Ubiquitous remote patient monitoring system

Bakker, P.G.M.

Award date:
2011

AME

TU/e Technische Universiteit Eindhoven University of Technology

Master's thesis

Ubiquitous Remote Patient Monitoring System

company confidential

Paul Bakker
November 2010 - August 2011

Supervisors:
dr. O.D. Amft (Eindhoven University of Technology)
ir. M.G.J.R. Stalpers (AME)

Abstract

Chronic Obstructive Pulmonary Disease (COPD), is a chronic lung disease, which causes limitations of airflow from and to the lungs of the patients. Physicians recommend COPD patients to quit smoking and follow a healthy and active lifestyle. Within the facilities of the pulmonary rehabilitation center patients are kept active, thanks to a tight schedule and intensive supervision. When patients are dismissed from the pulmonary rehabilitation center, the intensive supervision is lost. It will be harder for the patients to keep a healthy and active lifestyle.

This thesis proposes a remote patient monitoring architecture. A mobile measuring device can measure the patients' physical and physiological activity is created with this thesis. The architecture includes a generic wireless sensor node with various sensor specific electronics, a gateway and a server. The wireless sensor node measures different signals and transmits the measured data to the gateway. Algorithms that combine and analyze the measured sensor data run on the gateway. Energy saving algorithms, which extend battery lifetime should run on the sensor node. The gateway transfers the output of the algorithms to a server. Physicians can log onto the server and see the status of their patients.

The sensor node consists of a wireless ZigBee transceiver, a dedicated flash chip and a low power signal processor. The sensor node can store up to 10 hours of uncompressed sensor data. Electronics, which are specific to particular sensors, can be connected to the sensor node via the generic interface it offers. Each sensor node has an accelerometer on board to measure the patients' physical activity. These accelerometers can also provide input for power saving algorithms.

Design of sensor specific electronics was included in this project. The designed electronics can be used to measure the patients' respiration rate and saturation of peripheral oxygen (SpO_2). The functionality of the signal processor on the sensor node can be adjusted, such that the sensor node can use the connected sensor specific board.

An algorithm which reduces the energy consumption of the sensor node is also investigated. The respiration band consumes a substantial amount of power. To reduce the power consumption of the respiration band a dynamic sampling algorithm was created. The potential sensor energy consumption could be reduced by 85%, with a frequency accuracy loss of 22.5%.

Acknowledgements

The results of this theses were not possible without several people. First, I thank my supervisors, Micha Stalpers and Oliver Amft, for all their time and effort to support me during this project. I also thank Bram van Nunen for the time spent to support me during this project, and for providing feedback. I also like to thank my colleagues of AME which were involved with this project, especially I thank Bert van Moll and Ivo Ramaekers, for their support. Finally I thank my friends and relatives for their patience and understanding during this thesis project.

Contents

1 Introduction

This thesis project was carried out in the context of the iCare4COPD project. With the iCare4COPD project a measurement system, to monitor Chronic Obstructive Pulmonary Disease (COPD) patients, is to be created. This system should help motivate patients to maintain their level of physical activity and provide feedback to clinicians on the patients' status. It is assumed that coaching the patient to maintain their physical activity can prevent or shorten hospitalization.

The iCare4COPD project is a cooperative project between Applied Micro Electronics (AME), CIRO+, Philips and the (University of Technology Eindhoven) TU/e. AME is a company specialized in electronics and information technology. CIRO+ is an expertise center for chronic organ failure and incorporates a pulmonary rehabilitation for COPD patients. Philips wants to expand their expertise in medical equipment, in the area of COPD. The TU/e wants to publish high quality papers with the help of this project.

COPD is a progressive disease that makes breathing more difficult. "Progressive", in this context, means that the disease gets worse over time. The leading cause of COPD is cigarette smoking and long-term exposure to other lung irritants, such as, air pollution, chemical fumes and dust. Most people who have COPD, smoke or used to smoke for several years [6].

A symptom of COPD is airflow limitation on the pulmonary function, which results in shortness of breath and poor exercise capacity. The natural course of COPD is characterized by occasional sudden worsening of symptoms, called acute exacerbations [6]. Current treatment of COPD patients is preventing patients getting these acute exacerbations by stimulating a healthy and active lifestyle. When patients are at a pulmonary rehabilitation center they receive lots of supervision, which is lost when the patients are dismissed. When dismissed, the patient should continue maintaining that physical activity at home, which is hard without the intensive supervision.

Within this masters project a prototype system for collecting data on physical and physiological activity was created. Analysis on the influences of movement on the respiration sensors accuracy was performed. A dynamic sampling algorithm, to reduce energy consumption required by the sensor, was developed and evaluated. With the prototype the sensor data is transmitted via ZigBee (a wireless protocol) to a central point, the gateway. The gateway is a device, which can store the sensor data on a USB-stick.

The prototype measuring system can measure the sensors and can transmit the data up to 42 meters, or when worn in a pocket, the maximum range is 14.5 meter. The influence of motion on the accuracy of the respiration band is between 58% and 100%, with an average of 90%. The transition phases, the change of body posture, has the most influence on the accuracy of the sensor. The energy consumption of the dynamic sampling algorithm can be decreased to 15% of the original, with a inaccuracy of 22.5%.

1.1 Project description

This section describes briefly the project description of this master thesis. For the original thesis description please refer to appendix A.

The thesis project was divided into three main parts, an architecture, an implementation, and an algorithm development part. Within the architecture part of the project a global architecture for the iCare4COPD project was proposed. For the implementation part a prototype monitoring system was developed. The implementation of the prototyping monitoring system was divided in four main work steps:

- Investigate ubiquitous system architectures that address constraints of mobile distributed systems
- Develop an architecture for the sensor node
- Implement a prototype monitoring system, including hardware and software
- Evaluate the system in tests that simulate the later use in remote patient monitoring

The system has to cope with changing conditions during the patients daily life, in particular: erroneous sensor readings, sensor/communication failures, processing and power constraints. In addition, the system needs to be comfortable to wear and use for the patient. The architecture and implemented solution need to incorporate a possibility to quickly adapt the system for the use of different sensors (e.g. to measure the patients' heart rate).

The goal of the algorithmic part of the theses was to create a selective sampling routine, which the accuracy of the respiration data can be traded off against the available energy. From literature [4] it is known that body movement influences respiration sensors' signal. Therefore influences of motion on the sensors accuracy should be investigated.

After this thesis project, iCare4COPD project will continue. The iCare4COPD project will continue for 2.5 more years. After the iCare4COPD project, and when the results allows, a health care monitoring system will be developed for COPD patients.

1.2 Background information

The thesis project was performed at AME and was in collaboration with the TU/e and AME. The supervisor at AME was ir. M. Stalpers, the supervisor of the TU/e was dr. O. Amft. The architecture and implementation part of this thesis were supervised by AME and the algorithmic part by the TU/e.

AME

AME is a high tech company founded in 1996 at the high tech center of the Eindhoven University of Technology. Currently AME consists of 3 departments; RD&D electronics, SolvIT and Products. The thesis was conducted in the department SolvIT. The focus of SolvIT is on development of advanced embedded systems.

AME-SolvIT includes the focus on wireless communication based products. AME-SolvIT has as experiences the ZigBee wireless communication protocol. The roadmap of AME includes the use of ZigBee Cluster Library (ZCL) and the use of the EM357. The EM357 is the successor of the EM250 ZigBee transceiver chip from EMBER, which AME uses in current products [1].

TU/e

The Eindhoven University of Technology was founded as the Technische Hogeschool Eindhoven (THE) on June 23, 1956 by the Dutch government. It is located on its own campus in the center of Eindhoven. It is currently home to about 240 professors, 7200 students, 250 PDEng-students, 600 Ph.D. students, 200 post-doc students and 3000 regular employees. Yearly, the Eindhoven University of Technology produces almost 3000 scientific publications, 140 PhD-degrees, and 40 patents [18].

2 Architecture design and requirements

This section describes the global architecture of the remote patient monitoring system and the requirements of the sensor node. First, different use cases are described, followed by the architecture proposal. Finally the project scope and requirements are described.

2.1 Use cases

The system has to cope with different aspects. Each aspect sets its own requirements to the system. The aspects are divided among the different users of the system, which are, the patients, researchers and physicians.

2.1.1 Patients

The most important group of users are the patients. The patients need to wear and bare the system, the whole day, every day. Two aspects of the patients' life are considered below, which are the daily routine at home and a day away from home.

Daily routine at home

The patient should not become a prisoner in their own home, just because he is a COPD patient and need to wear the sensor node. Therefore sensor node should be wearable and non-obtrusive to allow patients to move freely. A sensor node, which is too large, would influence the patients' freedom considerably. Therefore a small sensor node is desirable.

