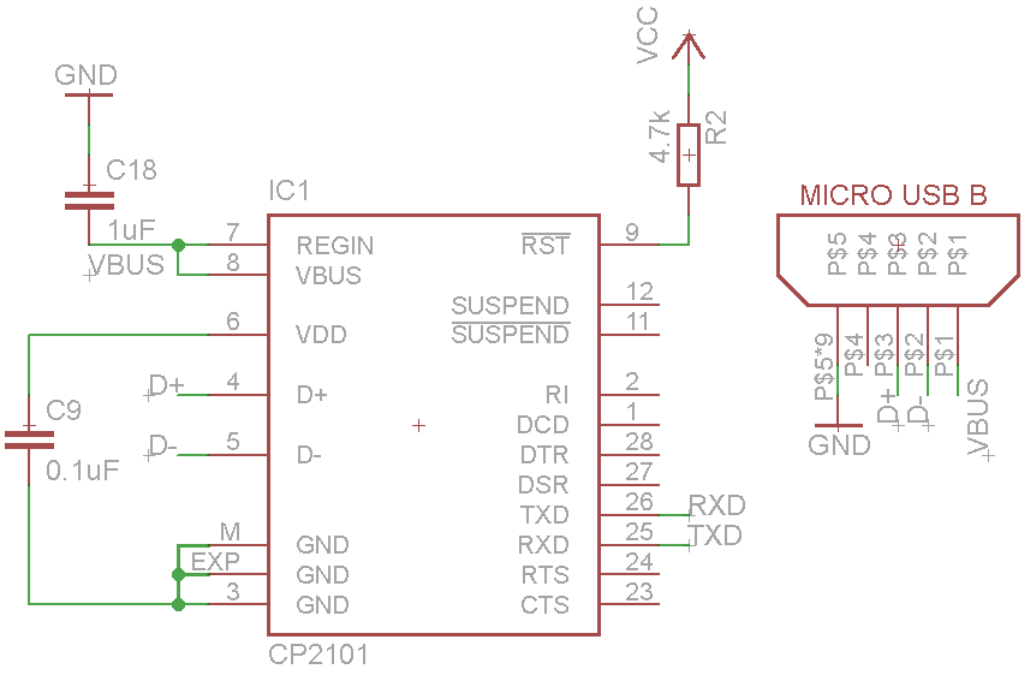


FIGURE 4.8: USB circuit

The MAX is also externally connected between pin 12 and 17, 11 and 15, and 10 and 16. The DB9 is externally connected between pin 7 and 8, grounded at pin five, and connected to RX and TX on pin three and two, respectively.

No further components than the MAX233CPP and DB9 connector, is required for the RS-232 interface.

4.7.2 Testing the computer interface

Testing the USB to UART bridge	
What is tested:	Testing if the bridge will connect the microcontroller to the computer.
How is it tested:	Connect the USB bridge as displayed in figure 4.8. Connect the circuit to the computer using a USB cable. Start up ”Termtite”. Send messages to the microcontroller.
Desired result:	The microcontroller should echo the messages back, adding ”From microcontroller: ” to each message.
Actual result:	The messages did not echo back anything, and the computer did not recognise the device.
Comments:	The USB to UART bridge failed to produce the correct result, and thus can not be used in the project.

TABLE 4.11: Testing the USB bridge

Testing the RS-232 interface towards Termite	
What is tested:	Testing if the RS-232 can transfer signals from the microcontroller to the computer, and back again.
How is it tested:	Connect the RS-232 interface as displayed in figure 4.9. Connect the circuit to the computer using a serial cable. Start up ”Termtite”. Send messages to the microcontroller.
Desired result:	The microcontroller should echo the messages back, adding ”From microcontroller: ” to each message.
Actual result:	Success, the microcontroller echoed back the messages to Termite.
Comments:	No comment.

TABLE 4.12: Testing the RS-232 interface

Testing the RS-232 interface towards SpiroPlot	
What is tested:	Testing if the RS-232 can transfer signals from the microcontroller to the computer program described in chapter 6.
How is it tested:	Connect the RS-232 interface as displayed in figure 4.9. Connect the circuit to the computer using a serial cable. Start up ”SpiroPlot”. Send a series of numbers, imitating a sine graph.
Desired result:	The program should plot the numbers into the graph.
Actual result:	Success. A screen shot of the result is displayed in figure 4.10
Comments:	No comment.

TABLE 4.13: Testing the RS-232 interface

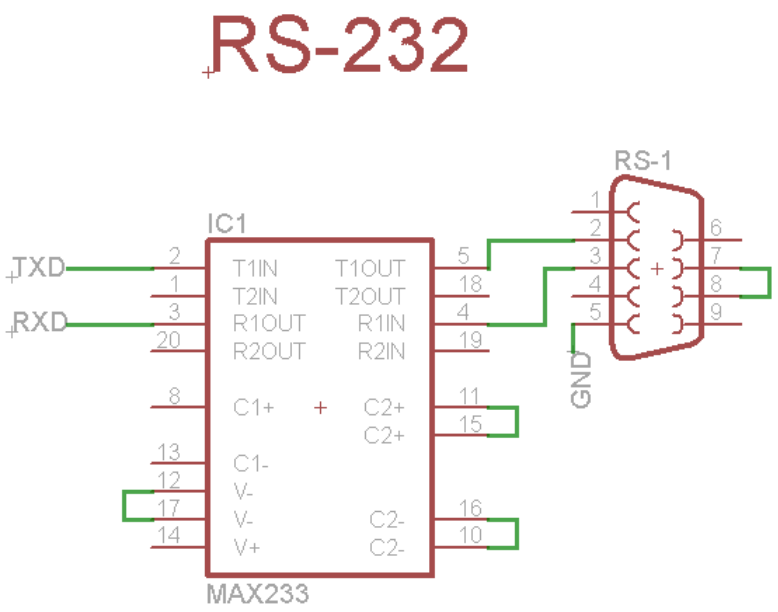


FIGURE 4.9: UART circuit - Remember to change the picture

4.8 Battery

A battery is needed in order to drive the circuit without connecting it to the computer. The battery can be charged using USB, which can charge it with 2.5W of power (0.5A @ 5V).

4.8.1 Schematic of the battery circuit

Figure 4.11 displays the schematic of how the battery is connected to the system. Charging voltage from the USB receptacle (5V) via two 0.3V Schottky diodes, is connected to the positive pole of the battery. The negative pole is connected directly to ground. The two Schottky diodes are necessary both to ensure correct polarity when connecting the battery, but also to transform the voltage from 5V to 4.3V that is the maximum charging voltage of the battery.

The positive pole of the battery is also connected to V_{cc}, to supply the rest of the circuit with power.

