



TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies ■

Case Study of Industry 4.0 Capability for Fume Exhaust Units

Master thesis

Study programme: N2612 – Electrical Engineering and Informatics

Study branch: 3906T001 – Mechatronics

Author: **Bc. Eliška Veisová**

Supervisor: Prof. Dr.-Ing. Alexander Kratzsch





Assignment for the master thesis

Course of studies: Mechatronics

Student's name: Eliška Veisová

Subject:

Case study of Industry 4.0 capability for fume exhausts units

Modern manufacturing technology are increasingly demanding a very defined process air to operate the product and system technology reliably. Particularly important is the detection of all hazardous materials, because this process step substantially influences the quality of process air. For an optimum detection, it is necessary to adjust a constant flow rates by user specification and over the operation time. The ULT AG manufactures filter devices for air purification and covers with their range among also other the application fields from laser shaping over electronic manufacturing till medical technology. In these master thesis should the possibility of integration this technology in highly interlinked production lines and visual human machine interfaces be investigated. The thesis should the starting points for the slogan "industry 4.0" and their technical possibilities" work out.

Assignment:

- Research of state of the art and customer requirements
- Definition of up to three functional elements
- Case study for one functional element:
 - Hardware and software requirements
 - Conception of a solution based on available circuit boards, e.g. Arduino, Raspberry Pi


Assessor: Prof. Dr.-Ing. Alexander Kratzsch (University of Applied Sciences)

Supervisor: Dr.-Ing. Stefan Jakschik (ULT AG Umwelt-Lufttechnik Löbau)


Date of issue: 24.03.2016

Date of submission: 25.07.2016

Registry-Number.: MA/EMIm14 – 08/16



Assessor



Dean

Declaration


I hereby certify that I have been informed that Act 121/2000, the Copyright Act of the Czech Republic, namely Section 60, Schoolwork, applies to my master thesis in full scope. I acknowledge that the Technical University of Liberec (TUL) does not infringe my copyrights by using my master thesis for TUL's internal purposes.

I am aware of my obligation to inform TUL on having used or licensed to use my master thesis in which event TUL may require compensation of costs incurred in creating the work at up to their actual amount.

I have written my master thesis myself using literature listed therein and consulting it with my supervisor and my tutor.

I hereby also declare that the hard copy of my master thesis is identical with its electronic form as saved at the IS STAG portal.

Date: 3-1-2017

Signature: 

Acknowledgements

I would like to express my sincere gratitude to Dr. Stefan Jakschik for the continuous support, patience, innovative ideas and for letting me be a part of his company for the duration of my master thesis writing.

I would like to thank my supervisor Prof. Dr.-Ing. Alexander Kratzsch for his insightful comments and guidance. My thanks also goes to Bc. Michal Dostálek for assistance with ARM programming, immense knowledge and encouragement.

Last but not least, I would like to thank my family for making my education possible and for supporting me throughout my life in general.

Abstract

This master thesis has two major purposes: (1) to investigate Industry 4.0 capabilities for fume exhaust units and (2) to create a prototype unit for needs of ULT AG company which complies with the Industry 4.0 rules and shows possibilities for future development in this area.

Industry 4.0 is considered as the fourth industrial revolution describing the future industry development trends leading to smart factories and intelligent production. Part of this thesis introduces basic principles and ideas of Industry 4.0 and presents various solutions related with the fume exhaust industry. As a reaction to increasing number of manufacturing processes employing Industry 4.0, the companies providing filtration solutions such as ULT AG are made to think about future form and features of their products.

The prototype unit uLite (ULT Air Quality Module) was created as a practical part of this thesis. Its purpose is to measure various parameters associated with the air quality and with the filtration unit itself, log them, save them to memory medium, present them to the user and share with other devices. The uLite unit core is based on ARM[®] Cortex[®]-M4 technology. The unit monitors the filtrated air quality and can detect if any harmful substances are left behind. It also has function to measure filtration unit vibration and possibility to connect external waterproof probe. Furthermore there is option to measure pressure difference. Two pairs of pressure differences can be measured with total of four inputs. The unit has its own power supply solution and it is enclosed in durable box.

Keywords: Industry 4.0, STM32F407, air quality sensors, digital and analogue sensors

Contents

1	Introduction	15
2	Industry 4.0	17
2.1	Internet of Things	17
2.2	Cyber-physical systems	17
2.3	Industry 4.0 basics	19
2.4	Industry 4.0 in air-filtration	20
2.4.1	Filter state and air quality monitoring	21
2.4.2	Mechanical state of the filter unit	21
2.4.3	Filter unit electrical properties	22
2.4.4	Fume exhaust unit control box	22
3	Embedded development introduction	23
3.1	Used resources	23
3.1.1	Programming IDE	23
3.1.2	Debugging	23
3.2	Hardware requirements for interfacing ICs	24
3.2.1	Pull-up and pull-down resistors	24
3.2.2	Decoupling capacitors	25
3.2.3	Analogue sensors interfacing techniques	25
3.3	Used communication buses	26
3.3.1	I ² C	26
3.3.2	Serial Peripheral Interface	28
3.3.3	1-Wire	31
4	Prototype measurement box unit uLite	34
4.1	Temperature/humidity sensor choice	34
4.1.1	SHT21 temperature and humidity sensor	34
4.1.2	DS18B20 temperature sensor	39
4.2	Air quality monitoring	43
4.2.1	Analogue sensors in uLite	43
4.2.2	Dust sensor GP2Y1010AU0F	43
4.2.3	VOC particle sensor MP502	46
4.2.4	Carbon Monoxide sensor MQ-7	48
4.2.5	Combustive gasses sensor TGS813	50

4.3	Pressure monitoring	51
4.3.1	AMS 5915 pressure sensor	51
4.4	Vibration monitoring	56
4.4.1	ADXL345 accelerometer	56
4.5	SD Card	61
4.5.1	SD card interface	62
4.5.2	FatFs	63
4.5.3	Commands for communication with SD card	63
4.5.4	SD Card initialization sequence	64
4.5.5	Communication with the SD card	66
4.5.6	SD card library for saving text files	67
4.5.7	Real time clock DS1307 IC	68
4.6	LCD display	72
4.6.1	LCD interface	72
4.6.2	SSD2119 TFT LCD driver	73
4.6.3	STMPE811 TFT touchscreen controller driver	75
4.6.4	Buttons	79
4.7	Ethernet connection	79
4.7.1	Transmission Control Protocol	80
4.7.2	ENC28J60 Ethernet controller	81
4.8	Base board	85
4.8.1	Base Board schematics	85
4.8.2	Base Board PCB	85
4.9	Power Supply board	86
4.9.1	Power Supply board schematics	86
4.9.2	Power Supply PCB	87
4.9.3	Power Supply accessories	87
4.10	Software for uLite box	88
4.10.1	Microcontroller	88
4.10.2	Program structure	89
4.11	Prototype encasing	91
5	uLite result presentation	97
6	Conclusion and perspectives	98
	Appendix A Contents of enclosed CD	99
	Appendix B Libraries on the attached CD	100
	Appendix C SD card commands	101
	Appendix D Saved files content	102
	Appendix E ASCII table	103
	Appendix F Function for creating buttons	104
	Appendix G Base board scheme	105

Appendix H	Power module scheme	106
Appendix I	STM32F407 clock configuration	107
Appendix J	uLite Connection Guide	108
Appendix K	Measuring phase flow graph	109
Appendix L	SHT21 library header	110
Appendix M	Wire Shark records	112
Appendix N	ENC28J60 SPI communication (Rigol)	113

List of Figures

Figure 1.1	Industrial revolutions	15
Figure 2.1	5C Architecture	18
Figure 2.2	Prototype unit requirements	22
Figure 3.1	Open drain (left) and push-pull output (right) comparison . .	24
Figure 3.2	Voltage follower (left) and non-inverting amplifier mode (right)	26
Figure 3.3	I ² C topology	27
Figure 3.4	I ² C protocol	27
Figure 3.5	classical SPI topology (left) and cascade configuration (right)	29
Figure 3.6	SPI Timing	30
Figure 3.7	Start of communication on 1-Wire	32
Figure 3.8	Sending 0/1 bit over 1-Wire	32
Figure 3.9	Reading 0/1 bit over 1-Wire	33
Figure 4.1	SHT21 temperature and humidity sensor interface	36
Figure 4.2	SHT21 trigger temperature measurement	37
Figure 4.3	SHT21 incoming data	37
Figure 4.4	SHT21 initialization and measurement flow diagram	38
Figure 4.5	DS18B20 interface	39
Figure 4.6	DS18B20 start sequence	40
Figure 4.7	DS18B20 initialization (left) and measurement (right) flow graph	42
Figure 4.8	DS18B20 probe	42
Figure 4.9	Dust sensor	44
Figure 4.10	Connection of the dust sensor	44
Figure 4.11	Pulse-driven wave form for IRED	45
Figure 4.12	Dust sensor response	45
Figure 4.13	Dust sensor measurement flow graph	46
Figure 4.14	MP502 VOC sensor	47
Figure 4.15	M502 recovery time	47
Figure 4.16	MP502 hair spray test	48
Figure 4.17	MQ-7 schematic	49
Figure 4.18	MQ-7 connection	49
Figure 4.19	TGS813 schematic	51
Figure 4.20	AMS 5915 pressure sensor interface	52
Figure 4.21	AMS 5915 communication example	54
Figure 4.22	Flow graphs of AMS 5915 initialization and measurement . . .	55
Figure 4.23	ADXL output vs. gravity force	57
Figure 4.24	Connection scheme of the ADXL345 accelerometer	58

Figure 4.25	ADXL345 data acquirement	59
Figure 4.26	ADXL initialization (left) and measurement (right) flow graphs	60
Figure 4.27	ADXL345 probe	61
Figure 4.28	ADXL345 probe - detail	61
Figure 4.29	SD card pinout when in SPI mode	62
Figure 4.30	LC Studio SDcard reader schematics	62
Figure 4.31	LC Studio SD card reader	63
Figure 4.32	SD card responses frames	64
Figure 4.33	Waking up the card and setting the SPI mode	65
Figure 4.34	Acquisition of the SD card data type	65
Figure 4.35	Part of the SD card initialization sequence, CMD58	66
Figure 4.36	Decoding of data saved from oscilloscope containing a string .	67
Figure 4.37	File initialization (left) and write to the file (right) flow graph	68
Figure 4.38	DS1307 connection	69
Figure 4.39	DS1307 request for register reading	70
Figure 4.40	DS1307 time data acquired	71
Figure 4.41	SSD2119 driver initialization	73
Figure 4.42	Flow diagram of displaying a single pixel	74
Figure 4.43	Resistive touch screen	76
Figure 4.44	STMPE811 Init	77
Figure 4.45	STMPE811 calibration	77
Figure 4.46	Reading touch position flow graph	78
Figure 4.47	TCP segment	80
Figure 4.48	ENC28J60 Ethernet controller break-out board	82
Figure 4.49	ENC28J60 interface	82
Figure 4.50	ENC28J60 communication flow graph	83
Figure 4.51	Web server created with ENC28J60	84
Figure 4.52	PCB bottom layer	85
Figure 4.53	PCB design	85
Figure 4.54	LM317 in adjustable regulator mode	86
Figure 4.55	PCB bottom layer	88
Figure 4.56	PCB top layer	88
Figure 4.57	PCB design	88
Figure 4.58	Switch and jack schematics + PCB design	88
Figure 4.59	STM32F407 pinout	89
Figure 4.60	Intilization (left) and preparation (right) phase flow graph . .	90
Figure 4.61	LCD interface pages	91
Figure 4.62	Position of individual elements in the box	92
Figure 4.63	uLite with the funnel	93
Figure 4.64	uLite side view	93
Figure 4.65	uLite top view	94
Figure 4.66	uLite side view 3	94
Figure 4.67	Connector detail	95
Figure 4.68	SD card detail	95
Figure 4.69	Power connection detail	95

Figure 4.70	Funnel	96
Figure 4.71	uLite inside wiring and the connection guide	96
Figure 5.1	uLite measurement phase	97
Figure 5.2	uLite measurement phase 2	97
Figure D.1	Example of the file on the SD card	102
Figure E.1	ASCII table	103
Figure G.1	Base board schema	105
Figure H.1	Power module scheme	106
Figure I.1	STM32F407 clock configuration	107
Figure J.1	uLite Connection Guide	108
Figure K.1	Flow graph of the measuring phase	109
Figure M.1	SD card responses frames	112
Figure N.1	ENC28J60 SPI communication (detail)	113
Figure N.2	ENC28J60 SPI communication	113

List of Tables

3.1	SPI modes	30
4.1	SHT21 temperature and humidity measurement parameters [16] . . .	35
4.2	Communication between DS18B20 and microcontroller	40
4.3	DS18B20 temperature register	41
4.4	AMS 5915-0350-D-B sensor specifications [2]	53
4.5	AMS 5915 data register	53
4.6	ADXL345 data registers	59
4.7	DS1307 registers [20]	70
4.8	STM32F4DIS-LCD display specifications	72
4.9	The functions of 8080 parallel interface	72
4.10	Fields in TCP header [36]	81
4.11	SPI instruction set for the ENC28J60 [35]	84
C.1	Commands for SD card initialization [48]	101

List of abbreviations

ABS	Akrylonitril-Butadien-Styren. 91
ADC	Analog-to-digital Converter. 24, 42, 45, 47, 49, 75, 76
ARP	Address Resolution Protocol. 81, 83
BCD	Binary Coded Decimal. 70
CAN	Controlled Area Network. 27
CO	Carbon Monoxide. 20, 47, 48
CPS	Cyber-physical Systems. 16–19
CRC	Cyclic Redundancy Check. 35, 63, 64, 83
DMA	Direct Memory Access. 82
ECN	Explicit Congestion Notification. 81
EEPROM	Electrically Erasable Programmable Read-only Memory. 27
FAT	File Allocation Table. 62, 65
FIFO	First In First Out. 77
FSO	Full Scale Output. 52
GPIO	General Purpose Input/Output. 82
GPL	General Public Licence. 22
HTML	HyperText Markup Language. 79
IC	Integrated Circuit. 25, 26, 30
IDE	Integrated Development Environment. 22
IIoT	Industrial Internet of Things. 16
IoT	Internet of Things. 16, 18, 19
IP	Internet Protocol. 78, 79
IRED	Infrared Emitting Diode. 42–44
IT	Information Technology. 14
LCD	Liquid Crystal Display. 27, 33, 36, 71–73, 75, 76, 98
LGA	Land Grid Array. 55
LSB	Least Significant Bit. 35, 40
M2M	Machine-to-machine. 16
MAC	Medium Access Controller. 79
MISO	Master In Slave Out. 27–29, 63
MMC	Multi Media Card. 27
MOSI	Master Out Slave In. 27–29, 63, 85

MSB Most Significant Bit. 35, 40, 53
OPC Open Platform Communications. 18
PCB Printed Circuit Board. 27, 30, 85, 87, 98
PET Polyethylene Terephthalate. 74, 75
PHY Physical Layer. 79, 83
RTC Real Time Clock. 67
SCL Synchronous Clock. 25, 26
SD Secure Digital. 7, 10–12, 27, 29, 60–66, 101, 112
SDA Synchronous Data. 25, 26
SDIO Secure Digital Input Output. 60, 61
SPI Serial Peripheral Interface. 9, 10, 12, 22, 27–30, 56, 60, 61, 63, 64, 72, 75, 81
SQL Structured Query Language. 98
SRAM Static Random Access Memory. 67
SS Slave Select. 27, 28
TCP Transmission Control Protocol. 79
TFT Thin Film Transistor. 7, 72, 74
TSC Touch Screen Controller. 75
UART Universal Synchronous/Asynchronous Receiver and Transmitter. 27
UDP User Datagram Protocol. 81
USB Universal Serial Bus. 27, 86
VOC Volatile Organic Compounds. 6, 20, 45, 46, 85

1 Introduction

Industrial production has changed significantly in last 300 years. Three significant industrial revolutions took place (figure 1.1). It has started with production mechanization using water and steam power in 18th century, followed by second industrial revolution in which had electricity boosted the industrial processes. The third industrial revolution brought the use of IT and automation technology to the manufacturing process. The progress is unstoppable and it was just a matter of time before next step in the industry process development comes.

The next industrial revolution is called Industry 4.0. For now Industry 4.0 doesn't bring anything so revolutionary technology-wise. Most of the technology is already around us in our daily lives and it has been for quite some time. The mobile phones, tablets and laptops with touchscreens connected to Internet 24/7 sharing data on cloud storage is today's standard. The 3D printer is already part of home workshops and autonomous robots are able to play football[†], piano [5] or chess [15]. The technologies have gotten reliable enough and became price reasonable, which allow their use in manufacturing processes. According to Moore's law [11] is the speed of electronics development increasing and so it is expected that more technology nowadays considered "futuristic" will be used in manufacturing process on daily basis.

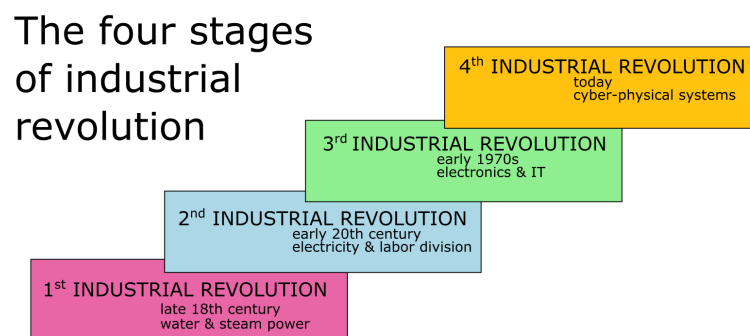


Figure 1.1: Industrial revolutions

In the last few years there has been a growing interest in the fourth industry revolution. The companies which are able to react to this new concept have change to improve the manufacturing process from all points of view. The optimization of manufacturing process brings faster production with preserved or increased accuracy

[†] www.robocup2015.org

together with very little waste of resources. The manufactures who decide to follow the Industry 4.0 ideas will have to make adjustments in the processes. If the process requires filtration solution, the choice falls on the filtration unit which follows the Industry 4.0 conventions.

ULT AG offers extraction and filtration technologies for various production processes. The offered compact and modular units are perfect basis for each filtration or extraction solution. The company also offers to modify and adapt the units to perfectly fit the costumer needs. The possibility of providing Industry 4.0 solutions would bring new opportunities to both ULT AG company and their customers.

One of the tasks of this thesis is to show ideas following Industry 4.0 which are applicable in the fume extraction industry. Those ideas are based both on the research on Industry 4.0 and on some of the competition companies solutions.

The task of the practical part of this thesis is to take the proposals and create a prototype unit connectible to the filtration unit. This unit is named uLite and it is able to collect various data, save them to memory medium and display results. The implementation of Ethernet interface is prepared both hardware-wise and software-wise. The core is based on ARM[®] Cortex[®]-M4 technology.

The remainder of this paper is organized as follows. In chapter 2 Industry 4.0 is introduced in details and there are various applications in the fume exhaust industry presented. The following chapter 3 contains theory of basic embedded development. This chapter is supporting the chapter 4 in which is the prototype unit development described in detail. Following Figure 5 contains the presentation of the final form of the unit. In chapter 6 there are conclusions and directions for future development.

2 Industry 4.0

Industry 4.0, also called the fourth industrial revolution, is a term describing today's changes in industry involving data exchange, contemporary automation and technologies. This term represents combination of industry, Internet of Things (IoT), Internet of Service (IoS) and cyber-physical systems.

2.1 Internet of Things

The Internet of Things is defined as a network of physical objects (embedded with electronics, sensors and network possibility) which can exchange information. This allows the objects to be sensed and controlled from a remote location. The integration of real world into the virtual reality brings better accuracy, efficiency and control. All of those factors contribute to the economic benefits. The “thing” in Internet of Things can be imagined as a vehicle with build-in sensors, biochips inside humans or animals controlling the life functions, intelligent thermostats, washers with wi-fi remote or even a whole building or cities. The “thing” can also be a production machine in a factory.

The IoT has been around for a while but in last years it is finding its way to industry production due to generally fast development of electronics. The use of IoT in industry gave rise to IIoT (Industrial Internet of Things). It extends the standard M2M (Machine-to-Machine) communication. When using the M2M the captured data are transmitted across a local area networks to trigger specific action or alert. The IoT includes properties and abilities of M2M but adds the significant value of interconnecting systems, customers, manufactures or whole enterprises over the Internet.

According to Maarten Botterman [10], the prediction for next 15 years is, that tens of billions devices will be wirelessly connected to the IoT. This will be possible only if the IPv6 protocol is implemented due to limited address space of currently used IPv4 protocol (“only” 4.3 billion unique IP addresses [17]).

2.2 Cyber-physical systems

The CPS (Cyber-physical Systems) are created by combination of physical components and embedded systems. The embedded systems provide computational power for algorithms and connectivity. Together they form a adaptable, resilient and safe

system which is a key for future technology development. The CPS are able to communicate and work with each other and with humans as well.

The CPS technology can bring significant improvement and optimization to every process. As stated by Jay Lee et al. in [26], the development is more or less still in the initial stage. The structure and methodology was defined to create a clean guideline for implementation of CPS in industry [31]. The structure is called 5C. The 5C architecture contains five necessary steps to fully implement CPS into the industrial process described in [50] by Dominick Vanthienen et al.:

- Connection
- Conversion
- Cyber
- Cognition
- Configuration

The 5C architecture can be visualized as a pyramid-shape diagram representing the data path. The data which are passed higher are reduced in size while their information value rises [30]. This is illustrated in figure 2.1 which was recreated based on illustration in an article written by Jay Lee et al. [30].

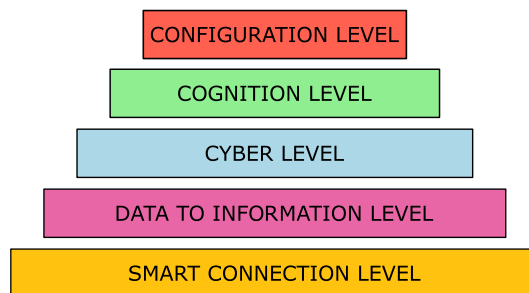


Figure 2.1: 5C Architecture

The data are collected in the first connection layer. In the second, conversion level, the data are processed and converted to information. As an example can serve collecting data on vibration of a drive which was described in [30] by Jay Lee et al. The data itself do not carry any information about the state of the machine but the algorithm can extract the status of the drive from the data. It then sends the status to the third layer. The cyber layer performs additional analysis. It can identify patterns in series of data using e.g. machine learning algorithms. Compared to the previous stage, the cyber level uses data collected from more devices to compare and add additional value to data. In cognition level the device can diagnose its own failures (or potential failures) and becomes aware of its own state [50]. In the last configuration layer is the machine able to track the state of its health. The machine itself can adjust its working load system. The result is a resilient system which is able to defend itself from failures [26]. Operators and factory managers can make informed decisions based on the data from the CPS.

This brings a new step in the maintenance model. At first there was the reactive model. In other words: wait for a machine breakdown and then repair it. Predictive maintenance model has brought improvement to this process. The machines are checked after a specific amount of time or cycle units and each time are the minor problems eliminated, preventing the machine breakdown. The ultimate goal is to implement predictive maintenance model. The machine is able to track its own health, can detect failures early on and send health monitoring information to the operation level. The maintenance is targeted and precisely planned to save both resources and human power.

2.3 Industry 4.0 basics

Industry 4.0 has several definitions and no exact steps for implementation of Industry 4.0 are given. However, there exist six principles to support and guide companies in identifying possible Industry 4.0 pilots which then can be implemented (described by Mario Hermann et al in [23] and by Hartmut Rauen in [37]).

- Interoperability
- Virtualization
- Decentralization
- Real-Time Capability
- Service Orientation
- Modularity

Interoperability

The connection of cyber-physical systems, humans and companies is one of the fundamental principles. All components should be able to communicate, exchange data and operate with each other through IoT. Holger Junker has in [27] interesting idea. According to the article, the ideal situation would be if all devices and services were able to communicate independently with one another, irrespective of manufacturer, operating system, hierarchy and topology. This is at some point already possible with use of e.g. OPC Unified Architecture, which is nowadays the preferred communication standard for Industry 4.0 [18]. As David Greenfield stated in [22], the OPC Foundation efforts to standardize interoperability are not new, but the importance of this step is increasing as the possibilities of Industry 4.0 and the Internet of Things are taken more seriously.

Virtualization

Cyber-physical systems can monitor the actual physical process. The data will be linked to the virtual model created by simulations. This link creates a virtual copy of the physical world.

Decentralization

When different systems within the company are allowed to make their own decision the overall response is faster. Cyber-physical systems are endowed with embedded computers capable of making the individual decisions. In case of any failure or warning, a report is sent to a higher level. Therefore, central planning and controlling is no longer needed (Schlick et al. [25]).

Real-Time Capability

Industry 4.0 makes a concentrated effort to gather the data in each process step and offers real-time feedback and monitoring. All the process parameters are tracked and processed. In case of a failure the plant can reroute products to another machine without stopping the manufacturing process.

Service Orientation

The services of companies, CPS and humans are available over the Internet of Services. They can be offered both internally or between companies.

Modularity

The importance of the ability to flexibly react to a change of requirements without too much effort is valuable. Modular systems enables easy adjustment in case of seasonal fluctuations or changed product characteristics (Hermann [23]). The individual modules are designed in such way that improvement, replacement or expansion is possible.

The vision of Industry 4.0 is a “Smart Factory”. The cyber-physical systems monitor manufacturing processes, collect data, create a virtual model of the whole factory and make decentralized decisions. Using IoT, the individual systems communicate with each other and also with humans.

2.4 Industry 4.0 in air-filtration

This chapter suggests Industry 4.0 ideas applicable for air filtration industry. At first sight, fume extraction units seem to be devices with straightforward function, but there exist a lot of ways how to extend their possibilities with use of different sensors and Internet connection. When proposing any idea, the main focus should always stay on the customer. The final product should be not only innovative, but also useful and reasonably priced. The proposed ideas are divided into three logical parts.

- Filter state and air quality monitoring
- Mechanical state of the filter unit
- Filter unit electrical properties

At the end of this chapter is the basic concept of uLite unit introduced.

2.4.1 Filter state and air quality monitoring

The most noticeable issue with the fume exhaust unit run is the necessity to regularly change filters. The question arises: what does regularly mean in this context? That strongly depends on not only what kind of process is the unit connected to but also on the operation time of the filter unit. The most primitive solution is to have a maintenance worker to perform a routine filter checks. The more automatized solution is monitoring of pressure drop in the filtration unit. This information is usable for determining the state of the filter. After the logic inside of the unit decides, that the filter is ready to be changed, a signal light on the unit is lid and the maintenance worker can change the filter. This approach has some drawback. One of them is stopping the filter unit while the production is running which can be in some cases costly.

Moving this approach one step forward, the unit can continuously observe the state of the filter and warn the user before the filter is clogged. The customer can order new filters in time and change the filter in convenient time. If the unit is connected to the Internet it can order new filters by itself or the company selling filtration units can observe the state of each filter and unit online and deliver the new filters to customer when needed.

The cartridge filter units with self cleaning filters don't require the filter change but a bin for collecting particles has to be regularly emptied. It is possible to use weight cells to obtain the bin mass. Emptying of the bin can be planned before its weight reaches the law given limits for lifting weights (the bin can be emptied without use of e.g. fork lift). If the collected data are fed to learning algorithm, the prediction of the next bin emptying can be calculated.

Another way how to tell if the filter (or rather whole fume exhaust unit) is working properly is to measure the air properties at the outlet of the unit. This brings safe assurance to the customer. Furthermore the detection of dust particles can detect smoke in the unit and together with values from temperature sensors it can trigger possible fire warning. The detection of combustive gases can prevent hazardous situations. A lot more sensors to monitor air quality can be deployed (humidity, CO, VOC etc.).

2.4.2 Mechanical state of the filter unit

To prevent damage and unexpected behaviour of the fume exhaust unit it is possible to employ sensors to detect if everything is working well from the mechanical point of view. The reed sensors are able to check if the doors of the unit are closed, tensometers or optical sensors can detect the presence of the filters. The unit shall not be started when one of the previous mentioned conditions is not met. The pressure sensors may help with identifying both open doors and missing filter situation as well as they can detect blocked hose on the filter unit inlet. The vibration of the blower drive may be measured using accelerometer. Increased vibration could produce unwanted noise and can show possible bearing or drive problems.

2.4.3 Filter unit electrical properties

Another way how to look at the filtration unit is from the electrical point of view. Data about drawn current and voltage can be processed and total power consumption can be calculated. The drawn current measurement can be used to detect possible current spikes which can mean drive malfunction. Along with current and voltage measurement a phase shift could be evaluated.

2.4.4 Fume exhaust unit control box

For the practical part of this master thesis a prototype of an unit connectible to the filtration unit is designed. The unit was named uLite (ULT Air Quality Module). This unit is able to measure, collect, process, save and display data (figure 2.2). The sharing of the data is ready to be implemented.

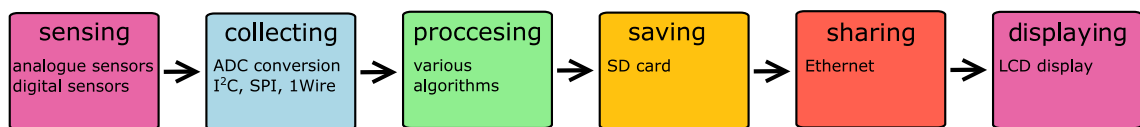


Figure 2.2: Prototype unit requirements

The motivation for creation uLite is to ensure that the quality of the filtrated air is acceptable from both costumer preferences and law regulations. The task of this box is not to measure exact values of gasses concentrations in the air but rather to ensure that the concentration is kept at minimal levels.

Furthermore the unit can be used to verify the active carbon filter. It is possible to determine the necessity to change the particle filter thanks to the increased pressure drop, but the active carbon filter has no such indicator. Additionally the unit can determine parameters of the polluted air entering the unit as arbitrary source of air can be brought to the unit inlet.

The unit is able to measure dust density as well as volatile particles, carbon monoxide and combustive gases concentrations. It is also capable of measuring temperature and humidity of the air on the unit inlet. Moreover there are pressure sensors for measuring pressure difference drops on two measuring points, accelerometer probe with magnet which can be attached to magnetic surface measuring vibrations and waterproof temperature probe are available.

3 Embedded development introduction

Development of embedded software applications requires a lot of both hardware and software theory knowledge. In this chapter there is a brief overview of hardware and software resources used for development followed by the theory which supports the next part of this thesis.

3.1 Used resources

3.1.1 Programming IDE

System Workbench for STM32 based on Eclipse IDE for C/C++ Developers in version Luna is used for the programming of the STM32F4 microcontroller. This program provides a software development platform for all STM32 microcontrollers. It already includes STM32 devices databases and libraries, source editor, linker, building tools (GCC-based cross compiler), debugging tools (OpenOCD) and flash programming tool. This IDE is free for commercial use, as long as the target hardware is based on an STM32 microcontroller and contains a few GPL-licensed components like compiler or debugger.

For the design of pinout, clock tree setting and peripherals setting STM32CubeMX software tool is used. This software works with STM32Cube HAL drivers, which are meant to ease the program migration across STM32 portfolio.

3.1.2 Debugging

Seleae 8-channel logic analyser and Seleae logic software version 1.1.15 are used to debug and decode the communication on different buses (I²C, SPI, 1-wire ...).

RIGOL DS1054 oscilloscope serves as an extended analyser of the signals. Unless stated otherwise, all graphs displaying signals are created from data collected by this oscilloscope. In this thesis, there are also screenshots of the Rigol screen to demonstrate the authentic bus decoding or signal time properties. For the communication, remote programming and advanced data logging are Ultra Sigma PC connectivity tool and Ultra Scope used (both freeware).

3.2 Hardware requirements for interfacing ICs

3.2.1 Pull-up and pull-down resistors

Output pins on microcontrollers fall into two categories: open drain (open collector) or push-pull. Push-pull output has two transistors (figure 3.1 on the right). Each has the capability of driving the output to the appropriate level. To make the output high, the top PNP transistor is on. On the other hand, when the output is set to low level the NPN transistor is on.

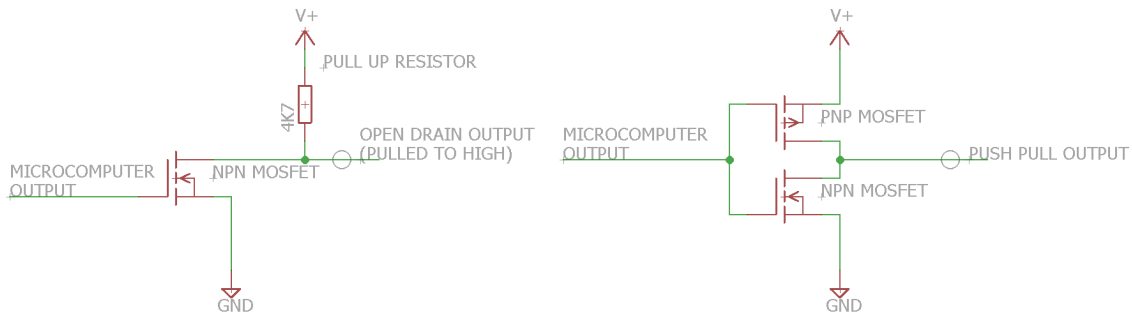


Figure 3.1: Open drain (left) and push-pull output (right) comparison

The open drain output lacks the PNP transistor driving the line to the high level (figure 3.1 on the left). To return the line into logical high state when the transistor is off, the pull-up resistor is deployed. If the line was not pulled by the resistor up or down, the pin would be left in floating state. This state is mostly not acceptable as the microcontroller might interpret the input value as either logical low or logical high. Microcontrollers have usually in-built programmable pull up/down resistors but it is not possible to choose their values. Hence, better results are obtained when using external components.