Day away from home

Patients should be possible to leave their home environment, out of the vicinity of the gateway, to do groceries or visit friends and relatives. The sensor node will then not be able to reach the gateway for a longer period of time. As a consequence, a mechanism must be developed to prevent data loss when the user moves out of the vicinity of the gateway.

2.1.2 Researchers

The researchers need to gather data using this system. The considered aspects of the researchers are: the system in use for research and the addition of extra sensors.

System use for research

For research purpose it is desired that all raw sensor data can be stored on the USB-stick. During the research stage of the iCare4COPD project, the data on the USB-stick should be anonymised from the actual patients' identity. It is assumed that each patient gets their own gateway, which implies that the gateway does not need to cope with a multi user use case. Each patient can have several sensor nodes, but the sensor nodes of one patient should only be able to transfer the data to one single gateway. Otherwise the gateway requires to be able to determine which sets of sensor nodes belong to an individual patient. This is required to be able to combine different sensor data.

Different sensors

During the research period of the iCare4COPD project it might be the case that different vital signs are interesting to be measured. It is undesirable that all the electronics of the sensor node should be adjusted because the researcher wants to use a different sensor. To cope with variation of sensors, one generic board and a different smaller, sensor specific, board is required. The behavior of the generic board should be adjustable to be able to use the sensor specific board. This will reduce the overall costs for electronics and development within the project, since parts of the system can be reused.

2.1.3 Caregiver

The final system should be able to provide the caregiver insights about the patients status. The considered aspect for the caregiver is that the final system is used by the health care provider.

System use by health care provider

The final system should notify the physicians about the status of the patients. The physician should be able to access the patients' data remotely and take appropriate actions based on this information.

2.2 Global architecture

The global architecture follows from the use cases described in Section 2.1. Figure 1 depicts the proposed global system architecture for the iCare4COPD project.

At the left most box in Figure 1 the home situation is shown. Within the home situation the patient wears the sensor node which communicates wireless, via ZigBee, with the gateway. Wireless transmission of the data should minimize the burden on the patient. With the

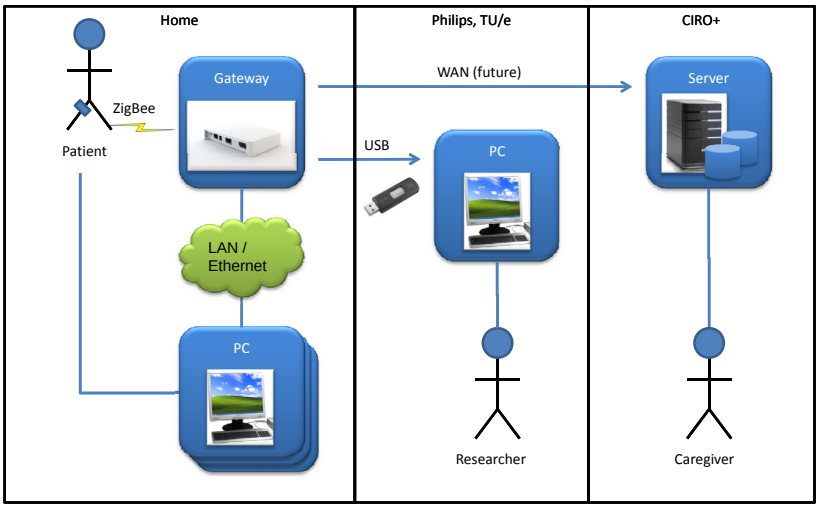


Figure 1: System architecture

iCare4COPD study patients should also be able to fill in questionnaires, which should be gathered by the gateway. These questionnaires can be filled in using a PC or laptop. The gateway in the final system should transfer the gathered data the server, however for research purposes the gathered data is stored on a USB-stick.

Patients should be able to receive feedback about their status via the gateway. Eventually the gateway could notify the patient via TV or computer. The patients receive feedback in their own home environment.

In this architecture there are four different main entities: the sensor node (shown in Figure 1, located on the patients leg), the gateway, the server, and PC's. Each entity has own purposes and constraints. The main purpose of the sensor node is to sample the sensors and transfer the data to the gateway. The gateway collects the data from the sensor node, perform the analysis algorithms and provides coaching and feedback to the patients. Eventually, the gateway should transfer the sensor data or the summarized results of the algorithms to the server. The server collects the data of several gateways and provide information of different patients to the physician. The PC's are used to analyze the data from the sensor nodes.

Resource constraints

Patients should be able to wear the sensors with the least impact on their daily life. Therefore the sensor node will be equipped with a small battery, to keep the sensor node small. This implies that the sensor node has limited energy. To reduce energy consumption the sensor node is equipped with a low power processor, which also limits the processing power of the sensor node. By rule of thumb, large storage capacity requires more power

than small storage capacity, therefore the sensor node is equipped with a small storage capacity.

The gateway is plugged into the mains socket and will be powered the whole year through. Therefore it is desirable that the gateway will be energy efficient. The gateway has to transfer the data to the server, or store it on a USB-stick. Therefore it is not required that the gateway has a large storage capacity.

To be energy efficient, the gateway will also be equipped with a energy efficient processor, which does limit the processing power of the gateway. The gateway will not be equipped with a high-end performance processor. These processors consume too much power.

Distributed processing

Because the sensor node is battery powered and equipped with a low power processor, the processing capabilities are limited. Algorithms that should run on the sensor node, should be algorithms that be aware of the power consumption. The algorithms can reduce the sensor data by combining the data locally or by sub-sampling the sensors.

The gateway has more processing power compared to the sensor nodes. Therefore the algorithms that combine sensor data and algorithms that determine the status of a patient, should run on the gateway. Coaching schemes and patient feedback should also provided by the gateway.

The server shows the status and data of several patients to the physician. The server would not have to perform many analysis algorithms, but would require a lot of storage capacity to store the patients' data.

Scalability

Several sensor nodes can transmit their sensor data to one gateway. However the number of sensor nodes is limited. The limiting factor is the bandwidth limitation of the wireless protocol. The theoretical maximum bandwidth of ZigBee is 250 kbps. Assuming the data rate in Appendix B (193 Byte/second) this would imply that at maximum 150 sensor nodes can transmit their data to one single gateway. However it is expected that 3-5 sensor nodes should be sufficient to monitor a patients' status, more than 3-5 sensor nodes would also be to invasive.

2.3 Project scope

The implementation part of this thesis will be limited to the measurement system, which involves the measurement of sensors and the transmission of the data to the gateway. The

measurement system consists of the sensor node, located on the patients' body and the communication with the gateway.

The main feature of the prototype design of this project is an unobtrusive, body worn sensor node that measures the users' activity and some of the users' vital signs. The main features of the prototype are:

1. Detecting physical activity: physical activity measured via an accelerometer
2. Detecting vital signs: measuring respiration rate and SpO_2
3. Wireless communication of sensor data via ZigBee

Figure 2 illustrates the data flow of the prototype. As Figure 2 depicts, the data flow goes from sensor to the sensor node and finally to the gateway. The electronics designed for the sensors (sensor specific boards) are physically located in the same housing as the sensor generic board. The sensor specific boards designed for the thesis are for the respiration sensor and the SpO_2 sensor. These are described in more detail in Section 3.4 and in Section 3.5.

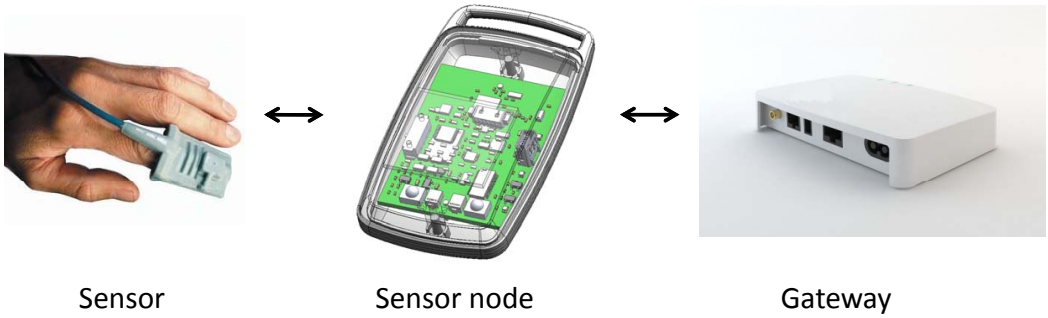


Figure 2: System data flow

The sensor node collects the data from the sensors' electronics. The raw measured data is transmitted via ZigBee to the gateway. The gateway is then responsible to process the data received from the sensors nodes. In the case of the prototype this will be limited to storing the data onto an USB-stick.