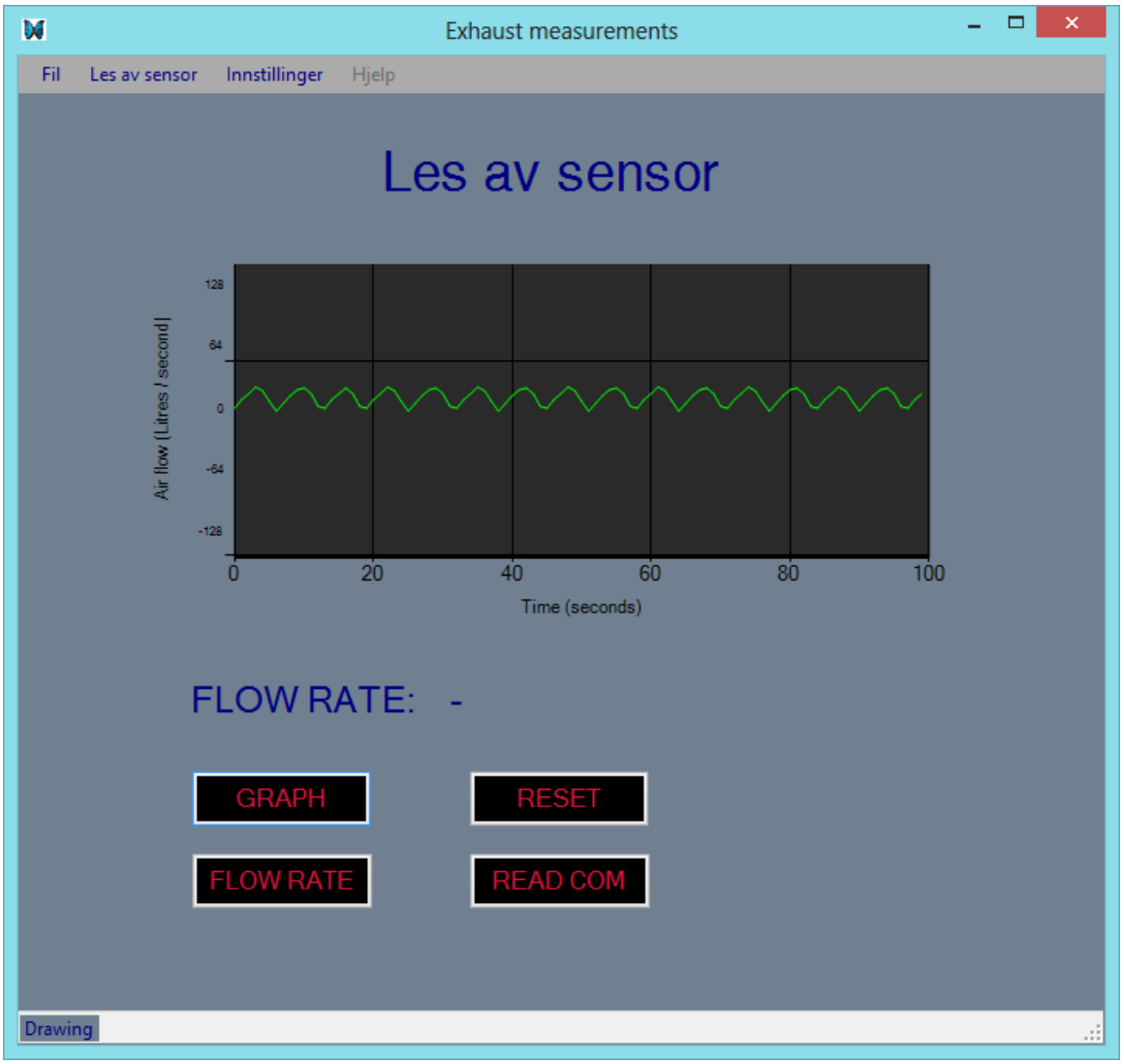


FIGURE 4.10: SpiroPlot draws data sent from the device.

4.8.2 Testing the Battery

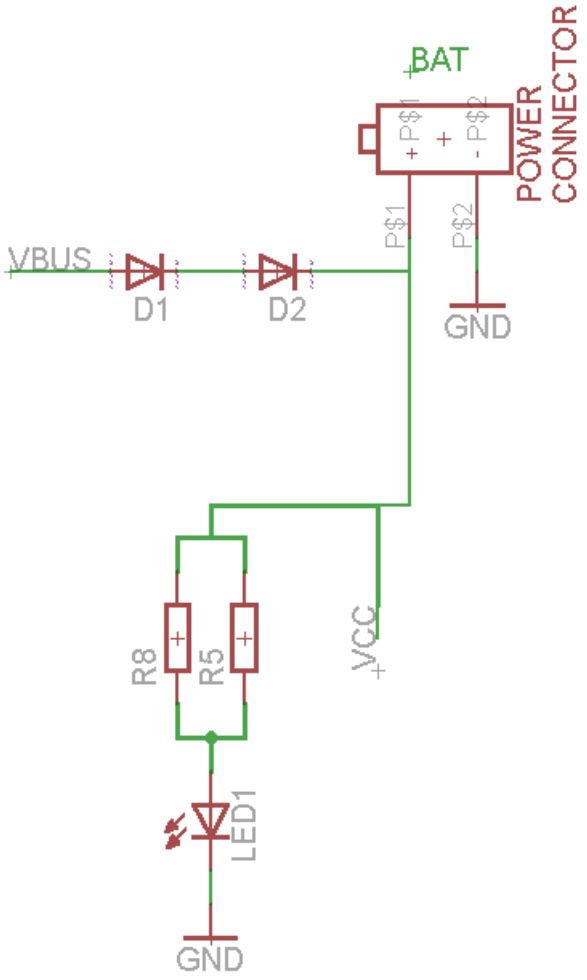


FIGURE 4.11: Battery circuit

Testing battery charge and discharge	
What is tested:	The battery's capabilities of charging and discharging.
How is it tested:	Connect the battery as displayed in figure 4.11. Charge it as much as possible. Measure the voltage over the battery when charging has stopped. Start discharging the circuit using an array of LEDs. When no LED is glowing, measure voltage over the battery.
Desired result:	The battery should be charged up to 4.35V before it automatically stops charging. It should also be discharged down to 3.3V before it stops discharging.
Actual result:	The battery stopped charging at 4.37V. It also stopped discharging at 3.27V.
Comments:	Using a battery of 2,300mAh is probably overkill. Around 1,000mAh should be sufficient.

TABLE 4.14: Charging/discharging main battery

Chapter 5

Result

After all the modules in the previous chapter was designed and tested, they were merged together into the printed circuit assembly shown in figure 5.1. Schematic and board design can be found in the next couple of pages.

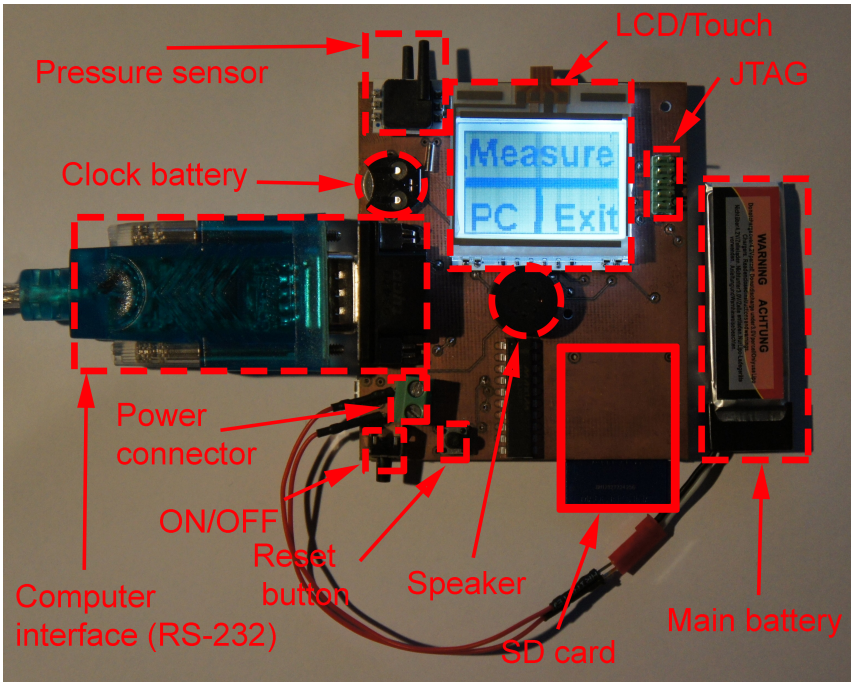
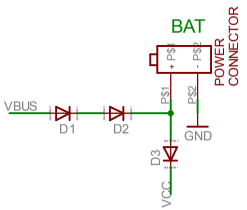
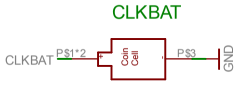