Every bus has its own capacitance including all pins, connection, wires or PCB traces capacitances. This capacitance varies with length of the bus and in most cases it is an unknown variable. Together with the pull-up resistor, the RC constant limits the bus capacitance rise time. If the selected pull-up resistor is too high, the line may not rise to the logical high level before it is pulled down again. With increasing number of devices on one line the capacitance increases and a lower value pull-up resistor should be considered. This leads to the fact that there is no universal value of pull-up resistor.

However, for most application is value between $3\text{ k}\Omega$ and $10\text{ k}\Omega$ sufficient. Power sensitive circuits may require higher values (up to $100\text{ k}\Omega$). On the other hand in speed sensitive circuit low values of pull up/down resistor are used ($1\text{ k}\Omega$ or even less). The lower value resistor increases current draw and negatively impact power consumption. The mostly used pull up/down resistor value is $4.7\text{ k}\Omega$.

The disadvantage of open drain output, or rather the advantage of push-pull output, is speed of the switching thus a higher frequency capability. The line is

driven both way whereas the line using open drain output can rise as fast as the RC constant allows. Push-pull can also provide more current.

3.2.2 Decoupling capacitors

The supply voltage is rarely stable in practice. Instead there are glitches, AC components or spikes. When analogue circuit is connected to the unstable power source, hum or cracking noise can be produced unintentionally. Digital circuits can be unstable and unpredictable. Decoupling capacitors are used to filter the power supply voltage and make it more stable. If voltage spike occurs, the capacitor absorbs the excess energy. In case of the sudden input voltage drops, the capacitor provides energy to make the voltage stable.

The method of decoupling is dependent on the frequency used in the circuit. The general recommendation for circuits with less than 50 MHz is to have one ceramic bypass capacitor between power supply pin and ground for each integrated circuit on the board as close to the IC as possible. The usual values are 0.1 μF or 0.01 μF . A larger electrolytic capacitor (up to hundreds μF) should be placed on the board as well.

Decoupling and grounding in high speed circuits is a complicated process on which many books (*High-Speed Digital Circuit* by Masakazu Shoji [42] or *Robust Electronic Design* by John R. Barnes [6]) has been written and it will not be discussed in this thesis.

The decoupling capacitors values and types are often recommended in the device datasheet.

3.2.3 Analogue sensors interfacing techniques

Analogue sensors produce variable output which can be voltage, current or resistance. In microcontroller systems output is usually converted to voltage in a suitable range for ADC (Analogue-to-digital Converter).

The inputs of ADC can be damaged by voltage signals higher than the specified ADC voltage range (in case of wrong design, circuit malfunction or power up/down phase). Most ADC inputs have internal diodes which start conduction in case that the voltage goes too high but the diodes are not designed to withstand large current for a longer period of time. The diodes are placed inside of the microcontroller and it is impossible to repair them. Therefore it is better to use another protection.

The simplest way of protecting inputs is use of operational amplifier. Operational amplifier gets saturated when its output gets to the supply rails. When the supply voltage is not higher than the maximal allowed input to the ADC, the operational amplifier is efficiently protecting ADC inputs. The operational amplifier also brings advantages to the analogue sensor interfacing circuit. The high impedance of the operational amplifier prevents current loading of the sensors. This forestalls heating of the sensors which can in some cases cause inaccuracy (e.g. when reading temperature).

If the output range of the analogue sensor is already suitable for the ADC converter is possible to use the voltage follower connection (figure 3.2 on the left). Because of the lack of feedback resistors the operational amplifier returns the same signal that is fed in. In case that the output range the analogue sensor has to be adjusted (scaled up/down), the connection as non-inverting amplifier (figure 3.2 on the right) is used. Equation 3.1 represents the voltage gain of such connection.

$$\frac{U_{out}}{U_{in}} = 1 + \frac{R_2}{R_1} \quad (3.1)$$

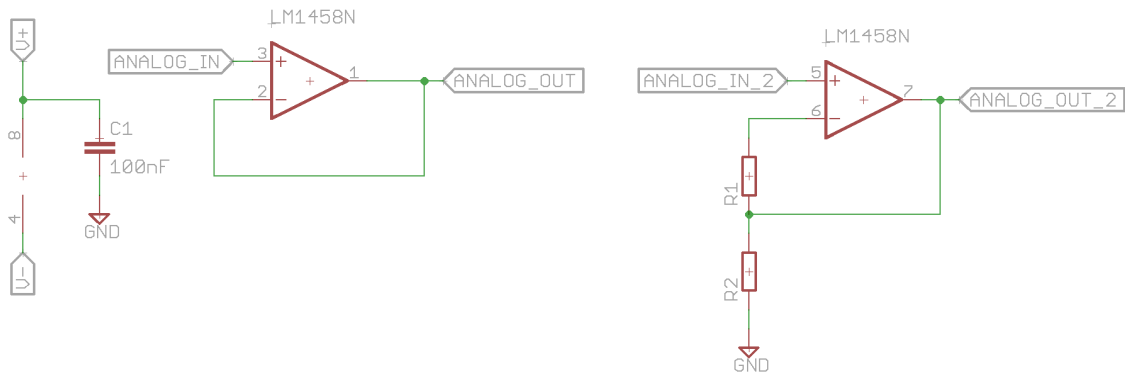


Figure 3.2: Voltage follower (left) and non-inverting amplifier mode (right)

3.3 Used communication buses

3.3.1 I²C

I²C bus was designed by Phillips originally for easy communication between integrated circuit and the same circuit board. I²C is short distance, serial communication and multi-master bus protocol. Nowadays it is not used only for IC communication on the same board but it also serves as connection to low-speed peripherals (which sensors usually are).

I²C topology

The line uses two wires for data transmission, one serial data line SDA and one clock line SCL. The slaves are connected in series and there can be arbitrary number of masters present on the line. The I²C topology is shown on figure 3.3.

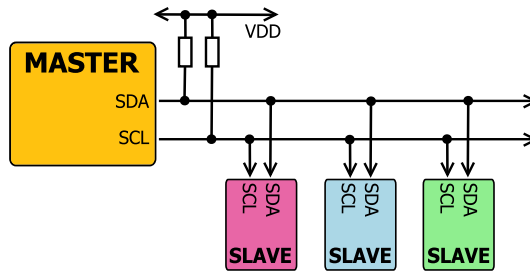


Figure 3.3: I²C topology

I²C data transmission

The communication is based on addressing individual devices. The basic design has 7-bit long address for each device with 16 reserved addresses. That leads to $2^7 - 16 = 112$ possible devices connected on one bus. Every device on the bus must have its own address. Some of the sold devices do not have an option to manually set the address and if there is a need of use more devices with the same address on one I²C bus a multiplexer has to be used (e.g. TCA9544A 4-Channel I²C and SMBus Multiplexer With Interrupt Logic). Each of the device connected to the line can be receiving or transmitting data.

The communication is initialized by the master. Before the master starts transmitting, the bus must be free (both SDA and SCL in logical high state). The master sends start condition which makes all the devices on the line listen on the data line for instructions. The start condition consists of pulling the SDA line low when the SCL is high. The master then provides clock on the SCL line. The data are valid as the clock pulse goes from low to high. The master sends a unique address of the device together with one byte informing the device whether the master wants to send or receive data from the addressed IC (1:read, 0:write). All the slave devices compare the send address with their own. If it doesn't match, they wait until the bus is released again. In case the device address matches, the IC sends back acknowledgement bit. After the master receives the acknowledgement bit it starts transmitting or receiving data. The communication proceeds until the master issues a stop condition (setting SDA line high when SCL is high) to make the bus free. To see frame of I²C communication protocol please refer to figure 3.4.

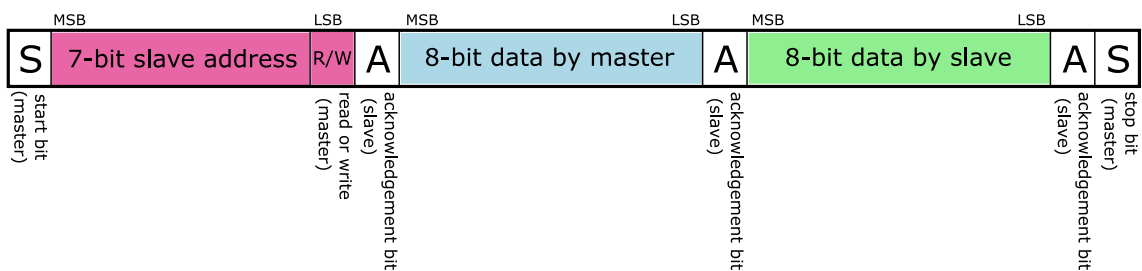


Figure 3.4: I²C protocol

The concrete example of the I²C communication can be found in section 4.1.1 (reading of the SHT21 sensor).

I²C hardware requirements

The I²C uses open drain connection, which means that the driver can pull the signal line low but it cannot drive it high so the pull-up resistor is required (discussed in section 3.2.1). If not said otherwise the ICs using I²C communication usually require decoupling capacitors across the power supply. The use of decoupling capacitors was covered in section 3.2.2.

The maximal length of the bus depends on its capacitance. According to the bus specification [41], the maximal capacitance for fast mode should not exceed 200 pF but it can go up 400 pF in normal mode.

I²C advantages and disadvantages

I²C maintains very long pin and signal count even when there are numerous devices connected on the bus. It supports multiple masters and incorporates acknowledge bits to confirm successful communication. On the other hand it increases the complexity of firmware and it requires pull-up resistors which are slowing the bus down, consume space on PCB and increase power dissipation. It can also be considered rather slow (standart speed I²C is 400 kHz, high-speed I²C operates at 1 MHz).

3.3.2 Serial Peripheral Interface

SPI (Serial Peripheral Interface) is a synchronous serial data interface developed by Motorola. The source of synchronization is the system clock generated by master. SPI can operate in full-duplex or half-duplex mode. Full-duplex communication between two devices means that both of them can send and receive data simultaneously. Half-duplex communication allows only one device at a time to transmit the data while the other device has to listen. SPI can transmit data with high speed (up to 100 MHz) over short distances and it is used for talking to a variety peripherals such as MMC or SD cards, LCD displays, flash and EEPROM memories, real-time clocks modules, Ethernet, USB, UART, CAN and many more. SPI has no formal definition, only a set of guidelines to follow. The concrete details for communicating with device using SPI are to be found in the device datasheet.

SPI topology

SPI is a single master protocol which allows multiple slaves connections. The traditional SPI topology and a cascade slave configuration are shown on figure 3.5. In case of tradition connection each slave has its own SS (slave select) line and it is connected to the master via two communication lines MISO (Master In Slave Out) and MOSI (Master Out Slave In). In case of cascade configuration all SS lines are connected together. Master output goes to the slave input which sends it to the next slave. The last connected slave sends the data back to the master. This way the data flows out of the master through each slave and back to the master. The configuration is sometimes referred as daisy-chained configuration. The advantage of this reduction of chip select pins and the different connection of MISO/MOSI lines

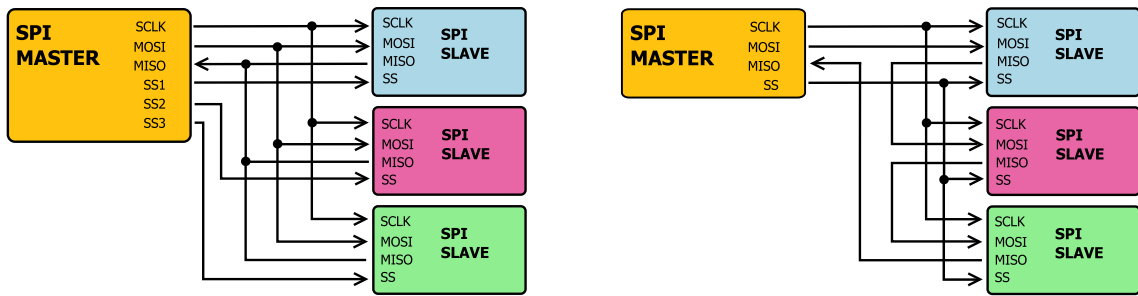


Figure 3.5: classical SPI topology (left) and cascade configuration (right)

can significantly reduce complexity of large systems. There is no complex system which requires SPI connection in this thesis, therefore there is only the traditional SPI topology used.

In the classical configuration the master can communicate with one slave at the time by use of the SS line. This line is usually active low, but it is always necessary to check this condition for specific device. SPI is called four-wire bus, but with increasing number of slaves, the number of needed SS lines arises. The line naming can be different for each device and the alternative names are listed in the bracket.

- SCLK (SPSCK,SCK): Serial Clock (output from master)
- MOSI (SDI): Master Output, Slave Input (output from master)
- MISO (SDO): Master In, Slave Out (output from slave)
- SS (CS,CE): Slave Select (active low/high)

SPI data transmission

There exist 4 different types of SPI clocking set by parameters CPHA (clock phase) and CPOL (clock polarity). The slave only accept the bit if there is a falling edge (CPHA = 1) or rising edge (CPHA = 0) of the clock signal detected. The two additional states are determined by the idle state of the clock (CPOL = 0 for active state 1 and CPOL = 1 for active state 0). The four SPI clocking types for PIC and ARM-based microcontrollers are usually referred as SPI Mode 0-3 (table 3.1). For the timing diagram please refer to figure 3.6 (figure is created based on the information in the SPI specification [24]).

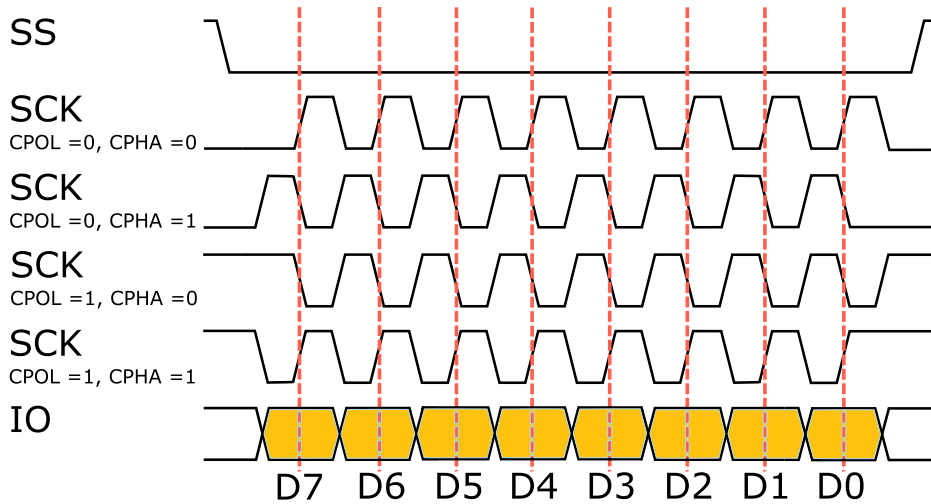


Figure 3.6: SPI Timing

Table 3.1: SPI modes

SPI MODE	Clock Phase	Clock Edge
0	0	1
1	0	0
2	1	1
3	1	0

The communication is always initialized by the master. The master configures the clock with frequency less or equal the device's maximal frequency. Then by use of SS pin selects a desired device. The devices which are not selected ignore the clock and MOSI signal and produce no MISO signal. The data transfer is organized by using shift registers with fixed word size (e.g. 8-bit, but it is not limited to only to this size). The master and slave shift registers are connected in a ring. The master shifts the register to MOSI line and slave shifts its register to MISO line. The process is repeated as long as there are data left to be send. Otherwise the master stops the clock, releases the SS line and the transmission is over. The detailed example of the SPI data transmission is described in section 4.5.4 (communication with SD card).

SPI hardware requirements

Generally speaking there is no need for pull-up resistor for SPI bus, because the driver use push-pull outputs so it can reset and set the lines internally (for details please refer to section 3.2.1). It is a good practice to put a pull-up resistor to the chip select lines so the line is hold up when not in use, preventing coincidently line pulling during power up and power down phases. In some applications pull-up is recommended for every line. For example for interfacing the SD card, a pull-up resistor on MISO line will prevent floating pins when SD card is not plugged in.

Adding the pull-up resistors to the SPI lines will not cause any problems and when it comes to the final design, it is always easier not to use the resistors then try to add some. Generally the pull-up resistor increases the robustness of the design.

ICs using SPI communication protocol often require decoupling capacitors on power supply lines (as explained in section 3.2.2).

The maximal length of SPI bus depends on the communication speed and on the concrete application but the fact that SPI is originally designed for communication of two ICs on a single PCB should be taken in account. The same restriction is valid for I²C, but the effect is weaker thanks to the significantly lower communication speed.

SPI advantages and disadvantages

SPI offers high speed full-duplex communication with word size up to 32-bits. SPI is not defined by formal standards, so the choice of content, purpose and size of the transferred word is arbitrary. The slave uses master clock so there is no need for external oscillators. On the other hand, there is no slave acknowledgement and no hardware flow control. It doesn't support multi-master configuration and compared to e.g. RS-232 can only transfer on short distances.

3.3.3 1-Wire

1-Wire is a half-duplex serial protocol bus developed by Dallas Semiconductors. It provides low-speed data and power over a single wire. The distinctive feature is the possibility to use only two wires thanks to the 800 pF capacitor in each 1-Wire device. This capacitor can store charge and power the device during periods when the data line is active.

Each device has 64-bit identification number. The first 8-bit of the identification number is a family code which defines the device functionality. The identification number is factory programmed, unique and not changeable.

1-Wire hardware requirements

The bus uses open drain connection so the pull-up resistor is required on the data line (section 3.2.1). Besides, there are no special HW requirements. When using the device in parasitic mode it is better to keep the wires short or to have a look at the signal on oscilloscope and adjust the pull-up resistor value. It has to be low enough so there is sufficient current for charging parasitic devices and high enough to pull the line low. The length of the bus is limited by its capacitance, in this case it should not exceed 10 nF. If the capacitance is too high, the signal cannot change fast enough to transport data (each 1-Wire slave adds 30 pF, typical cables has around 60 pF/m).

1-Wire data transmission

Usually there is not a 1-Wire interface in microcontrollers and STM32F407 is no exception. In order to communicate it is necessary to use software based signalling. The communication always starts with a master sending a reset pulse that pulls the line low at least for $480\ \mu\text{s}$ (figure 3.7). After the master releases the line, external pull-up pulls it back to high level. If there is a device connected to the 1-Wire bus it responds after a short period of time (max $60\ \mu\text{s}$) with a presence pulse which is shorter than the initial master pulse (up to $240\ \mu\text{s}$). The communication can start after the slave releases the line.

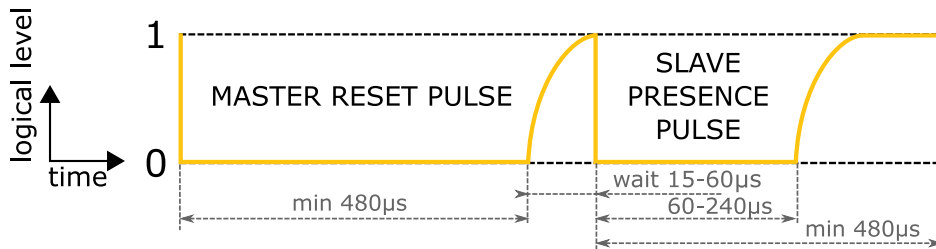


Figure 3.7: Start of communication on 1-Wire

Sending one and zero bits is done in $60\ \mu\text{s}$ time slots (figure 3.8). To write zero, the line must be pulled down for the whole time. Slave is sampling around $30\ \mu\text{s}$ and it reads zero. In order to write one, the master must pull time line down for maximum of $15\ \mu\text{s}$ and then release it. The pull-up resistor returns the line back to high level and the slave detects logical one.

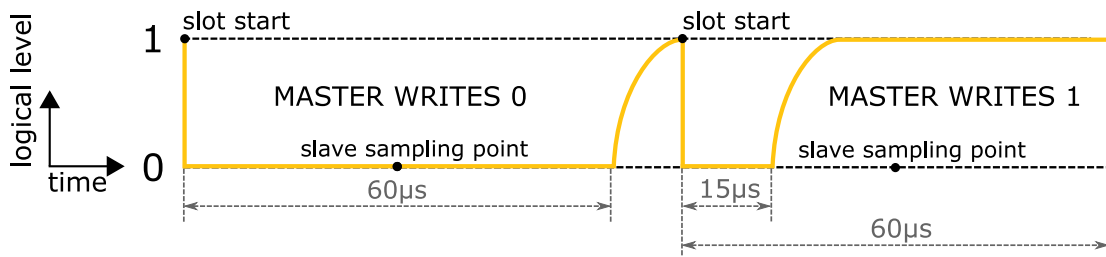


Figure 3.8: Sending 0/1 bit over 1-Wire

The reading is slightly more complicated (figure 3.9). The master has to pull down the line for about $1\ \mu\text{s}$ and then releases the line. If the slave send zero, the line stays low for the whole time slot at the master samples the logical zero. If the slave wants to transmit logical one, it doesn't pull the line down and after the master short initial pulse, the line return to logical one. The master then samples this value. The master can sample in time from $15\ \mu\text{s}$ to $60\ \mu\text{s}$. When sampling close to the lower limit it is necessary to check if the pull-up resistor has enough time to pull the line back up.

Every communication on 1-Wire bus must consist of three parts. First comes the reset, next there is a ROM command enabling addressing slaves, followed by the function command sequence.

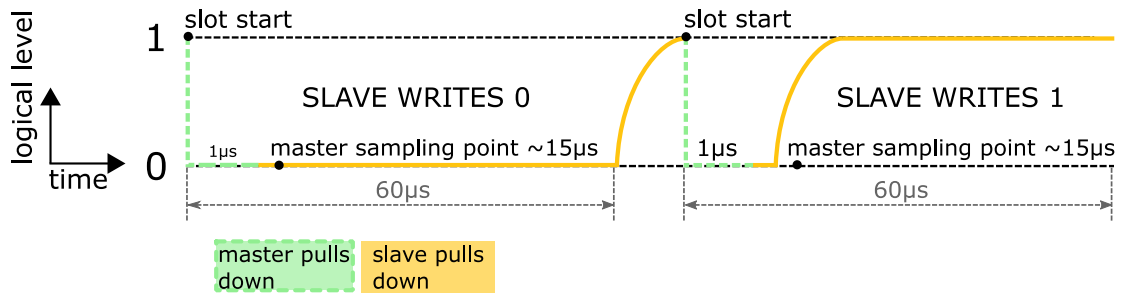


Figure 3.9: Reading 0/1 bit over 1-Wire

The detailed example of 1-Wire communication can be found in section 4.1.2 (communication with DS18B20 temperature sensor).

4 Prototype measurement box unit uLite

The practical application of this thesis is to design a unit fulfilling requirements stated in section 2.4.4 on page 22. The box is designed from the very beginning, starting with the choice of an appropriate microcontroller, platform for development, sensors and other peripherals. The program meets the needs and wishes of ULT AG company. The unit is encased in a customized project box with its own supply power solution and user interface in form of touch LCD display.

4.1 Temperature/humidity sensor choice

The temperature and humidity of filtrated air is valuable information especially when units are connected to extensive heat producing process such as soldering or welding.

Every application using sensors brings different requirements to its parameters. For simple measurement of ambient air temperature a more affordable sensor with lower accuracy is sufficient. For measuring hot air at the unit input it is important to have a sensor which is designed to measure such temperatures and preserves desired accuracy at the used range. If the temperature was a controlled variable, a high precision measurement would be required.

As the representative of relatively cheap sensor with quite high accuracy and measurement speed was chosen STH21 temperature sensor from Sensirion. The second sensor selected for the use in uLite is DS18B20 from Maxim. This sensor advantage is possibility to run without external power (less demanding interface design).

The STH21 sensor is used inside the air channel, measuring temperature and humidity of the air. The DS18B20 is designed as optional connectible waterproof temperature probe.

4.1.1 SHT21 temperature and humidity sensor

SHT21 is a temperature and humidity sensor from Sensirion [16]. The sensor is used in industry applications requiring higher temperature and humidity accuracy (table 4.1). SHT21 comes in a small Dual Flat No Leads package (3x3 mm foot print and 1 mm height), provides calibrated linearised signal which is transferred on the I²C bus. For the purpose of this master thesis, a SHT21 breakout board was purchased. This board contains already soldered SHT21 sensor and decoupling

capacitors. The reason for use of the breakout board is that the soldering of Flat No Leads package is not possible by hand as it has all contacts on the bottom of the package.

Table 4.1: SHT21 temperature and humidity measurement parameters [16]

Humidity		
	Condition	Value [RH]
Resolution	12 bit	0.04
	8 bit	0.7
Accuracy	typ	$\pm 2\%$
	max	$\pm 5\%$
Temperature		
	Condition	Value [$^{\circ}\text{C}$]
Resolution	14 bit	0.01
	12 bit	0.04
Accuracy	typ	$\pm 0.3\%$
	max	$\pm 1.5\%$

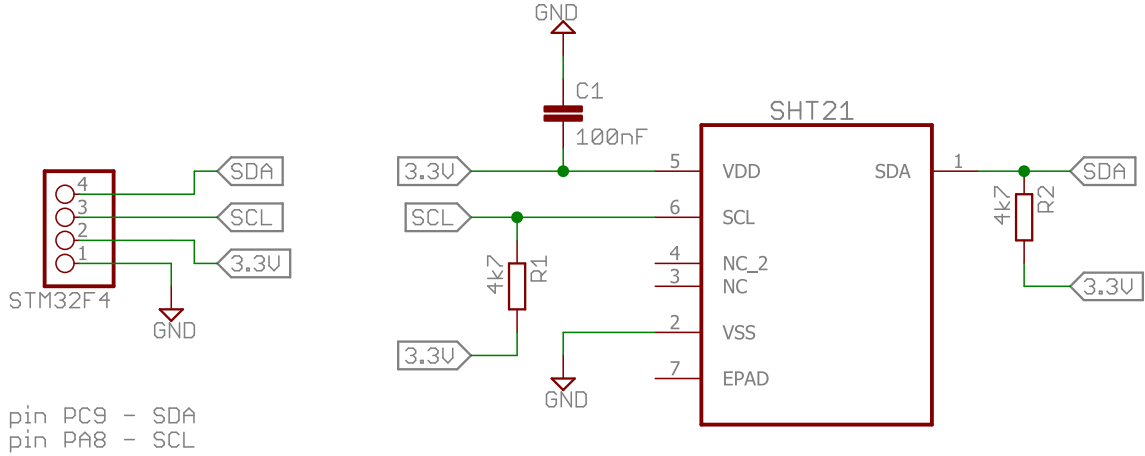
The humidity is measured with use of capacitive relative humidity sensors. They are composed of a substrate (glass, ceramic, silicon) with thin film of polymer or metal oxide between conductive electrodes. The change of dielectric constant is proportional to the relative humidity of the air surrounding the sensor. For the temperature measurement there is a silicon band-gap temperature sensor. The principle of this common form of temperature sensor is that the forward voltage of a silicon diode is temperature dependent.

SHT21 interfacing

The transfer of the data between SHT21 sensor and microcomputer is provided by I²C bus (for more information refer to section 3.3.1). In figure 4.1 there is the scheme of used connection.

SHT21 communication

The datasheet provides device address, which is not adjustable in this case. The sensor has two modes in which can run the measurement: *hold master* mode and *no hold master* mode. In the *hold master* mode is the I²C blocked during measurement and no other communication can be processed. This mode is not suitable for applications with more devices on the bus therefore not usable for the program created as a part of this thesis. In the *no hold master* mode the measurement is triggered, the line is free for the whole measurement time and master has to poll the sensor for



EPAD is internally connected to the VSS. The NC pads must be left floating.

Figure 4.1: SHT21 temperature and humidity sensor interface

the measurement results. This mode is used in the program, but the *hold master* mode is also implemented in the library for possible future use.

One temperature reading, processing of the data and calculation of the temperature follows. Figure 4.2 shows how the I²C device address (0x40)_h and a write bit (denoted as *W*) is send to trigger temperature measurement in the *hold master* mode (0xE3)_h. On following figure 4.3 is I²C device address and a read bit (denoted as *R*) send to the sensor which then returns the measured data.

The first two bytes (0x6A)_h MSB and (0x18)_h LSB contain the temperature data (last byte is CRC). Creating a 16-bit number by their combination yields result:

$$S_t = \overbrace{(0110 \ 1010)}^{(0x6A)_h} \overbrace{(0001 \ 1000)}^{(0x18)_h} = (27160)_d$$

If the two last bits are different from zero it is necessary to negate them. The calculation of temperature according to the datasheet is presented in equation 4.1.

$$T = -46.85 + 175.72 \cdot \frac{S_t}{2^{16}} = -46.85 + 175.72 \cdot \frac{27160}{2^{16}} = 25.9 \quad [^{\circ}\text{C}] \quad (4.1)$$

The acquirement of humidity has similar steps. The only difference is address of the register triggering the humidity measurement and the equation 4.2. The variable S_{rh} is calculated the same way as S_t .

$$R = -6 + 125 \cdot \frac{S_{rh}}{2^{16}} \quad [\%] \quad (4.2)$$

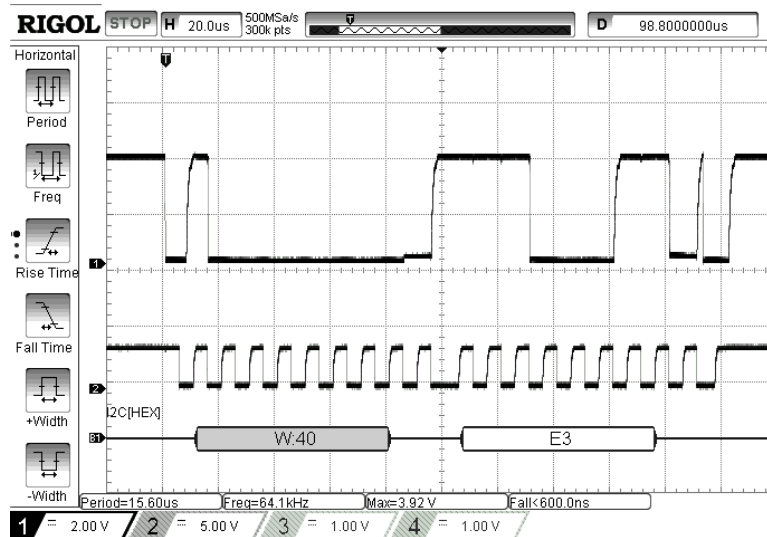


Figure 4.2: SHT21 trigger temperature measurement

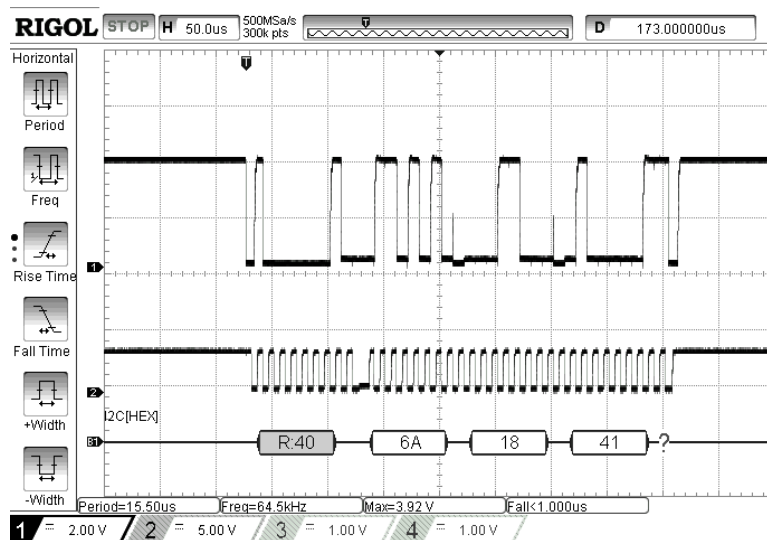


Figure 4.3: SHT21 incoming data

SHT21 library

The library contains functions to adjust all parameters of the sensor so the full potential of this sensor can be used. However, there is no opportunity to change parameters of the sensor as the user interface of the LCD display doesn't enable it. The only chance how to change the measurement parameters is directly in the program (default values of resolution are set to 13-bits for temperature measurement and 10-bits for humidity measurement). The internal heater of the sensor serves for self testing. The library also enables check of the supply voltage which is relevant for battery operated devices.

The flow graph of initialization of the sensor and temperature/humidity measure-

ment can be found on figure 4.4. To see an example of the header file please refer to appendix L. For all files, full library and for comments please see the attached CD (files *SHT21.c*, *SHT21.h* and *SHT21_systemInclude.h*).

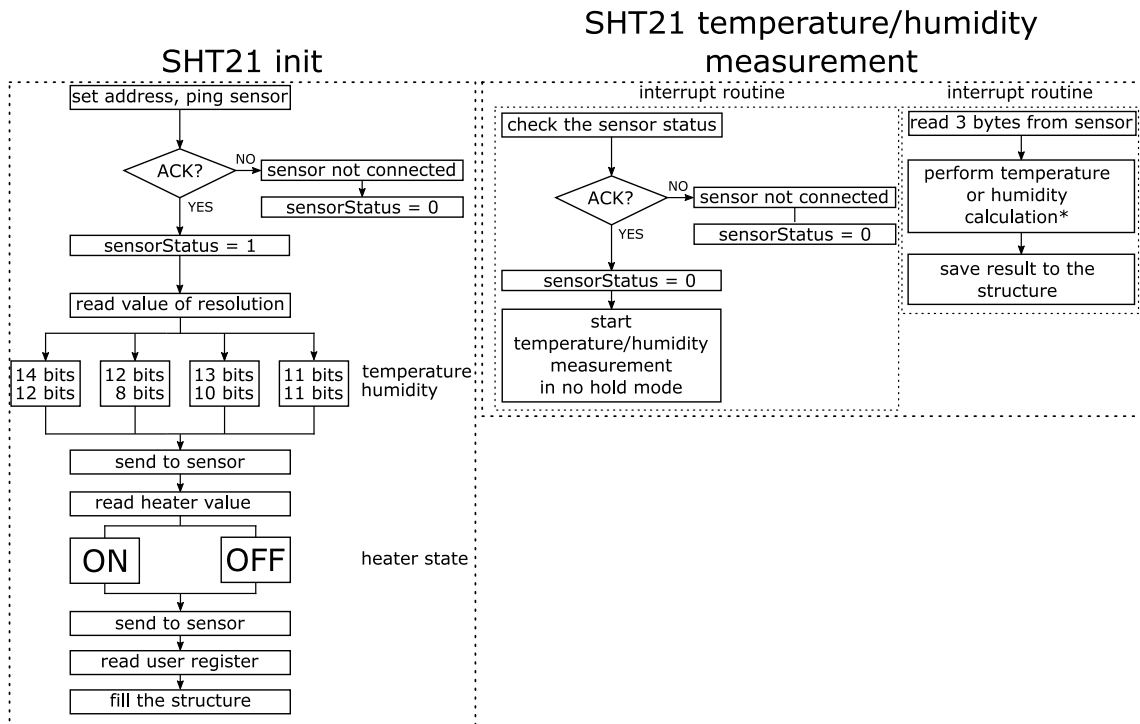


Figure 4.4: SHT21 initialization and measurement flow diagram

All the information relevant to the sensor are stored in a structure with following arrangement (code 4.1).