2.4 Requirements of the sensor node

In this section the most important requirements of the system will be discussed. The most important requirements for the sensor nodes prototype are shown in Table 1.

Table 1: Main Requirements

Requirement	**Reasoning**
Measure physical and physiological activity	Reduced physical and physiological activity might be important parameters to predict exacerbations
Sensor node be smaller than 8 x 6 x 2 cm^3	The sensor node should not become too large to wear, otherwise patients might be less cooperative to wear the sensory system. The chosen size is approximately the size of a mobile phone.
It should be possible to connect different sensors without redesigning the whole system	This should decrease development time and cost to integrate different sensors. During the research project it is unknown which sensors should be used.
Reprogram DSP	For research purposes it should be possible to change the algorithms, which runs on the DSP of the sensor node (e.g. to test a sub-sampling algorithm)
Battery lifetime should exceed 12 hours	A short battery lifetime would have a too large impact on the daily activity on the patients
A full day of stored sensor data should be transmitted within 4 hours	At night it is expected that the patient does not wear the sensor nodes and the system is in vicinity of the gateway. It should be possible to transmit the data of a full day measuring within 4 hours such that the data transfer could be done at night. Also, a higher data throughput increases the battery life, since the radio is on for a shorter period of time.
The sensor node should be able to store up to 6 hours of raw sensor data.	This enable the patients to be out of the vicinity of the gateway for a longer period of time

The points, which are critical to quality are:.

- Energy consumption (battery lifetime)
- Measurement robustness
- Mechanics and usability
- Data throughput
- Communication reliability
- Measurement accuracy

The energy consumption should be taken into account with the prototype, because the prototype will form a basis of input for the final product. The final product should be very energy efficient. Data throughput and communication efficiency contribute to the energy efficiency. It is expected that the transceiver consumes a large part of the energy; therefore it is desired to make efficient use of the radio.

Communication reliability is important because it is not allowed to lose measurement data due to communication. Unreliable communication will result in gaps in the data. These gaps might conceal critical information on which algorithms can be based on. The measurements should also have a sufficient accuracy to allow features to be extracted to base the algorithms on.

3 System implementation and design

This section describes the implementation of the sensor nodes generic board, and the design of sensor specific boards including, the design for the respiration band and the SpO_2 sensor. Results are given at the end of each section.

3.1 Generic sensor board

The generic sensor board measures the sensors (electronics) and transmit the data to the gateway. This section describes the design and implementation of the generic sensor board.

3.1.1 Related work

Much research in the field of body area networks has focused on issues related to wireless sensor design, sensor miniaturization, low-power sensor circuitry and signal processing. M. Chen et al. [3] present an overview of body area networks. Body Area Networks (BAN) seldom work alone, sensor data almost always need to be combined. The design of intra-BAN communication, radio communication of about 2 meters around the human body, is critical [3]. In our architecture the gateway gathers the data and performs the analysis algorithms.

The BodyANT is especially designed for monitoring of daily life activity. The BodyANT shows that a battery lifetime can be achieved up to 5 days while continuous transmitting accelerometer data at 32 Hz. The BodyANT requires retransmitting of data to lower the loss of data, which can result up to 15% for individual activities [10].

The MITes, another body sensor node, is based on a Nordic nRF24E1 chipset, which is a system on chip (processor and transceiver in one). The drawback of using an integrated chipset is that the processors performance is relatively low. Much processing resources are used up for handling wireless communication [9]. The design of the BSN Node is most similar to the design proposed in this work. By using 20 pins connector, various digital and analogue interfaces are provided with the BSN [9]. The BSN node is designed to run TinyOS. The BSN Node also has external flash which is connected to the microcontroller of the BSN. However, the proposed has more onboard memory required to be used for a longer period of time, when the sensor node is not in the range of the gateway.

3.1.2 Starting point

First a pre-study was conducted on different types of wireless transmission protocols. This pre-study included communication speed and power consumption comparison. The compared transceivers, including the transceivers used in the related work, had their own positive and negative points. AME's experience with ZigBee was decisive.

The starting point of the generic sensor node is that the wireless transceiver is the EM357 from EMBER. The EM357 is a ZigBee (IEEE 802.15.4) transceiver. This starting point is imposed by the foreseen road map of AME. AME wants to expand its ZigBee product portfolio in health care market. AME currently works with the predecessor of the EM357, the EM250. AME replaces the EM250 for the EM357, because the EM357 is more energy efficient, and has a better range than the EM250. Like the EM250 the EM357 is a system on chip. The EM357 has an ARM Cortex-M3 based processor on board. The fact that the EM357 has its own processor on board brings opportunities for different architectural topologies.

3.1.3 Design Principles

This paragraph describes the design principles and the consequences they impose. The design principles were:

Generic vs. specific All hardware components that are required by the sensor should be placed on the sensor specific board. The components required to collect and transmit sensor data should be placed on the generic board.

It is desirable to a make one generic board for all sensors. This reduces the price of production, compared to making a dedicated sensor node board for all different sensors.

Miniaturization Smaller components are preferred over larger components, to keep the footprint down.

Battery lifetime A long battery lifetime is desired, components with low power consumption are preferred.

Flexibility The Digital Signal Processor (DSP) should be reprogrammable. Although it is assumed that the algorithms that run on the DSP are limited in complexity.

Aim for standardized interfaces This enables a better interchangeability of the sensor node components and interchangeability of different sensors.

Digital interface preferred Digital interfaces enable interchangeability and by rule of thumb sensors with a standardized digital interface require the least amount of additional hardware to connect. Moreover, their usability for engineers is better, because often digital interfaced components have similar interfaces. Analog sensors might require an analog filter before any useful information can be extracted. By rule of thumb, digital interfaced sensors are more energy efficient than analog sensors.

3.1.4 Sensor node architectural design

To increase the battery lifetime of the sensor node, the majority of the time, the EM357 needs to be in low power mode. The EM357 would consume too much power when it would sample the sensors. A low power DSP is needed to sample the sensors and to reduce the energy consumption. The choice of the DSP is described in Section 3.1.5.

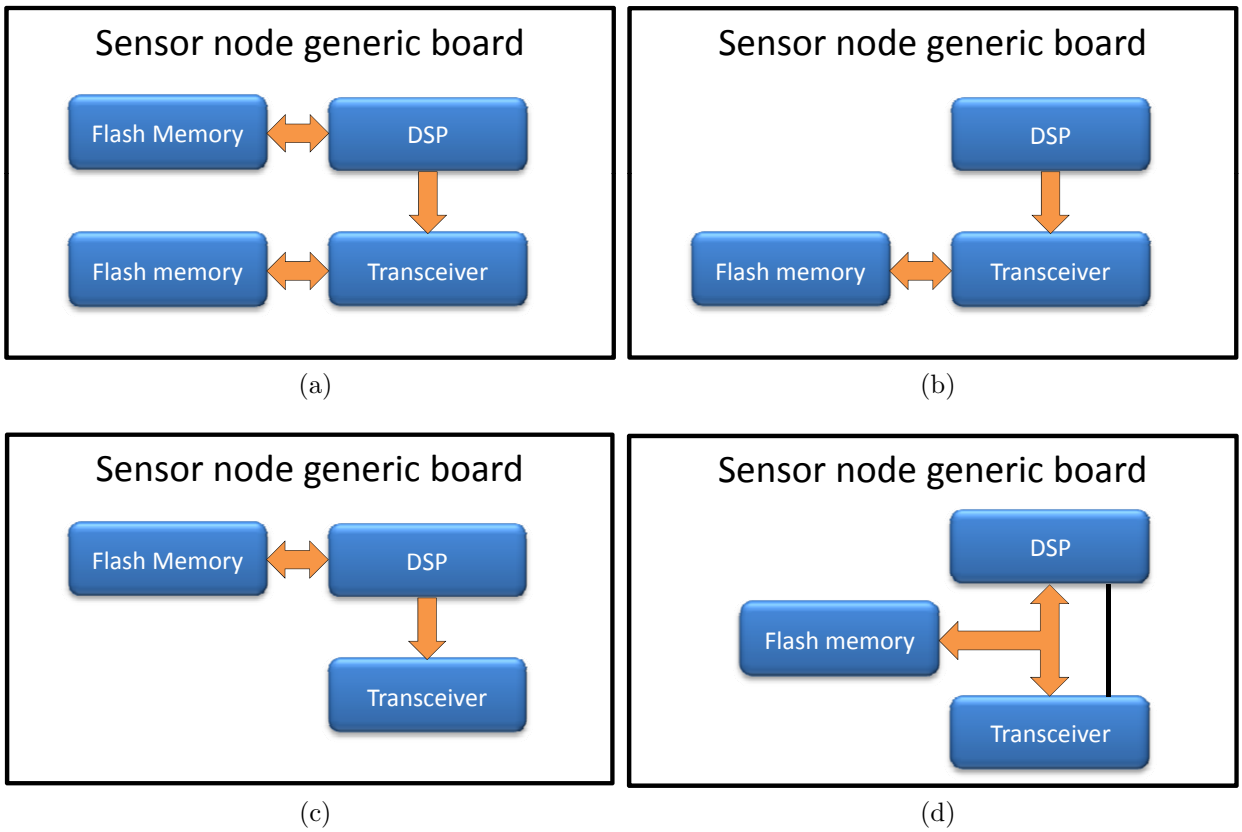


Figure 3: Considered sensor node architectural topologies: 3a: a design with two external flash chips, 3b: the design with the flash located at the transceiver, 3c: the flash located at the processor side 3d: the design where both the processor and transceiver can access the external memory.