FIGURE 5.1: PCB markup

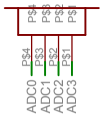
Main Battery



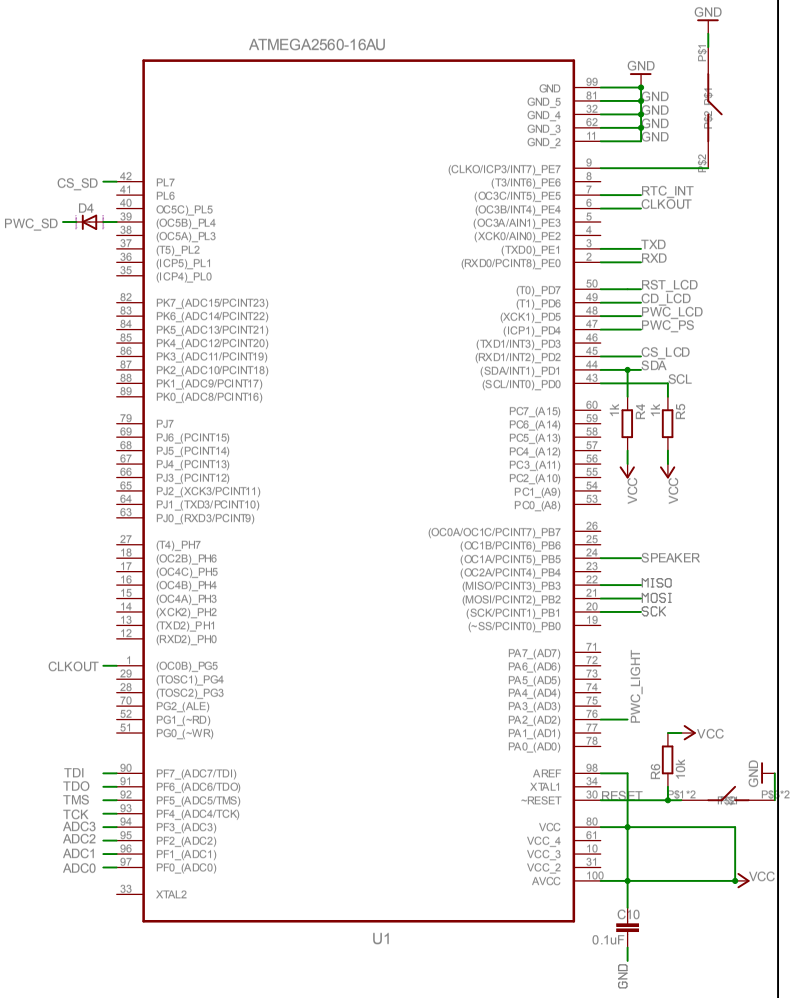
Clock Battery



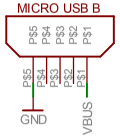
Touch



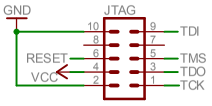
MicroController



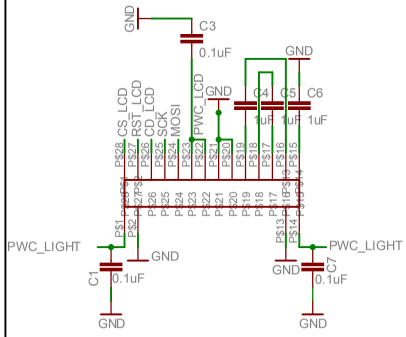
USB



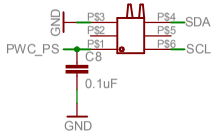
JTAG



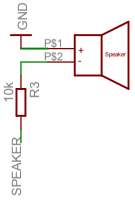
LCD



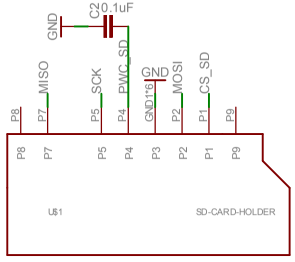
Pressure Sensor



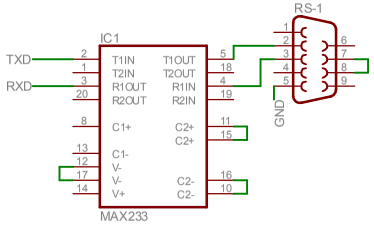
Speaker



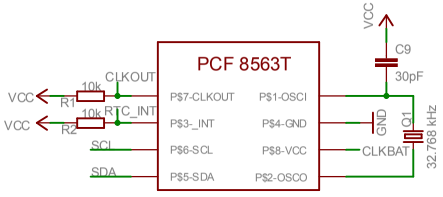
SD-Card



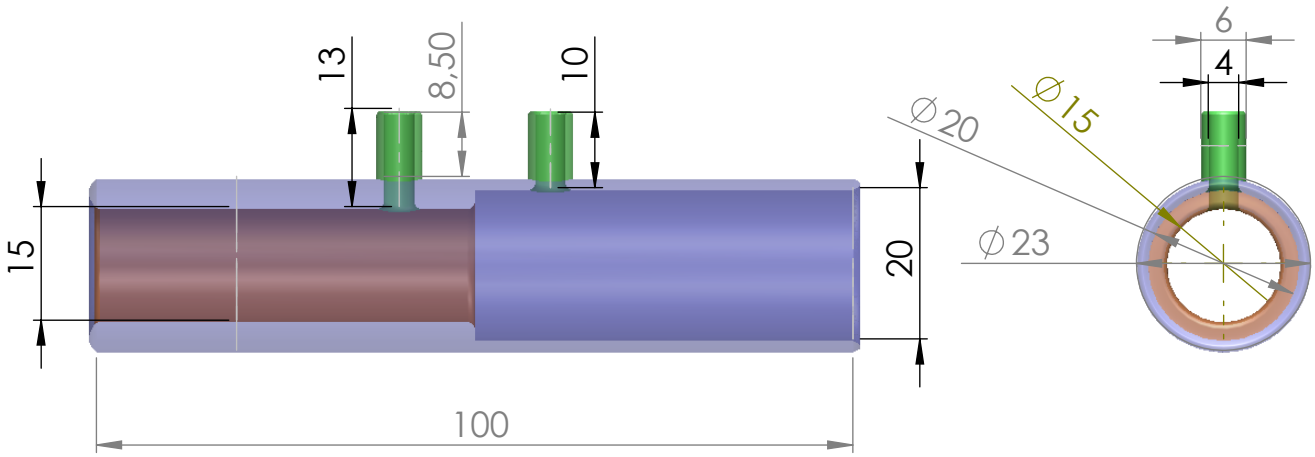
RS-232

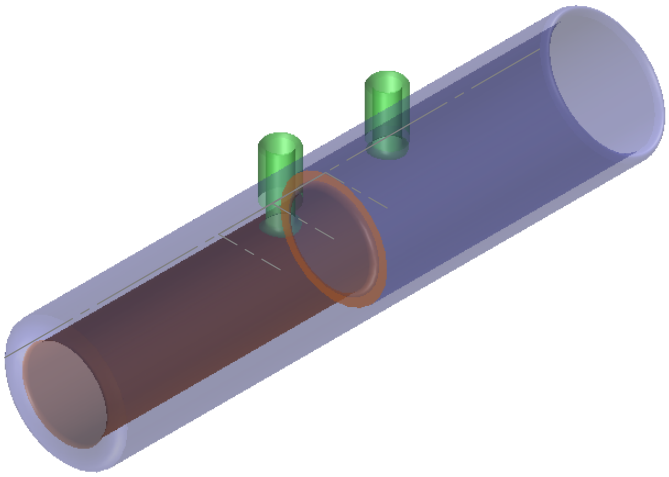


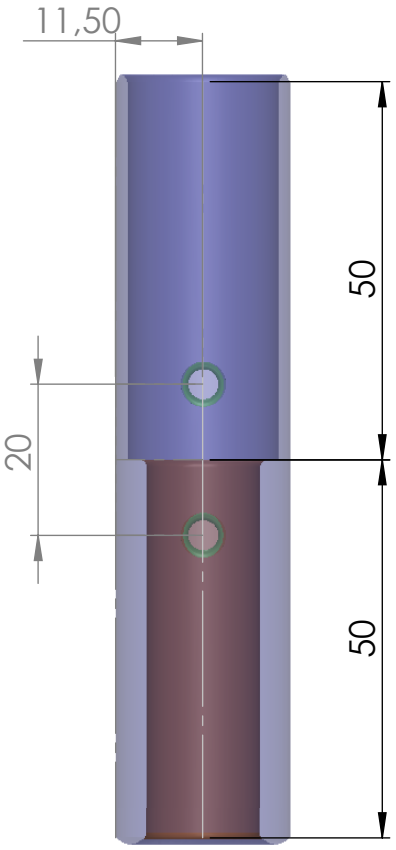
Real Time Clock



Size: A3	Department of Engineering Cybernetics		
Engineer:	Ove-Joakim Istad	Date:	21.12.2013 18:57:54
		Aproved:	
Description:	Spirometer Master project in Engineering Cybernetics	Page:	1/1
Filename:	SPIRO_v4	Revision:	2.0







UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:		DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION	
	NAME	SIGNATURE	DATE	TITLE:		
DRAWN	Ove-J. Istad		07.03.13			
CHK'D						
APPV'D						
MFG						
Q.A				DWG NO.	BlowPipe	A4
			WEIGHT:	SCALE:1:1	SHEET 1 OF 1	

This resulted in a fully functioning spirometer, capable of measuring the pulmonic abilities of a patient, with great accuracy and repeatability. The interface, a 102 x 64 pixel backlit LCD, is intuitive and simple, with a touch screen, enabling feedback to and from the user.

Storing the measurement was suppose to be done on a SD memory card, but because of difficulties explained in chapter 5.6.1, the readings was rather transferred to a computer using a serial cable.

In order to receive data from the device, a computer program was created. This is explained in detail in chapter 6.

The original 2,300 mAh battery used in the module testing, was exchanged to one with only 520 mAh capacity. As the device has a power consumption of 20mA*3V = 60 mW, the battery can supply the circuit with power for 520/60 = 8 hours of continuous use, or almost two years using the device for 5 minutes every week.

Figure 5.4 shows the device ready and operational. The picture is taken at start-up, giving you three options, as explained below.

5.1 Functional description

Figure 5.6 shows how the device can operate in normal mode, that is without a system fault or wrong operating.

The device starts by pressing the button on the bottom. This will wake up the system, using an interrupt directly to the microcontroller. After short time, the main menu will appear. The first option, ”Measure”, allows the user to quickly begin a measurement. A series with three short beeps, following one large beep lasting for 6 seconds will signalize the user to start blowing into the blow pipe (as shown in figure 5.5). The exhaust should start after the third short beep, and last out the entire 6 second period, if possible.

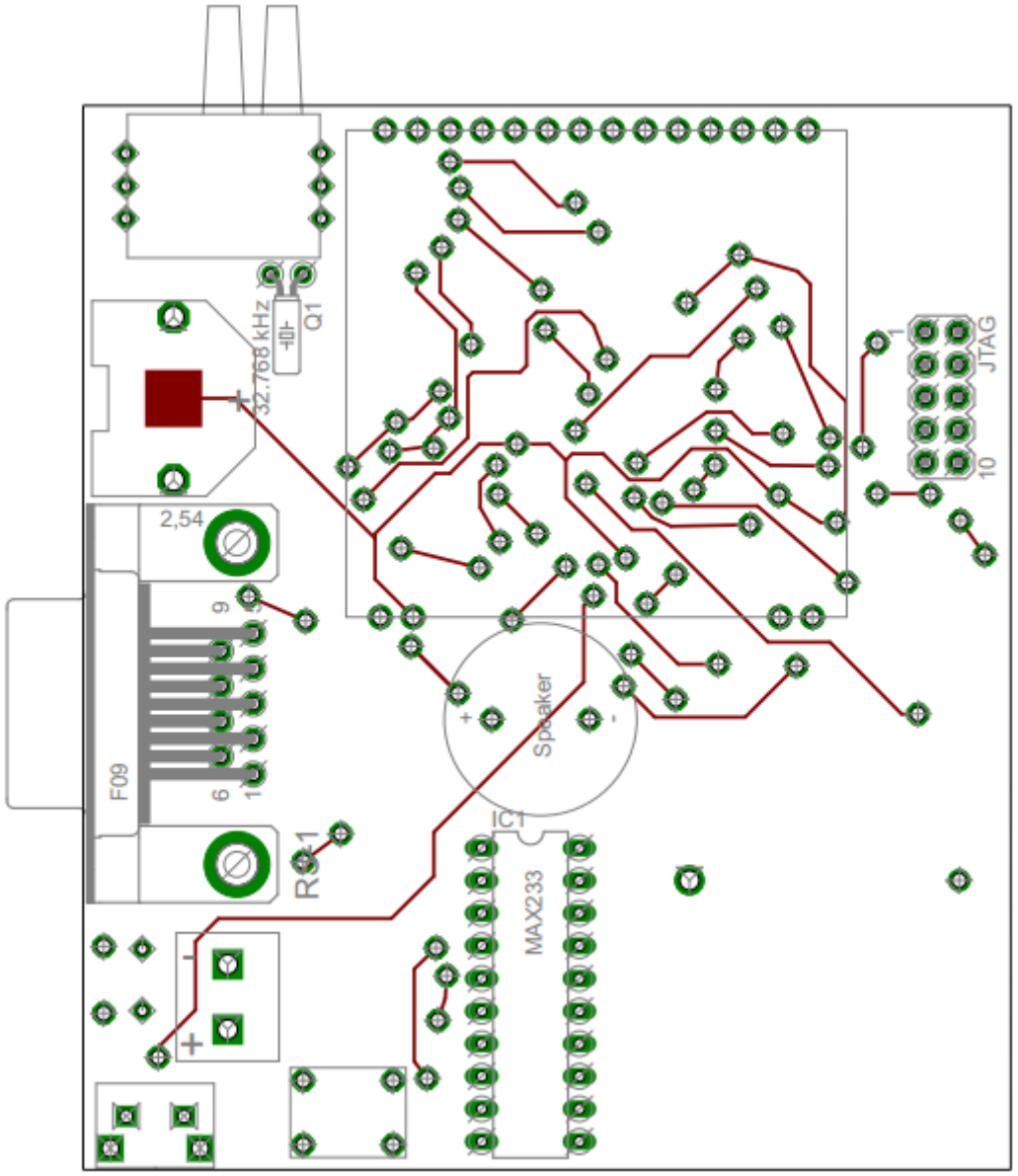


FIGURE 5.2: PCB front

If the user want to transfer this measurement to the computer, the second option ”PC” can be selected. It simply transfer all sample points to the computer program, if a measurement occurred since last power on. If not, the device will signal the user by showing the message ”No data available”.

The third and final button in the main menu, ”Exit”, will set the device in an idle state, only to be waken up again by pressing the power button.