Code 4.1: Data holding structure for SHT sensor

```

struct SHT_parameters {
    uint16_t writeAddress;
    uint16_t readAddress;
    uint8_t resolutionTemp;
    uint8_t resolutionHum;
    // status of the heater, 0 = OFF, 1 = ON
    uint8_t heater;
    // battery status 0: VDD > 2.25V; 1: VDD < 2.25V
    uint8_t battery;
    // results of the last temperature and humidity reading
    float lastTempReading;
    float lastHumReading;
    // 1 = sensor connected, 0 = sensor connection problem
    uint8_t sensorStatus;
};
  
```

* conversion process can be found in section 4.1.1

4.1.2 DS18B20 temperature sensor

The DS18B20 digital thermometer is commonly used temperature sensor. The advantage of this sensor is the used communication bus 1-Wire designed by Dallas Semiconductors. This bus enables to connect devices with use of two wires only (data line and ground line). The device can run in parasite mode which reduces the need of external power therefore eliminating the need of external components for interfacing. Each sensor has unique 64-bit serial code, which allows multiple devices to work on one line. The sensor comes in several packages (TO-92, 8-pin SO, 8-pin μ SOP) and it is also possible to get the sensor in waterproof probe version. This probe is used as an optional external temperature sensor for uLite unit.

DS18B20 interfacing

Interfacing of the sensor requires minimum external components. The only necessary part is a pull up resistor on the data (figure 4.5). Details about use of pull-up resistors can be found in section 3.2.1.

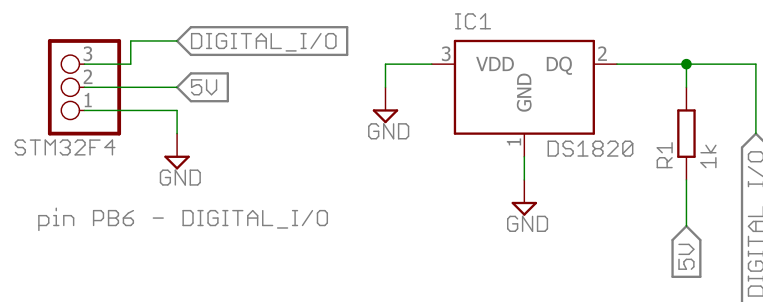


Figure 4.5: DS18B20 interface

DS18B20 communication

The basics of 1-Wire communication were introduced in the section 3.3.3. As an practical example of the communication on the 1-Wire bus, there is a start sequence caught on oscilloscope (figure 4.6). From the times displayed on the starts and ends of the pulses, it is possible to calculate their width. The reset pulse has 484 μ s, the sensor is waiting for the response for 40 μ s and the presence pulse is 114 μ s. All the measured values fall inside the definition range of 1-Wire bus.

Rigol DS1054 can't decode 1-Wire bus so the logic analyser from Saleae logic is used to capture the communication between the microcomputer and the sensor (table 4.2). The white rows are commands from master, the blue rows contains data from sensor.

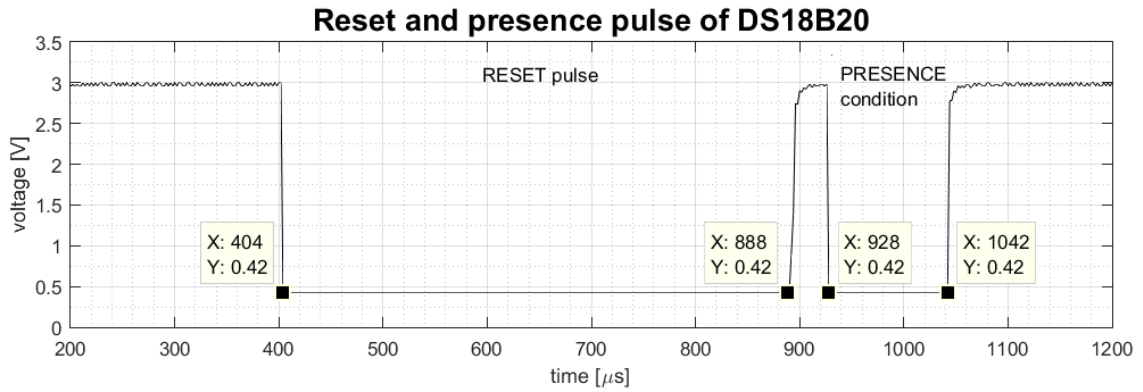


Figure 4.6: DS18B20 start sequence

Table 4.2: Communication between DS18B20 and microcontroller

Time [s]	Detail	Data
9.489004875	Reset Pulse	
9.489525375	Presence Pulse	
9.491332375	Read Rom Command	(0x33) _h
9.492534375	ROM Family Code	(0x28) _h
9.492934125	ROM Code	(0x754F1B3) _h
9.495331750	ROM CRC	(0x87) _h
9.498053125	Reset Pulse	
9.498571125	Presence Pulse	
9.500379875	Match Rom Command	(0x55) _h
9.501575500	ROM Family Code	(0x28) _h
9.502785875	ROM Code	(0x754F1B3) _h
9.510001375	ROM CRC	(0x87) _h
9.511193250	Data (Trigger T measuring)	(0x44) _h
10.27150625	Reset Pulse	
10.27202037	Presence Pulse	
10.27382875	Match Rom Command	(0x55) _h
10.27502012	ROM Family Code	(0x28) _h
10.27622650	ROM Code	(0x754F1B3) _h
10.28344000	ROM CRC	(0x87) _h
10.28463562	Data (Read scratchpad)	(0xBE) _h
10.28581925	Data	(0xA4) _h
10.28621912	Data	(0x01) _h
10.28661887	Data	(0x4B) _h
10.28701875	Data	(0x46) _h
10.28741862	Data	(0x7F) _h
10.28781850	Data	(0xFF) _h
10.28821837	Data	(0x0C) _h
10.28861837	Data	(0x10) _h
10.28901837	Data	(0xDA) _h

The temperature is passed as 16 bit sign-extended two's complement number in first two bytes of returned data. The structure of data containing information about temperature is shown in table 4.3.

Table 4.3: DS18B20 temperature register

MSB								LSB							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
S	S	S	S	S	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}

In this case has the LSB value $(0xA4)_h$ and MSB is equal to $(0x01)_h$.

$$\underbrace{0000}_{(0xA4)_h} \underbrace{0001}_{(0xA4)_h} \quad \underbrace{1010}_{(0xA4)_h} \underbrace{0100}_{(0xA4)_h}$$

When the data are acquired with 12 bit resolution all bits are valid. The first bit of MSB is a sign bit ($S = 0$ for positive numbers, $S = 1$ for negative numbers). According to the datasheet [21], the decimal part is computed from the last four bits as:

$$(0.0100)_b = (0.25)_d$$

The whole number part is evaluated from bit four to bit ten as:

$$(0011010)_b = (26)_d$$

The sign bit is zero, so the value is positive and the final result is 26.25°C

DS18B20 library

In this case not only a library for communicating with the sensor was written. It was necessary to write a library for 1-Wire protocol as it is not integrated in the SM32F4 the same way as e.g I²C is. This library contains functions for sending and receiving bytes with carefully timing as described in the section 3.3.3. The library can be found on the attached CD (files *OneWire.c* and *OneWire.h*) as well as the library for DS18B20 (files *DS18B20.c* and *DS18B20.h*)

The flow graphs of the initialization process and measurement can be found on figure 4.7.

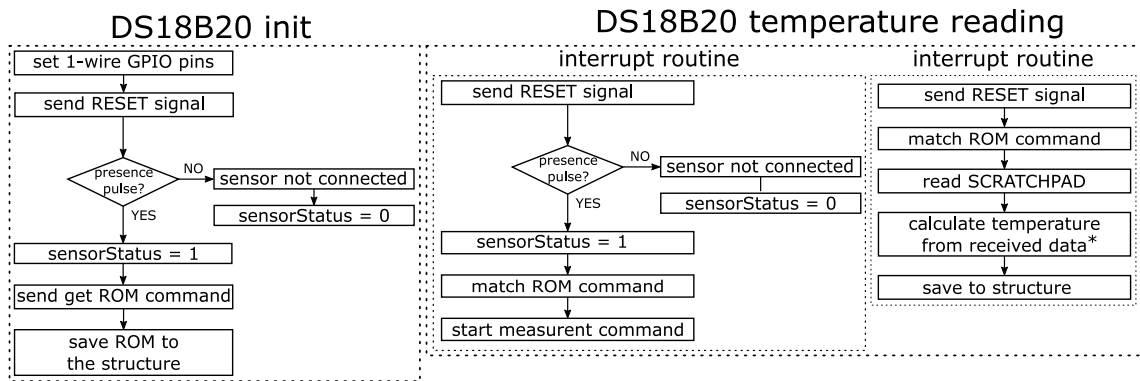


Figure 4.7: DS18B20 initialization (left) and measurement (right) flow graph

All the data in relevant to the DS18B20 are saved in a structure (code 4.2).

Code 4.2: Data holding structure for DS18B20 sensor

```

struct DS18B20_parameters {
    // unique identification code
    uint8_t ROM_NO[8];
    // last measured value of temperature
    float lastTemperatureReading;
    // resolution used for measurement
    uint8_t resolution;
    // 1: sensor connected, 0: sensor connection error
    uint8_t sensorStatus;
};

```

DS18B20 probe

The waterproof probe is provided by the manufacturer. To connect it to the uLite box, it was only necessary to solder a plug socket connector (figure 4.8).



Figure 4.8: DS18B20 probe

* conversion process can be found in section 4.1.2

4.2 Air quality monitoring

The primary motivation for building uLite unit is to bring the possibility of air quality measurement. The data can be use to prove that the filtrated air meets law regulations. The filtration units are designed to purify the air and the sensors are used to detect substances which should not be present. Therefore even cheaper sensors with a low measurement precision can be used as the interest lies in the presence of the substance and not so much in the exact concentration. The same applies for the gas sensors which are not calibrated therefore not suitable for measurement of exact concentration.

4.2.1 Analogue sensors in uLite

The analogue sensors used in uLite are read with use of operational amplifier. The description of this method is in section 3.2.3. There is a structure created for each analogue sensor holding the last measured value. The structure is created even though it has only one element to preserve the program convention (code 4.3).

Code 4.3: Example of data holding structure for analogue sensors

```
struct Dustsensor {  
    uint16_t lastValueRead;  
};
```

STM32F407 has three 12-bits ADC converters with 15 channels. The program uses two ADC converters. The first one is used for reading gas sensors and the second one is reserved for reading values of the dust sensor. All ADC conversions are triggered by interrupts.

4.2.2 Dust sensor GP2Y1010AU0F

The compact optical dust sensor GP2Y1010AU0F (figure 4.9) from Sharp is used to detect presence of dust in the air. The sensor uses optical sensing system. An internal infra-red diode (IRED) and an phototransistor are diagonally arranged in this device. The LED creates short light pulses and the reflected infra-red light is opening the photo-transistor accordingly. This sensor can detect particles of minimal size PM2.5 (2.5 μm).

Dust sensor GP2Y1010AU0F interfacing

The IRED diode is internally connected to PNP transistor and to switch it on the pin has to pulled down. According to the datasheet, the IRED diode needs 20 mA but the current drawn was over 40 mA so the pin should not be pulled down by the microcontroller pin as there is a danger of destroying the 20 mA tolerant microcontroller pin. The switching is solved with an additional transistor (figure 4.10).



Figure 4.9: Dust sensor

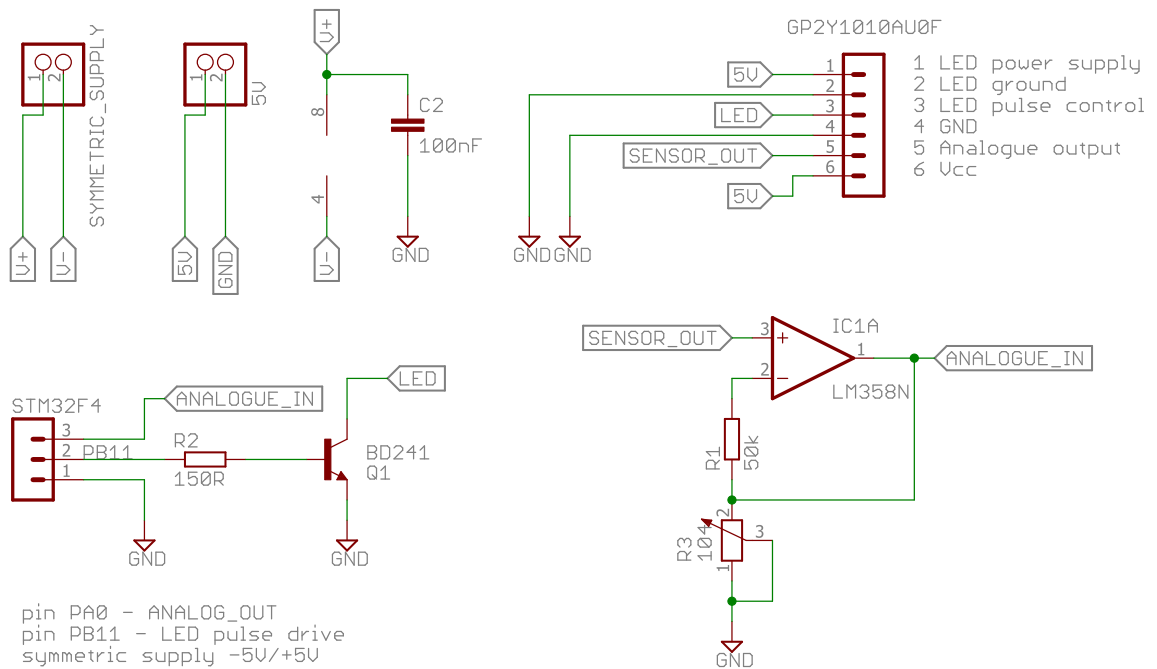


Figure 4.10: Connection of the dust sensor

The pulse cycle for IRED diode should be 10 ± 1 ms and the pulse width 0.32 ± 0.02 ms. The sampling should take place in the first 28 ms of the pulse width. The generated pulses for the dust sensor are on figure 4.11.

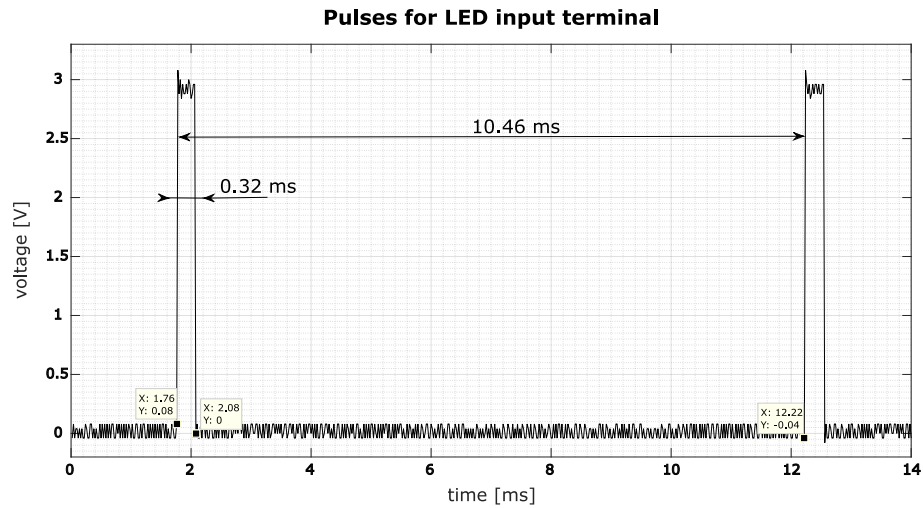


Figure 4.11: Pulse-driven wave form for IRED

The output is a voltage signal between 0-3.3 V depending on the dust air density. This range was achieved by setting the trimmer R3 in the operational amplifier circuit. Before this adjustment was the range smaller and the full scale of ADC converter would not be used. On the figure 4.12 it is possible to observe one measurement cycle of the dust sensor under two conditions.

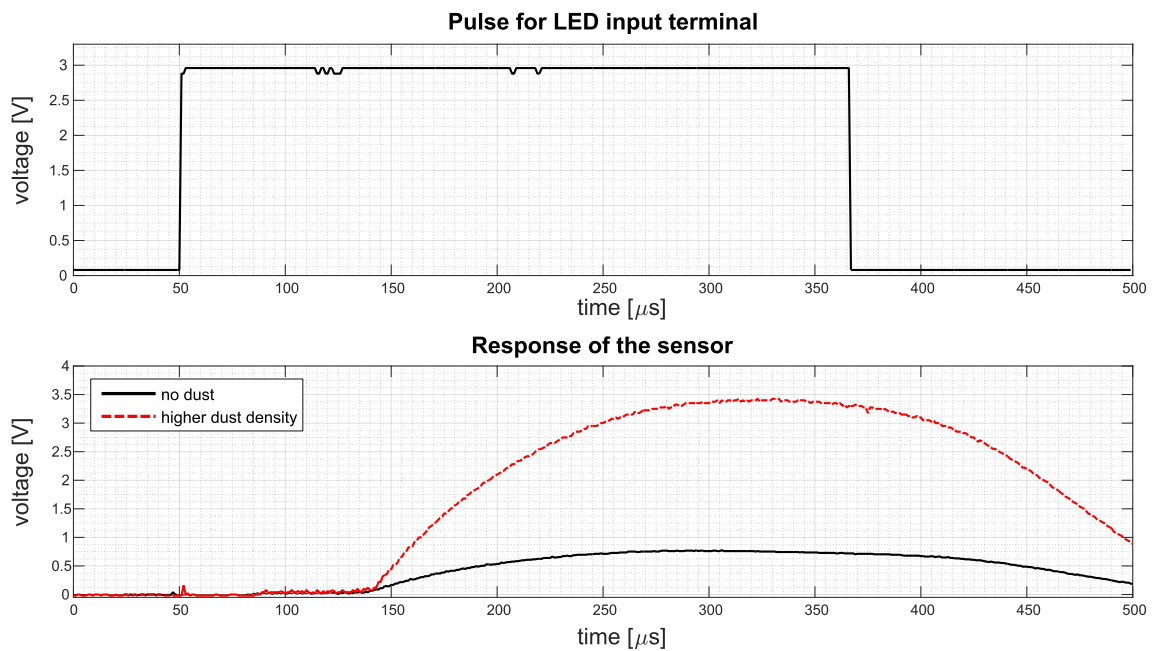


Figure 4.12: Dust sensor response

Dust sensor measurement

The pulse-driven wave form for the IRED was created by two timers with periods 0.32 ms and 10 ms that periodically switch the output pin on and off. When the pin

goes high, the ADC conversion is switch on and the data are sampled. After the end of the pulse the ADC conversion stops and the array with the measured data is searched for the highest value. This value is then declared as the result and save in the structure.

The flow diagram of the measuring process is on figure 4.13. The waiting times are handled as interrupts from timers to save computing power of the microcontroller.

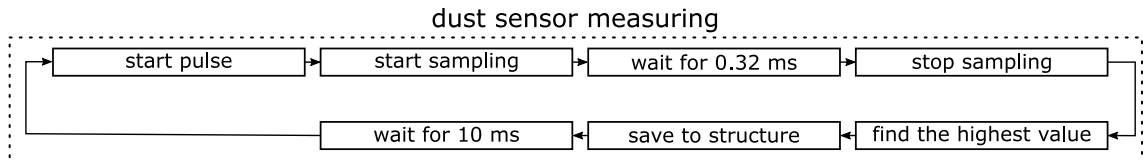


Figure 4.13: Dust sensor measurement flow graph

4.2.3 VOC particle sensor MP502

Volatile organic compounds (VOC) are organic chemicals with a high vapour pressure at temperature around 20 °C. The VOC can be both natural in origin or human made. They can be harmless or can be considered health risk. As examples of VOC commonly present in air is possible to mention acetone, formaldehyde, toluene, benzene, methylene chloride, ethylene glycol ...

VOC particles are one of the parameter which states the air quality and to which can law regulation apply as well. The motivation to measure VOC is to verify function of the carbon filter because in contrast to the dust filters, the wear out of the carbon particles filters is not easily detectable.

Either way is the goal to detect any traces of VOC particles in the air and take appropriate action without too much dependency on what VOC and in which concentration is present.

The MP502 gas sensor from Winsen (figure 4.14) has high sensitivity to organic gases such as toluene, benzene or methanol. The detection range is from 0 ppm to 50 ppm. The sensor contains internal heater which needs 5 V to operate and it's power consumption is 300 mW. The heater has to be on for the whole operation time.

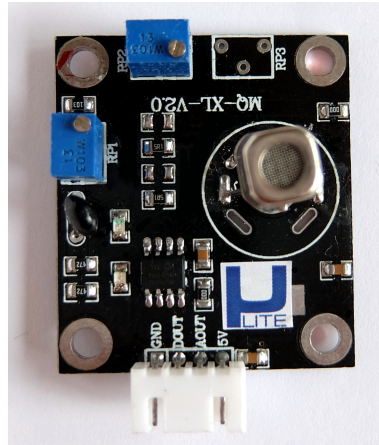


Figure 4.14: MP502 VOC sensor

MP502 interfacing

The sensor comes with already prepared breakout board. The sensor came with complete circuit solution. It is possible to adjust load resistor using one of the trimmer on the board. The resistor value was adjusted to scale the output values from 0 V to ca. 3.5 V. To protect the analogue pin, the sensor is read with use of the operational amplifier (section 3.2.3).

The sensor has quick response and recovery time about 20 s . The measurement was made by covering the sensor with ethanol and then moving the sensor into clean air. Find the result of this measurement on figure 4.15. For the next test, a Syoss Hold & Flex hair spray containing besides other ingredients dimethyl ether, alcohol denat, aminomethyl propanol, perfume and benzene alcohol was sprayed into the vicinity of the sensor. The result is on figure 4.16. The resulting VOC particle level was holding on until the window was opened.

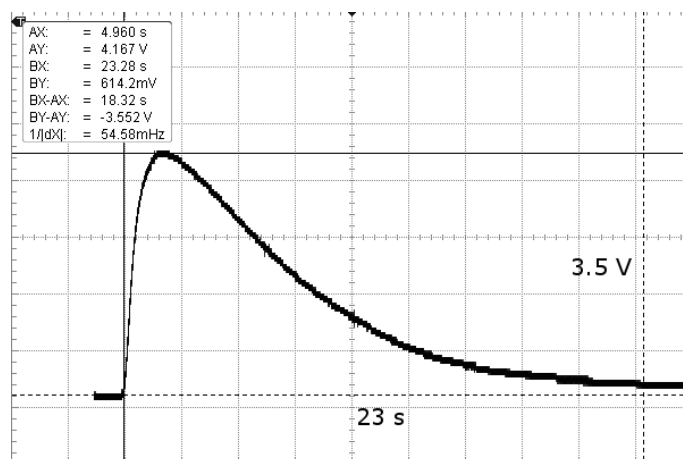


Figure 4.15: M502 recovery time

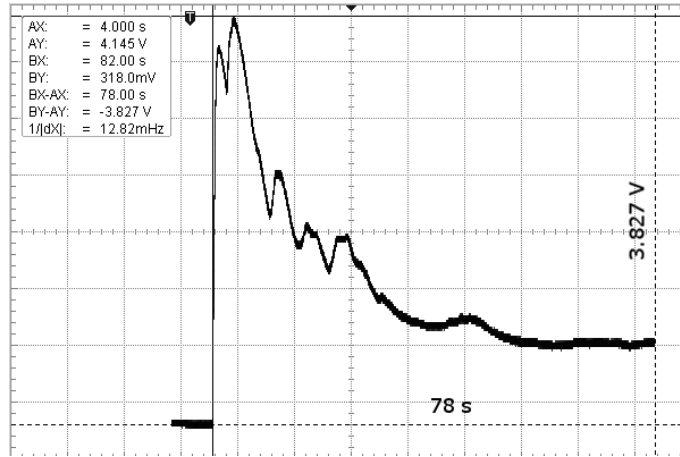


Figure 4.16: MP502 hair spray test

This sensor doesn't require any special measuring process. The value of the ADC converter can be read at any time. The value is saved in the structure.

4.2.4 Carbon Monoxide sensor MQ-7

Carbon monoxide (CO) is a odorless, colorless poisonous gas. It is produced by the incomplete burning of liquid, solid, gaseous fuels and all materials containing carbon. It is a leading cause of chemical poisoning in both houses and workplaces where it is produced by several industrial processes [12]. CO competes with oxygen and binds to haemoglobin which prevents sufficient oxygen from reaching the tissues of the human body. The levels of concentration up to 25 ppm are not harmful. The concentration of about 50 ppm corresponds to the normal state of CO contained in a body of a heavy smoker. The concentrations above 60 ppm brings health problems such as headache, shortness of breath. In concentrations around 500 ppm brings coma, unconsciousness and death.

MQ-7 analogue semiconductor sensor is used for detection of CO (10-1000 ppm). It also has high sensitivity to hydrogen (H₂) and methan (CH₄). The sensitive material is tin oxide (SnO₂) which has high reactivity to gasses with a low oxidation number (also called reducing gasses) such as methane or CO at relative low operating temperatures. The principle is detecting the conductivity change of n-type semiconductor material while the surface reacts with gasses. The conductivity is low when exposed to clean air, because the conduction electrons are bound to surface oxygen. On the other hand, when exposed to the reducing gasses, electrons are no longer bound to the surface state and the conductivity increases [39].

For the purpose of this master thesis, the sensor is not calibrated. The sensor output characteristic is not linear and it is dependent on the temperature and humidity, so the exact values of concentration are not reachable. However when taken in account the previously mentioned effects of even rather low concentration of CO on human body, the sensor is sufficient for this application. If the CO sensor detects increased concentration of CO, H₂ or CH₄, the action should be taken.

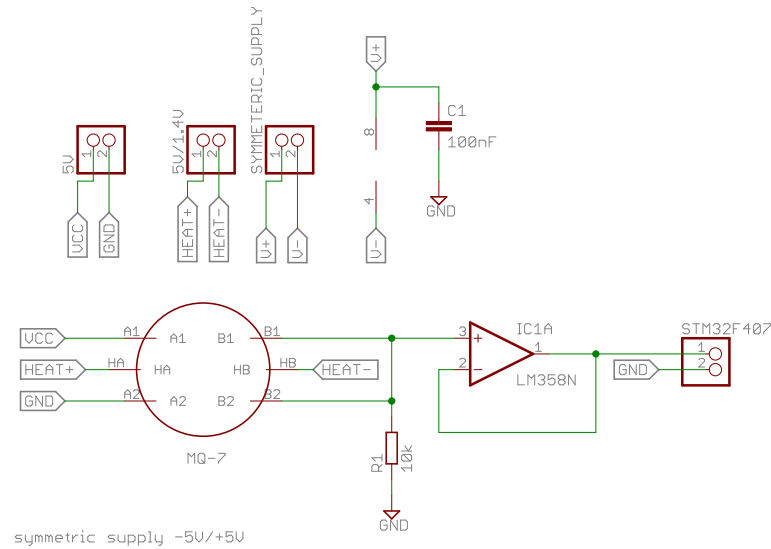


Figure 4.17: MQ-7 schematic

MQ-7 interfacing

According to the datasheet [46], before using this sensor in regular operation it is necessary to let the sensor “bake” (it is connected to the supply voltage for at least 48 hours before first use) to eliminate any remaining moisture or contamination from the manufacturing process. After this initial pre-heat there is a shorter preheating before any measurement (500 s) recommended.

In the normal operation requires MQ-7 the internal heater cycling voltage phases. In the low temperature phase (heater is powered by 1.4 V for 90 s) the detection of CO is carried out. In the high temperature phase (heater is powered by 5 V for 60 s) the gases which has absorbed at low temperature are cleaned.

The sensor has six pins. There is the basic test circuit of MQ-7 on the figure 4.18. Two of the pins are used as a supply power for the heater, two pins are sensor power supply and the last two are analogue output from the sensor. The datasheet recommends 10 kΩ for the R_L .

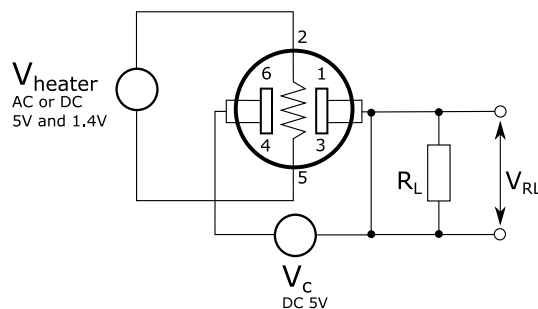


Figure 4.18: MQ-7 connection

There are few adjustments done to this circuit (figure 4.17). The switching of 5 V and 1.4 V is solved in the power supply circuit (section 4.9). For the reading

of the analogue sensor is used connection with operational amplifier (explained in 3.2.3).

MQ-7 measuring

The values read from the sensor are only valid in the non-cleaning phase. The information about in which state the sensor is can be found in the structure (code 4.4).

Code 4.4: Data hodlijg structure for MQ-7

```
struct COsensor {
    uint32_t lastValueRead;
    //phase = 0 : cleaning phase
    //phase = 1 : measuring phase
    uint8_t phase;
};
```

When the sensor is in the cleaning state, the value of the ADC is not processed.

4.2.5 Combustive gasses sensor TGS813

Smoke which is involved in any combustion or burning consists of gases such as carbon dioxide, carbon monoxide, methane, propane, butane and so on. Filter units are often connected to industrial processes producing smoke and if there exists a way how to tell presence of any of the previously mentioned gasses, the function of the filter or the whole unit can be verified. The TGS813 has high sensitivity to methane (CH₄), propane (C₃H₈) and butane (C₄H₁₀). It also detects traces of hydrogen (H₂), ethanol (C₂H₆O) and carbon monoxide (CO). The internal structure is similar to the carbon monoxide sensor. The sensitive element is tin oxide (SnO₂) as well. When it comes to the contact with any gaseous element, the internal resistance can drop as low as 20 times to its normal value. The initial preheating phase of this sensor took 7 days and before every new measurement there is approximately 1 minute long.

TGS813 interfacing

In comparison with the MQ-7 sensor is the interfacing easier. The testing scheme is the same as for the MQ-7 sensor (figure 4.18) with only one modification. The heater doesn't not require voltage changing over time. This simplifies the connection scheme (figure 4.19).

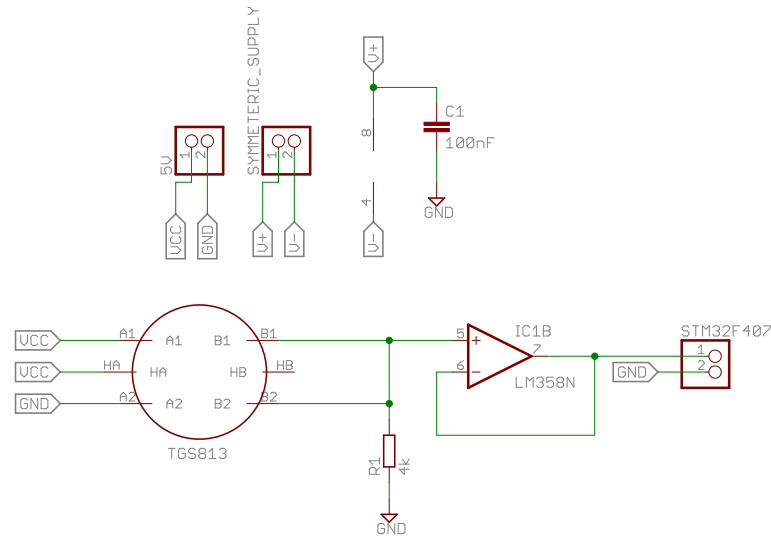


Figure 4.19: TGS813 schematic

TGS813 sensor doesn't require any special measuring process. The value of the ADC converter can be read at any time and it is stored in the corresponding structure.

4.3 Pressure monitoring

Monitoring of pressure drops plays an important role in filter unit functions. The pressure can be measured on the filtration unit inlet and after the filter (before the fan). This data can be used to calculate a filter usage according to the equation 4.3 in which is Δp representing the pressure drop on the filter.

$$\text{Filter usage [\%]} = \frac{\text{actual } \Delta p - \Delta p_{\text{new filter}}}{\Delta p_{\text{used filter}} - \Delta p_{\text{new filter}}} \quad (4.3)$$

Furthermore the blocked inlet hose can be recognised. In case that the difference of pressure drop on the fan and the pressure drop on the filter exceeded some defined value, that is possibility of foreign object in the inlet hose. The pressure difference measuring can also be used for filter presence check. If the revolution of the fan drive are higher than zero and the pressure drop over the filter is lower than defined value, the filter may be missing or it is not correctly inserted. In case that the pressure drop over filter is higher than defined value but the pressure drop before fan is not increased, the door of the filter unit is not closed or sealed properly. All the threshold values have to be measured and will be slightly different for individual filter unit constructions.