Figure 3 depicts four different topologies, which were considered as possible architectural designs for this project. The components are shown in the blocks. The orange arrows depicts the dataflow over Serial Peripheral Interface (SPI) busses. The black line represents an extra wire used to claim the shared bus. These four architectural designs are compared to each other in the next pages.

Dataflow

In all topologies shown in Figure 3, the DSP samples the sensors. With topology (a), the DSP first stores the sampled data to the flash, connected to the DSP. When the transceiver can transmit data to the gateway, the DSP first retrieves the data from the flash and then transmits the data to the transceiver. When the transceiver fails to transmit the data to the gateway, the transceiver can store the data in its own flash, to retransmit later. Or the DSP can decide when to transfer the data to the transceiver. The transceiver should then decide if it can transmit the data to the gateway or if it should store it into flash. In topology (b), the DSP directly transmits the sampled data to the transceiver and the transceiver can transmit the data or store it in flash to retransmit it later. With topology (c), data needs to be stored by the DSP and sent to the transceiver. The transceiver must notify the DSP on the status of the data; otherwise the DSP would not know what to do with the data, if it can be discarded or retried. In topology (d), the DSP sets the data into flash, where the transceiver can retrieve the data from, to transmit.

PCB size

The Printed Circuit Board (PCB) size is deduced from the number of components (assuming the same components are used) and the number of busses. Only topology (a) requires two flashes, therefore it is expected that this topology requires the largest PCB size. All other topologies have 1 flash and two busses, except topology (d), which has only 1 bus. However topology (d) requires two extra wires to claim the bus. By requiring two extra SPI busses on the DSP, the DSP requires to have more SPI connections available or, one SPI bus need to be shared with other components. Sharing the SPI bus reduces the available bandwidth for each component. Topology (c) requires a larger DSP, compared to topology (b) and (d), because it requires extra SPI busses on the DSP, or the SPI bus needs to be shared.

Power consumption

The majority of the energy goes to the transceivers radio. With topology (a), it is possible that data will be stored twice, once by the DSP and once by the transceiver. The transceiver can enable the radio when there is enough data to transmit. In topology (b), the data needs to be stored at the transceiver side. This requires more power than storing the data at the DSP side, since the transceiver consumes more energy than the DSP. With this topology

the radio of the transceiver can also be efficiently used. With topology (c), the data is stored in the flash at the DSP side. Each time the DSP sends something to the transceiver, the transceiver needs to enable its radio, which is bad for the battery lifetime. Topology (d) requires only one store and one read to flash. The radio of the transceiver can be used efficiently when there is enough data and the sensor node is in vicinity of the gateway.

PCB costs

The PCB cost is composed out of the number of components and the price of the components. All topologies have the same amount of components, except for topology (a). When assuming the same flash is used topology a is most expensive. However when smaller flashes, which are cheaper, are used the PCB of topology (a) would be cheaper.

Software complexity

It is expected that topologies (a) and (b) have similar software complexity. With topology (c) the DSP needs to keep track of the data transmitted, which imposes extra complexity regarding the DSP and communication with the transceiver. However this topology would simplify the software on the transmitter. Topology (d) requires a bus claiming protocol, which imposes the most software complexity.

Trade-off

Table 2 depicts the trade-off matrix of the sensor node generic board topologies. A '-' depicts a negative rating, the '+' a positive rating and the □ a neutral rating. As the tradeoff matrix shows, the PCB size is positive rated for topology (b) and (d). Topology (c) is rated neutral because this topology requires a larger DSP. And topology (a) got a '-', because it requires two flash chips.

Flexibility rates the different possible data flows. Topology (b) and (c) have the least flexibility since with these topologies the dataflow is fixed, it is always the same component that stores it to external flash. Topology (d) is rated neutral because in software it can also be the case that the DSP first transmits the data to the transceiver and the transceiver stores it into flash. However this would not be logical, since this would consume much power, since the transceiver is awake.

The PCB size and software complexity are the two aspects of most interest. The power saving, that would be implied by the architecture, is calculated in Appendix D. The impact of letting the transceiver store the data in flash only consists of 3.4% of the total energy consumption. Taken these points into expect and by looking at the rating in the tradeoff matrix, topology (b) is chosen to be the preferred topology.

Table 2: Trade off of different architectural topologies

	topology (a)	topology (b)	topology (c)	topology (d)
PCB size	-	+	□	+
Power consumption	-	□	+	+
Flexibility	+	-	-	□
PCB costs	-	□	□	□
Software complexity	+	+	□	-

3.1.5 Hardware design

The hardware of the generic node has the architecture as shown in Figure 4. As Figure 4 shows, the central components of the generic sensor node are the transceiver and the DSP. The transceiver is connected to external memory, the real time clock and the buttons. The DSP is connected to the sensors and the battery measurement circuit. The sensors are the accelerometer and via the header connected sensors. The header connects the generic sensor board with the sensor specific sensor boards. The design also includes a voltage regulator, which regulates the voltage for all the components of the sensor board. The sensor board also incorporates a battery charge circuit, which enables the use of rechargeable batteries. More information regarding the voltage regulator, and battery charge circuit, battery measurement circuit can be found in Appendix E.

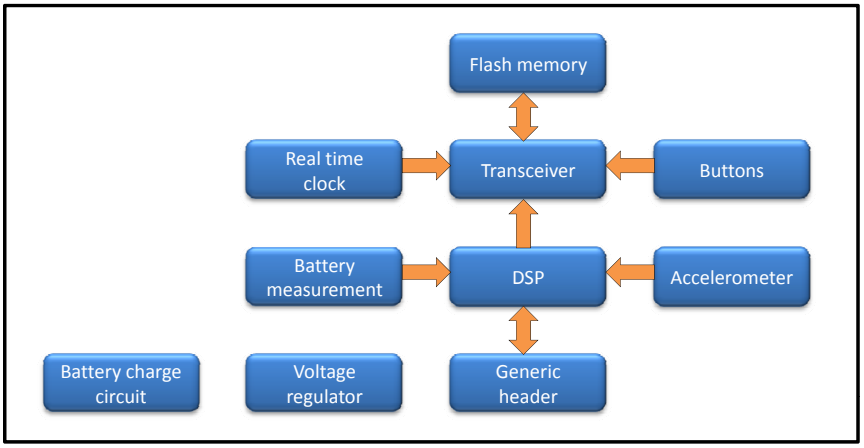


Figure 4: Hardware architecture sensor generic board

Housing

Figure 5 depicts the sensor generic board, with a sensor specific board, in the casing. The housing is the MINITEC EL from OKW. The dimensions of this housing is 78 x 48 x 20 mm^3, which meets the size requirements set in Section 2.4. This casing was chosen because

it was known that it does not cause any allergic reactions on patients. As can be seen from Figure 5 the housing has a round shape, which makes wearing the sensor less of a burden.