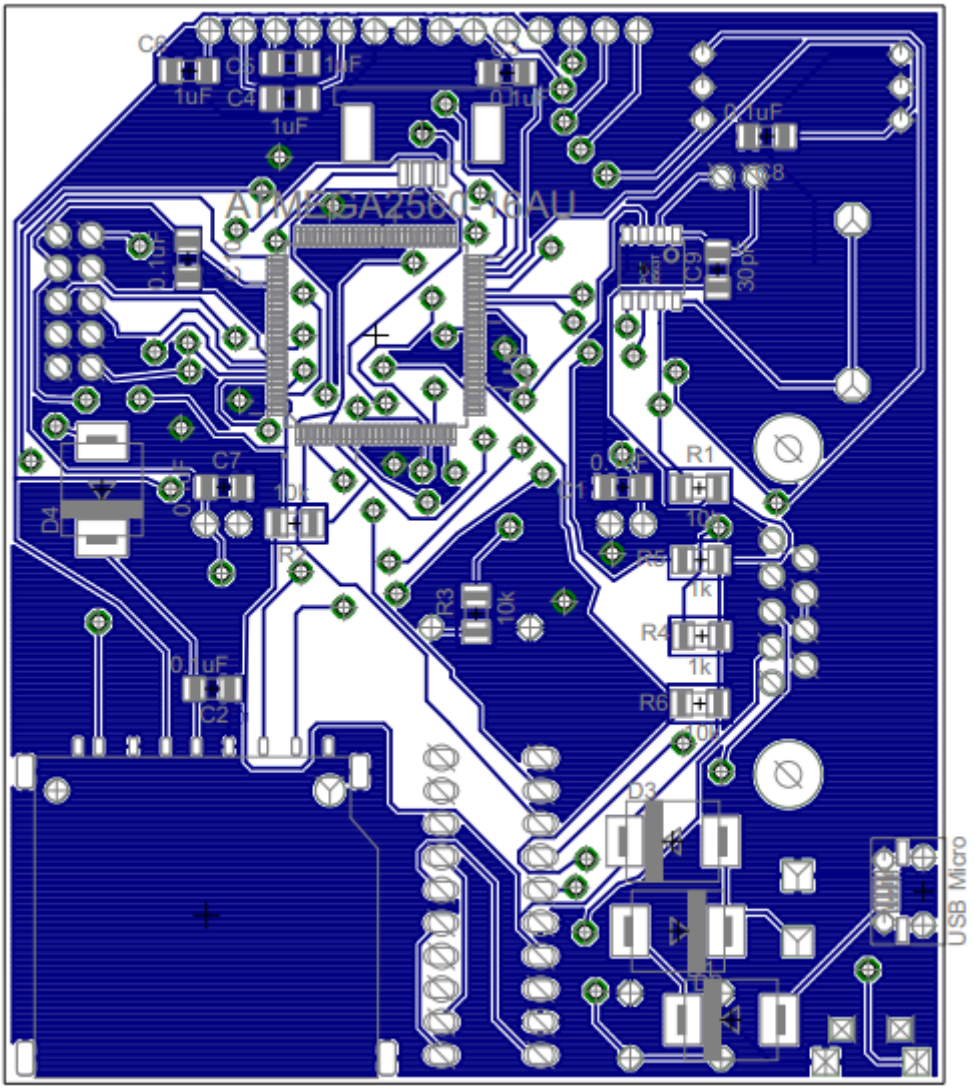


FIGURE 5.3: PCB back

5.2 Schematic of complete design

The next page displays the schematic of the entire design. In order to improve readability, all wires are given a name as depicted in the schematic. These names are then referenced in all the connection points of the circuit. As an example, V_{cc} is connected both to the JTAG interface, the main battery, MAX233CP, RTC and the microcontroller. This can be shown by a wire named V_{cc}.