4.3.1 AMS 5915 pressure sensor

AMS 5915 is a digital pressure sensor series with amplifier and digital output (I²C) characterized by high precision measurement. The principle of sensing element is

based on piezoresistive measuring cell. The sensor consists a resistor on the diaphragm of the cell's silicon chip. When there is a pressure applied it causes a deflection and the resistance changes. The advantage of this function principle is high resistance to chemicals, relatively high output signal and possibility to measure within small ranges with high accuracy. In addition to the pressure difference measurement, the sensor also provides temperature measurement used for pressure measurement temperature compensation. The measured value is accessible so it can be used as additional monitoring of the temperature inside of the filter unit (close to the control board on which are those sensors usually mounted). The suitable pressure range is selected as -350 mbar to 350 mbar which complies with the AMS 5915-0350-D-B type.

AMS 5915 interfacing

The sensor comes in common dual in-line package (DIP) and it doesn't need any additional components to function except the pull-up resistor on the I²C bus lines. The connection of the sensors to the microcomputer is shown on figure 4.20. On the schematic, two sensors are connected to demonstrate the possibility of connecting more device with different addresses to one I²C bus.

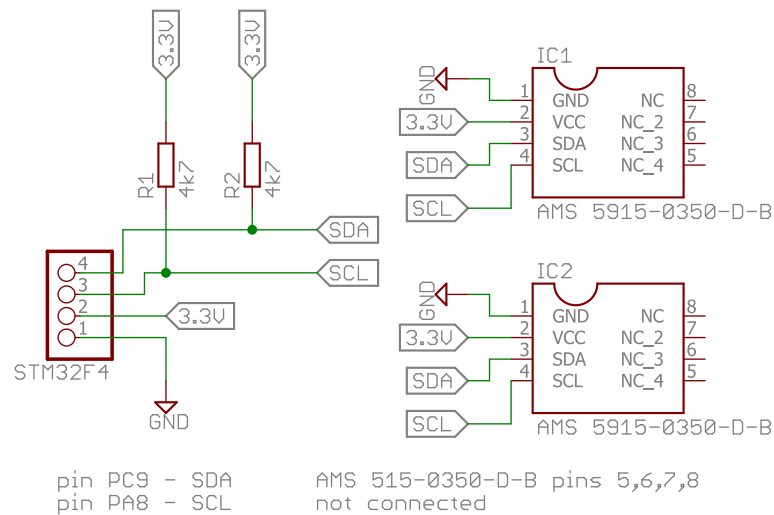


Figure 4.20: AMS 5915 pressure sensor interface

Table 4.4: AMS 5915-0350-D-B sensor specifications [2]

AMS 5915-0350-D-B	
Pressure range	±5 PSI
Communication	I ² C
Accuracy of pressure measurement	±0.5% FSO
Overall accuracy	±1% FSO
Temperature measurement error	±3% FSO
Resolution A/D converter	14 bit
Resolution pressure signal	12 bit
Resolution temperature signal	11 bit

AMS 5915 communication

The address of this sensor is programmable in all seven bits. The programming of the address is done with a special software from HJK Sensoren + Systeme GmbH & Co. KG. The three ordered sensors came with pre-setted addresses $(0x01)_h, (0x02)_h, (0x03)_h$. The communication with this sensor is limited only to addressing the sensor. The communication is started always by master by sending the address of desired sensor. When the is sensor addressed it returns four data bytes (figure 4.21). The address and every data byte (except for the last one) is followed by the acknowledgement bit. The structure of returned data is in table 4.5.

Table 4.5: AMS 5915 data register

1st byte MSB pressure								2nd byte LSB pressure							
x	x	5	4	3	2	1	0	7	6	5	4	3	2	1	0
3rd byte MSB temperature								4nd byte LSB temperature							
7	6	5	4	3	2	1	0	7	6	5	x	x	x	x	x

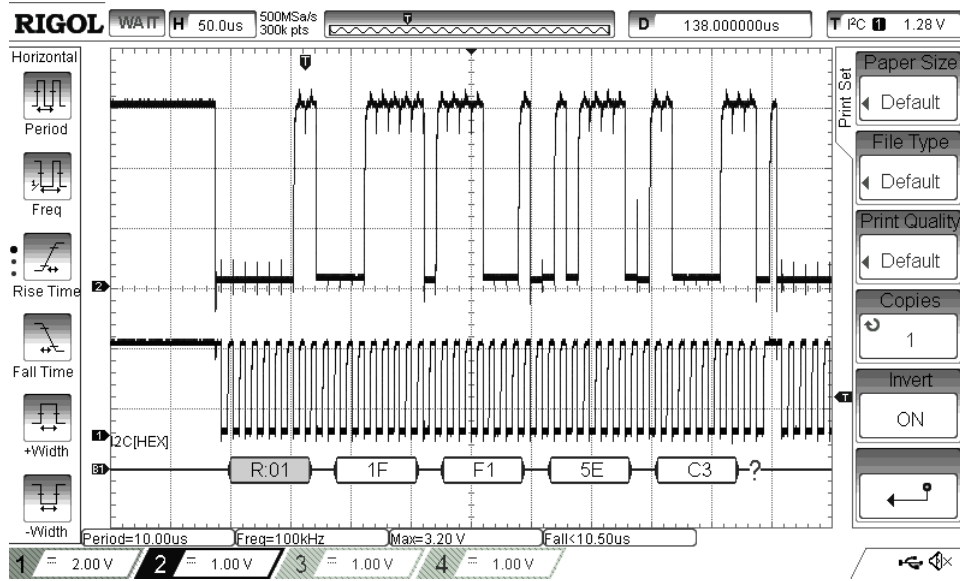


Figure 4.21: AMS 5915 communication example

From the figure 4.21 it is possible to read data bytes $(0x1F)_h$, $(0xF1)_h$, $(0x5E)_h$, $(0xC3)_h$. The pressure measurement is acquired by ignoring the first two bits of the MSB byte and using the value in an equation provided in datasheet.

$$(1FF8)_h = (0001 \ 1111 \ 1111 \ 0001)_b = (8177)_d$$

The first two bits are in this case already zero. The equation for pressure calculation is taken from the datasheet (equation 4.4.)

$$p = \frac{Dig_{output} - Dig_{output_{min}}}{Sensitivity} + p_{min} \quad \text{with} \quad Sensitivity = \frac{Dig_{output_{max}} - Dig_{output_{min}}}{p_{max} - p_{min}} \quad (4.4)$$

The constants used in the equation 4.4 are specific for each AMS 5915 sensor type and they can be found in the datasheet [2].

$$\begin{aligned} Dig_{output_{min}} &= 1638 \\ Dig_{output_{max}} &= 14745 \\ p_{max} &= 350 \text{ mbar} \\ p_{min} &= -350 \text{ mbar} \end{aligned}$$

Please refer to equation 4.5 and 4.6 for the calculation of the physical value of measured pressure difference. The resulting pressure difference is really small, because there were no connections to either one of the inputs and the pressure

difference should be theoretically zero. The small offset is negligible for given range.

$$\text{Sensitivity} = \frac{Dig_{output_{max}} - Dig_{output_{min}}}{p_{max} - p_{min}} = \frac{14745 - 1638}{350 - (-350)} = 18.7243 \quad (4.5)$$

$$p = \frac{Dig_{output} - Dig_{output_{min}}}{\text{Sensitivity}} + p_{min} = \frac{8177 - 1638}{18.7243} + (-350) = -0.7746 \text{ mbar} \quad (4.6)$$

For the calculation of temperature the first eleven bits of the two last data bytes are used.

$$(5EC3)_h = (0101 \ 1110 \ 1100 \ 0011)_b$$

Before calculating the physical value it is necessary to disregard the last five bites (table 4.5). To ignore the last five bits shifting to right by 5 bits is carried out.

$$(0101 \ 1110 \ 1100 \ 0011)_b \gg 5 = (0010 \ 1111 \ 0110)_b = (758)_b$$

The relation for temperature calculation is in equation 4.7. The numerical solution is in the equation 4.8.

$$T = \frac{Dig_{output}(T) \cdot 200}{2048} - 50 \quad [^{\circ}\text{C}] \quad (4.7)$$

$$T = \frac{758 \cdot 200}{2048} - 50 = 24.02 \quad [^{\circ}\text{C}] \quad (4.8)$$

AMS 5915 library

This sensor doesn't allow any changes of the parameters so the initialization of the sensor and the reading of the values are both simple (figure 4.22).

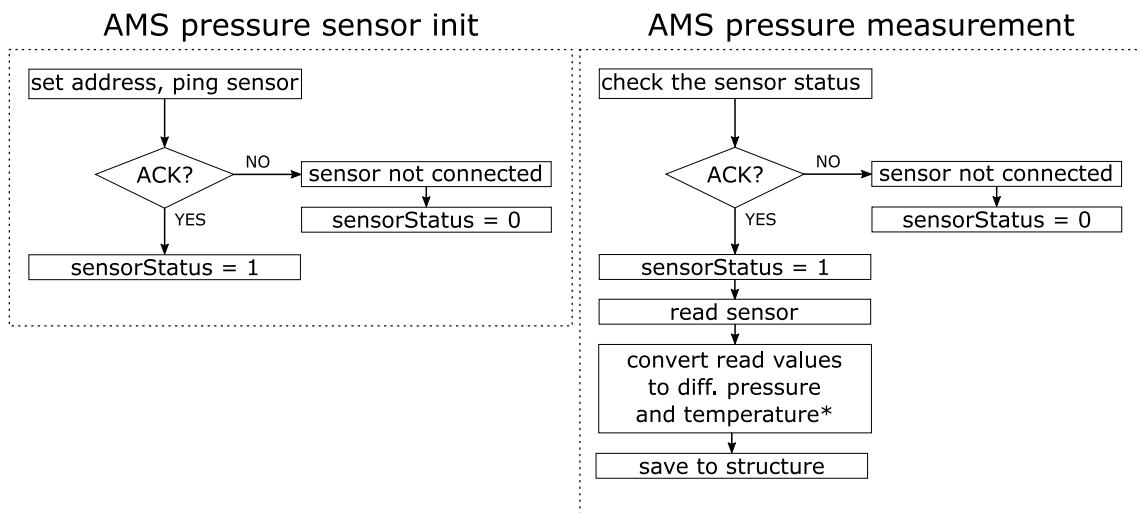


Figure 4.22: Flow graphs of AMS 5915 initialization and measurement

* conversion process can be found in section 4.3.1

The structure contains sensor address, results of the last differential pressure and temperature reading and the status of the sensor (code 4.5).

Code 4.5: Data holding structure for AMS5919

```
struct AMS_parameters
{ uint8_t address;
  // result of the last temperature reading
  float lastTempReading;
  // result of the last pressure reading
  float lastPressureDifReading;
  // 1: sensor connected, 0: sensor disconnected
  uint8_t sensorStatus;
};
```

4.4 Vibration monitoring

Monitoring of the drive or unit vibration can help determine the state of the DC motor in the blower. If there are data about vibration of the drive in time available, it is possible to investigate the vibration spectrum in which various problems of the drive can be found. To collect the data an accelerometer is used.

The measurement of drive vibration should be done as close to the drive as possible. Ideal case would be the accelerometer placed on the drive itself but uLite is designed as a mobile measurement unit and the run of a filtration unit is not possible with the unit door open. Therefore an accelerometer probe which can be attached to an arbitrary magnetic surface is designed.

4.4.1 ADXL345 accelerometer

The ADXL345 is low-power 3-axis accelerometer with selectable sensitivity ranges ($\pm 2G$, $\pm 4G$, $\pm 8G$, $\pm 16G$), resolution (10,11,12,13-bit) and data rate ranging (from 10 Hz to 3200 Hz). The sensor consists of a micro structure on a silicon substrate suspended by springs from poly-silicon* which allows it to deflect smoothly in any direction. The deflection causes a change in capacitance between plates which are attached to the structure and fixed plates. This change is converted to an output voltage proportional to the acceleration on every axis. The accelerometer was purchased on the break-out board because it only comes in the package which is manually unsolderable (14-terminal LGA). ADXL345 offers change of all measurement parameters and provides some interesting additional functions. It is possible to combine ranges and resolutions, detect free fall or send interrupts triggered by various motions (detect of the presence, lack of movement, double tap etc.). It contains self-testing function as well.

* *“Polycrystalline silicon, also called poly-silicon or poly-Si, is a high purity, polycrystalline form of silicon, used as a raw material by the solar photovoltaic and electronics industry.”*
https://en.wikipedia.org/wiki/Polycrystalline_silicon

The output response of the sensor relative to the orientation to gravity is shown on figure 4.23 (redrawn based on figure in device datasheet [19]).

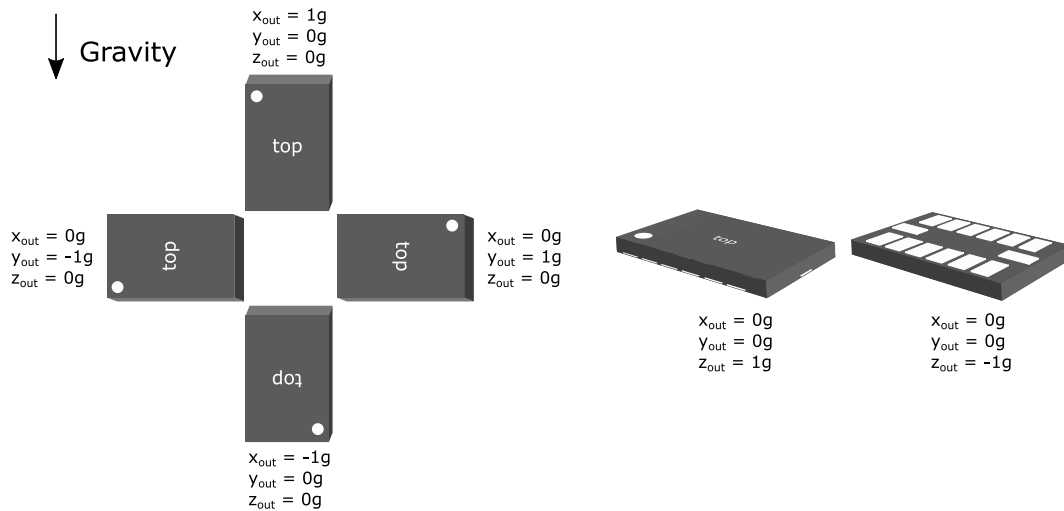


Figure 4.23: ADXL output vs. gravity force

One of the value is always going to have an offset equal to ± 1 . There is a mark on the accelerometer probe which guides the user to place the accelerometer in the “correct” position but there is no guarantee that the user positions the accelerometer according to the instruction. Therefore is no correction of the offset value in the program and this operation has to be done in the data processing procedure. The accelerometer has registers in which is possible to set offsets for all axis in case that it is mounted on a surface and change of its position in respect to gravity field is not expected.

ADXL345 interfacing

ADXL345 offers both I²C and SPI communication. Because the sensor is going to be used in form of a probe, the I²C bus is selected. The reasons are that I²C bus generally allows communication over longer distances and it also needs less conductors. To activate the I²C communication, the CS pin has to be connected to VDD. The device has option to choose from two I²C addresses by setting the pin SDO high or low. In this case it is connected to the high logical level (figure 4.24) setting the accelerometer address to $(0x1D)_h$ (with a write bit 0 translates the address to $(0x3A)_h$, similarly with a read bit 1 is the address $(0x3B)_h$). The I²C lines use 4.7 k Ω pull-up resistors (explained in section 3.2.1) and the power line is decoupled with capacitors (see section 3.2.2).

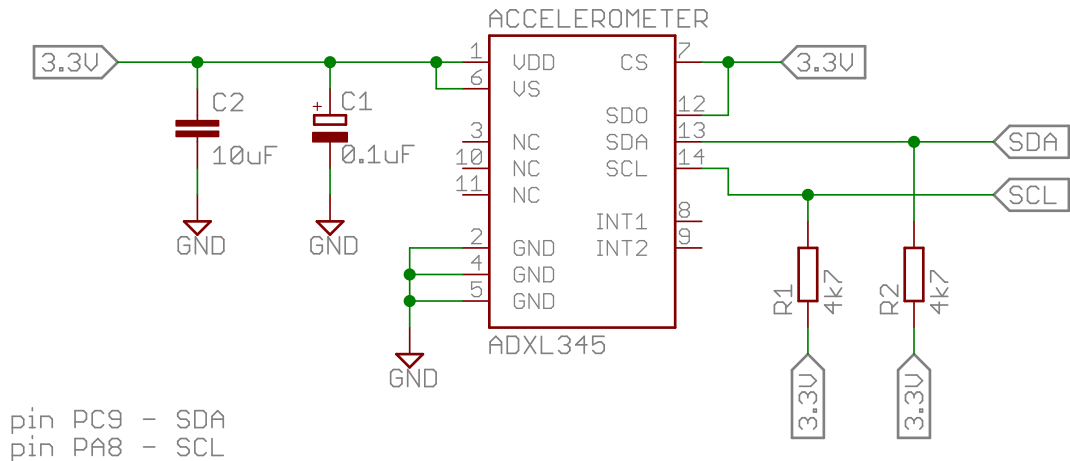


Figure 4.24: Connection scheme of the ADXL345 accelerometer

ADXL345 communication

Even though the sensor offers wide variety of measuring modes the application of measuring vibrations for the analysis of a drive requires simple continuous measurement mode. Individuals measurements do not have to be triggered and the value can be read at any time. It is recommended to use the memory burst reading because when a single-byte reading operation is performed, the remaining data in the sensor FIFO are lost. The only requirement on reading of the sensor running in continuous measuring mode is that there must be at least 5 μ s between individual readings.

Six bytes are read in memory burst mode from registers $(0x32)_h$ to $(0x37)_h$ (figure 4.25) using the accelerometer address and a read bit $(0x1D)_h$. Due to the fact the accelerometer is actually a probe with cable length of 2 m it is possible to notice a decreased quality of the signal. Despite this fact there are no missed acknowledgements and the communication is therefore still reliable.

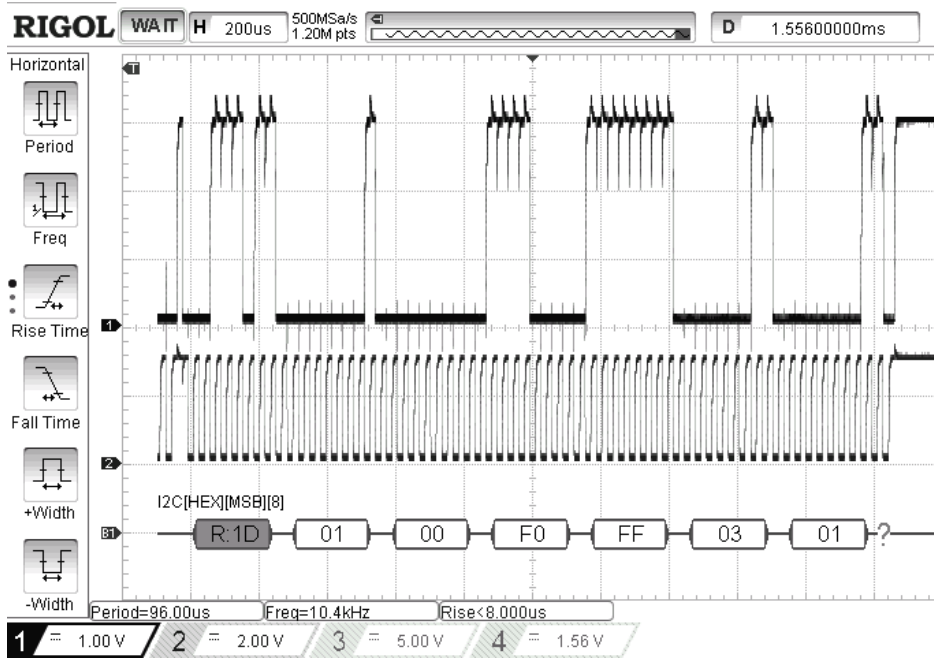


Figure 4.25: ADXL345 data acquisition

The output data is two's complement format with following arrangement (table 4.6):

Table 4.6: ADXL345 data registers

Register address	Description
(0x32) _h	X-Axis Data 0
(0x33) _h	X-Axis Data 1
(0x34) _h	Y-Axis Data 0
(0x35) _h	Y-Axis Data 1
(0x36) _h	Z-Axis Data 0
(0x37) _h	Z-Axis Data 1

To calculate the physical value it is first necessary to conjugate LSB and MSB bytes for each axis data (equation 4.9).

$$\begin{aligned}
 \text{X axis data: } & \overbrace{(0000 \ 0000)}^{(0x00)_h} \overbrace{(0000 \ 0001)}^{(0x01)_h}{}_b = (1)_d \\
 \text{Y axis data: } & \overbrace{(1111 \ 1111)}^{(0xFF)_h} \overbrace{(1111 \ 0000)}^{(0xF0)_h}{}_b = (-16)_d \\
 \text{Z axis data: } & \overbrace{(0000 \ 0001)}^{(0x01)_h} \overbrace{(0000 \ 0011)}^{(0x03)_h}{}_b = (259)_d
 \end{aligned} \tag{4.9}$$

The scale factor for given range of measurement can be found in the device datasheet and it is equal to 3.9. The value is also scaled to get the value in g or in m/s^2 when using SI units (equation 4.10). The position of the sensor during the measurement can be determined from figure 4.23.

$$\begin{aligned}
 \text{X axis data:} & \quad 1 \cdot \frac{3.9}{1000} = 0.0039 \text{ m/s}^2 \\
 \text{Y axis data:} & \quad -16 \cdot \frac{3.9}{1000} = -0.0624 \text{ m/s}^2 \\
 \text{Z axis data:} & \quad -259 \cdot \frac{3.9}{1000} = 1.0101 \text{ m/s}^2
 \end{aligned}
 \tag{4.10}$$

ADXL345 library

Because of the huge possibilities of this sensor, only functions directly needed for the vibration measurement in this particular application are implemented. The complete library can be found on the attached CD (files *ADXL345.c* and *ADXL345.h*).

In the sensor initialization process (figure 4.26 on the left) is the range of measurement set to $\pm 2g$ and the sensor the continuous measurement mode is started. The resolution of $\pm 2g$ range is set to 10-bits by default. The flow graph of the reading algorithm is on figure 4.26 (on the right).

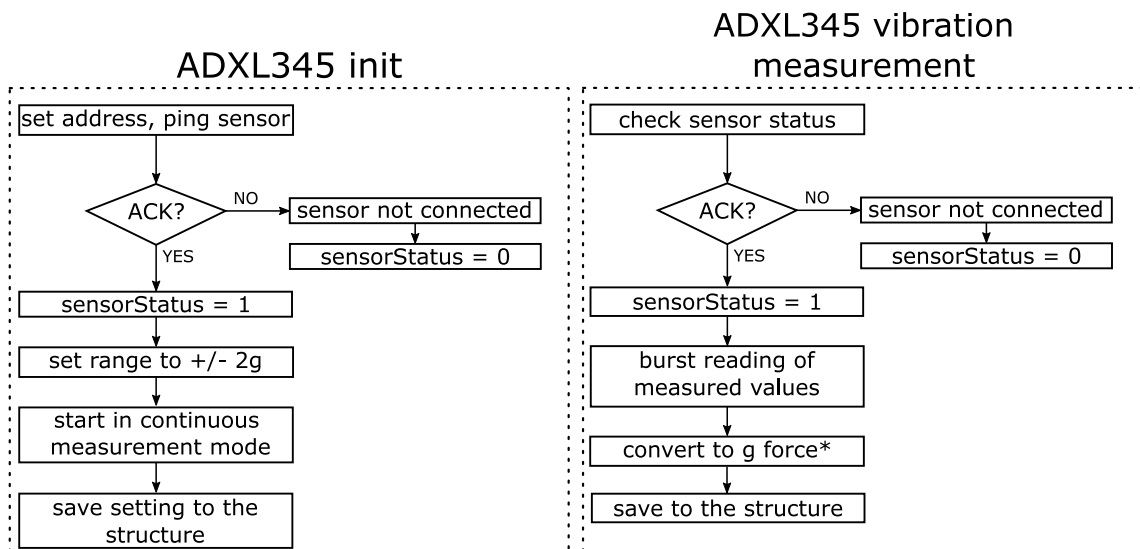


Figure 4.26: ADXL initialization (left) and measurement (right) flow graphs

The structure for this sensor contains addresses, the range (to make the library usable also for another application), last values of the reading and sensor status indicator (code 4.6).

* conversion process can be found in section 4.1.1

Code 4.6: Data holding structure for ADXL345

```
struct ADXL345_parameters {
    uint16_t writeAddress;
    uint16_t readAddress;
    // selected range, can only take values of type range_ADXL345
    (RANGE_2G, RANGE_4G, RANGE_8G, RANGE_16G)
    range_ADXL345 range;
    // results of the last reading
    float xReading;
    float yReading;
    float zReading;
    // 1 = sensor connected, 0 = sensor connection error
    uint8_t sensorStatus;
};
```

ADXL345 probe

The probe was created using the break-out accelerometer board, 4-core conductor, heat shrinking tube, electrical tape, neodymium magnet, plug socket connector and stickers (figure 4.27 and 4.28).



Figure 4.27: ADXL345 probe



Figure 4.28: ADXL345 probe - detail

4.5 SD Card

MicroSD Card from Kingston (8 GB) is used to log and backup the measured values. The communication with a SD card can be managed in two modes: SDIO and SPI mode. Both SDIO and SPI mode run on similar frequencies (40 kHz for SD, 50 kHz for SPI [48]). The difference is the bus width and the fact that SDIO protocol uses

separate lines for commands and data. The 4-bit or 8-bit width SDIO is four or eight times faster than SPI. The advantage of SPI is that the protocol is widely used and well documented while SDIO is not. The developed application doesn't require high speed data transfer as the measurements are performed every so often and the size of the data is small. For given reasons is SPI communication protocol selected.

It is possible to connect individual SD card pins directly to the pins on the STM32F4 discovery board but to make the manipulation easier, a SD Card interface board by LC Studio was used (figure 4.31). The pinout of a classical SD card operating in SPI mode is shown on figure 4.29.

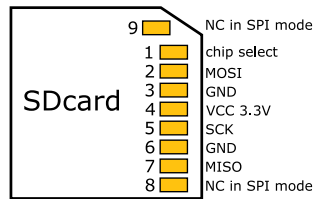


Figure 4.29: SD card pinout when in SPI mode

4.5.1 SD card interface

The break-out board from LC studio offers SPI and has pull-up resistors on all SPI lines and two decoupling capacitors on the power supply lines. There is also an option to power this line from 5 V supply thanks to the 3.3 V regulator LM1117. The schematics of this break out board is on figure 4.30. The mirrored output pins on the card reader header connector are useful for debugging as they allow to connect oscilloscope probes or logic analyser without the necessity of using breadboard.

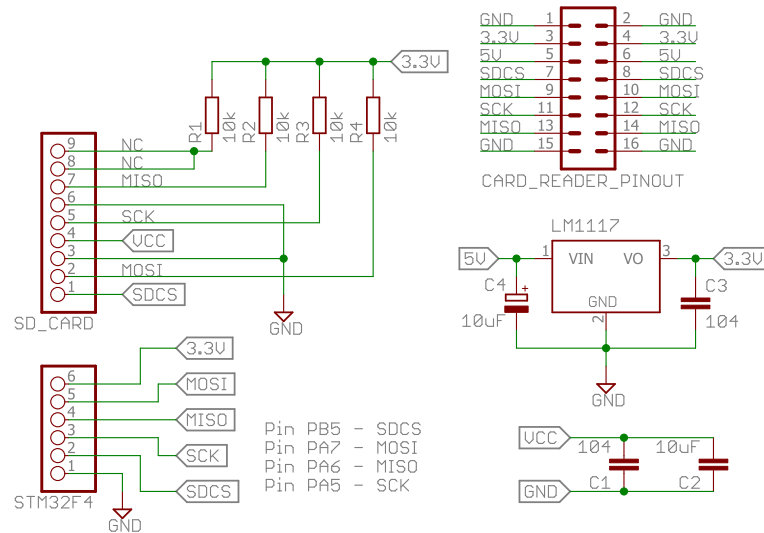


Figure 4.30: LC Studio SDcard reader schematics

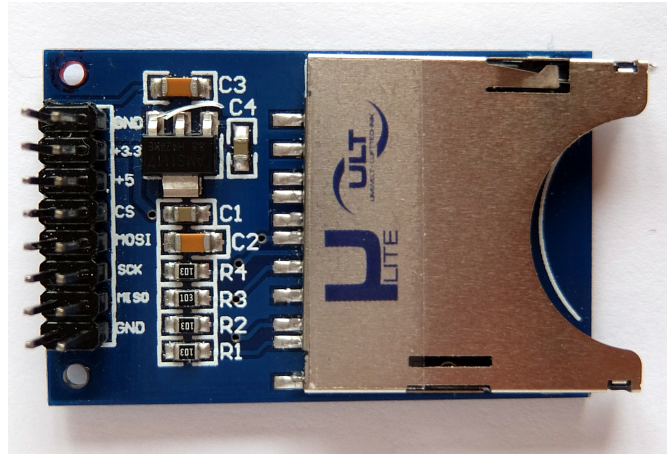


Figure 4.31: LC Studio SD card reader

4.5.2 FatFs

FAT file system (File Allocation Table) is used to allow access to files and directories. To manipulate with the directory structure and to handle files generic FAT file system module FatFs* is applied. The FatFs specification [13] says “FatFs is a generic FAT/exFAT file system module for small embedded systems. The FatFs module is written in compliance with ANSI C (C89) and completely separated from the disk I/O layer. Therefore it is independent of the platform. It can be incorporated into small microcontrollers with limited resource, such as 8051, PIC, AVR, ARM, Z80, 78K and etc.”

Among others it offers following features:

- Windows compatible file systems including FAT sub-types FAT12, FAT16 and FAT32.
- Long name support.
- RTOS support.
- Unlimited number of open files and up to 10 connected volumes.
- Volume size up to 2T bytes with sector size up to 4K bytes.

The code is free to use for both personal and commercial products without any restrictions.

4.5.3 Commands for communication with SD card

The commands used for generic initialization of SD card are described in table C.1 in appendix C. The table was created based on information in SD card specification [48] and [49]. The command structure of SPI communication is defined in the standards [14] as follows.

* http://elm-chan.org/fsw/ff/00index_e.html

The command has always six bytes. The first byte is command number with added value of $(0x40)_h$. The middle four bytes can be filled with an argument. The last byte is CRC which is by default switched off in the SPI (but the six bytes still have to be send).

The SD card is ready to receive a command when it drives MISO line high. To wait for response “I don’t care” value $(0xFF)_h$ is send to repeat the clock cycles. The response time is eight bytes at maximum (if the response doesn’t came after eight clock cycles there is a problem with communication). In case the first bit of the response is high, the low seven bits are error flags. Response can take up to two bytes, depending on the command. To end communication the CS chip select pin has to be set high and at least eight cycles of clock with value on MOSI line $(0xFF)_h$ has to be send.

The card can respond using several response formats. The most important are R1, R3 and R7 response formats (figure 4.32). The most common is R1 response (value of R1 $(0x00)_h$ means successful), frames R3 and R7 are used for the SD card set up.

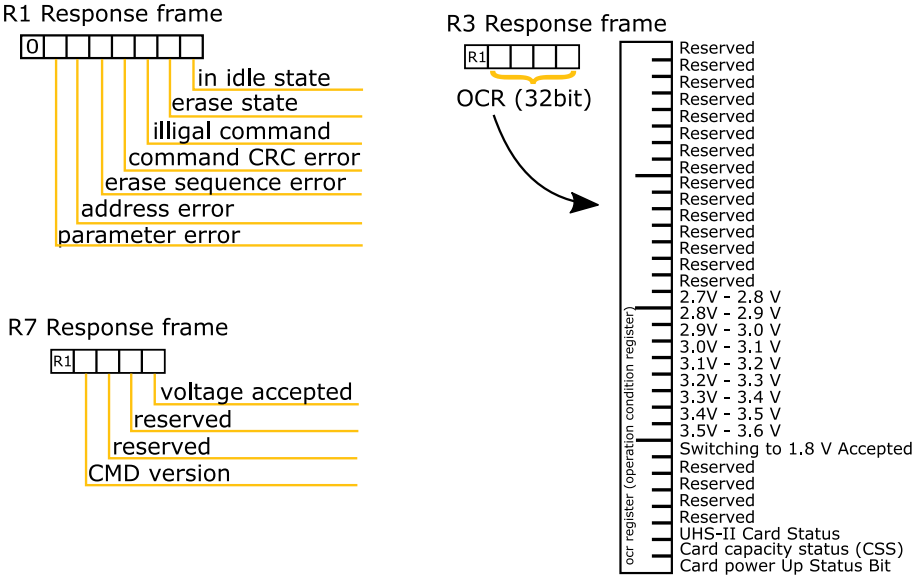


Figure 4.32: SD card responses frames

4.5.4 SD Card initialization sequence

After the power reset the SD card enters its native operating SD mode. There exist a sequence of steps to start communicating with the SD card via SPI bus. First it is necessary to wait a moment (most sources recommend 1 ms or more) after the power switches up to allow stabilization. To “wake” up the SD card it is necessary to send at least 74 dummy clock cycles. Eleven 8-bit clock cycles is send to full-fill this condition (figure 4.33).



Figure 4.33: Waking up the card and setting the SPI mode

After the SD has been waken up, the chip select pin CS is set to low which enables the card. Another two dummy clock cycles (which are always send prior to a command to allow the card to prepare for incoming data) follow and a command CMD0 is send. The structure of the command:

0x40 0x00 0x00 0x00 0x00 0x95

The $(0x95)_h$ is a pre calculated checksum CRC valid for this command [14]. The “I don’t care” bits with value of $(0xFF)_h$ are send to shift the register and read the data send by the card. This value is send until the SD card responds. The SD card returns response R1 with *In Idle State* bit high. The card enters the SPI mode and CRC wont be evaluated anymore. To end the communication an end-of-command clocking with value $(0xFF)_h$ is send and the chip select pin is set to high again.