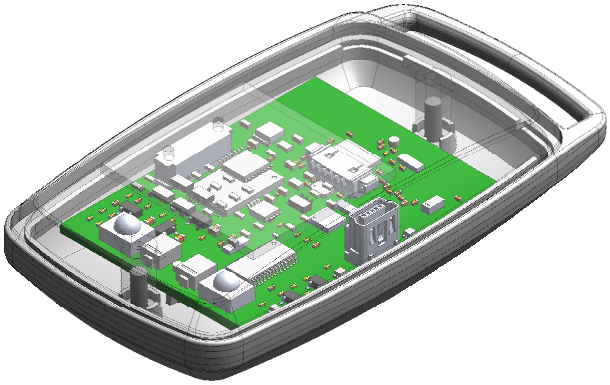


Figure 5: 3D drawing of the generic board with sensor specific board (SpO_2)

Header

The header is shown in Figure 6 and offers the following functionality to the sensor specific board:

- 2,7 V and 3.6V / 5V Power supply
- Ground power
- 1 counter capture pin
- 3 wired SPI
- 4 ADC pins
- 4 GPIO pins

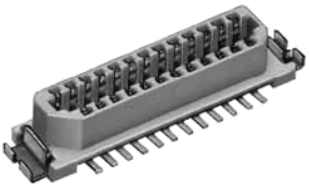


Figure 6: DF9 connector

Battery choice

It was required that the sensor node will be battery powered. The chosen battery is a 3.6V Lithium Polymer (LIPO) battery with a capacity of 300 mAh. This battery has a discharge capacity of 300mA. A LIPO battery was chosen due to the high capacity to weight and size ratio. This particular battery was chosen because of its size. The size of this battery enabled the placement of the battery to be located underneath the PCB in the casing.

DSP

To increase the battery lifetime of the sensor nodes, the sensor nodes should consist of low power components. Selecting a low power processor seemed to be a logical choice. Processors from the MSP430 series from Texas Instruments (TI) are known to be very energy efficient, when the task is not computational intensive and when the processor can sleep most of the time [7]. It was expected that only non computational intensive processing algorithms would run on the DSP. It was expected that, within the eventual sensor node, the algorithms would be optimized and would be able to run on the DSP without help of any Operating System (OS).

AME also has experience with the MSP430 family. For this reasons the MSP430 family was chosen. The considered processors are listed in Table 3. As can be seen in Table 3 the computational power is bound to 16 or 25 Million Instructions Per Second (MIPS). This is the maximum frequency of the clock and the instruction execution can be done in a single clock period.

Table 3: List of considered MSP430 processors

MSP430 type	Size (mm3)	RAM	Flash	GPIO (max)	SPI	ADC	MIPS
F2013	4,4x5x1.15	128B	2kB	10	1	1 (16bit)	16 Mhz
F2132	5 x 5 x .9	512B	8kB	24	2	8 (10bit)	16 MHz
F5310	7 x 7 x .9	6 kB	32 kB	31	2	6 (10bit)	25 MHz
F2410	10 x 10 x 1.4	4 kB	56 kB	49	4	8 (12bit)	16 MHz

The MSP430F5310 would be the preferred DSP, however the delivery time of the MSP430F5310 would be too long for the duration of the thesis project. The long lead time is because this is a new product of TI. The MSP430F2013, which AME uses in different other products, was considered to be too weak for this project. The MSP430F2410, with a footprint of 10 x 10 mm^2 was considered to be too large. The chosen processor is the MSP430F2132. As soon as the lead time of the MSP430F5310 becomes shorter it might be interesting to reconsider the DSP choice.

External flash

The flash was required to have enough storage capacity to store several hours of raw measurement data. The chosen external flash is the AT45DB642DT, which is a 64Mbit flash. The external flash has a footprint of 13.4 mm x 8 mm. The flash can store up to 11 hours of raw measurement data (assuming 696 kB/hour is generated as described in Appendix B). When data compression algorithms would be used it is expected that the flash can store more hours of measurement data.

Real time clock

The real time clock (RTC) is connected to the EM357. Two real time clock (RTC) chips were considered, namely the DS3234SN and the DS2417P. The chosen RTC was the DS2417P, which is a one wire RTC. This RTC was chosen due to its small footprint. The real time clock was required to be able to synchronize time. This was required to determine the actual time when a sample is taken. The RTC deviates maximum 2 minutes per month, which calculates back to 4 seconds per day.

3.2 Software design

This section describes the software design of the sensor node and gateway. A framework for the DSP has been created to provide a generic software interface for the developer. This helps to be able to quickly adopt the system to use other sensors.

3.2.1 Design principle

The DSP samples the sensors and transmits it via the EM357 to the gateway, where it is stored on a USB-stick. For the developer of the system the communication between DSP and gateway is transparent. It does not matter for the developer that programs the DSP, if the DSP physically communicates with the EM357 or directly to the gateway. Eventually the USB-stick will contain the measurements that the DSP transferred to the transceiver.

Software framework DSP

The design of the software framework, which will be on the DSP, is shown in Figure 7. This design has a layered structure and can be divided in the following blocks:

- Configuration
- Communication

- Sensors
- Functional
- HAL (Hardware Abstraction Layer)

Figure 7: Class diagram design of the DSPs framework

The configuration block contains the board configuration and the message configuration. These configurations can be used globally throughout the project. The board configuration contains definitions to abstract the code from the physical hardware configuration. The board configuration links functionality to the actual hardware configuration. The message configuration file specifies the messages, which are sent from the transceiver to the DSP.

For the communication, a queue and a host abstraction are made. The main function block, pushes measurement data into the queue. The queue combines the data into packets. When a packet is full a Cyclic Redundancy Check (CRC) will be added, which is used to make communication between transceiver and DSP more reliable. The host-file abstracts the functionality to transmit data to the transceiver. It also contains a small buffer. Whenever communication between DSP and transceiver does not succeed and the packet has to be retransmitted, the packet from this buffer is taken.

The sensors block abstracts the functionality from the type of sensor and gives a generic interface to the rest of the framework. The framework can then use this block to sample the sensors. When a new sensor is introduced, code that reads out the sensor need to be created and only small modifications in the main of the framework is required to be changed to be able to use the sensor.

The functional block contains the part that measures the battery and controls the LED. The LED can be used to notify the user about the status of the DSP, for instance when the battery is empty.

The HAL contains the code, which is directly related to the hardware. The hardware components available are:

General Purpose Input Output (GPIO) GPIO can be used to control the output and read the digital value of the input of the pins.

Timer The timer used to indicate the time on the DSP.

Serial Peripheral Interface (SPI) SPI is the used digital communication interface.

Counter The Counter is the connected to a hardware counter. The hardware counter/-timer can be read with this functional block.

Analog The Analog block reads out the analog voltage from the analog enabled pins.

State diagram DSP

The main function block as shown in Figure 7 contains the state machine of the DSP. Figure 8 depicts this state machine.

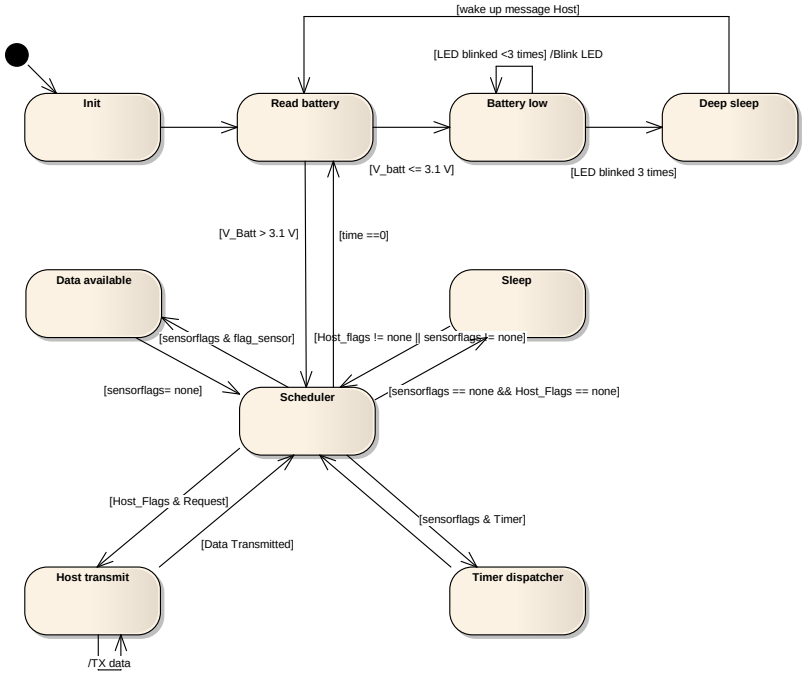


Figure 8: State machine of the DSP

When the system is started, the peripherals are initialized and the battery is read. When the battery voltage is low the system will blink the LED three times and go into deep sleep state. In deep sleep, the system consumes the least amount of power. This is important because the battery of the system is a LIPO battery. If the protection circuit of the LIPO fails the battery could get broken because LIPO's are vulnerable to deep discharge.