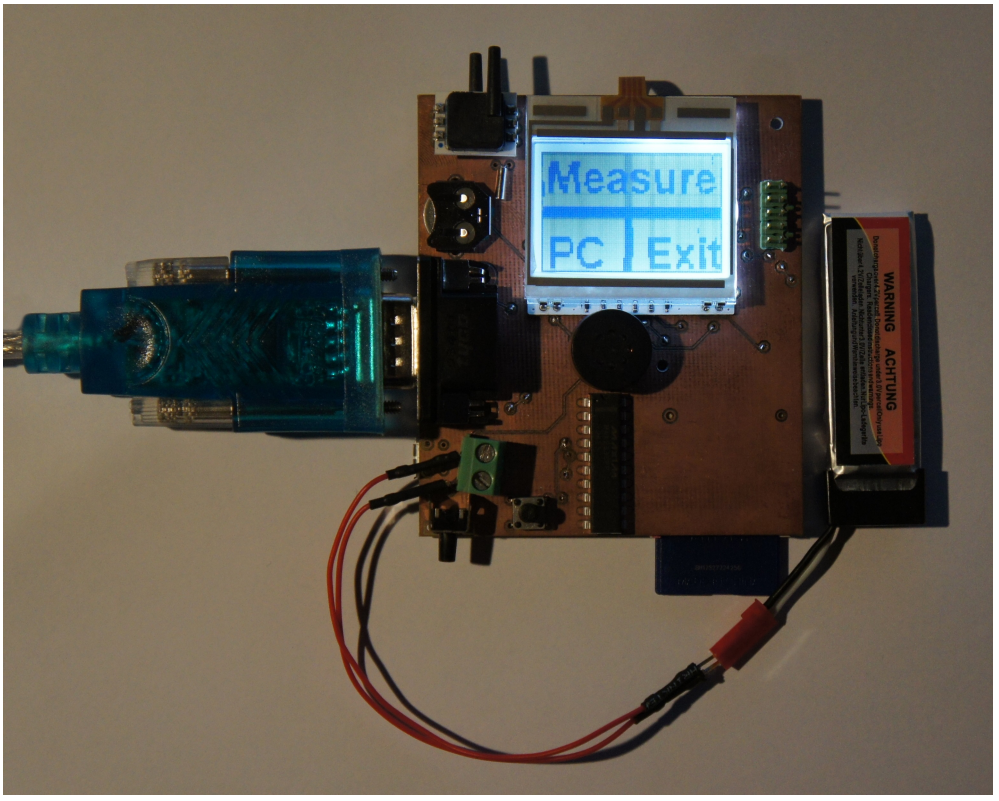


FIGURE 5.4: Finished PCB

FIGURE 5.5: Blow pipe

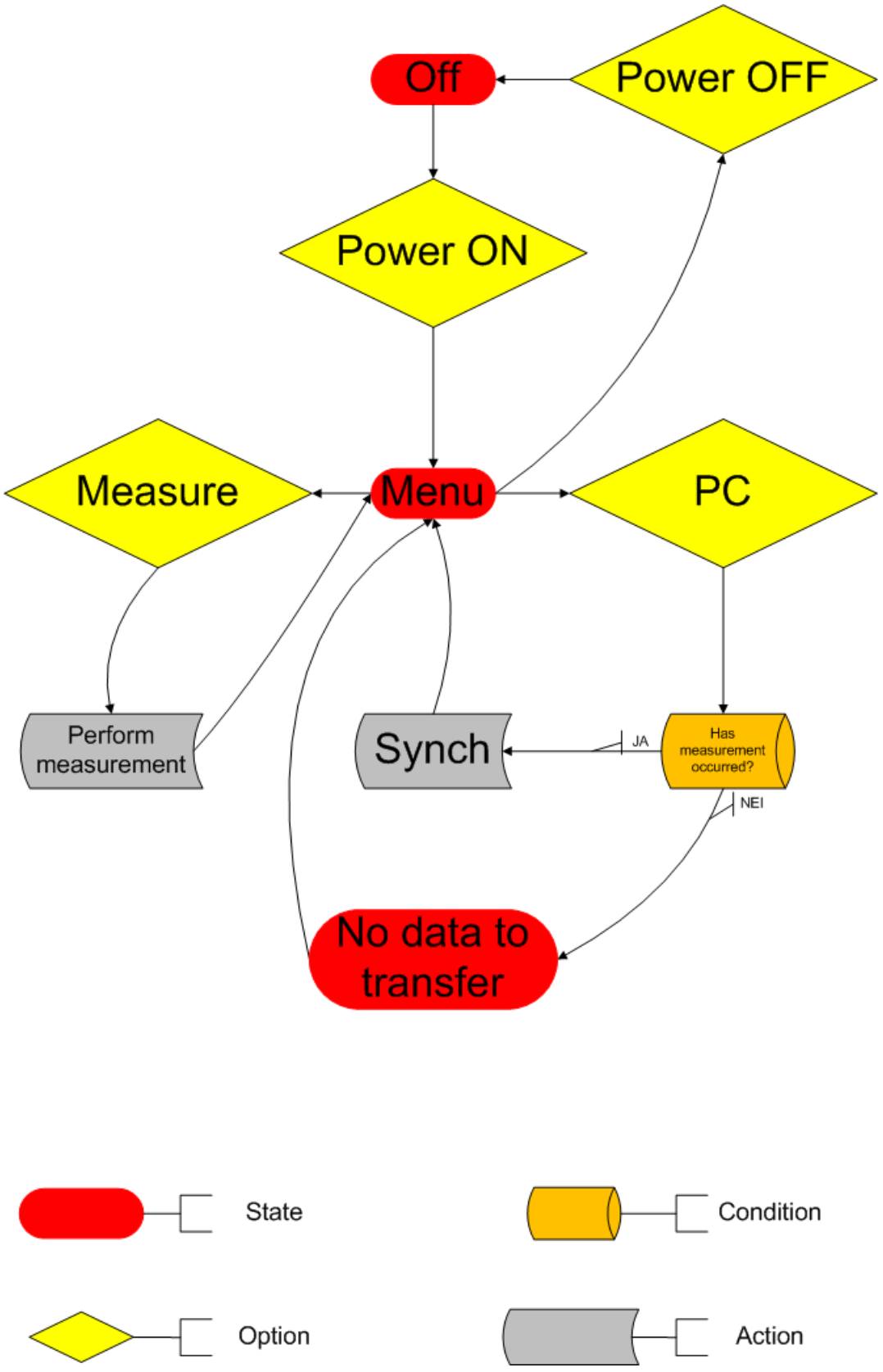


FIGURE 5.6: Flow chart

5.3 Class and block diagram

Figure 5.7 displays a class diagram of all the different c-files and functions used to make this device. Below is a short description of what the different c-files does:

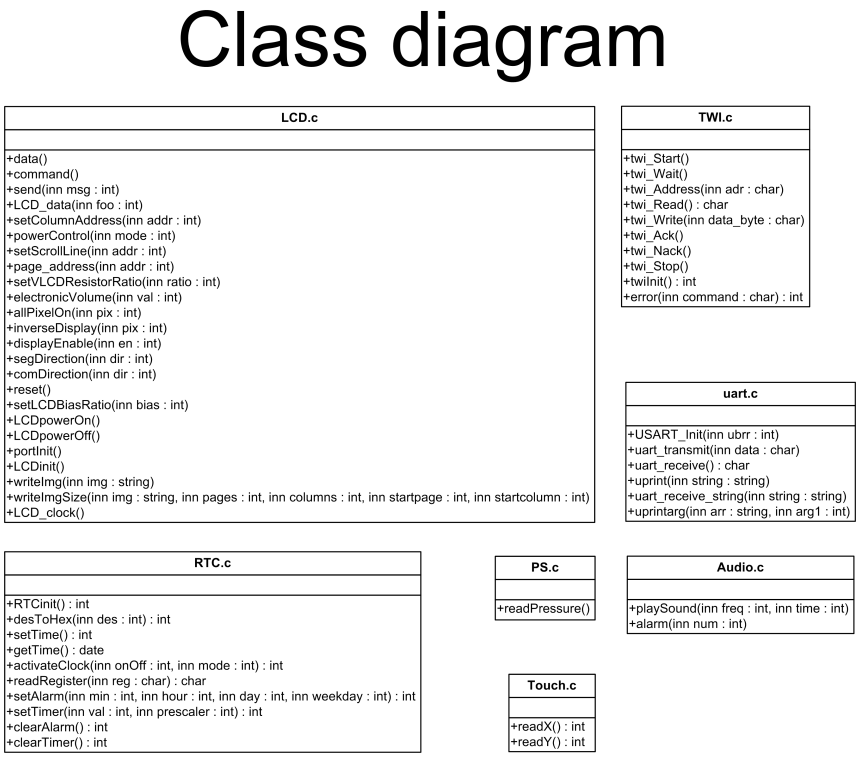


FIGURE 5.7: Class diagram

- spiro.c - Main method for starting up the device and displaying the menu.
- twi.c - Adds functions for communicating on the TWI-bus.
- uart.c - Include functions for communicating with the computer using UART.
- rtc.c - Include functions for communicating with the RTC.
- sd.c - Include functions for communicating with the SD card.
- lcd.C - Include functions for sending and displaying images on the LCD.

- touch.c - Include functions for receiving touch input.
- ps.c - Include functions for communicating with the pressure sensor.
- audio.c - Include functions for playing sound.
- config.h - Includes all the constants used in the code.
- menu1.c - Includes the pictures used by the LCD.
- diskio.c, ff.c, ffconf.h and integer.h are files used by the FatFs - SD card file system.

5.4 Power supply

As the enlightened sensor might have noticed, the power to the different IC's was not guided directly from the power supply, but rather via the microcontroller's I/O ports.
This alternative approach have been used in order to implement a software power control system. Using simple commands, the IC could be switched on and off from the microcontroller.
The author is well aware that this is not a common way to do it, but in this case, a software controlled power switch seems to be a good and working solution. The microcontroller's I/O ports could divert up to 40 mA of current each pin, and 100/200 mA each port (depends on which port) (Ref: Atmega2560 data sheet, page 367,368 on the CD). Since the highest current drawn from one pin is 30 mA (back light panel), and this has been given a separate port, the current limit will not be exceeded.

5.5 Power calculations

This section gives an overview of how much power each module use, and how much power is drawn altogether from the battery.

Microcontroller: 14mA @ 5V and 8MHz gives 70mW in active mode.

7.5µA @ 3V gives 22.5µW in power-down mode.

Pressure sensor: 3mA @ 3V gives 9mW.

Back light panel: 40mA @ 3.3V gives 132mW.

LCD: 250µA @ 3.3V gives 0.825mW.

SD: 80mA @ 3.6V and 25Hz gives 288mW.
Note: A current use of 80mA exceeds the maximum current drawn from each pin. This should be changed in a new version of the circuit board.

RTC:
N/A since the clock battery drives this IC.

MAX233:

15mA @ 5V, 2kΩ load on both inputs gives 75mW.

Touch panel: 25mA @ 3.3V gives 82.5mW.

Speaker: 0.5W according to the data sheet.

5.6 Problems that occurred

In this section, the most severe problems that occurred will be examined and explained how one can possibly solve them. Many of the problems were not solved during this project because of lack of time, and because of the miller machine encountered an error multiple times, lastly in the last month of submission of this thesis.

5.6.1 SD card

The SD card only operates with a voltage level around 3.3 Voltage. Even though the SD module worked in the module test phase, it did not work when connecting the battery to the circuit. Using a multi meter, it was discovered that the Schottky diode, intended to lower the voltage level onto the SD card, did not lower the voltage enough.

A possible solution is either to add another diode in series with the previous, or to change the entire diode solution in the design, with a voltage divider (two resistors in series, where V_{in} is given over both, and V_{out} is given over one of the resistors 5.8).

5.6.2 RTC

The RTC stopped working in the middle of testing the finished circuit assembly board. It was examined using a multi meter, but the voltage levels at each pin was as they supposed to be. A possible solution can be to exchange both the RTC chip, the crystal, and the capacitor and resistors connected to this module. This will rule out any errors due to faulty components.

The design and software is not thought of as a cause of error because it worked at the beginning of the test period.

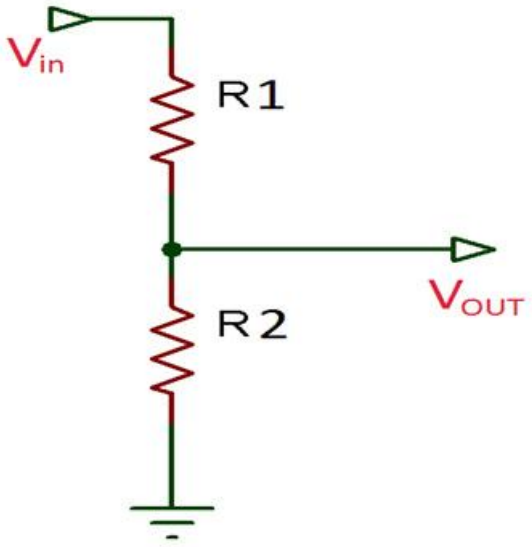


FIGURE 5.8: Voltage divider, as an alternative to diode voltage drop

5.6.3 Power button

The power button displayed in figure 5.1 toggles between turning on the device, and sending it to a low powered standby mode. It is a simple design where a press of the button should connect an interrupt pin to ground, and releasing it should tie it to V_{cc} again. This option faulted because in the original schematic, the foot print of the button was mirrored. In addition, no pull-up resistor was included in the design to prevent the voltage at the interrupt port to float.