To be able successfully write and read data it is necessary to check what type of card is used so it can be used with correct parameters. To inquire the card type, the command CMD8 is send with argument $(0x000001AA)_h$ and pre-calculated CRC value $(0x87)_h$ (figure 4.34).

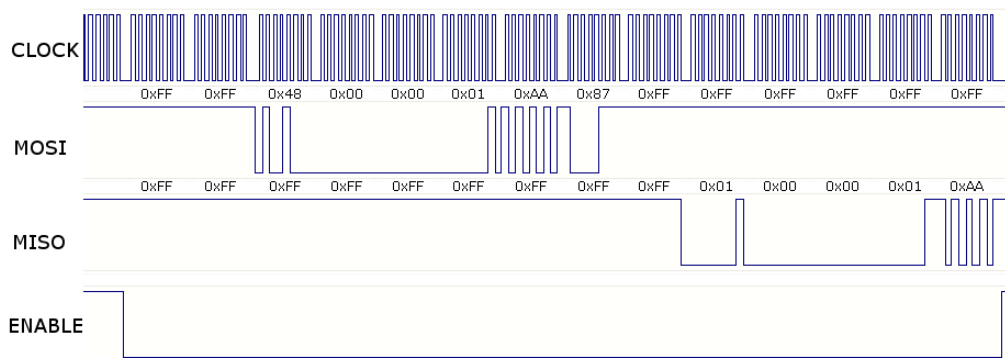


Figure 4.34: Acquisition of the SD card data type

If the card rejects the command with illegal command error $(0x05)_h$, the card type is SDCv1 or MMCv3. If the command is accepted, the R7 response is returned. In this particular case (figure 4.34) is the returned value

0x00 0x00 0x01 0xAA

which shows that the type of the used SD card is SDCv2 [49].

Second to last part of the initialization is sending command CMD58 to read CCS (Card Capacity Status) bit of OCR (figure 4.32 and 4.35).

0x51 0x00 0x00 0x00 0x00 0x01

The card responds with R3 containing CSS bit low. This means the card is Standard Capacity SD Memory Card. If the CSS bit was high, the card would be High Capacity SD Card.

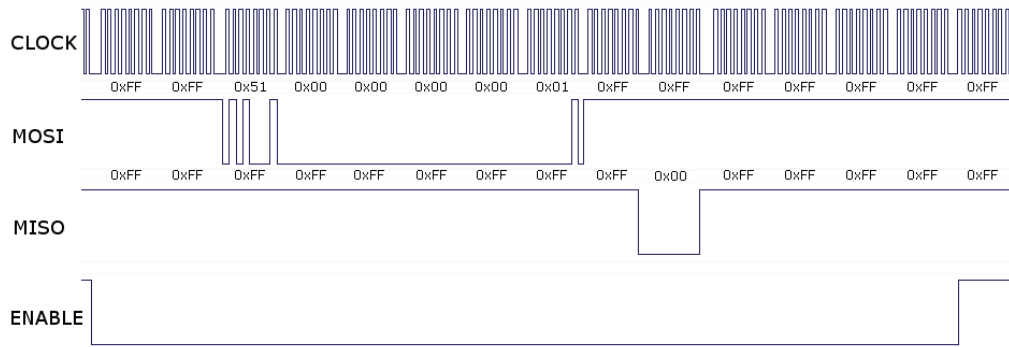


Figure 4.35: Part of the SD card initialization sequence, CMD58

The last command forces block size to 512 bytes so it is possible to work with FAT file system on this type of SD card. This initialization has prepared the card for communications (SPI mode and block size configured) and also provided a information about card (type and size) to microcontroller.

4.5.5 Communication with the SD card

After the low level SD card driver is initialized it is possible to start using the FatFs library (mentioned in section 4.5.2) containing functions which can mount a medium, open file, write strings into file, close files etc. The use of those function is straightforward (code 4.7). The data string produced and saved by the code is caught on oscilloscope (figure 4.36) and the decoding of such string is demonstrated. The ASCII table was used to decode individual letters (appendix E).

Code 4.7: Use of FatFs

```

FATFS FS; //storage medium variable
FIL fil; // variable for file
FRESULT fres;// variable for saving results of operations
// mount the card
f_mount(&FS, "SD:", 1);
// try to open file */
if ((fres = f_open(&fil, "SD:test_file.txt", FA_OPEN_ALWAYS |
FA_READ | FA_WRITE)) == FR_OK) {
    //format string
    sprintf(buffer, "ULT :");
    f_puts(buffer, &fil);
    f_close(&fil);
}
//unmount the card
f_mount(NULL, "SD:", 1);

```

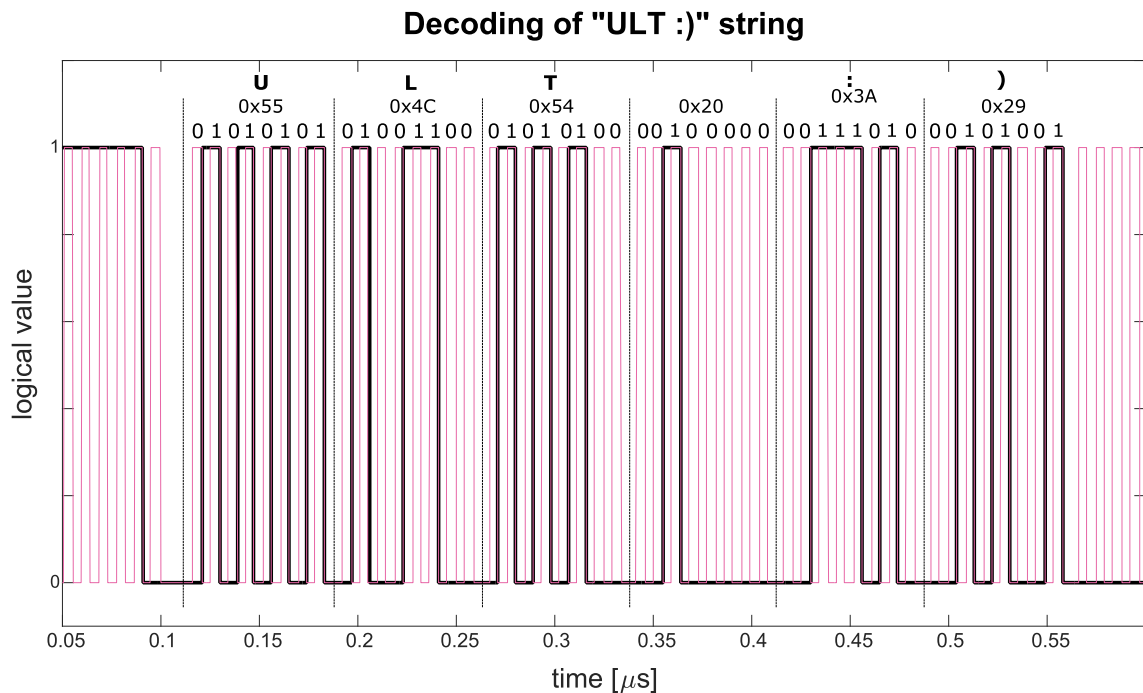


Figure 4.36: Decoding of data saved from oscilloscope containing a string

4.5.6 SD card library for saving text files

To store the measured value, an interrupt caused by timer is triggered and a function to write measured data to the end of the corresponding file is called. Every digital sensor has its own text file with header containing the information about sensor, date and time, units etc. The analogue sensors evaluating air quality share one file. Example of a created file can be found in appendix D on figure D.1.

The flow graphs of initialization on a file and write of a single result of the measurement can be found of figure 4.37.

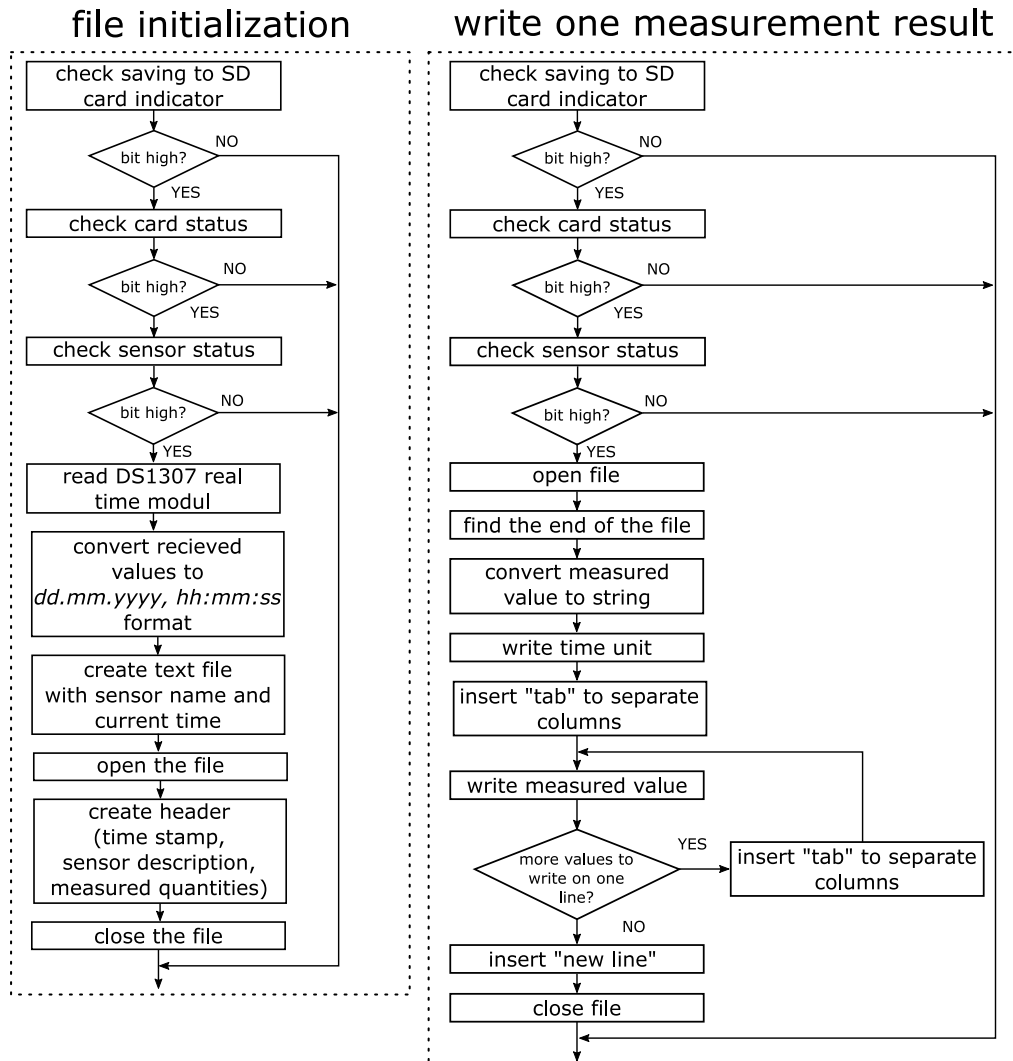


Figure 4.37: File initialization (left) and write to the file (right) flow graph

The libraries containing functions for formatting the text output can be found on the attached CD (files *write_sd.c* and *write_sd.h*) as well as the low level driver and FatFs files (*fatfs_sd.c*, *fatfs_sd.h*, *diskio.c*, *diskio.h*, *ff.c*, *ff.h*, *ccsbc.c*, and *ffconf.h*).

4.5.7 Real time clock DS1307 IC

There is a real time stamp in every measurement file. To acquire the time information RTC (Real time clock) DS1307 is used. STM3F4xx devices has internal calibrated oscillator but the external oscillator is more accurate and the whole microcontroller doesn't have to be powered all the time.

The DS1307 is low-power, full binary-coded decimal clock/calendar real-time clock with 56 bytes of non-volatile SRAM with I²C communication. The correction

of leap year and date adjustment for months with less than 31 days is implemented by the manufacturer. The DS1307 has build-in circuit detecting loss of supply power and switches to the backup supply power automatically. The backup supply can be provided by any standard 3 V lithium cell or any other energy source. According to the datasheet [20] the DS1307 will be back-up for 10 years when a lithium battery with at least 48 mAh is used.

DS1307 interfacing

The interfacing DS1307 real-clock to the microcontroller is shown on figure 4.38. The DS1307 communicates over I²C and requires standard external 32.768 kHz oscillator with specified load capacitance of 12.5 pF.

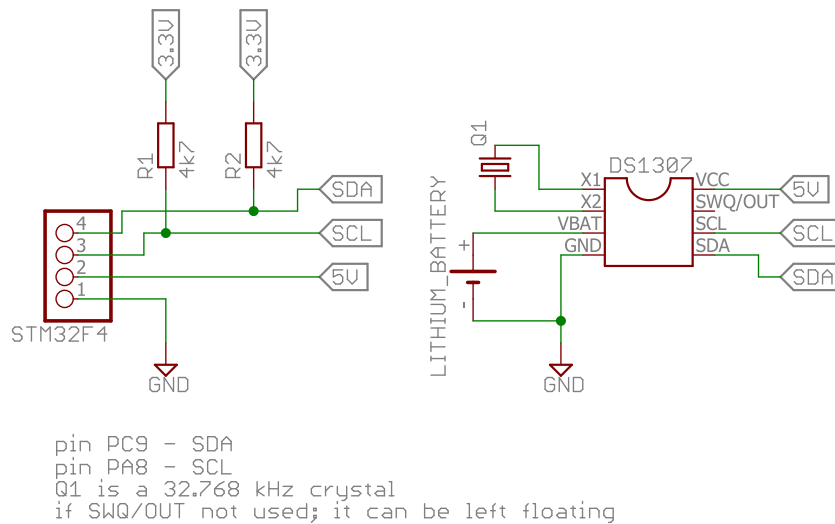


Figure 4.38: DS1307 connection

DS1307 communication

The data are saved in seven registers with addresses (0x00)_h to (0x07)_h (table 4.7). The seventh bit of the first register (0x00)_h *CH* has to be cleaned to enable the oscillator. The time can be read both in 12 and 24 hours format.

Table 4.7: DS1307 registers [20]

Function	Address	Bits							
		7	6	5	4	3	2	1	0
Seconds	0x00h	CH	10 Seconds				Seconds		
Minutes	0x01h	0	10 Minutes				Minutes		
Hours	0x02h	0	12	10 Hour	10 Hour	Hours			
			24	PM/AM					
Day	0x03h	0	0	0	0	0	Day		
Date	0x04h	0	0	10 Date			Date		
Month	0x05h	0	0	0	10 Month	Month			
Year	0x06h	10 Year				Year			

On first application of power is the device set to 01.01.2000, 00:00:00. It is necessary to set the current time and calendar date. This setting is necessary only once as the backup power wont allow any data loss. After the initialization, the current time and date can be read. The I²C address of the device is (0x68)_h. The example of reading the time is on figures 4.39 and 4.40.

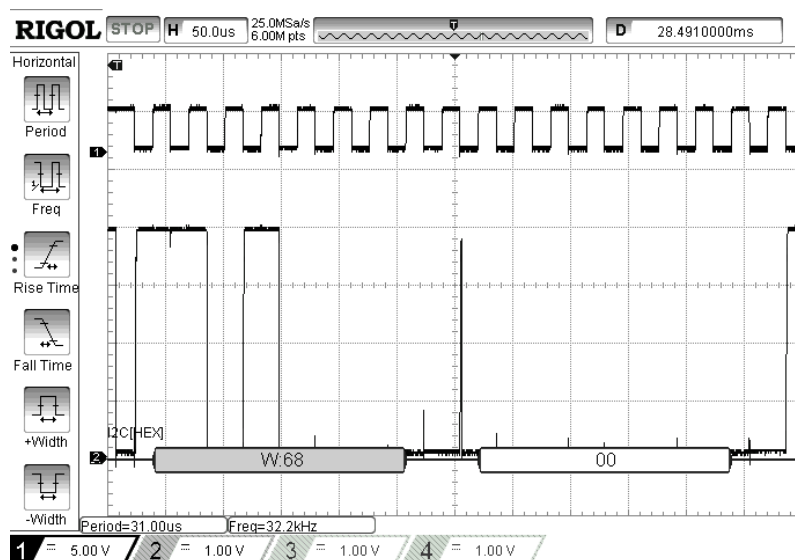


Figure 4.39: DS1307 request for register reading

The time data acquisition process precedes request for reading registers. The request is composed of sending addressing the first register (0x00)_h with device address (0x68)_h and write bit *W* (as seen on figure 4.39). This is followed by the device address with read bit *R* and the device then starts returning the data. In this case a multi-memory read approach is used so it is not necessary to address

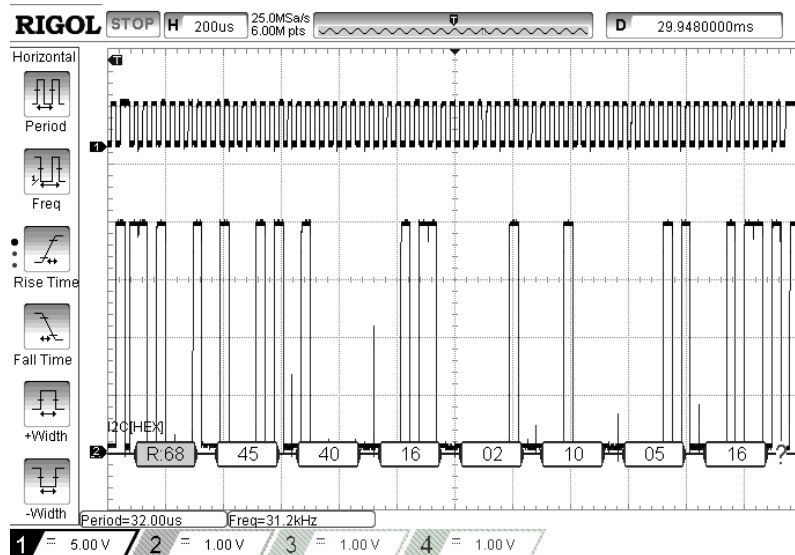


Figure 4.40: DS1307 time data acquired

each individual register for the full time and date acquisition. With the use of the table 4.7, it is possible to read the returned data as follows.

16 : 40 : 45, *second day (tuesday)*, 10.5.201

Please note that the oscilloscope is set to show hexadecimal values for example the value of hours “16” is in fact $(0x16)_h$ which translates as $(22)_d$. This is consequence of the data saved in the register in BCD (Binary Coded Decimal) format. Thus is necessary to convert the values into the required format to enable processing of the time data and to prevent wrong reading. The BCD is a system of writing numbers which assigns a four-digit binary code to each number from zero to nine. The numbers larger than nine are expressed digit by digit. For example number $(25)_d = (11001)_b$ is in BCD expressed as:

$$\begin{array}{cc} \underbrace{0010} & \underbrace{0101} \\ (2)_d & (5)_d \end{array}$$

The BCD offers compromise before machine-readable and human-readable numerals. The conversion (code 4.8) is done by adding of the two nibbles (four bites) from which the more significant nibble must be multiplied by ten (to move one order up).

Code 4.8: Conversion of BCD format to decimal integer

```
// returns converted BCD number
uint8_t bcd2int(uint8_t bcd) {
    uint8_t dec = 10*(bcd>>4);
    dec += bcd & 0x0F;
    return dec;
}
```

4.6 LCD display

For the representation of the measured data and for communication between user and the microcontroller a 3.5 inch touch LCD display was selected (table 4.8).

For the connection of the display to the microcontroller was used STM32F4DISCOVERY Base Board.

Table 4.8: STM32F4DIS-LCD display specifications

STM32F4DIS-LCD	
Size	3.5 inches LCD board
Driving IC	SSD2119
Display format	320x240
Color	262K colors
Interface	16-bit 8080 parallel system interface
Touch screen	4-wire resistive touch screen
Driving IC for touchscreen	STMPE811

4.6.1 LCD interface

The display is connected using 16-bit 8080 parallel interface. The signals for controlling LCD are divided into two types: control signals and data signals. In this particular case there are sixteen data lines. The four control signals WR, RD, DC, CS (all active low) define the type of operation. Combination of different levels of those signals generates desired operation (table 4.9).

Table 4.9: The functions of 8080 parallel interface

WR	RD	DC	CS	operation
1	0	0	0	read 8-bit command
1	0	1	0	read 8-bit parameter
0	1	0	0	write 8-bit command
0	1	1	0	write 16-bit display data

CS (chip select) signal line must be pulled low to perform any operation. RD and WR are read and write lines. In other words, if the write operation is required, the WR line must be pulled down and the RD has to be hold high and similarly for the read operation. The DC line determines if the data are intended to be written into the display data RAM or into the internal command register.

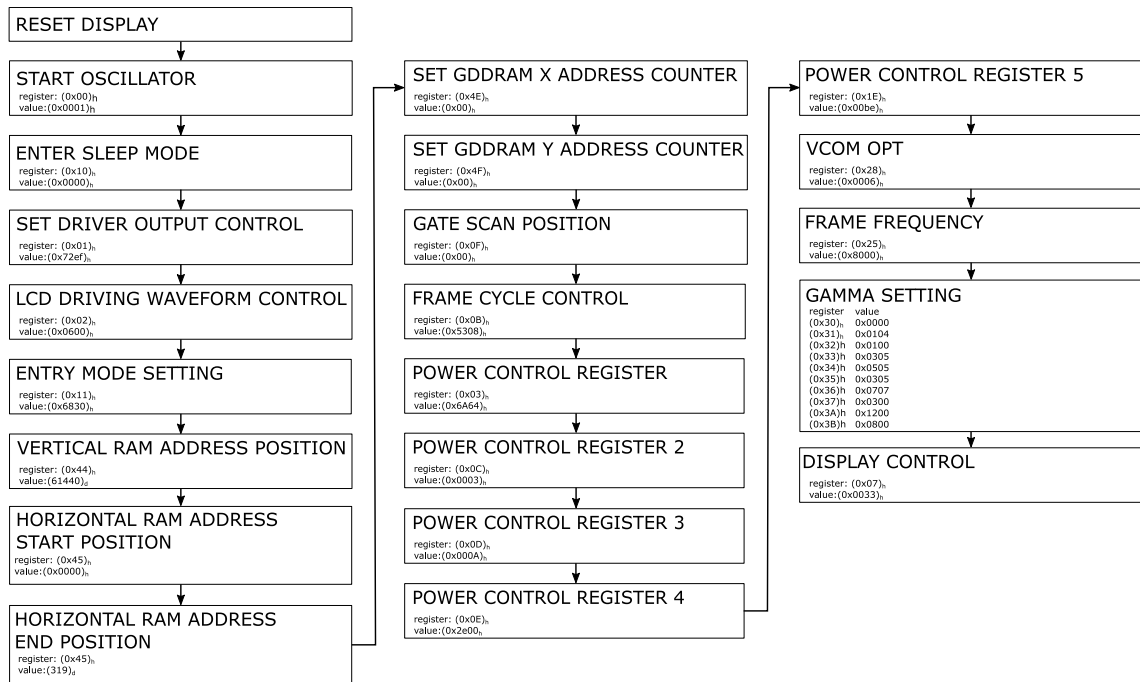


Figure 4.41: SSD2119 driver initialization

4.6.2 SSD2119 TFT LCD driver

SSD2119 is an all in one TFT LCD driver which can be used to drive panel with up to 262k color and resolution of 320x240 pixels. The LCD display can be interfaced via 8/9/16 bit 6800 parallel interface, 8/9/16-bit 8080 parallel interface or SPI.

The driver comes with the LCD display on the board but an initialization is needed. The LCD is designed to be used with 16 bit 8080 parallel interface thus it is necessary to set the SSD2119 driver to the same mode. As can be seen from the LCD internal connection scheme [45], SSD2119 is already connected in 16-bit 8080 parallel mode. The mode selection pins PS[3:0] are set to combination (0010)_b using combination of resistors. According to the table 6-3 in SSD2119 datasheet [43] is this combination used for 16-bit 8080 parallel mode therefore correct for this application.

The initialization flow graph can be seen on figure 4.41. The initialization starts with resetting the display. The oscillator is started and display is put into the sleep mode because some of the register are not accessible in the normal mode. Driver output control register sets color mode to RGB (contrary to default BRG setting). Waveform control register activates the line inversion to help to suppress the flicker [4]. The entry mode register deactivates pins HSYNC/VSYNC which under normal conditions take control of position synchronization. Instead is the window of the RAM buffer defined using fixed values. Gate scan position assures that the first line is on the very top of the display. This parameter can be used e.g. to protect a header (including e.g. company logo or time and date) which should remain the same all the time the LCD is on from overwriting. Rest of the registers was set to with help of the datasheet to achieve good displaying quality without flickering.

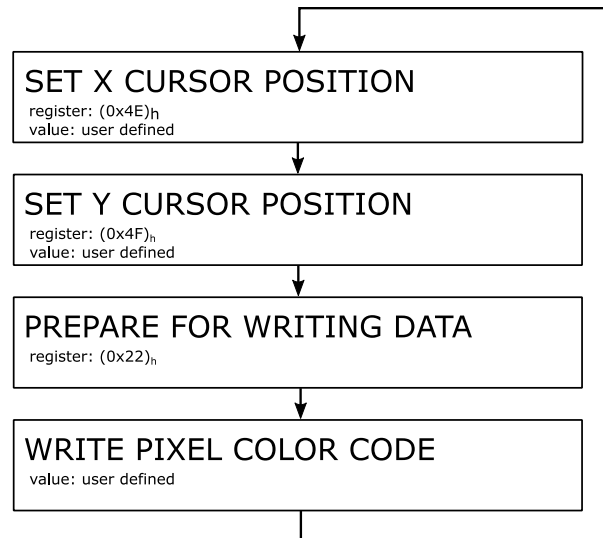


Figure 4.42: Flow diagram of displaying a single pixel

After the initialization the LCD is ready to display a pixel with given color on given position. The process of displaying a pixel is shown on figure 4.42.

The display with given initialization requires color in RGB565 color code thus is necessary to convert the values before sending them to the display (code 4.9).

Code 4.9: Conversion from RGB888 to RGB565

```

RGB565(R ,G, B) (((R)& 0xF8) << 8) | (((G) & 0xFC) << 3) | (((B) &
0xF8) >> 3))
  
```

Graphic elements

After the function for writing a single pixel with defined color on the desired position on the screen is ready, it is possible to start creating functions to display text and graphic. There exist two options how to solve this problem. First one is to use prepared commercial libraries such as uGFX*, SEGGER emWin† or StemWin‡. With use of the prepared libraries is the displaying of various graphic elements easy but they also have drawbacks. The first two mentioned are not free for commercial use and StemWin is not an open source project. Hence a simple small graphical library is written. The functions display text, buttons and various geometric figures. There is an example of drawing line with given parameters in the snippet 4.10).

* <http://ugfx.org/>

† <https://www.segger.com/emwin.html>

‡ www.st.com/resource/en/application_note/dm00089670.pdf

Code 4.10: Drawing line on the display

```
#define SSD2119_X_RAM_ADDR_REG    0x4E
#define SSD2119_Y_RAM_ADDR_REG    0x4F
#define SSD2119_RAM_DATA_REG      0x22

void LCD_WriteRAM_Prepare(void){
    LCD_CMD = SSD2119_RAM_DATA_REG;}

void LCD_WriteReg(uint8_t register, uint16_t value){
    LCD_CMD = register; // choose register to write into
    LCD_Data = value;} // send the required value

void LCD_WriteRAM(uint16_t RGB_Code){
    LCD_Data = RGB_Code;} // write into 16-bit register of LCD RAM

void LCD_SetCursor(uint16_t Xpos, uint16_t Ypos){
    LCD_WriteReg(SSD2119_X_RAM_ADDR_REG, Xpos); // set X address of
    the cursor
    LCD_WriteReg(SSD2119_Y_RAM_ADDR_REG, Ypos);} // set Y address of
    the cursor

void LCD_drawHorizontalLine(uint16_t Xpos, uint16_t Ypos, uint16_t
length, uint16_t color){
    uint32_t i = 0;
    LCD_SetCursor(Xpos, Ypos); // set cursor to the start of the line
    position
    LCD_WriteRAM_Prepare(); // prepare for writing into the LCD RAM
    for(i = 0; i < length; i++) {
        LCD_WriteRAM(color);
    }
}
```

4.6.3 STMPE811 TFT touchscreen controller driver

4-wire resistive touch screen

The 4-wire resistive touch screen consists of three layers (figure 4.43). There is a rigid substrate made usually from glass or acrylic. Surfaces are coated with a transparent conductive film (e.g. indium tin oxide). The conductive layers are normally not touching thanks to the insulating spacers on the edges and in-between the two layers. There is a flexible membrane made from PET (polyethylene terephthalate).

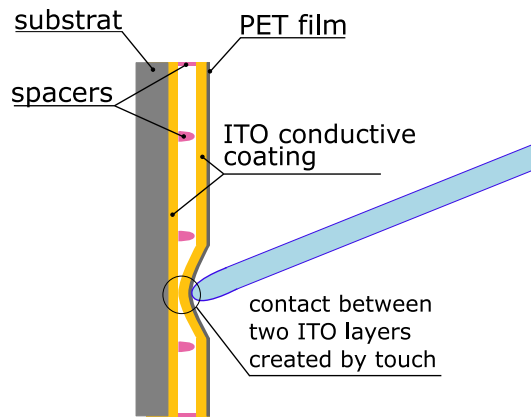


Figure 4.43: Resistive touch screen

If there is a touch present on the PET layer, a contact between the two layers is made. The 4-wire technology uses pair of electrodes for each layer. The electrodes are often referred as bus-bars. The bus-bars are placed perpendicularly and the four wires are referred as X+,X-,Y+ and Y-. The reading of the touch position in y axis is done by driving Y+ high, Y- low and reading X+. Similarly the x coordinate can be obtained. The STMPE811 driver, if properly initialized, has ability to measure those voltages and convert them to a digital values thanks to the build-in ADC.

STMPE811 initialization

The STMPE811 can communicate on both I²C and SPI bus, but the wiring on the LCD display board makes only I²C bus accessible. The address of the driver is (0x82)_h and the driver is running on a different I²C bus then the sensors (marked as I²C3 in the STM32F407 peripherals list). The reason is that if the touch occurs at time of any other I²C communication already in progress, the reaction on touch can appear as delayed. In the first step of the initialization (figure 4.44) are the ADC and TSC (Touch Screen Controller) clocks activated. The next step, setting the ADC, is crucial for correct working of the touch screen controller. The ADC resolution, conversion time and clock speed are set to 12-bits, 80 clock cycles and 3.25 MHz.

Next step is configuring TSC. In the datasheet note there is a recommendation for setting the settling time for panels larger then six inches as 1 ms. Since the used LCD panel is half size, the settling time was selected as 500 μs. The resulting position is set to be calculated as an average of last four readings and will be treated as single point reading. This setting shows the best result concerning the precision and speed of the touch screen response. The range of the pressure measurement is set high and the accuracy low because in the given application only detection of the touch is necessary and not the evaluation of the pressure used (the evaluation of pressure force is used e.g. in painting programs). The maximal driving current is set to maximal value of 50 mA because the unit is not battery driven and there is no need for extreme power savings. At last the interrupts are enabled. The controller has one dedicated pin output to generate interrupts in case of touch event so the

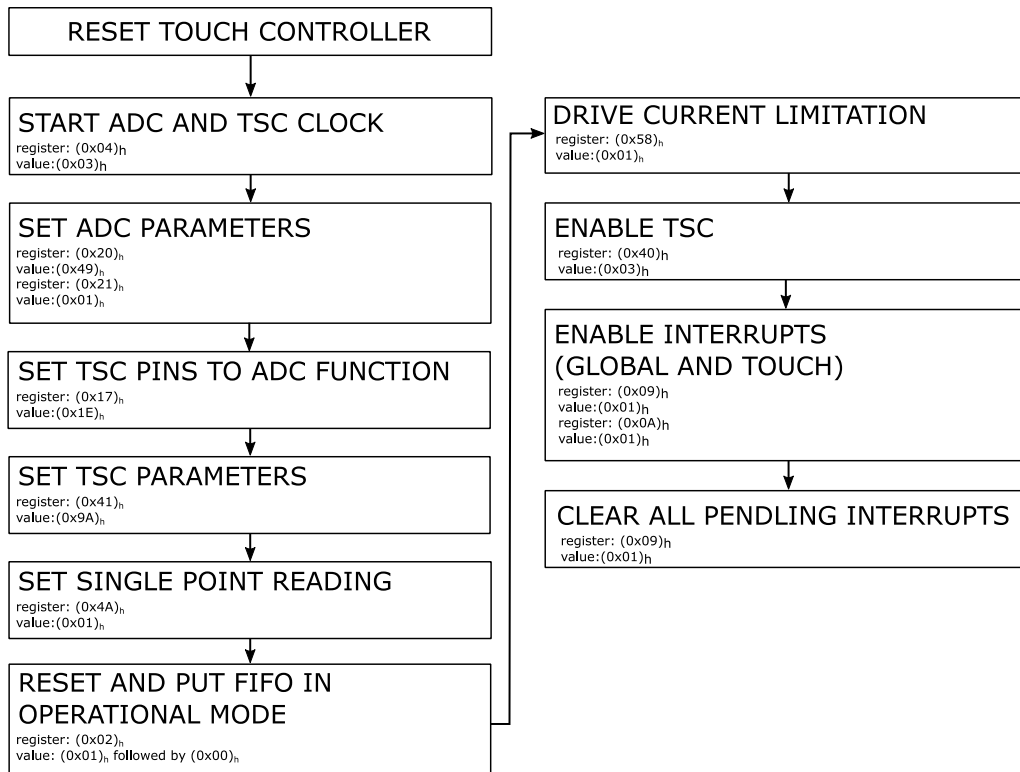


Figure 4.44: STMPE811 Init

microcontroller doesn't have to read new values of ADC converter all the time. The interrupt line is set to be active low and set to level interrupt. The same setting has to be done in the microcontroller interrupt control.

STMPE811 calibration

Since the controller has no information about the size and resolution of the connected display, the calibration must be done manually. The raw value returned by the touch controller in the corners was measured (figure 4.45). The LCD dimensions are known, the output of the ADC is linear and the correct position can be calculated by cross-multiplication.