When the battery voltage is above a predefined value, in this case 3.1 Volt, the system will continue in the scheduler state. The scheduler state contains all the logic to determine the actions required. When the timer is increased, the main is notified. The main will change its state into the timer dispatcher state. In the timer dispatcher state every function, which depends on the timer is called. When there is data available at the sensors the state changes into data available. Within the data available state the sensors are sampled. The sampled data will be put into the queue. When the queue is full the system will go into the Host transmit state, where the system will transmit the data via SPI to the transceiver.

3.2.2 EM357

The EM357 should receive data from the DSP and transmit it to the gateway. This section describes the implementation of the EM357. The implemented state diagram is shown in Figure 9.

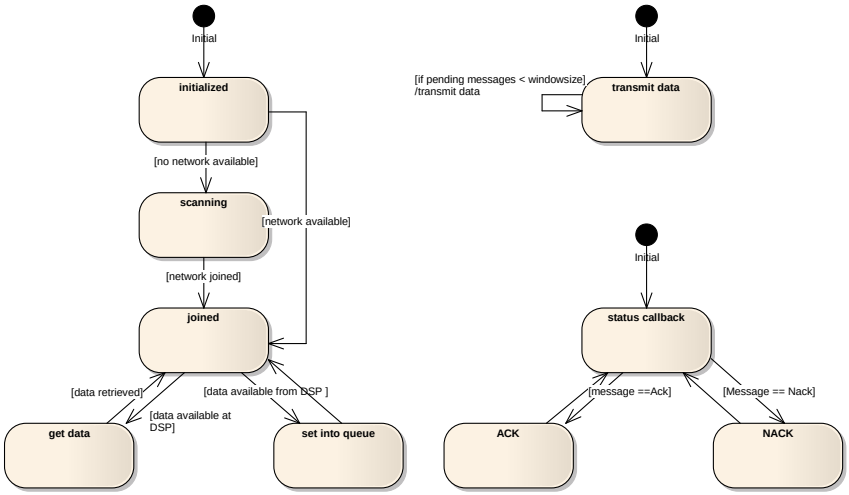


Figure 9: State diagram EM357

As Figure 9 shows, the state diagram is divided into 3 individual parts. The state diagram on the left is the main state diagram. The main state diagram controls the all over behavior of the system. After initialization and there is no network available, the EM357 will scan and try to join a network. When the EM357 is already in a network the state will change to joined without scanning for a network.

When, in the joined state, the EM357 receives a signal from the DSP that there is data available, the EM357 retrieves that data. In the get data state the EM357 will gather the data from the DSP. The gathered data is then pushed into a queue.

The state diagram indicated in the top right checks if there is data in the queue that needs transmitting. Data is only transmitted when the amount of pending messages is smaller than the maximum pending amount (window size).

The state diagram in the right bottom corner handles the message callbacks. The system generates message callbacks when data messages are successfully transmitted or when this is not the case. The message state, as used in the concurrent sending protocol (see below), is changed with information of this callback (Ack / Nack).

Communication with DSP

When the DSP indicates that there is data available, the EM357 retrieves data from the DSP. Listing 1 shows the pseudo code, which retrieves data from the DSP. The EM357 will gather data from the DSP until the indication pin is set low. When the pin is set low the EM357 inspects the packet to see if the packet is received successfully. This is done with a CRC check. When the packet is received successfully, the EM357 sends an Ack back to the DSP.

```
1 while (pin ==high)
2   data[i] <- SPI-Read;
3
4 size <- extract Size(data)
5 crc_msg <- extract CRC (data)
6 crc_calc <-calculate CRC(data)
7
8 crc_msg == crc_calc
9   SPI-write <- ACK
```

Listing 1: EM357 retrieving data from DSP

Communication with gateway

As argued in Appendix C.3.1, a concurrent transmission policy is required and pushing the data from sensor node to the gateway is the desired transmission method. To be able to send messages concurrently a data structure is required to keep track of the messages status. The data structure is shown in Figure 10. The data structure contains a Transaction Sequence Number (TSN) and a state. The state can be Pending, Ack and Nack, indicating if the packet is yet to be transmitted, already has been Acknowledged, or if it received a timeout.

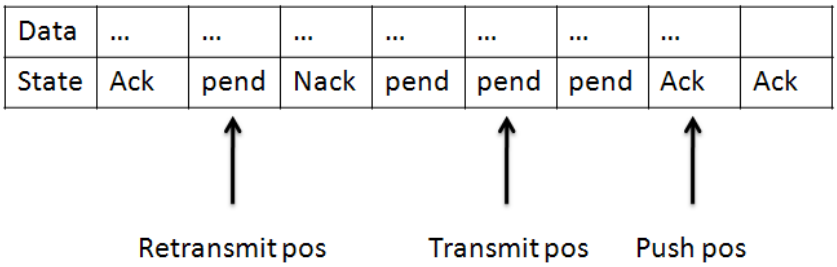


Figure 10: Data structure used for concurrent pushing of data from the sensor node to the gateway

The packets are put into a queue. The queue has three position pointers, indicating the retransmit position, transmission position and the push position. Whenever a new data element is put into the queue this element is put in the place where the push position is referring to. After the element is put into the queue the push position is incremented. Whenever an Acknowledge is received from a packet the state of that packet is changed to Ack. The retransmit position is updated to the next position where no Ack is located. This is required because the acknowledgements can be received out of order. Simply moving the retransmit position every time an Ack is received can result in a wrong indication of which data needs to be retransmitted. Whenever the update scheme, which moves the retransmit position, encounters a Nack, the system will set the transmission position to the retransmit position, all states in between is set to pending again. By resetting the transmit position to retransmit position the window is reset. The EM357 will try to resent everything in the window, including the Nack'ed message.

```
1 while (queue[retransmit_pos].state == ACK)
2   retransmit_pos <- retransmit_pos + 1;
3
4 if (queue[retransmit_pos].state == NACK)
5   reset window;
```

Listing 2: pseudocode of the update of the retransmit position

Whenever a packet is transmitted, the transmission position (Transmit pos) is incremented. A packet can only be transmitted when the difference between Retransmit pos and Transmit pos (the amount of pending messages) is smaller than a predefined window size.

Transmission speed measurements are conducted and can be seen in Appendix H. The maximum data throughput archived was 8 kB/s, which is 64 kbps, initially this was 2.3 kbps. With the improvement of transmission speed the bottleneck was moved to the gateway, which could not keep up with the amount of incoming messages.

3.2.3 Gateway

The software for the gateway is based on software available by AME. A class was added, which captures and stores the incoming messages. The function, which stores the data, checks if the incoming message is a known type of message. If the message type is not known, the message is ignored, if known the TSN is checked. If the TSN is already known then it is assumed that the packet is already in the buffer. The data in the packet is extracted and the origin of the packet is checked. When the origin data is known, the gateway knows how the data looks like and stores the data accordingly.

Data on the USB stick is stored XML based format. XML is a standard to store data. This is chosen because XML offers a generic implementation of storing data. When the gateway encounters a power down, all data in buffers (located in RAM) is lost and the XML file is not closed properly. The XML file should be finished manually.

3.3 Results

The hardware of the generic sensor board is shown in Figure 11. Here we can clearly see the buttons (green), the generic header for sensor specific boards (red), the DSP (yellow), the transceiver (blue) and the flash (purple).

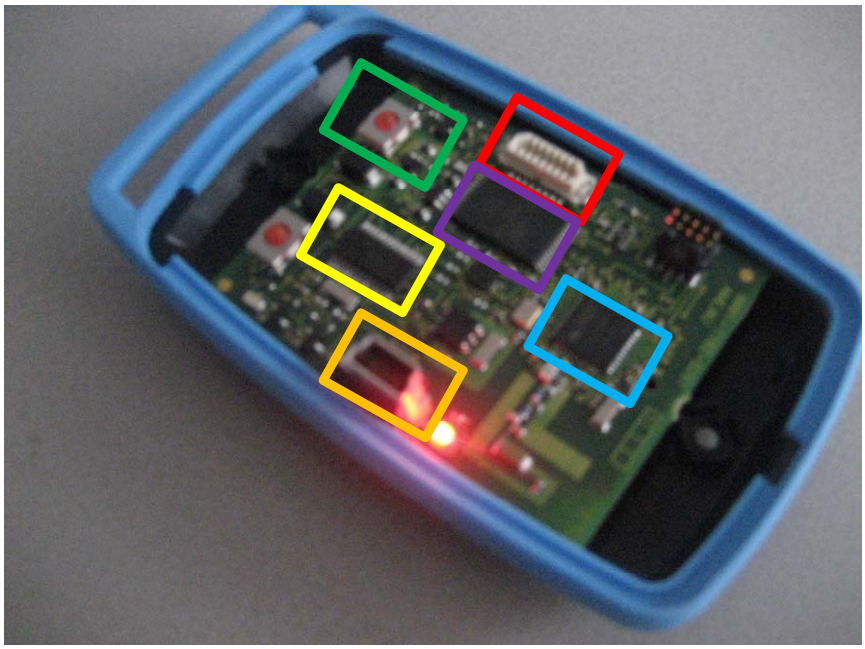


Figure 11: The sensor generic board

Due to a small design mistake in the electronics of the generic sensor board, prevented reprogramming the DSP via the generic header. This was caused because some ports were shared with other components, interfering with the programming line. In a next version this should be resolved.