Resolving these two mistakes ought to make the ON/OFF-button work. This has been done using "rat's-nest" assembly, because of the milling machine was out of order.

5.6.4 USB

Originally the project should include a USB interface, because it is more common than a serial interface using RS-232. This was, however, discarded as it seemed impossible (at least for the author) to solder the UART-to-USB-bridge onto the circuit board.

In hindsight, this could have been completed either by finding a similar component with another package (preferably a through-hole component), letting someone more experienced solder the component, or by using a so called ”pick-and-place-machine”. Because of simplicity and the fact that this project was not meant to necessarily use the state of the art interfaces, the already tested RS-232-interface was used instead.

An alternative approach to this problem could have been to use a microcontroller with a built in USB-interface. This would have made the solution easier and quicker to build (less soldering), cheaper because of fewer components and less complex. Unfortunately this was not thought of when designing the solution. To the author, this latter solution seems to be the better choice for the device, and should be implemented in an eventual new version.

5.6.5 Speaker

The current design implements a resistor in series with the speaker. This was mistakenly done because the speaker's data sheet specified that a maximum 0.5 W of power could be sent through the speaker.
With an impedance of 8 ohm and a 3.3 V power supply yield 1.3 W, which is 0.7 V larger than maximum specified power. When measuring the power used by the speaker, it never exceeded 0.5W, so the resistor could be removed.

5.6.6 On/off-button

[SKRIV MER HER]

5.6.7 Voltage levels

[SKRIV MER HER]

Chapter 6

Computer Program

In order to set up the spirometer properly, and to communicate with the hospital, a computer program was made. The spirometer is connected to a computer using a serial cable.

When the computer program has been started, the following options are available:

- Setup the device
- Backup data and synchronize the clock
- Instant plotting of exhaust speed
- Setup the program

Figure 6.1 displays a screen dump of the computer program during real time measurement. Figure 6.2 displays the setting window, and figure 6.3 displays the file menu.

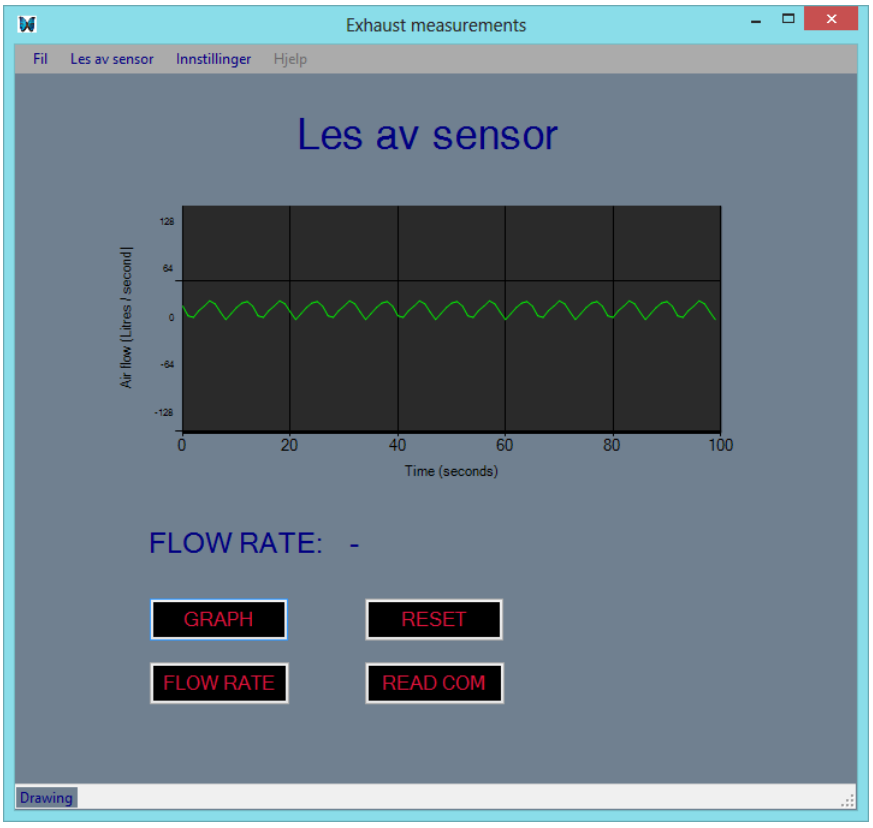


FIGURE 6.1: Computer program screen dump

6.1 Setup the device

In order to make it easier to set up the device, this is done from the computer. The following options can be added

- Name of user
- How often the patient should perform a new reading
- How often backup should be performed
- Alert the doctors office either every time backup is performed, or if the patients health is significantly decreasing.
- E-mail address to the doctors office

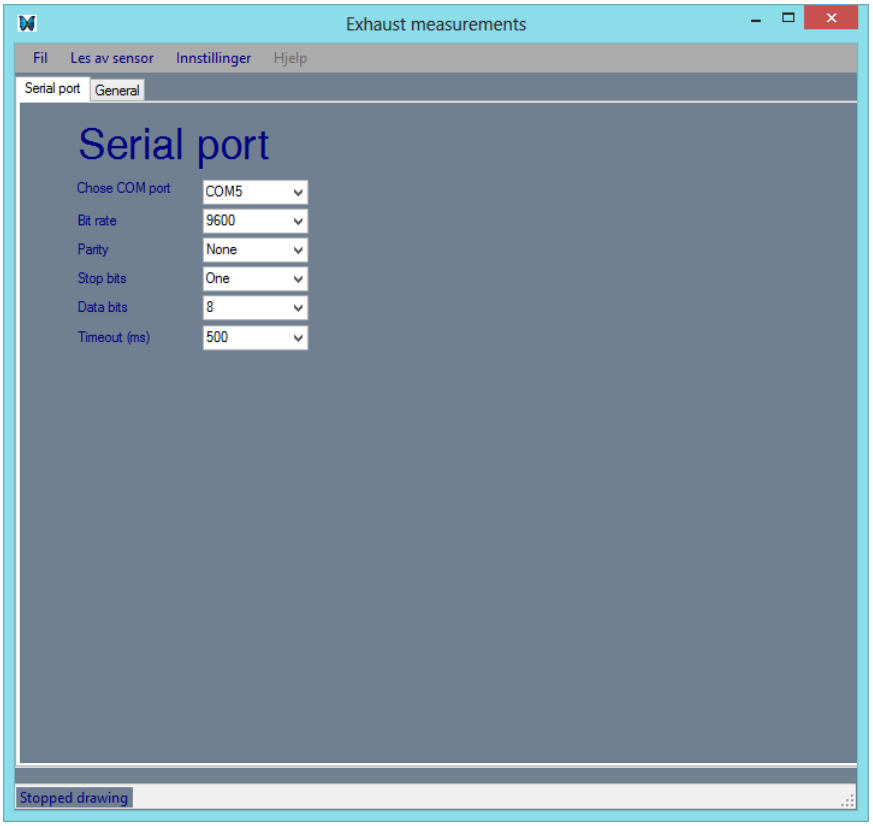


FIGURE 6.2: Computer program screendump

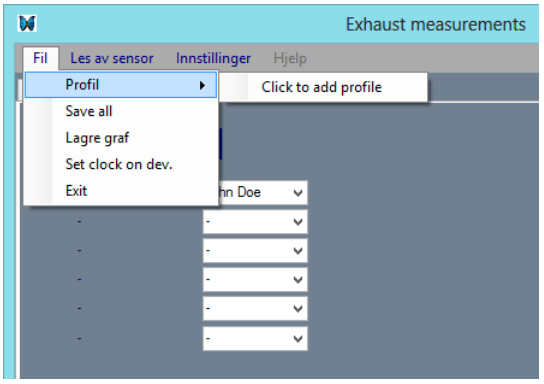


FIGURE 6.3: Computer program screendump

The spirometer has a built in real time clock/calendar with an alarm function. By setting the reading and backup limit, the spirometer will alert the user to do this using an alarm.