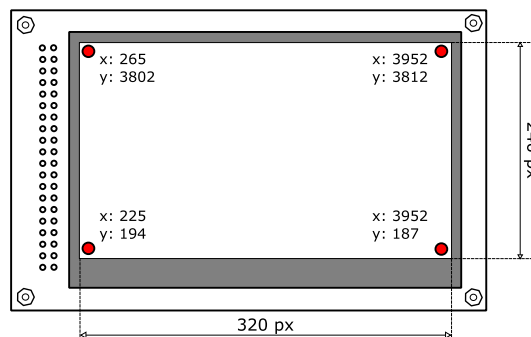


Figure 4.45: STMPE811 calibration

Moreover the corrected values are scaled to bring the beginning of the coordinate system into the upper left corner so it corresponds to the beginning of coordinate system.

STMPE811 reading touch position

The flow graph of reading the touch position is on figure 4.46. In the interrupt routine is the position stored in the touch controller FIFO read, converted and scaled. The defined buttons are tested one by one. If the button is enabled, the position of the touch is compared with the inside area of the button. If it corresponds, the button is marked as pressed and the routine ends, interrupts flags are cleared and the microcomputer waits for another interrupt.

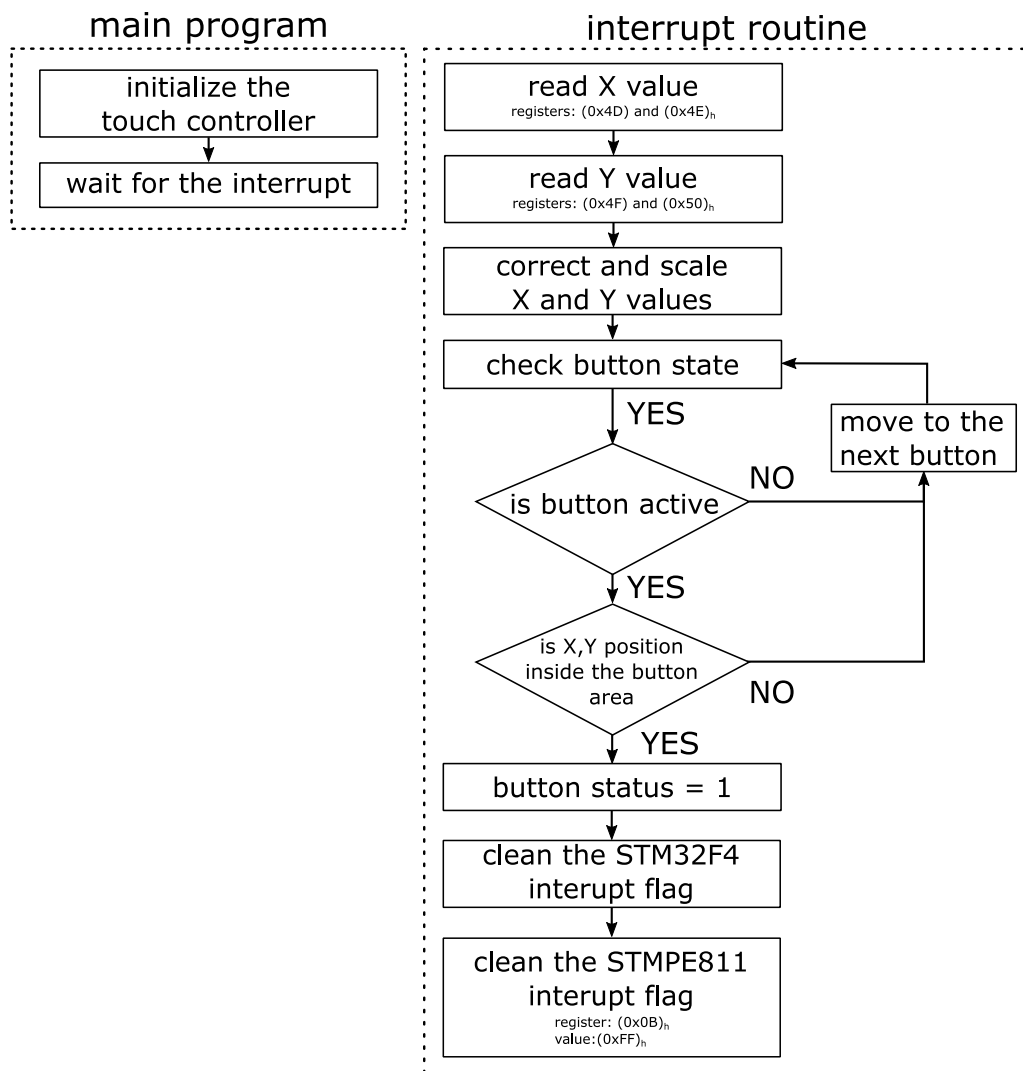


Figure 4.46: Reading touch position flow graph

4.6.4 Buttons

After the proper initialization of both LCD display and the touch controller it is possible to create an interface for communication with the user. One of the element of such interface is a button. As mention in section 4.6.2, no prepared graphical library was used thus the buttons have to be designed. Every button is an element of a structure with following arrangement (code 4.11).

Code 4.11: Structure for holding button parameters

```
struct buttons {
    // position of the button left upper corner on the display
    uint16_t xPosition;
    uint16_t yPosition;
    // size of the button
    // if ySize or xSize == 0 the size is calculated automatically
    // based on the button text
    uint16_t ySize;
    uint16_t xSize;
    uint16_t backColor;    // color scheme for the button
    uint16_t textColor;   // color scheme for the button
    sFONT *fonts;         // font size
    char text[40];        // text of the button
    // area in which is the button active
    uint16_t xTouchMin;
    uint16_t xTouchMax;
    uint16_t yTouchMin;
    uint16_t yTouchMax;
    uint8_t volatile enable; // enable = 1 -> button is active
    uint8_t volatile status; // status = 1 -> button is pressed
};
```

The function which takes elements of the structure and uses them to display the button on the display is demonstrated in appendix F (code F.1).

4.7 Ethernet connection

The created program for transmitting data via Ethernet demonstrates the possibilities of the Ethernet controller. However, it is not fully implemented into uLite unit and the transmission of measured data via Ethernet to another device is not yet possible. The first reason is that the receiving application for the transmitted data is not written. It is possible to send data and display them in the web browser on given IP (Internet Protocol) address but it is not possible to save them. Another problem is connected with use of the STM32F407 Discovery Board. The

Ethernet module is communicating via SPI and requires fast responses from the microcontroller. On the STM32F407 Discovery Board there are several peripherals already connected to the SPI bus (accelerometer, gyroscope and audio driver). All the connected devices increase the RC constant which is slowing down the bus, delay responses of the microcontroller and cause lost of packets and thus very unreliable communication which freezes occasionally. Moreover the used breakout board with ENC28J60 Ethernet controller also contains resistors and capacitors which makes the communication even more unpredictable. The program works without problem on modules such as STM32Fxxx Nucleo Boards which are mend to be used for deployment and it would work with the microcontroller placed on a dedicated board as well.

4.7.1 Transmission Control Protocol

The used Ethernet controller already contains the Medium Access Controller layer (MAC) and Physical Layer (PHY). Thanks to the build-in network stack and transmitting/receiving buffers the controller takes care about most parts of the communication. The following chapter covers only basics of TCP (Transmission Control Protocol) protocol which was used to support HTML (HyperText Markup Language) application layer.

TCP is a transport layer supported by IP. It offers connection-oriented, reliable, full-duplex and error checked delivery of data between applications communicating over an IP network. The connection-oriented means that a virtual connection must be establish before any data can be transmitted [32]. Every transmission is acknowledged by the receiver and if the sender doesn't receive the acknowledgement within a specific time frame, the data are resended. The communication with use of TCP is full-duplex. The data travel in segments containing the data bytes and control information identifiers. Example of such segment is on figure 4.47 (redrawn based on [36]). The description of individual parts of the TCP header are in table 4.10.

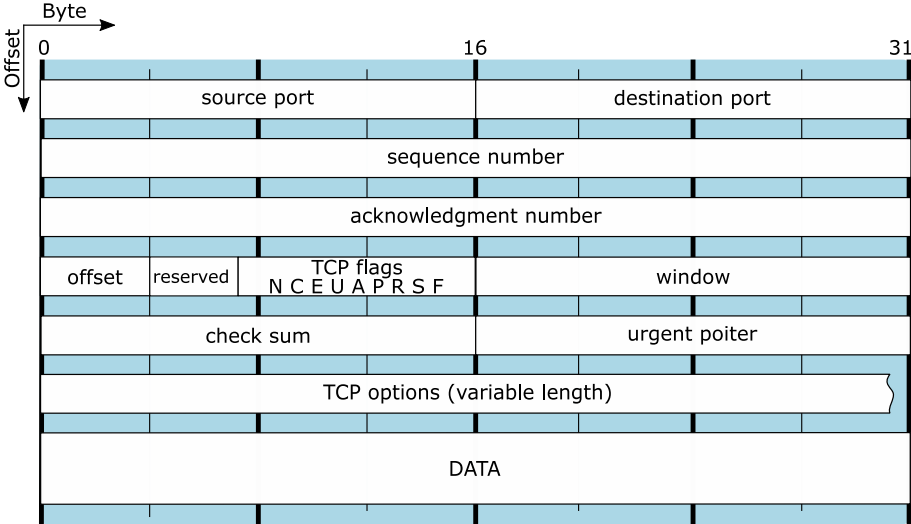


Figure 4.47: TCP segment

Table 4.10: Fields in TCP header [36]

Field	Length	Function	
source port	16 bits	The port number at the source side.	
destination port	16 bits	The port number at the receiver side.	
sequence number	32 bits	Identifies the data segment position in the stream of already send segments.	
ack. number	32 bits	Acknowledged sequence number.	
offset	4 bits	Number of 32-bit words in the TCP header.	
reserved	3 bits	Reserved for later use.	
flags	9x1 bit	NS	ECN-nonce (Explicit Congestion Notification) concealment protection (experimental).
		CWR	If set, the received TCP segment has responded in congestion control mechanism.
		ECE	Indication of TCP seed ECN capability.
		URG	If set, the urgent pointer field contains data for the receiver.
		ACK	If set, the acknowledgement field contains data for the receiver.
		PSH	If set, the data shall be send without buffering.
		RST	If set, the sender is requesting connection reset.
		SYN	If set, the synchronization of sequence number between two nodes is required by the sender.
		FIN	If set, the send segment is last in a sequence and communication should be closed.

TCP implementation

Before sending any data with use of TCP protocol a correct header has to created. For the creation of valid packets is used a library created by Guido Socher “IP, ARP, UDP and TCP functions” which is available under GLP V2 license. Both files contained in this library can be found on the attached CD (*ip_arp_udp_tcp.h* and *ip_arp_udp_tcp.c*). It is possible to use arbitrary TCP/IP stack such as lwIP*.

4.7.2 ENC28J60 Ethernet controller

The ENC28J60 (figure 4.48) is a stand-alone Ethernet controller with SPI interface. This controller is used in form of a break-out board commonly sold for developing

* <http://en.wikipedia.org/wiki/LwIP>

applications with Ethernet connection. It already contains all external elements needed for interfacing the controller (crystal, resistors, capacitors and RJ-45 connector). The ENC28J60 meets all the IEE 802.3 specifications. It also provides an internal DMA (Direct Memory Access) for faster data transfer and internal checksum calculator. It has two dedicated LEDs to indicate network activity indication. This Ethernet controller doesn't come with its own MAC address. It can be set manually but if the device was used commercially it would be necessary to purchase a registered MAC address (block of 4096 MAC addresses costs about \$665*). As

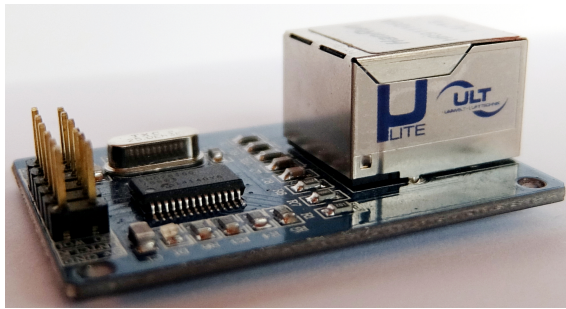


Figure 4.48: ENC28J60 Ethernet controller break-out board

already mentioned, the break-out board already contains all necessary components to make the ENC28J60 chip work. It is connected to microcontroller SPI pins and to one GPIO pin to accept interrupts from ENC28J60 (figure 4.49).

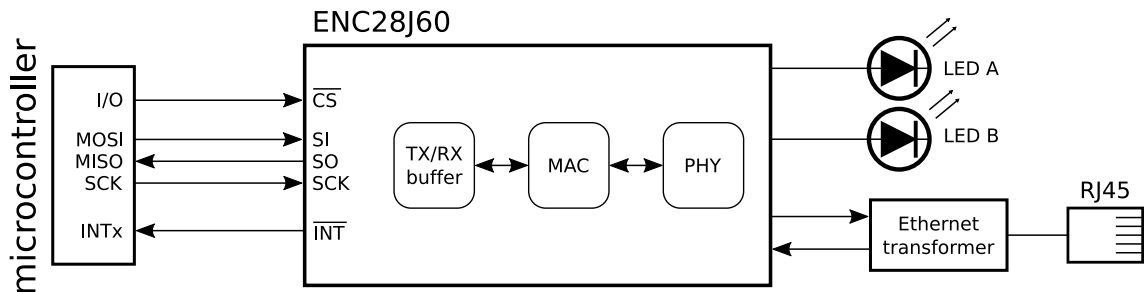


Figure 4.49: ENC28J60 interface

ENC28J60 Ethernet controller communication

There are three memory types in ENC28J60 Ethernet controller: control registers, Ethernet buffer and PHY registers.

The control registers are divided into four banks. Each bank is 32-bytes long. The control registers are directly read and written to by the SPI interface.

The 8Kbytes Ethernet buffer contains transmit and receive buffer. The sizes of those areas are programmable via SPI by the host controller (the part of the buffer which is not programmed as a receive buffer is automatically considered to be a

* <https://standards.ieee.org/develop/regauth/oui36/index.html>

transmit buffer). The receive buffer is a circular FIFO buffer completely managed by the hardware. Both buffers are addressed with several pointers indicating start, end or read position (for the complete list see the device datasheet [35]). Those buffers can be only accessed with use of read buffer memory and write buffer memory SPI commands (table 4.11).

The PHY registers are used for PHY module control and status retrieval. Those registers are not accessible through the SPI interface and they must be addressed with Media Independent Interface Management (MIIM) which is part of the MAC.

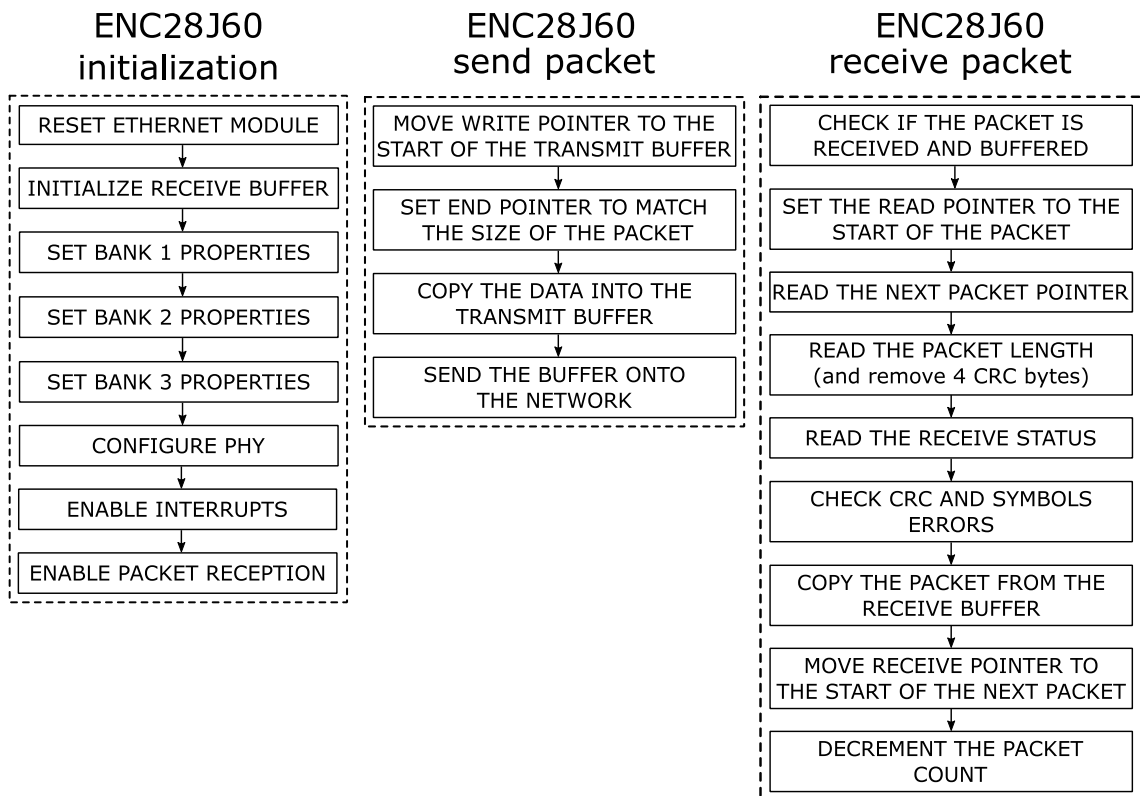


Figure 4.50: ENC28J60 communication flow graph

The initialization of ENC28J60 (figure 4.50) starts with the chip reset. The receive buffer is initialized for 16-bit transfers. In Bank 1, it is possible to set filters for packets. In this case, only ARP (Address Resolution Protocol) packets are allowed for broadcast. Bank 2 contains registers for allowing and setting up the MAC parameters. The automatic padding of packets to 60 bytes and CRC control is enabled. In Bank 3, the actual MAC address is saved. The PHY layer is configured, the interrupts are enabled, and the reception of the packet is allowed.

When sending a packet, the pointer is moved to the start of the transmit buffer and set to the end pointer according to the packet length. The packet data are then copied into the buffer and sent out. When reading a packet, it is first checked if it is received and buffered correctly. The pointer to the start of the following packet is read and saved. The length of the incoming packet is also read and shortened by four bytes containing CRC. The receive status vector is read and checked.

for errors. If there are no errors, the data are copied from the receiver buffer and the read pointer is moved at the beginning of the next packet (using the previously saved position) to free the memory. The packet counter is decreased to indicate successful operation.

Table 4.11: SPI instruction set for the ENC28J60 [35]

Instruction name	Byte 0		Byte 1
	Opcode	Argument	Data
Read Control Register (RCR)	0 0 0	a a a a a	N/A
Read Buffer Memory (RBM)	0 0 1	1 1 0 1 0	N/A
Write Control Register (WCR)	0 1 0	a a a a a	d d d d d d d d
Write Buffer Memory (WBM)	0 1 1	1 1 0 1 0	d d d d d d d d
Bit Field Set (BFS)	1 0 0	a a a a a	d d d d d d d d
Bit Field Clear (BFC)	1 0 1	a a a a a	d d d d d d d d
System Reset Command (SRC)	1 1 1	1 1 1 1 1	N/A

a = control register address, d = data

Simple web server with ENC28J60 Ethernet controller

To demonstrate the function of the ENC28J60, a simple web server is created. The data can be accessed with the IP address and password which was set in the program. The result is captured on figure 4.51.

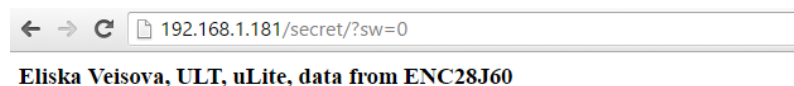


Figure 4.51: Web server created with ENC28J60

The debugging of Ethernet is not possible with the available oscilloscope and logic analyser. However it is possible to observe the packets with software Wireshark (an open source packet analyser). The communication was recorded using this software and the file is available on the attached CD. Data about one packet can be found in appendix M on figure M.1.

Even though the communication cannot be decoded with use of the oscillator it is possible to observe the communication signal. On figure N.1 in appendix N the negative effect of additional devices connected to the SPI bus on the Discovery

Board can be observed. The MMOSI (green) line rise time is very low and despite the microcontroller efforts to communicate the bits are not read correctly. The ENC28J60 evaluates this situation as if the microcontroller was not connected at all and ends the communication. On figure N.2 in appendix N it is possible to see this effect in a large scale.

4.8 Base board

4.8.1 Base Board schematics

Base Board was designed to bring all the parts described in the previous sections together on one PCB board. The schematics can be found in appendix G (figure G.1). The components are easily connectible thanks to the pins on the Base Board. The board contains all the necessary circuit elements. The components are placed in such way that they are close to the point of their working position. The sensors evaluating air quality are located close to the air channel, the external sensors are situated the near of the connectors and the pinheads for connection with microcontroller are right beside the STM32F407 Discovery Board kit.

4.8.2 Base Board PCB

The PCB (figures 4.52 and 4.53) was manufactured as one sided PCB with green unsolderable mask and white silkscreen labelling the components positions by company APAMA. Each element has its own connection cable with exact length to make the connections inside the box more transparent. The board dimension is 97x66 mm. The diameter of the wires is 1.17 mm and 2.54 mm crimp connectors housing were used. The VOC sensor and the dust sensor are connected with JST ZHR-6 and JST XHP-4 Wire-to-board connectors.

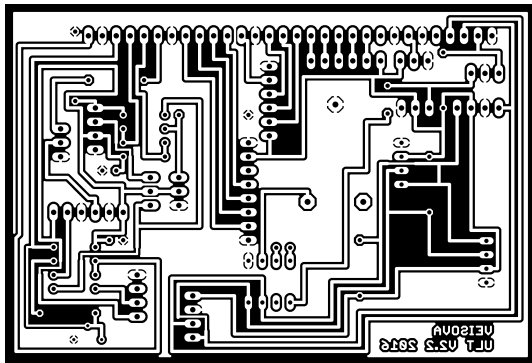


Figure 4.52: PCB bottom layer

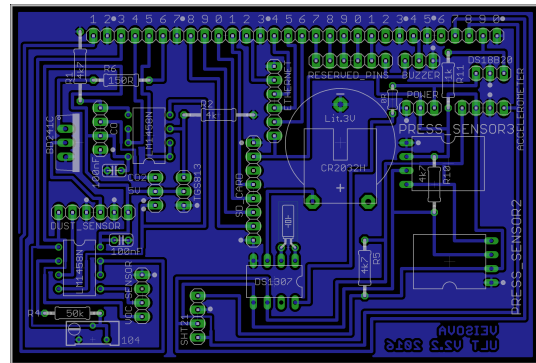


Figure 4.53: PCB design

4.9 Power Supply board

In the development phase was the STM32F407 Discovery Board powered by the same USB cable which was used for programming the microcontroller. The parts requiring external power were powered by laboratory power source with possibility of voltage setting and current limitation. To make the project independent of the laboratory source and USB power, the power supply solution is designed.

The components on the base board are power by either 3.3 V or 5 V. One of the gas sensor (CO) requires specific voltage cycling of 1.4 V and 5 V. Due to the use of operational amplifiers it is necessary to provide symmetrical power supply. All the requirements are summed up in the following item list.

- 5 V supply for gas sensors, dust sensor, DS18B20 temperature sensor, real time clock DS1307, LCD display.
- 3.3 V supply for SHT21 temperature sensor, ADXL345 accelerometer, AMS5915 pressure sensor, SD card reader, LCD display.
- Symmetric supply for operational amplifiers ± 5 V.
- Alternating voltage 5 V/1.4 V for the heater of CO sensor.

4.9.1 Power Supply board schematics

The schema of the power module can be found in appendix H (figure H.1). The main power input to the unit is 12 V provided by switch mode power supply for 240 V AC input with 12 V direct voltage output, maximum output power of 2 A and standard 5.5/2.1 mm output connector. It is possible to power the uLite unit with up to 30 V. There is a medium time lag 2 A fuse on the input of the power module board to protect the rest of the circuit. The input voltage is regulated by the positive adjustable voltage regulator LM317TS in To-220 package to approximately 8 V with maximal output current of 1.5 A. This step prevents overheating of the following regulators and provides stable power. The nominal output voltage of this regulator is selected by means of a resistive divider (equation 4.11 and figure 4.54).

$$U_{out} = U_{ref} \cdot \left(1 + \frac{R_2}{R_1}\right) + I_{adj} \cdot R_2 \quad (4.11)$$

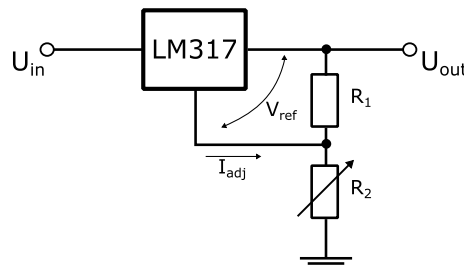


Figure 4.54: LM317 in adjustable regulator mode

As stated in the datasheet [33], LM317 provides an internal reference voltage U_{ref} of 1.25 V. The current I_{adj} is designed to be very low with maximum value of 100 μ A. Therefore the error term containing I_{adj} in equation 4.11 can be neglected. For this application was R1 selected as 470 Ω and R2 was set with use of multimeter to achieve approximately 8 V on the output. The capacitors C6 and C7 are placed to smooth drops in supply voltage lines and to filter 50 Hz. Capacitance C5 improves transient response of the regulator. There are two fixed voltage regulators REG1117, one providing 3.3 V and the second 5 V. These voltage regulators require input and output capacitors (C1, C2, C3, C4) [1]. There is one more REG1117 regulator designed to cycle voltages for the CO gas sensor. As the transistor T1 BC817 opens and closes, the ratio of the resistor divider is changed, setting either 1.4 V or 5 V to the output.

Symmetrical power supply can be created by use of specially designed silicon chips such as TLE2426* from Texas Instruments, but those chips can only handle low current of maximum 40 mA. The chips that can handle higher current gets expensive, too complicated, can't provide voltage high enough or come in hand unsolderable packages. The cheap elegant solution is to use operational amplifier in comparator mode (in this case TDA2008V). The two closely matched resistors create a voltage divider at the non-inverting input to the operational amplifier. C10 and R9 forms a low pass filter to eliminate noise internally generated by operational amplifier. C8 and C9 stabilize the output. This connection is designed to provide current up to 3 A.

4.9.2 Power Supply PCB

The PCB was designed as two layer board with dimension of 58x64 mm. The final design can be seen on figures 4.55, 4.56 and 4.57. The board was manufactured by company APAMA with green solder mask coating and white silkscreen on both sites.

4.9.3 Power Supply accessories

To make the powering of the unit user friendly, a button switch and a power jack are mounted to the side of the box. They are fixed with a help of two small PCBs with holes for screws (figure 4.58). Those PCBs were manufactured in home laboratory and the dimension of each PCB is 18x20 mm. This systems allows reliable connection of the switch and power jack to the power supply module.

* <http://www.ti.com/product/TLE2426>

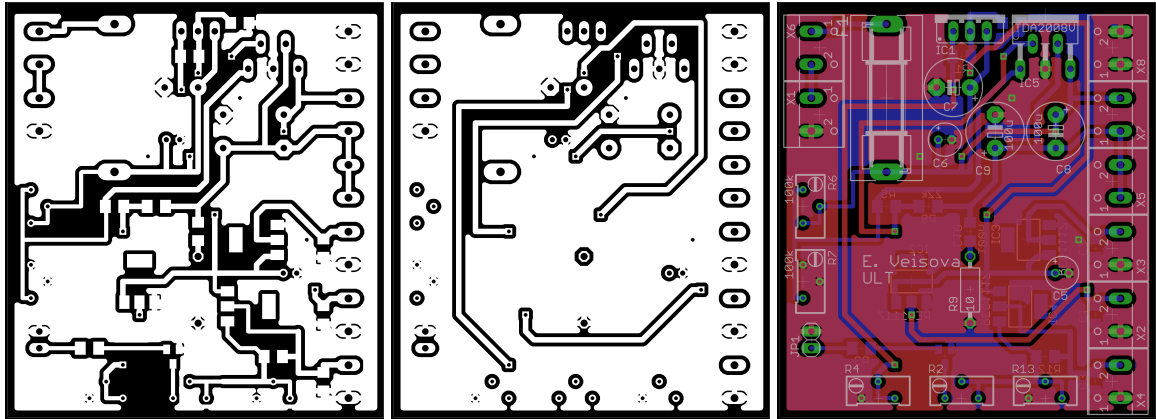


Figure 4.55: PCB bottom layer Figure 4.56: PCB top layer Figure 4.57: PCB design layer

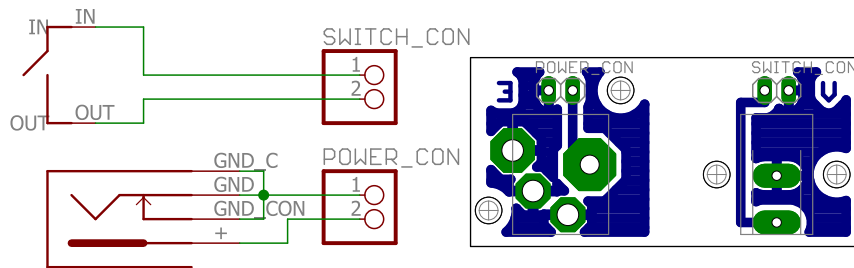


Figure 4.58: Switch and jack schematics + PCB design

4.10 Software for uLite box

4.10.1 Microcontroller

The software solution is based on STM32F407VGT6 in LQFP100 package on the Discovery Board (please see the microcontroller specification in datasheet [44]). The Discovery Boards kits are complete solution for the development of STM32 based application. They have integrated debugger and programmer which makes them ideal for prototyping.

On the other hand when developing on Discovery board it is necessary to carefully select pins and prevent collision with devices already mounted on the discovery board (accelerometer, audio sensor, digital microphone etc.). The used pinout is on figure 4.59. The core clock is set to 168 MHz. The whole clock configuration setting is on figure I.1 in appendix I.

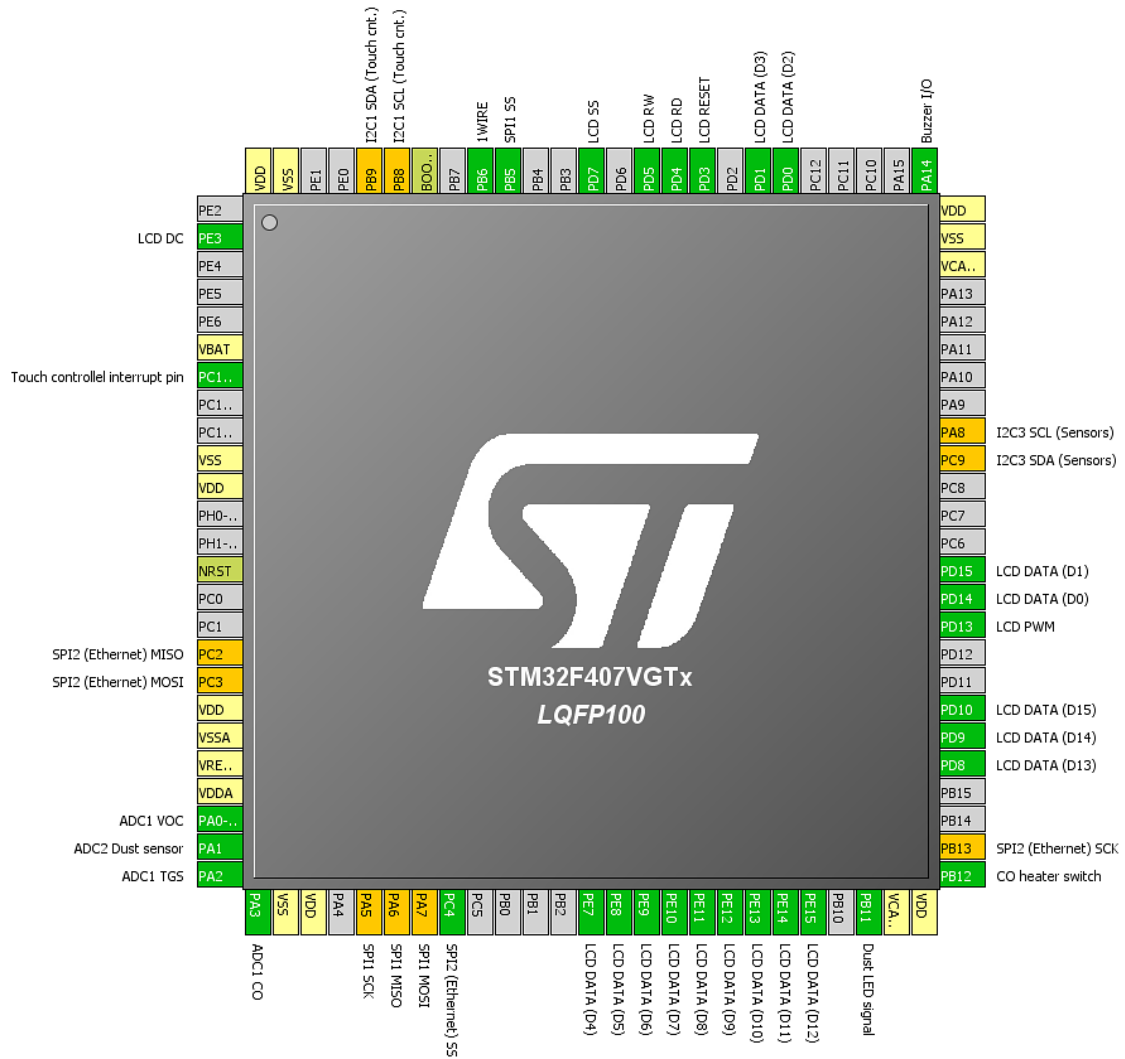


Figure 4.59: STM32F407 pinout

4.10.2 Program structure

The program is composed of three main parts. In the first part the initialization of microcontroller internal and external peripherals take place (figure 4.60 on the left). The second parts prompts user to start the measurement with selected parameters (figure 4.60 on the right) and the third part is the measurement itself (figure K.1 in appendix K).