The software framework on the DSP requires 6.8 kB of flash and 372 bytes of RAM, this excludes the queue. The DSP has 512B RAM and has 8kB of flash, which results that only 1.2 kB of flash and 140 bytes of RAM is available for the queue and for algorithms. However, the code of the framework is not optimized. Optimizing the code of the framework could increase the available RAM. Because the available RAM is only 140 bytes, it would also be recommended to reconsider the DSP in the next version.

A range of 42.5 meter was achieved. This range was tested in the hallway of AME with the sensor nodes within sight. When the sensor node is put into the pocket of a person the range decreases to 14 meter. This decrease is caused by the absorbance of the signal by the human body.

3.4 Respiration sensor board

To measure the respiration rate, the respiration band from ambulatory monitoring is used. This is an elastic band containing an insulated wire. In this section the circuitry and software for the respiration band is discussed. First background information of what can be found in literature about the sensor is given, followed by the hardware and software design. Finally results of the respiration sensor is given.

3.4.1 Background information

The respiration band can encircle the chest and/or abdomen. The induction of the band depends on the enclosed cross section of the band. By connecting this band to an LC-oscillation circuit the oscillation frequency can be determined. The changes in oscillation frequency indicate changes in the enclosed cross. By measuring both chest and abdominal cross sectional areas normal breathing and airway obstruction can be detected [4]. The expected impedance of the respiration band is approximately 6.1 μH.

When two bends are used, the bands can influence each other. If the modulation is excessive, the two oscillators will lock onto the same frequency and independent measurement of the 2 bands will no longer be possible. Possible locking of the oscillation frequency can be avoided if the two oscillator frequencies are properly chosen [4]. When two bands are measured and the measured values are multiplied by weighted constants and added together, this would result in a quantitative estimation of the change in the subjects' lung volume [16].

3.4.2 Hardware design

For the respiration band a simple LC-oscillator has been designed. The circuit for the respiration sensor is shown in Figure 12.

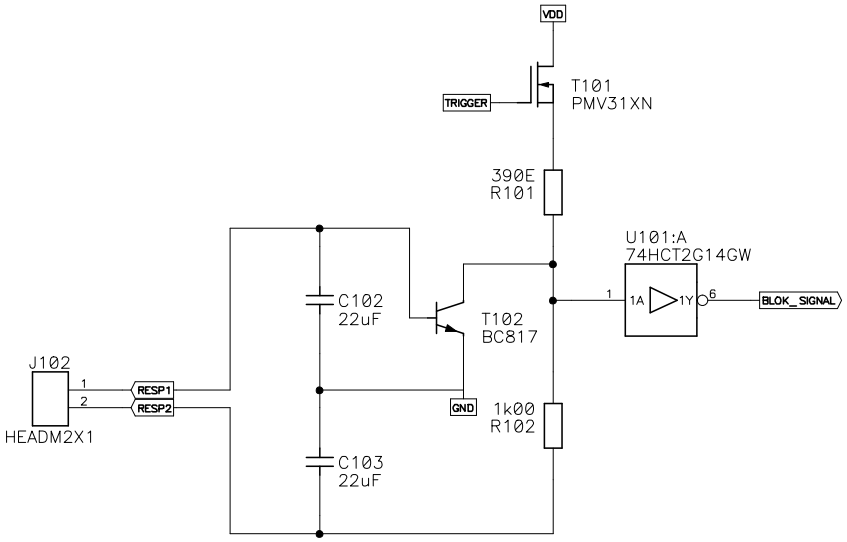


Figure 12: Colpitt oscillator circuit for respiration band

This design is a Colpitt's oscillator in combination with an inverter. The inverter acts like a Schmitt trigger and transforms the sinusoidal waveform of the oscillator into a block function. The change of inductance of the respiration band will influence the frequency of the oscillator, described by Equation 1. The oscillation frequency of the oscillator can be adjusted by changing the capacitance of C_1 and C_2.

$$f_0 = \frac{1}{2\pi\sqrt{L \cdot \frac{C_1 \cdot C_2}{C_1 + C_2}}} \tag{1}$$

The implementation of this circuit has two headers; one to connect the respiration band to the circuit, the other to connect this sensor specific board to the sensor node generic board.

Before implementing the circuit, the circuit has been simulated with the assumption that the impedance of the respiration band is similar as found in literature (6.1μH see Section 3.4.1). The simulation results of the circuit are shown in Figure 13. However this simulation results are from before the inverter. The inverter will make the signal more square and will increase the peak-to-peak value of the signal.

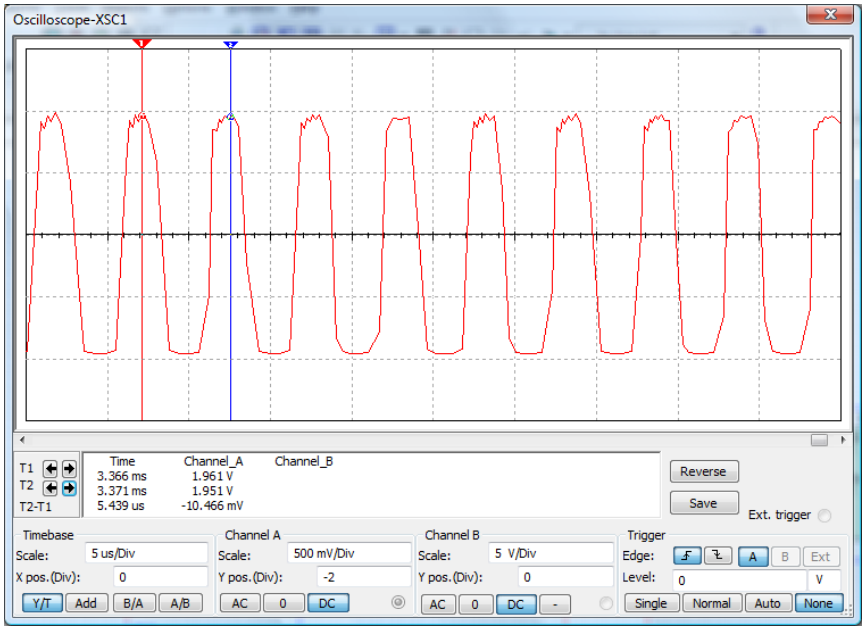


Figure 13: Simulation results of the oscillation circuit

As can be seen in Figure 13 the peak of the oscillation circuit is around 1.7 Volt, and the minimum is located at 38mV, the frequency of the circuit is around 170 kHz. Within the datasheet of the inverter the following thresholds for the falling and rising are stated. The 2.7 V voltage supply is an interpolated result of the two above stated voltage supplies. As can be seen from Table 4 the simulated voltage output would be sufficient to drive the inverter.

Table 4: Voltage thresholds inverter

Voltage supply (Volt)	**Negative going threshold (Volt)**	**Positive going threshold (Volt)**
4,5	0,87	1,58
5,5	1,11	1,78
2,7	0,438	1,22

Stability issues

With the band we did not encounter any stability issues. However, when this does occur a Clapp-oscillator could be considered. The advantage of the Clapp-oscillator, in comparison with the Colpitts-oscillator, is that variations of the load have less impact on the oscillation. This has a positive effect on the oscillation stability and accuracy. The resonance frequency of a Clapp oscillator can be calculated by Equation 2.

$$f_0 = \frac{1}{2\pi\sqrt{L \cdot \frac{1}{\frac{1}{C_1}\cdot\frac{1}{C_2}\cdot\frac{1}{C_3}}}} \tag{2}$$

3.4.3 Software design

The output of the inverter will be connected to a hardware counter of the DSP. Every time the software framework calls the timer callback function is called a predefined amount of times, the respiration sensor need to be measured. The difference between the previous amount of the hardware counter and the current amount is determined.

3.4.4 Results

Figure 14 shows a scope plot of the respiration band circuit. The top graph shows the raw signal, which will go into the DSP. Below the respiration frequency is shown. The breathing cycles are clearly shown in this bottom graph. At the center of the graph, where there is a gap, the breath was held. At the end of the frequency graph a movement artifact can be seen, which was expected since long wires were used.