6.2 Back up data and synchronize the clock

From time to time the spirometer needs to back up the measurements. This is done simply by connecting the spirometer to the computer and starting the program.

Data existing on the SD card but not in the computers backup file, will automatically be transferred to the computer.

Depending on the setting, an email with the new patient data will be sent to the doctors office using the Advanced Encryption Standard(AES).

In addition to backup the data, the RTC is synchronized with the computer clock.

6.3 Instant plotting of exhaust speed

If desirable, real-time measurement of the exhaust speed can be performed by selecting the ”Sensor Reading”-tab.

Chapter 7

Discussion

7.1 Repeatability

In order for the device to be considered reliable, it has to have a high repeatability, i.e. to give similar result when multiple measurements are performed. A simple test was performed where the author blew into the blow pipe ten times with five minutes intervals.
The test results was as following:

Testing the devices repeatability	
Test number	Result
1	585
2	573
3	583
4	590
5	568
6	570
7	523
8	589
9	565
10	607

Table 7.1: Testing the devices repeatability

This test gave the average of 575 and a maximum deviation of 9%.

According to the American Thoracic Society (ATS), there are three criteria that must be fulfilled in order to get a result accepted by the ATS[12]:

1. Minimum of 3 acceptable FVL attempts
2. The 2 largest FVC's are within 0.15L of each other
3. The 2 largest FEV1's are within 0.15L of each other

In addition, according to a study, ninety percent of patients are able to reproduce FEV1 within 120 ml (6.1%) and FVC within 150 ml (5.3%)[13].

The deviation of this device are slightly above the acceptable criteria for repeatability, and should be improved.

7.2 Battery capacity

As mentioned in chapter 4.8, using a battery of 2,300 mAh was indeed overkill, as it was more expensive, took up more space and used longer time to charge up than if a battery of smaller capacity was chosen. At the time, this battery seemed to be a good choice, but after measuring how much power the device actually used, this extra capacity can be reduced with 90-98%.

One of the local hobby store in Trondheim could sell a battery of 65 mAh for about 10$, or about a third of the price of the original 1300 mAh battery. This battery could supply the device with continuous power for 2-3 hours, enough for several months use (5 minute usage each day).

7.3 Circuit errors

A number of errors with the ciruit did occur. This it thoroughly explained in section 5.6, and the suggestions given there should be tested and implemented.

Chapter 8

Conclusion

This project proved to solve its initial problem, namely to perform regular measurements of the users exhaust capacity. This was performed using a medium grade pressure sensor, measuring the pressure before and after a contraction of a blow pipe.
Using a touch panel mounted on an LCD, the user could initiate the measurement and send the data to a computer using a standard serial port (RS-232).
This data could be viewed in a specially designed computer program called ”SpiroPlot”, and then saved locally on the computer.
The computer program could also calculate the ”Forced Vital Capacity”, and ”Forced Expiratory Volume in One Second”, two parameters used to diagnose COPD.

Chapter 9

Further work

Improved LCD:
The LCD could be improved with a larger display with higher resolution, for a better user experience. This will probably raise the total cost and complexity, but it might be worth it.

Swipe gestures:
Adding swipe gestures, i.e. press-and-drag movement on the touch screen, will further improve the user friendliness of the device. It could, for example, be used for navigating back in the menu.

Casing:
Although planned, no casing was constructed for the circuit board. Designing and building this will make the finished product easier to handle and protect the components inside. It is an absolute must, should the device be produced for the open marked, and design and ergonomic should be vital focus areas.

Change pressure sensor:
If the measurements have too poor resolution, a pressure sensor of premium grade could be used instead of the one used in this project. This will raise cost, but improve accuracy.

Improve tubing:
The tubing constructed for this project, could introduce turbulence in the air flow through the tube. The inner profile of this tube could be changed from orthogonal corners, to a more aerodynamic shape, reducing turbulent flow.

Calibrating the device:
No exact calibration has been done in this project, because it was only a proof of concept. Should the device be sold on the open marked, exact calibration is indeed necessary. Maybe the local hospital could be helpful with this, since they probably already have calibrating tool in use for their own spirometers.

On/off button:
The on/off button used in this project is a software controlled switch. There is no large problem with using a hardware based button instead, should the power consumption be reduced to zero when not in use.

Implement USB interface:
As mentioned in section 5.6.4, the USB interface did not function as planned. In order to construct a fully functioning device, USB functionality should be improved and implemented using the method described in the same section.

Improve computer program:
The computer program ought to be improved with the following specifications:

- Encrypting data gathered and implement e-mail transfer of this data.
- Ability to save profile settings.
- Adding calibrating options.
- Better error handling.
- Ability to set clock on device (should the device have a functioning RTC).
- Ability to save the graph into multiple formats.
- Create a help menu.

Appendix A

Content on CD

- Copy of this report, in vivid colors
- C source code
- Design drawing of the tube
- Schematic and board layout of solution (SD.sch, SD.brd)
- Images of the resulting circuit board, tube and menu
- ATmega2560 16AU datasheet
- SD card holder datasheet
- HDI Pressure Sensors datasheet from Sensortechnic
- SD-card holder datasheet
- Shottky diode datasheet
- Micro USB type B Recepticle datasheet
- Speaker datasheet
- CP2102 - USB2UART datasheet
- PCF8563 - RTC datasheet

- Dogs graphics series LCD datasheet
- Coin cell holder datasheet
- FatFs - Generic FAT File System Module
- Using the TWI module as I2C master (Application note)
- TWI library

Bibliography

[1] Sørlandet sykehus hjemmeside @ONLINE, December 2012. URL http://sshf.no/kols.

[2] Us department of health & human services @ONLINE, December 2012. URL http://www.nhlbi.nih.gov/health/health-topics/topics/copd/.

[3] Aktiv med kols @ONLINE, December 2012. URL http://aktivmedkols.no/boer-du-oppsoeke-lege/hva-er-kols/.

[4] What adults with chronic obstructive pulmonary disease (copd) should know about 2009 h1n1 flu, Dec 2009. URL http://www.cdc.gov/h1n1flu/guidance/copd.htm.

[5] Copd and the pneumonia vaccine: What you need to know, Nov 2012. URL http://copd.about.com/od/copdtreatment/p/pneumoniavaccin.htm.

[6] Vaccination of adults with asthma and copd, Aug 2010. URL http://onlinelibrary.wiley.com/doi/10.1111/j.1398-9995.2010.02462.x/full.

[7] Spirometry @ONLINE, January 2013. URL http://en.wikipedia.org/wiki/Spirometry.

[8] Copd @ONLINE, January 2013. URL http://en.wikipedia.org/wiki/Chronic_obstructive_pulmonary_disease.

[9] Peter J. Barned. Managing chronic obstructive pulmonary disease. 2nd. ed.:95, 2002.

[10] Frank Zhao. Fatfs, May 2013. URL http://elm-chan.org/fsw/ff/00index_e.html.

[11] Bernoulli's principle @ONLINE, December 2012. URL http://en.wikipedia.org/wiki/Bernoulli's_principle.

[12] M. R. Miller et.al. Ats/ers standardization of lung function testing: Standardization in spirometry. *Euro Respir J 2005*, 26:319–338, 2005.

[13] Paul L. Enright et.al. Repeatability of spirometry in 18,000 adult patients. *American Journal of Respiratory and Critical Care Medicine*, 169:235–238, January 2004. URL http://www.atsjournals.org/doi/full/10.1164/rccm.200204-347OC#.Uvgkh3-oE2s.