In the flow diagrams there are used elements described in the previous chapters. For example the “digital sensors init” refers to section 4.1.1 for initialization of the SHT21 sensor and similarly for all digital sensors. The same principle also applies for reading of the buttons (section 4.6.3). In the flow diagram is the process simplified to make the flow diagrams more readable (each button touch triggers an interrupt which would disrupt the flow diagram).

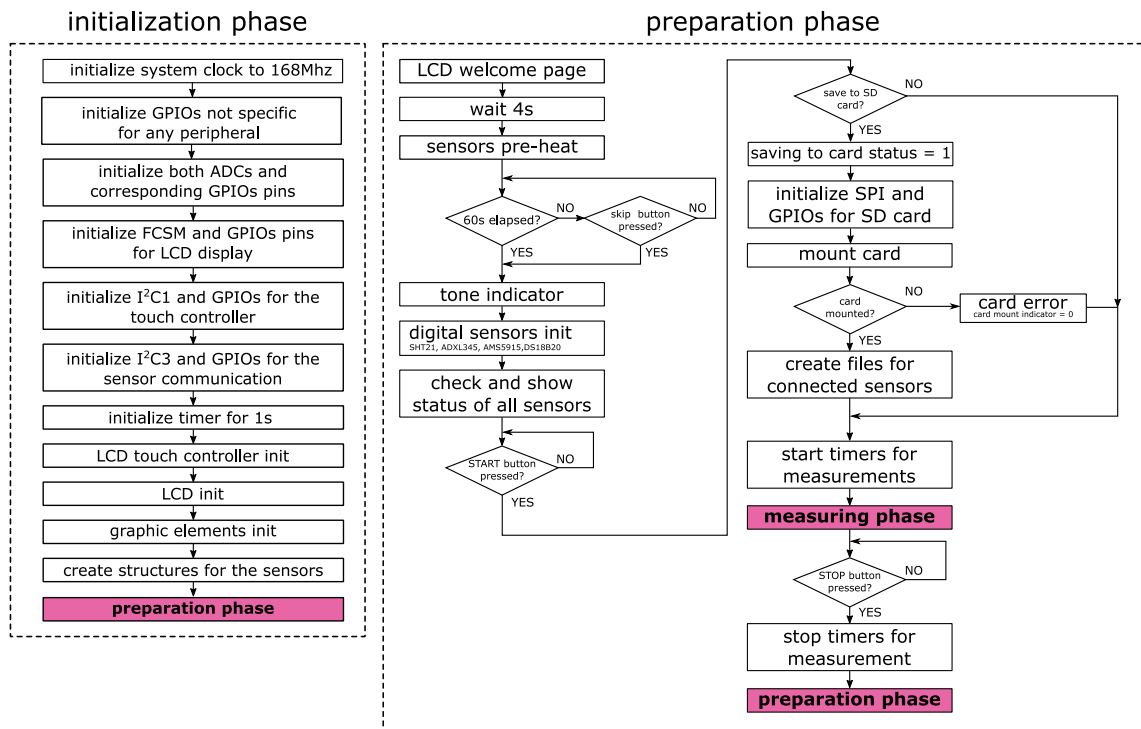


Figure 4.60: Initilization (left) and preparation (right) phase flow graph

The flow diagram corresponds to the frames of LCD interface on figure 4.61. After power up there is a welcome page with the very basic information about the project (figure 4.61a). After a short while, the one minute countdown starts (figure 4.61b). The one minute is for heating up the gas sensors but it is possible to skip this phase by pressing the “Skip!” button (user is not interested in the measurement with gas sensors or the unit has been switched off recently and the sensors didn’t have time to cool down). Either way there is a beep tone to indicate the end of the waiting phase. The program enters the preparation phase (figures 4.61c and 4.61d). The user is informed about the state of the connected sensors. The saving to the SD card can be switched on and off. After the “Start measurement!” button is pressed, the program leaves the preparation phase and it moves into the measurement stage (figures 4.61e and 4.61f). In the upper left corner there is time, date and the information about SD card status displayed. There is also the “STOP” button which stops the measurement phase and returns the program into the preparation phase. Below the horizontal line there are the measured values displayed. In case that any digital sensor is not connected, the user is informed. As mentioned before the analogue sensors are not calibrated therefore the actual measured value is not displayed. The results are interpreted in form of so to speak “loading bar” which is customized for every sensor and shows any increased reading.



Figure 4.61: LCD interface pages

4.11 Prototype encasing

For easy manipulation is the unit enclosed in 200x150x70 mm universal ABS prototype box. The box is modified to meet the requirements of the project. Inside the box there is plexiglass partition diving the electronics and sensor area and protecting

the electronics from possible negative impact of the humidity or dust in the air. Its position is secured with a silicon which also adds insulation effect. A white sticker is placed on the plexiglass to shelter the dust sensor from the power supply board light emission. The wires for connecting sensor are passed through the plexiglass via rubber grommets. The position of each element is shown on figure 4.62.

The necessary openings for allowing access to the SD card, Ethernet port, for connecting of sensors were drilled. All the parts inside are secured with crews to prevent unwanted movement. The connection of uLite to a filtration unit is designed as a funnel with a flexible pipe (figure 4.70). The funnel has neodymium magnets to stick to the filtration unit surface and it covers the air outlet of the filtration unit.

A logo for uLite is designed and placed on the box as well as the labels to each element on the box. A connection guide is printed and placed onto the inner side of the box lid (figure J.1 in appendix J).

The finished box is displayed on figures 4.63 - 4.69. The inside of the box is displayed on figure 4.71. For more detailed photos of both uLite interior and exterior please see the content of the attached CD (appendix A).

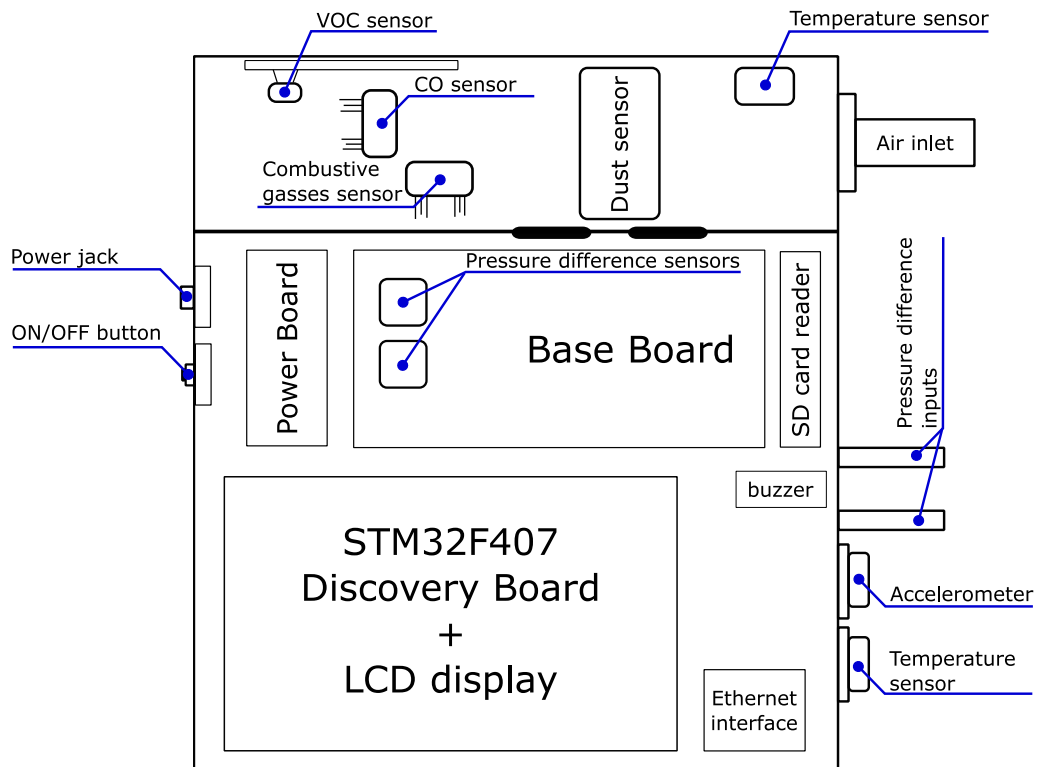


Figure 4.62: Position of individual elements in the box

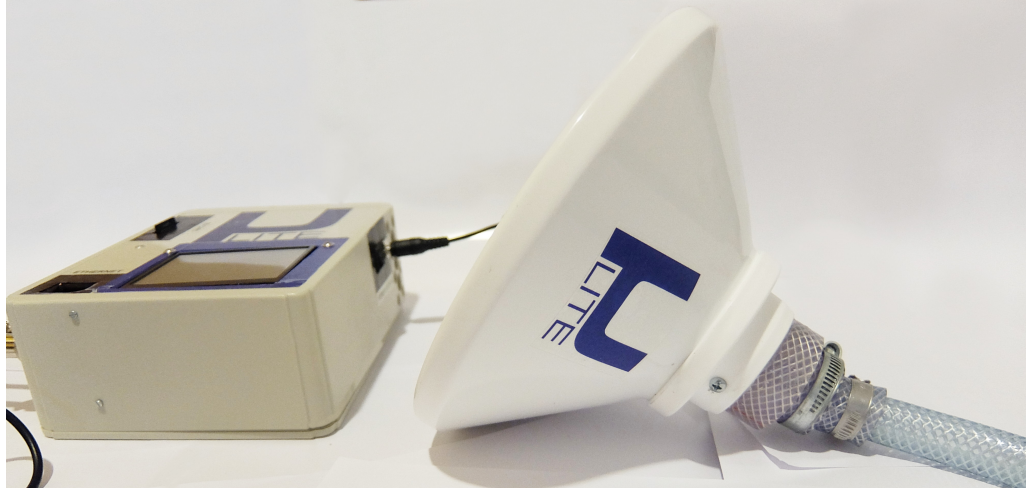


Figure 4.63: uLite with the funnel

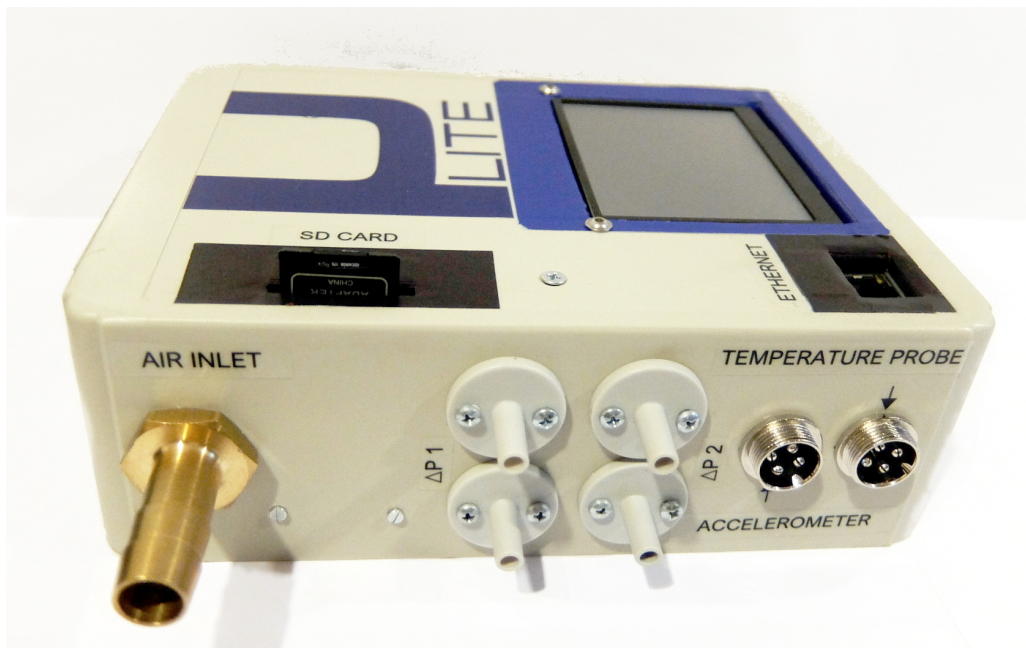


Figure 4.64: uLite side view



Figure 4.65: uLite top view



Figure 4.66: uLite side view 3



Figure 4.67: Connector detail



Figure 4.68: SD card detail

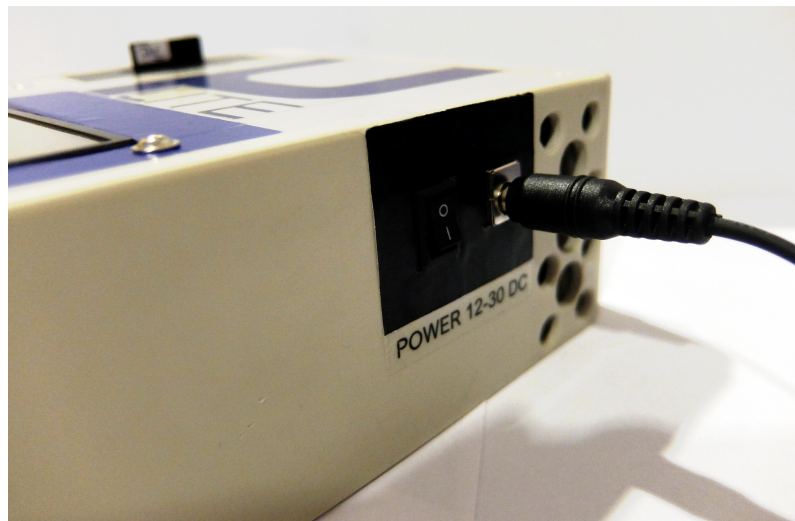


Figure 4.69: Power connection detail



Figure 4.70: Funnel

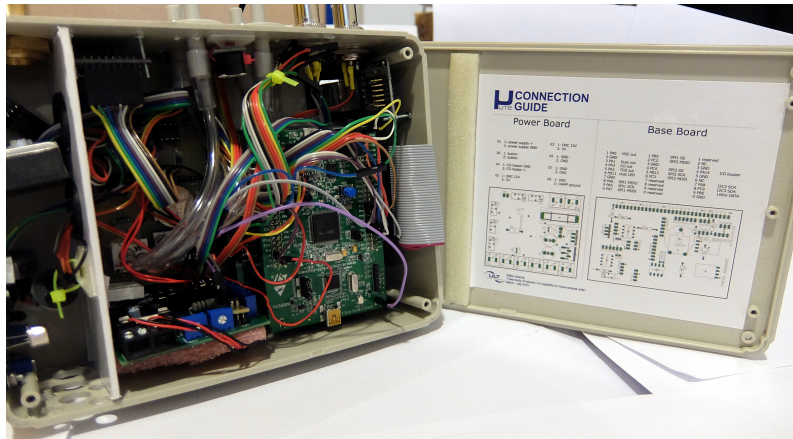


Figure 4.71: uLite inside wiring and the connection guide

5 uLite result presentation

The test measurement was carried out with use of different sources of chemicals and heat sources. The The output files can be found on the attached CD (appendix A). The photos from the uLite test run can be found on figures 5.1 (clean air) and 5.2 (polluted air). uLite measurement unit function is verified and it is ready to be tested in the real application on the filtration unit.

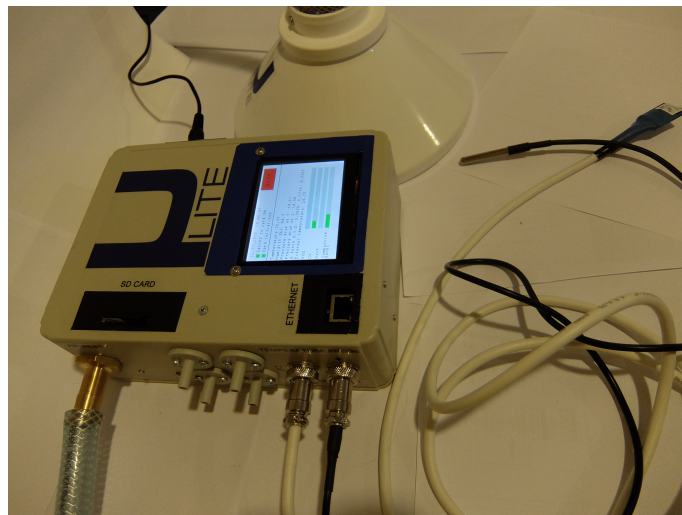


Figure 5.1: uLite measurement phase



Figure 5.2: uLite measurement phase 2

6 Conclusion and perspectives

Industry 4.0 is undoubtedly going to gain more and more significance in the manufacturing industry. In the beginning of this thesis is the general concept of Industry 4.0 introduced and the concrete applications for the air filtration industry are discussed. Some of the proposals are chosen and used in the practical part of this thesis.

The result of the practical part, uLite unit, presents the use of embedded technologies for implementation of Industry 4.0. The unit is capable of measuring, logging and displaying various parameters. Both software and hardware are designed from the beginning. The encasing, name and logo are also created exclusively for this thesis. Every step of the development is described in detail and the documentation can serve for future research. In most cases the libraries are universal and can be used in arbitrary project.

uLite unit should inspire future projects. But to be able to use the full potential of the unit, the most important step is to create an application for receiving and processing the data send from uLite via Ethernet. This application can take a form of SQL database of simple C# windows application.

The next step in development would be to move the microprocessor to dedicated board. Also the Ethernet and SD card connectors should be placed on the same board as well as the LCD display interface. The air quality sensors may be placed on an extra PCB board which would lower the amount of needed wire connections.

Currently the LCD user interface allows a basic control of the uLite unit. The future improvement could be to implement function which enables advance setting of the measurement parameters (such as resolution of the sensors or measuring period/time). Moreover, a mobile application for a smart phones and tablets can replace the LCD display.

A Contents of enclosed CD

CD	
├─ Master_thesis_Latex...	This master thesis is written in \LaTeX . This folder contains all source files and figures used in the thesis.
├─ Master_thesis_PDF....	Please find the thesis text in pdf form here.
├─ Ethernet interface.....	Standalone program for communication via Ethernet, WireShark session log and additional screens.
│ ├─ src	
│ ├─ inc	
│ ├─ project	
│ └─ additional_files	
├─ uLite_HW.....	The directory contains schematics and PCB layouts for Power Board and Base Board (in form of both Eagle and Gerber files) as well as all schematics used in this thesis.
│ ├─ eagle_parts_schematics	
│ │ ├─ sensors	
│ │ └─ others	
│ ├─ power_board	
│ │ ├─ eagle_files	
│ │ └─ gerber_files	
│ ├─ base_board	
│ │ ├─ eagle_files	
│ │ └─ gerber_files	
├─ uLite_measurements...	This folder contains measurements examples.
├─ uLite_photos.....	There are photos of the complete uLite unit.
├─ uLite_SW	This directory contains the software solution for uLite with form of source files, libraries and project which is possible to import into AC6 SW4STM32 IDE.
│ ├─ Drivers	
│ │ ├─ CMSIS	
│ │ └─ STM32F4xx_HAL_Driver	
│ ├─ Inc	In this directory there are headers for all .c files used in the program solution. Detailed description off the files in individual folder can be found in appendix B.
│ │ ├─ lcd	
│ │ ├─ others	
│ │ ├─ peripherals_init	
│ │ ├─ SD_card	
│ │ └─ sensors	
│ ├─ Src	This directory contains all libraries used in the program.
│ └─ SW4STM32	The folder contains .cproject which can be imported into AC6 SW4STM32 IDE.

B Libraries on the attached CD

Inc	
_ lcd	
_ lcd_display.h.	Contains function for LCD initialization and displaying of graphics elements and text.
_ lcd_touch_controller.h.....	Function for initialization of the touch controller and for reading of the buttons.
_ fonts.h.	Set of several fonts defined in form of ASCII tables converted to array of hex (some created by a freeware GLCD Font Creator).
_ buttons.h.	Setting of buttons parameters.
_ others	
_ OneWire.h.	Contains function for communication over One Wire with arbitrary device which supports this bus.
_ DS1307.h.	RTC initialization and functions for time reading and handling of the read values.
_ peripherals_init	
_ adc_init.h	
_ gpio_init.h	
_ i2c_init.h	
_ spi_init.h	
_ timer_init.h	
_ SD_card	
_ diskio.h.	FatFs: Common include file for FatFs and disk I/O module.
_ fatfs_sd.h.....	Low level driver for SD card.
_ ff.h.....	FatFs: Common include file for FatFs and application module.
_ ffconf.h.....	FatFs: Configuration file for FatFs module.
_ files_headers.h.....	Headers for measurement files.
_ write_sd.h.....	Contains function for formatting the measurement results and saving them to SD card.
_ integer.h.	FatFs: Integer type definition for FatFs.
_ sensors	
_ ADLX345.h.	Accelerometer.
_ AMS5915.h.	Pressure sensor.
_ DS18B20.h.....	External temperature sensor.
_ SHT21.h.	Temperature and humidity sensor.
_ SHT21_systemInclude.h....	Sensor register addresses.
_ stm32fxxx_hal.h	
_ stm32f4xx_it.h	
_ stm32f4xx_hal_conf.h	

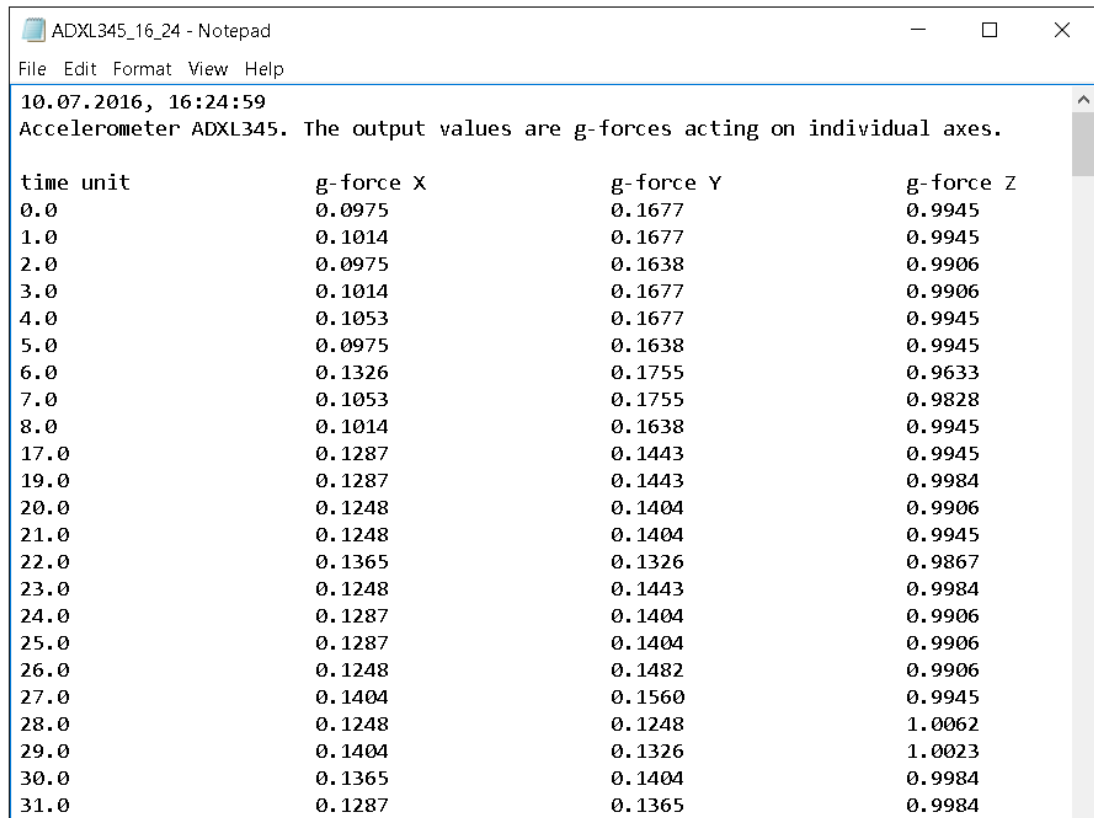
C SD card commands

Table C.1: Commands for SD card initialization [48]

Index	Argument	Response	Data	Name	Function
CMD0	-	R1	-	GO_IDLE_STATE	Software reset
CMD1	-	R1	-	SEND_OP_COND	Initiate initialization process
ACMD41 [†]	Y	R1	-	APP_SEND_OP_COND	Initiate initialization process (SDC only)
CMD8	Y	R7	-	SEND_IF_COND	Check voltage range (SDC v2 only)
CMD9	-	R1	Y	SEND_CSD	Read CSD (Card Specific Data) register
CMD10	-	R1	Y	SEND_CID	Read CID (Card Identification Data) register
CMD12	-	R1	-	STOP_TRANSMISSION	Stop data reading
CMD16	Y	R1	-	SET_BLOCKLEN	Change R/W block size
CMD17	Y	R1	Y	READ_SINGLE_BLOCK	Read a block
CMD18	Y	R1	Y	READ_MULTIPLE_BLOCK	Read multiple blocks
CMD24	Y	R1	Y	WRITE_BLOCK	Write a block
CMD25	Y	R1	Y	WRITE_MULTIPLE_BLOCK	Write multiple blocks
CMD55	-	R1	-	APP_CMD	Leading command for ACMD<n> command
CMD58	-	R3	-	READ_OCR	Read OCR (Operation Condition Register)

[†] ACMD<n> is a command sequence of CMD55 and CMD<n>

D Saved files content



ADXL345_16_24 - Notepad

File Edit Format View Help

10.07.2016, 16:24:59
Accelerometer ADXL345. The output values are g-forces acting on individual axes.

time unit	g-force X	g-force Y	g-force Z
0.0	0.0975	0.1677	0.9945
1.0	0.1014	0.1677	0.9945
2.0	0.0975	0.1638	0.9906
3.0	0.1014	0.1677	0.9906
4.0	0.1053	0.1677	0.9945
5.0	0.0975	0.1638	0.9945
6.0	0.1326	0.1755	0.9633
7.0	0.1053	0.1755	0.9828
8.0	0.1014	0.1638	0.9945
17.0	0.1287	0.1443	0.9945
19.0	0.1287	0.1443	0.9984
20.0	0.1248	0.1404	0.9906
21.0	0.1248	0.1404	0.9945
22.0	0.1365	0.1326	0.9867
23.0	0.1248	0.1443	0.9984
24.0	0.1287	0.1404	0.9906
25.0	0.1287	0.1404	0.9906
26.0	0.1248	0.1482	0.9906
27.0	0.1404	0.1560	0.9945
28.0	0.1248	0.1248	1.0062
29.0	0.1404	0.1326	1.0023
30.0	0.1365	0.1404	0.9984
31.0	0.1287	0.1365	0.9984

Figure D.1: Example of the file on the SD card

E ASCII table

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0		[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Figure E.1: ASCII table [†]

[†] <https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/ASCII-Table-wide.svg/2000px-ASCII-Table-wide.svg.png>

F Function for creating buttons

Code F.1: Function for drawing buttons

```
void createUserButton(struct buttons *buttons_t){
uint8_t textLength; // holds the length of the button text
uint8_t charWidth; // holds the width of the char for given font
uint8_t xSize; // X size of the button
uint8_t ySize; // Y size of the button
uint8_t offset = 30; // the space between the text and border
// set font and color scheme for the button
LCD_SetFont(buttons_t->font);
LCD_SetBackColor(buttons_t->backColor);
LCD_SetTextColor(buttons_t->textColor);
// in case the user did not enter the exact X size of the button,
// calculate it
if (buttons_t->xSize == 0) {
charWidth = buttons_t->font->Width; // width of the char in pixels
textLength = strlen(buttons_t->text); // length of the given text
// calculate the width of the whole string in pixels
textLength = textLength*charWidth;
// add space between the text and button border
xSize = textLength+offset;
} else {
xSize = buttons_t->xSize; // in case X size is defined
}
// in case the user did not enter the exact Y size of the button,
// calculate it
if (buttons_t->ySize == 0) {
ySize = LINE(3); // define Y size as 3xheight of the font
} else {
ySize = buttons_t->ySize; // in case Y size is defined
}
// display the button on given position with given (calculated) size
LCD_DrawFullRect(buttons_t->xPosition,buttons_t->yPosition,xSize,ySize);
LCD_SetBackColor(buttons_t->backColor); // set color of the text
LCD_SetTextColor(buttons_t->textColor);
// display the string aligned in the middle of the button
LCD_DisplayString(buttons_t->yPosition+LINE(1),
(uint8_t*)buttons_t->text,buttons_t->xPosition+(offset/2));
//calculate the active area of the button and save it into the structure
buttons_t->xTouchMin = buttons_t->xPosition;
buttons_t->xTouchMax = buttons_t->xPosition+xSize;
buttons_t->yTouchMin = buttons_t->yPosition;
buttons_t->yTouchMax = buttons_t->yPosition+ySize;
}
```

H Power module scheme

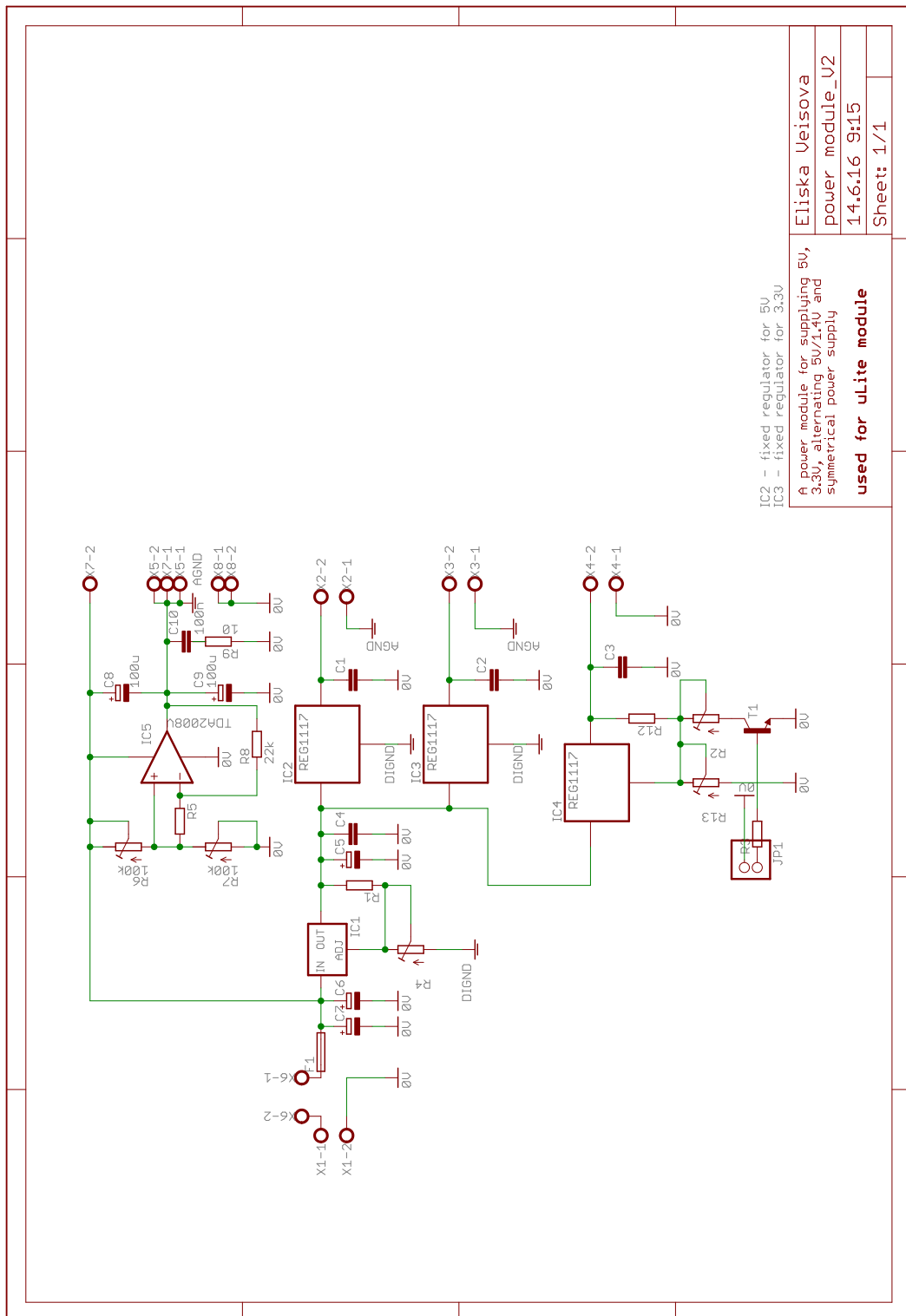


Figure H.1: Power module scheme

I STM32F407 clock configuration

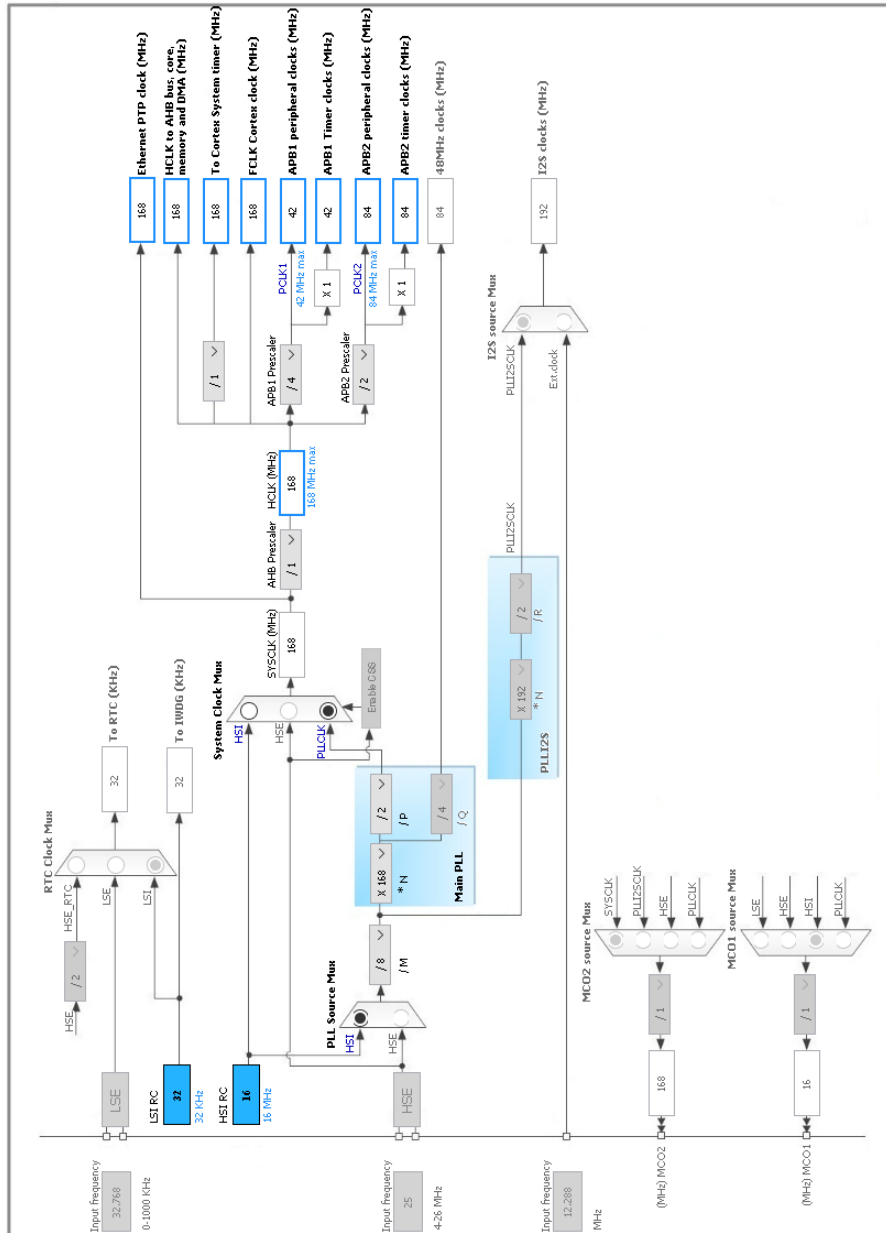


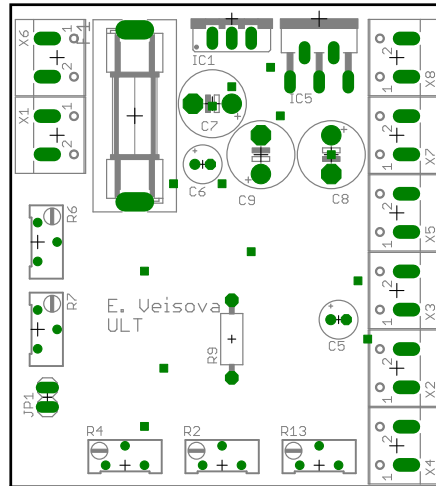
Figure I.1: STM32F407 clock configuration

J uLite Connection Guide



Power Board

- X1 1: power supply +
2: power supply GND
- X6 1: button
2: button
- X4 1: CO heater GND
2: CO heater +
- X2 1: DNC 12V
2: 5V
- X3 1: DNC 12V
2: 3V
- X5 1: GND
2: GND
- X7 1: GND
2: DNC
- X8 1: DNC
2: OAMP ground



Base Board

- 1 PA0 VOC out
- 2 GND
- 3 PA1 Dust out
- 4 PA3 CO out
- 5 PA2 TGS out
- 6 PB11 dust LED
- 7 GND
- 8 PA6 SPI1 MISO
- 9 PA5 SPI1 SCK
- 0 PA7 SPI1 MOSI
- 1 PB5 SPI1 SS
- 2 PC2 SPI2 MISO
- 3 GND
- 4 PA4 SPI2 SS
- 5 PB13 SPI2 SCK
- 6 PC3 SPI2 MOSI
- 7 reserved
- 8 PC9 I2C3 SCK
- 9 PB6 I2C3 SDA
- 0 GND 1Wire DATA

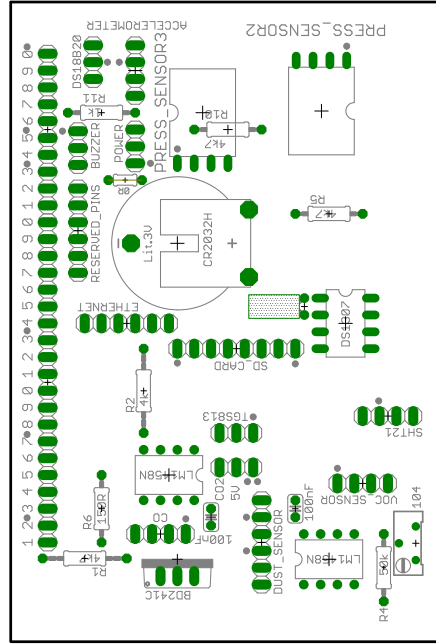


Figure J.1: uLite Connection Guide

K Measuring phase flow graph

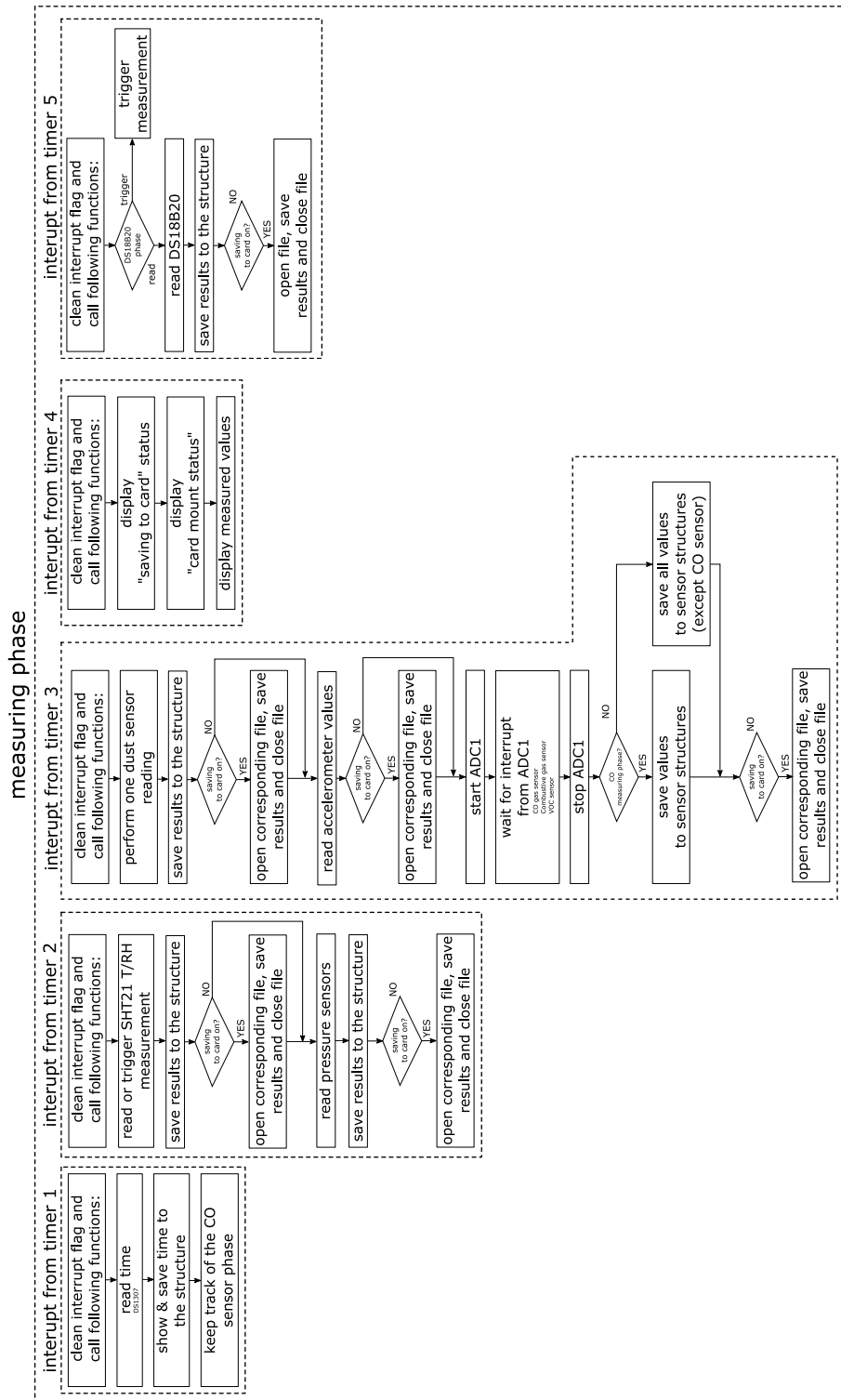


Figure K.1: Flow graph of the measuring phase

L SHT21 library header

```
#include <SHT21_systemInclude.h> // includes defines for measuring times,
    resolutions values, addresses...
struct SHT_parameters { // structure for holding parameters of each
    sensor
    uint16_t writeAddress;
    uint16_t readAddress;
    uint8_t resolutionTemp;
    uint8_t resolutionHum;
    uint8_t heater; // status of the heater, 0 = OFF, 1 = ON
    uint8_t battery; // battery status 0: VDD> 2.25V; 1: VDD<2.25V
    float lastTempReading; // results of the last temperature reading
    float lastHumReading; // results of the last humidity reading
};
extern struct SHT_parameters SHT_parameters_t;
extern uint8_t buffer[16]; // buffer for storing values read from the
    sensor
/* Definition of possible commands */
typedef enum {
    TRIGGER_t_MEASUREMENT_HOLD_MASTER = 0xE3, // 1110 0011 227
    TRIGGER_t_MEASUREMENT_NO_HOLD_MASTER = 0xF3, // 1111 0011 243
    TRIGGER_rh_MEASUREMENT_HOLD_MASTER = 0xE5, // 1110 0101 229
    TRIGGER_rh_MEASUREMENT_NO_HOLD_MASTER= 0xF5, // 1111 0101 245
    WRITE_USER_REGISTER = 0xE6, // 1110 0110 230
    READ_USER_REGISTER = 0xE7, // 1110 0111 231
    SOFT_RESET = 0xFE, // 1111 1110 254
} SHT21_COMMANDS;
/* possible temperature and humidity combinations */
typedef enum
{
    TEMP14HUM12 = 0,
    TEMP12HUM8 = 1,
    TEMP13HUM10 = 2,
    TEMP11HUM11 = 3,
} Resolution_TEMP_HUM;

//////////////////////////////////USER FUNCTIONS//////////////////////////////////

/* @brief Sensor init (default addresses, default resolution temperature
    14bits, humidity 12bits, heater off)
* @param *SHT_parameters_t : pointer to a structure with sensor
    parameters*/
void initializeSensor(struct SHT_parameters *SHT_parameters_t);
/* @brief Reads the humidity and stores the result in the structure
```

```

* @param *SHT_parameters_t : pointer to a structure with sensor
  parameters*/
void humidityReading(struct SHT_parameters *SHT_parameters_t);
/* @brief Reads the temperature and stores the result in the structure
* @param *SHT_parameters_t : pointer to a structure with sensor
  parameters*/
void temperatureReading(struct SHT_parameters *SHT_parameters_t);
/* @brief Reads user register content and fills the structure with data
* @param *SHT_parameters_t : pointer to a structure with sensor
  parameters*/
void readUserRegister(struct SHT_parameters *SHT_parameters_t);
/* @brief Sets resolution of temperature and humidity measurement
* @param *SHT_parameters_t : pointer to a structure with sensor
  parameters
* @param res_value      : chosen combination of the resolutions, shall
  have values          TEMP14HUM12,TEMP12HUM8,TEMP13HUM10,TEMP11HUM11
*/
void setResolution(struct SHT_parameters *SHT_parameters_t,
  Resolution_TEMP_HUM res_value);
/* @brief Toggles the heater
* @param *SHT_parameters_t : pointer to a structure with sensor
  parameters
* @param heater           : set 0 to stop the heater, 1 to start the heater
*
  any other value wont do anything
*/
void heaterToggle(struct SHT_parameters *SHT_parameters_t, uint8_t
  heater);

////////////////////////////////////INTERNAL FUNCTIONS////////////////////////////////////

float calculateHum(uint8_t buff[16]);
float calculateTemp(uint8_t buff[16]);
void sendToSensor(struct SHT_parameters *SHT_parameters_t, SHT21_COMMANDS
  byteToSend);
void readFromSensor(struct SHT_parameters *SHT_parameters_t, uint16_t
  number_of_bt);
void waitTillTheEndOfConverstionTemp(struct SHT_parameters
  *SHT_parameters_t);
void waitTillTheEndOfConverstionHum(struct SHT_parameters
  *SHT_parameters_t);
void recieveTemperature(struct SHT_parameters *SHT_parameters_t);
void recieveHumidity(struct SHT_parameters *SHT_parameters_t);

```

M Wire Shark records

Wireshark - Packet 151 - wireshark_pcapng_7048B914-7A8C-41E0-AF63-F3E7A5C995A5_20160719002346_a11964

▼ **HTTP/1.0 200 OK**
 [HTTP/1.0 200 OK]
 [Severity Level: Chat]
 [Group: Sequence]
 Request Version: HTTP/1.0
 Status Code: 200
 Response Phrase: OK
 Content-Type: text/html
 Content-Length: 57
 [Content length: 57]
 Pragma: no-cache
 Connection: About to close
 \r\n
 [HTTP response 1/2]
 [Time since request: 0.013368000 seconds]
 [Request in frame: 147]
 [Next response in frame: 153]

▼ **Line-based text data: text/html**
 <h3>Eliska Veisova, ULT, ulite, data from EIK28160</h3>

Wireshark - Packet 151 - wireshark_pcapng_7048B914-7A8C-41E0-AF63-F3E7A5C995A5_20160719002346_a11964

[Coloring Rule Name: HTTP]
 [Coloring Rule String: http || tcp.port == 80 || http2]
 Ethernet II, Src: 3comCorp_03:04:06 (00:01:02:03:04:06), Dst: AsrockIn_85:98:ad (d0:50:99:85:98:ad)
 Destination: AsrockIn_85:98:ad (d0:50:99:85:98:ad)
 Address: AsrockIn_85:98:ad (d0:50:99:85:98:ad)

 = LG bit: Globally unique address (factory default)
 = IG bit: Individual address (unicast)
 Source: 3comCorp_03:04:06 (00:01:02:03:04:06)
 Address: 3comCorp_03:04:06 (00:01:02:03:04:06)
 = LG bit: Globally unique address (factory default)
 = IG bit: Individual address (unicast)
 Type: IPv4 (0x0800)
 Internet Protocol Version 4, Src: 192.168.1.181, Dst: 192.168.1.102
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 Total Length: 207
 Identification: 0x00e1 (225)
 > Flags: 0x00
 Fragment offset: 0

▼ **Frame 151: 221 bytes on wire (1768 bits), 221 bytes captured (1768 bits) on interface**
 Interface id: 0 (\Device\NPF_{7048B914-7A8C-41E0-AF63-F3E7A5C995A5})
 Encapsulation type: Ethernet (1)
 Arrival Time: Jul 19, 2016 00:23:52.601953000 Central Europe Daylight Time
 [Time shift for this packet: 0.000000000 seconds]
 Epoch Time: 146880632.601953000 seconds
 [Time delta from previous captured frame: 0.007921000 seconds]
 [Time delta from previous displayed frame: 0.007921000 seconds]
 Frame Number: 151
 Frame Length: 221 bytes (1768 bits)
 Capture Length: 221 bytes (1768 bits)
 [Frame is marked: False]
 [Frame is ignored: False]
 [Protocols in frame: eth:ethertype:ip:tcp:http:data-text-lines]
 [Coloring Rule Name: HTTP]
 [Coloring Rule String: http || tcp.port == 80 || http2]
 Ethernet II, Src: 3comCorp_03:04:06 (00:01:02:03:04:06), Dst: AsrockIn_85:98:ad (d0:50:99:85:98:ad)
 Internet Protocol Version 4, Src: 192.168.1.181, Dst: 192.168.1.102
 Transmission Control Protocol, Src Port: 80 (80), Dst Port: 63392 (63392), Seq: 1, Ac

0000 d0 50 99 85 98 ad 00 01 02 03 04 06 08 00 45 00 .P.....E.
 0010 00 cf 00 e1 00 00 40 06 f4 dc c0 a8 01 b5 c0 a8@.....
 0020 01 66 00 50 f7 a0 06 d9 78 f1 a6 66 30 bb 50 18 .f.P...x..f0.P.
 0030 02 00 59 ea 00 00 48 54 54 50 2f 31 2e 30 20 32 ..Y...HT TP/1.0 2
 0040 30 20 20 4f 4b 0d 0a 43 6f 6e 74 65 6e 74 2d 54 00 OK...C ontent-T
 0050 79 70 65 3a 20 74 65 78 74 2f 68 74 6d 6c 0d 0a ype: tex t/html..
 0060 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 Content- Length:
 0070 35 37 0d 0a 50 72 61 67 6d 61 3a 20 6e 6f 2d 63 57..Prag ma: no-c
 0080 61 63 68 65 0d 0a 43 6f 6e 65 63 74 69 6f 6e ache..Co nnection
 0090 3a 20 41 62 6f 75 74 20 74 6f 20 63 6c 6f 73 65 : About to close
 00a0 0d 0a 0d 0a 3c 68 33 3e 45 6c 69 73 6b 61 20 56<h3> Eliska V
 00b0 65 69 73 6f 76 61 2c 20 55 4c 54 2c 20 75 4c 69 eisova, ULT, uli
 00c0 74 65 2c 20 64 61 74 61 20 66 72 6f 6d 20 45 4e te, data from EN
 00d0 43 32 38 4a 36 30 3c 2f 68 33 3e 0d 0a C28160</ h3>..

Figure M.1: SD card responses frames

N ENC28J60 SPI communication (Rigol)

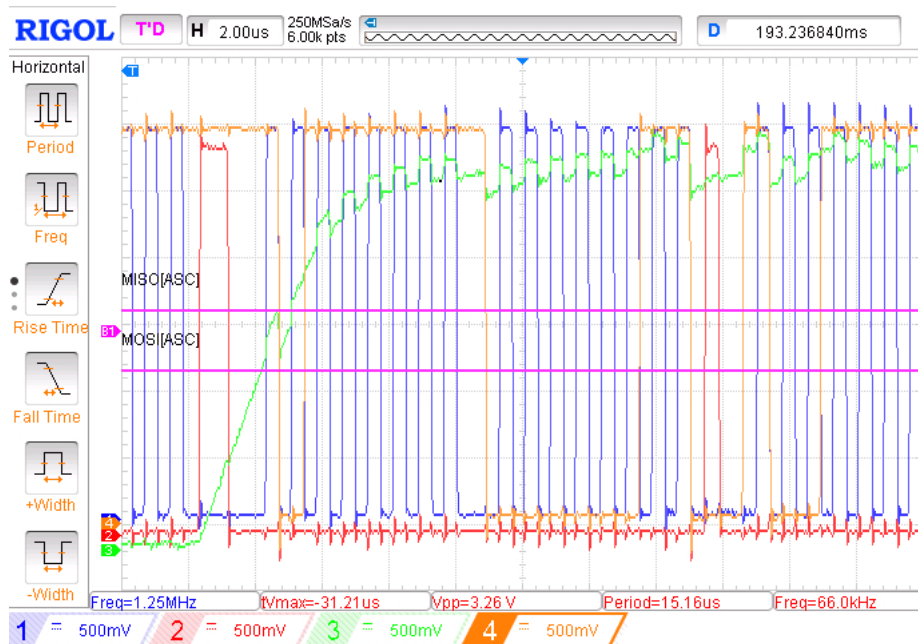


Figure N.1: ENC28J60 SPI communication (detail)

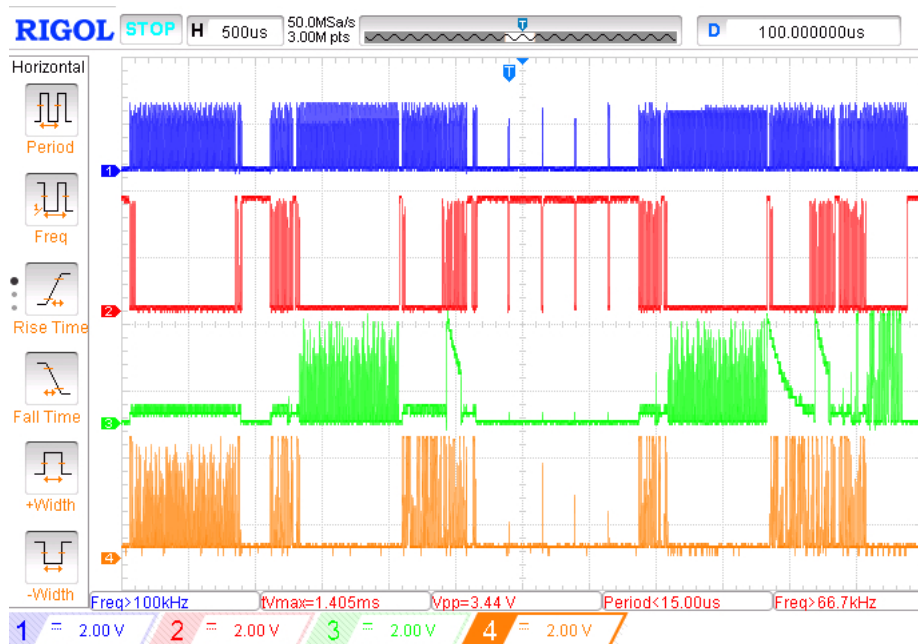


Figure N.2: ENC28J60 SPI communication

Bibliography

- [1] *800mA and 1A Low Dropout Positive Regulator*. Available at: <http://www.ti.com/lit/ds/sbvs001d/sbvs001d.pdf> (Accessed 14.6.2016). Texas Instruments. 2004.
- [2] *AMS 5915 Amplified pressure sensor with digital output (I2C)*. Available at: http://www.analogmicro.de/_pages/sens/ams5915/ams5915_data_sheet.pdf (Accessed 10.4.2016), Rev. 2.1. HJK Sensoren + Systeme. Apr. 2015.
- [3] *Application note of Sharp dust sensor GP2Y1010AU0F*. Available at: http://www.sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y1010au_appl_e.pdf (Accessed 10.4.2016), Sheet No.: OP13024EN. Sharp. 2006.
- [4] David Armitage, Ian Underwood, and Shin tson Wu. *Introduction to microdisplays*. John Wiley and Sons, Ltd, 2006. ISBN: 9780470852811.
- [5] Abhishek Attal et al. *Piano Playing Robot*. Vol. 3. International Journal of Engineering Research and Technology, May 2014. ISBN: 22780181.
- [6] John R. Barnes. *Robust Electronic Design*. 1st edition, 360 pages. Springer US, 2004. ISBN: 1-4020-7830-7.
- [7] Jaap Bloem et al. *No more secrets with Big Data Analytics*. Available at: http://vint.sogeti.com/wp-content/uploads/2013/11/Sogeti_NoMoreSecrets.pdf (Accessed 20.5.2016). Sogeti LINE UP boek en media bv Groningen, 2013.
- [8] Jaap Bloem et al. *The Fourth Industrial Revolution: Things to Tighten the Link Between it and ot*. Available at: <http://www.fr.sogeti.com/globalassets/global/downloads/reports/vint-research-3-the-fourth-industrial-revolution> (Accessed 20.5.2016). Sogeti LINE UP boek en media bv Groningen, 2014.
- [9] Jaap Bloem et al. *THINGS — Internet of Business Opportunities*. Available at: <http://www.fr.sogeti.com/globalassets/global/downloads/reports/vint-research-1-things-internet-of-business-opportunities.pdf> (Accessed 20.5.2016). Sogeti LINE UP boek en media bv Groningen, 2013.
- [10] Maarten Botterman. *Internet of Things: an early reality of the Future Internet, WORKSHOP REPORT*. Available at: http://cordis.europa.eu/pub/fp7/ict/docs/enet/iot-prague-workshop-report-vfinal-20090706_en.pdf (Accessed 28.5.2016). O'Reilly Media, May 2009.
- [11] Encyclopædia Britannica. *Moore's law*. Available at: <http://www.britannica.com/topic/Moores-law> (Accessed 20.5.2016). Encyclopædia Britannica Inc., 2014.
- [12] *Carbon Monoxide in the workplace*. Available at: www.iapa.ca/pdf/carbon_monoxide_feb2003.pdf (Accessed 4.5.2016). IAPA - Industrial Accident Prevention Association. 2008.
- [13] Elm Chan. *FatFs - Generic FAT File System Module*. Available at: http://elm-chan.org/fsw/ff/00index_e.html (Accessed 22.5.2016).

- [14] SanDisk Corporation. *SanDisk miniSD: Card Product Manual*. Available at: <http://alumni.cs.ucr.edu/~amitra/sdcard/Additional/ProdManualminiSDv1.1.pdf> (Accessed 22.5.2016). Nov. 2003.
- [15] Timothee Cour, Rémy Lauranson, and Matthieu Vachette. *Autonomous Chess-playing Robot*. Ecole Polytechnique, July 2002.
- [16] *Datasheet SHT21: Humidity and Temperature Sensor IC*. Available at: https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity_Sensors/Sensirion_Humidity_Sensors_SHT21_Datasheet_V4.pdf (Accessed 10.4.2016), Version 4. Sensirion. May 2014.
- [17] Joseph Davies. *Understanding IPv6 (3rd Edition)*. ISBN: 0735659141. O'Reilly Media, 2012.
- [18] Karl-Heinz Deiretsbacher et al. *OPC Unified Architecture Pioneer of the 4th industrial (r)evolution*. Available at: https://jp.opcfoundation.org/wp-content/uploads/2014/03/OPC_UA_I_4.0_Pioneer_US_v2.pdf (Accessed 20.5.2016). 2015.
- [19] *Digital Accelerometer ADXL345*. Available at: <http://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL354.pdf> (Accessed 23.6.2016). Analog Devices. 2009.
- [20] *DS1307 64 x 8, Serial, I2C Real-Time Clock*. Available at: <http://datasheets.maximintegrated.com/en/ds/DS1307.pdf> (Accessed 26.5.2016), Rev. 3. Maxim Integrated Products, Inc. 2015.
- [21] *DS18B20 Programmable Resolution 1-Wire Digital Thermometer*. Available at: <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf> (Accessed 10.4.2016), Rev. 4. Maxim Integrated. 2015.
- [22] David Greenfield. *Industry 4.0 and OPC UA*. Available at: <http://www.automationworld.com/industry-40-and-opc-ua> (Accessed 20.5.2016). 2014.
- [23] Mario Hermann, Tobias Pentek, and Boris Otto. *Design Principles for Industrie 4.0 Scenarios: A Literature Review*. Available at: <http://www.leorobotics.nl/sites/leorobotics.nl/files/bestanden/2015%20-%20Hermann%20Pentek%20%26%20Otto%20-%20Design%20Principles%20for%20Industrie%204%20Scenarios.pdf> (Accessed 20.5.2016). Encyclopædia Britannica Inc., Jan. 2015.
- [24] Texas Instruments. *User guide: KeyStone Architecture Serial Peripheral Interface (SPI)*. Literature Number: SPRUGP2A, Available at: <http://www.ti.com/lit/ug/sprugp2a/sprugp2a.pdf> (Accessed 6.4.2016). Mar. 2012.
- [25] Schlick J. et al. *Industrie 4.0 in der praktischen Anwendung*. 2014.
- [26] Shi J et al. *A survey of cyber-physical systems, In: International conference on wireless communication and signal processing (WPC)*. Nov. 2011.
- [27] Holger Junker. *Interoperability is the Key for IoT and Industrie 4.0*. Available at: <http://www.maintworld.com/Applications/Interoperability-is-the-Key-for-IoT-and-Industrie-4.0> (Accessed 20.5.2016). Nov. 2015.
- [28] Prof. Dr. Henning Kagermann, Prof. Dr. Wolfgang Wahlster, and Dr. Johannes Helbig. *Recommendations for implementing the strategic initiative INDUSTRIE 4.0*. Available at: http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderseiten/Industrie_4.0/Final_report_Industrie_4.0_accessible.pdf (Accessed 4.5.2016). National Academy of Science and Engineering, Apr. 2013.

- [29] *L3GD20 MEMS motion sensor: three-axis digital output gyroscope*. Available at: <http://www2.st.com/resource/en/datasheet/l3gd20.pdf> (Accessed 10.4.2016), Rev 2. STMicroelectronics. 2013.
- [30] Jay Lee and Behrad Bagheri. *Big future for cyber-physical manufacturing systems*. Available at: <http://www.designworldonline.com/big-future-for-cyber-physical-manufacturing-systems/> (Accessed 26.5.2016). Sept. 2015.
- [31] Jay Lee, Behrad Bagheri, and Hung-An Kao. *A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems*. Available at: www.sciencedirect.com (Accessed 26.5.2016). NSF Industry/University Cooperative Research center on Intelligent Maintenance Systems, University of Cincinnati, Elsevier, Oct. 2014.
- [32] Candace Leiden and Marshall Wilensky. *TCP/IP For Dummies 6th Edition*. Wiley Publishing. 2009. ISBN: 78-0-470-45060-4.
- [33] *LM317 3 Terminal Adjustable Regulator*. Available at: <http://www.ti.com/lit/ds/symlink/lm317.pdf> (Accessed 14.6.2016). Texas Instruments. 2014.
- [34] *LM75A Digital Temperature Sensor and Thermal Watchdog With Two-Wire Interface*. Available at: <http://www.ti.com/lit/ds/symlink/lm75a.pdf> (Accessed 10.4.2016). Texas Instruments. 2014.
- [35] Microchip. *ENC28J60, Stand-Alone Ethernet Controller with SPI Interface*. Available at: <http://ww1.microchip.com/downloads/en/DeviceDoc/39662e.pdf> (Accessed 20.7.2016).
- [36] Lydia Parziale et al. *TCP/IP Tutorial and Technical Overview*. Available at: <https://www.redbooks.ibm.com/redbooks/pdfs/gg243376.pdf> (Accessed 20.7.2016). Dec. 2006.
- [37] Hartmut Rauen. *Industrie 4.0 in practice – Solutions for industrial applications*. Available at: <http://industrie40.vdma.org/documents/4214230/5356229/Industrie%204.0%20in%20practice%202016/7fa35030-9456-4de4-8f55-fbd7380d8cf4> (Accessed 20.5.2016). VDMA, 2016.
- [38] Michael Rüßmann et al. *Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries*. Available at: https://www.bcgperspectives.com/content/articles/engineered_products_project_business_industry_40_future_productivity_growth_manufacturing_industries/ (Accessed 12.5.2016). Web article on www.bcgperspectives.com, Apr. 2015.
- [39] Díaz Delgado Rül. *Tin oxide gas sensors: an electrochemical approach*. Universitat de Barcelona, 2002.
- [40] Dr. Ralf C. Schlaepfer, Markus Koch, and Dr. Philipp Merkofer. *Industry 4.0: Challenges and solutions for the digital transformation and use of exponential technologies*. Available at: <http://www2.deloitte.com/content/dam/Deloitte/ch/Documents/manufacturing/ch-en-manufacturing-industry-4-0-24102014.pdf> (Accessed 6.5.2016). The Creative Studio at Deloitte, Apr. 2015.
- [41] NXP Semiconductors. *UM10204 I2C-bus specification and user manual*. Available at: http://www.nxp.com/documents/user_manual/UM10204.pdf (Accessed 1.4.2016 Rev. 6). Apr. 2014.
- [42] Masakazu Shoji. *High-Speed Digital Circuits*. 1st edition, 360 pages. Addison Wesley Publishing Company, Apr. 1996. ISBN: 020163483X.
- [43] *SSD2119 - 320 RGB x 240 TFT LCD Driver Integrated Power Circuit, Source and Gate Driver and RAM*. Available at: <http://www.hpinfotech.ro/SSD2119.pdf> (Accessed 14.6.2016), Rev 1.4. SOLOMON SYSTECH. 2009.

- [44] *STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM-based 32-bit MCUs*. Available at:
http://www.st.com/content/ccc/resource/technical/document/reference_manual/3d/6d/5a/66/b4/99/40/d4/DM00031020.pdf/files/DM00031020.pdf/jcr:content/translations/en.DM00031020.pdf (Accessed 10.4.2016), Sheet No.: DocID018909 Rev 12. STM. 2016.
- [45] STMicroelectronics. *STM32F4DISLCD Rev History and Block Diagram*. Available at:
wiki.hevs.ch/uit/images/e/ef/STM32F4DIS-LCD_REV1.0.pdf (Accessed 27.6.2016). Oct. 2012.
- [46] *Technical data MQ-7 Sensor*. Available at:
<https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf> (Accessed 10.4.2016). Hanwei Electronics CO. 2002.
- [47] *TGS 813 - for the detection of Combustible Gases*. Available at:
www.figarosensor.com/products/813pdf.pdf (Accessed 10.4.2016), Rev. 2. Figaro TGS813. 2012.
- [48] SD Group (Panasonic SanDisk Toshiba) and SD Card Association. *SD Specifications Part 1: Physical Layer Simplified Specification*. Version 2.00 Available at:
https://www.sdcard.org/downloads/pls/pdf/part1_410.pdf (Accessed 22.5.2016). Sept. 2006.
- [49] SD Group (Panasonic SanDisk Toshiba) and SD Card Association. *SD Specifications Part A2: SD Host Controller*. Version 2.00 Available at:
https://www.sdcard.org/developers/overview/host_controller/simple_spec/Simplified_SD_Host_Controller_Spec.pdf (Accessed 22.5.2016). Feb. 2007.
- [50] Dominick Vanthienen, Markus Klotzbucher, and Herman Bruyninckx. *The 5C-based architectural Composition Pattern: lessons learned from re-developing the iTaSC framework for constraint-based robot programming*. May 2014. ISBN: 20353928.