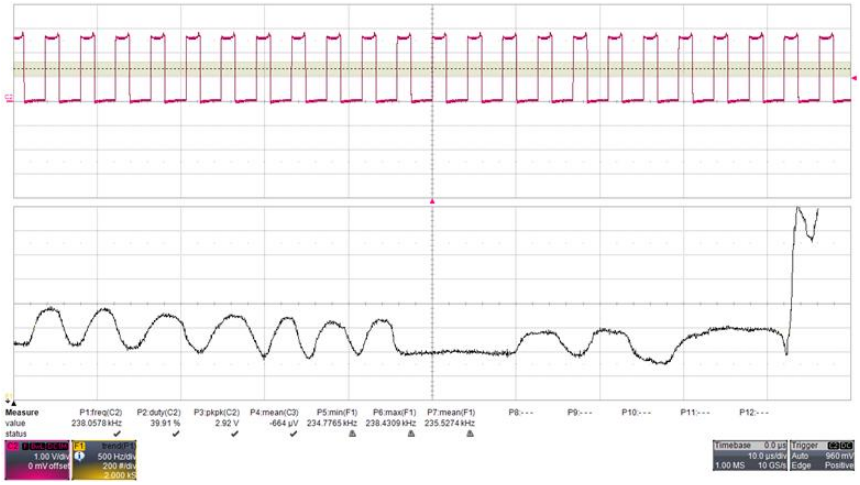


Figure 14: Results of the respiration sensor board

3.5 SpO_2 sensor board

This section describes the design of the SpO_2 sensor. The hardware of the SpO_2 is based on the TMS320VC5505 DSP Medical Development kit from Texas instruments [11].

3.5.1 Background information

The SpO_2 sensor consists of two LED's, a red colored LED and an infrared colored LED. This sensor is typically placed on the index finger. The red and infrared light passes through the index finger. Hemoglobin in the blood absorbs the red and infrared light, the amount of light absorbed is an indication of the amount of hemoglobin in the blood. Hemoglobin is the protein in the blood, which transports oxygen from the lungs to the rest of the body.

Hemoglobin molecules reflect more red light when they are oxygenated, whereas, the reflection of infrared light increases with de-oxygenated hemoglobin molecules [15]. Reddy et al [14] formulates the formula to calculate the $\%SpO_2$ as follows:

$$\%SpO_2 = 110 - 25 \cdot Q \tag{3}$$

Where,

$$Q = \frac{V_{pR}/V_{DCR}}{V_{pIR}/V_{DCIR}} \tag{4}$$

In Equation 4 the symbols represent:

- V_{pR} = peak-to-peak value of the red light signal
- V_{DCR} = DC part of the red light signal
- V_{pIR} = peak-to-peak value of the infrared light signal
- V_{DCIR} = DC part of the infrared light signal

The peak-to-peak value of the light is caused by the different states in cardiac cycle. Figure 15 depicts the cardiac cycle and indicates where the systole and diastole state begins and end. The systole state is the contraction phase during the cardiac cycle. The diastole phase is when the heart fills with blood.

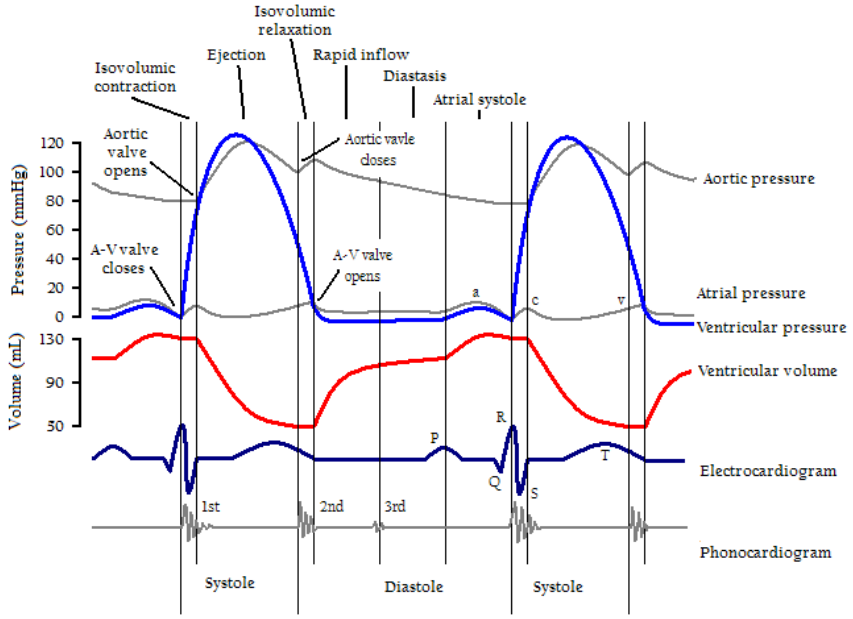


Figure 15: cardiac cycle, [5]

3.5.2 Hardware design

The hardware architectural design is shown in Figure 16. In the architecture in Figure 16 the connector on the right is the generic sensor board connector and the connector on the left is the sensor connector. To be able to use the SpO_2 sensor appropriately the LED's intensity can be controlled by the DA5886, a digital-to-analog converter. The LED's intensity needs to be controlled to prevent the photosensitive diode of being saturated the whole time. To determine the SpO_2 value the peak-to-peak value and the DC value is required. Both can be extracted from the raw signal indicated in the architecture. The amplified signal is the signal after DC correction; here the signal is amplified and the peak-to-peak signal can be extracted more easily.

3.5.3 Software design

Every 2 times that the timer callback is called by the framework, the SpO_2 is measured. To take a measurement of the SpO_2 sensor the red LED is enabled first and the output is sampled. After sampling the output, the red LED is disabled and the infrared LED is enabled. The output is sampled again and finally the infrared LED is disabled. Between enabling the LED and sampling the output a delay of 450 μs is set. This is the same "on time" as described in the architectural description of the SpO_2 board from TI [11].

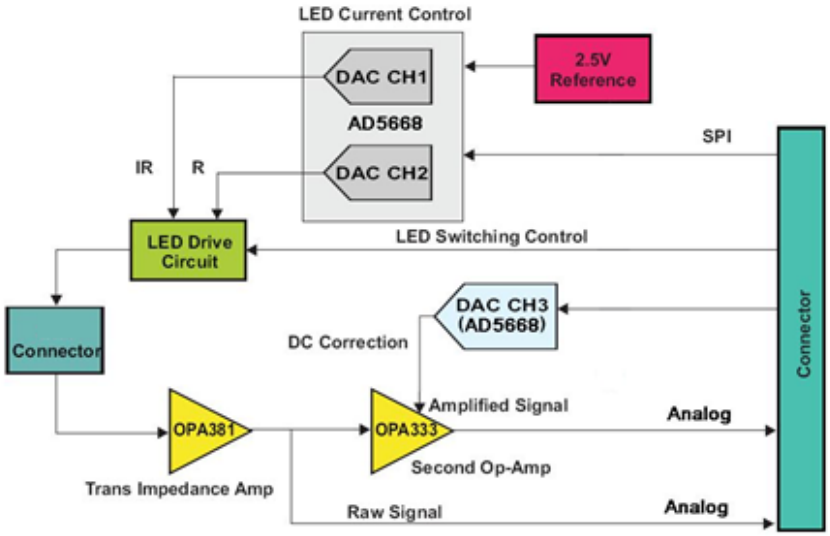
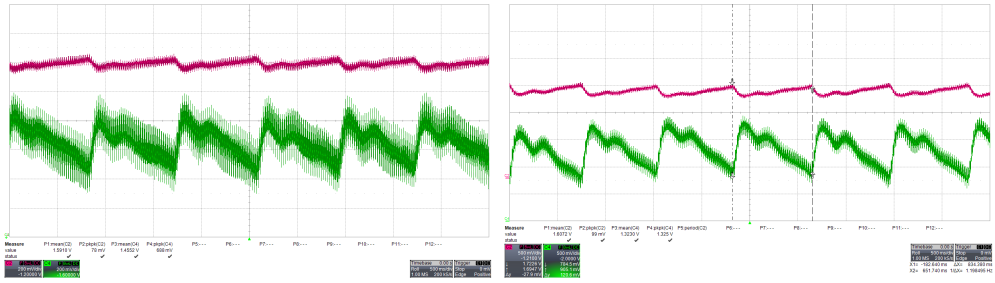


Figure 16: hardware architecture SpO_2 [11]

3.5.4 Results

The result of the measurement circuit is shown in Figure 17. With this measurement the LED of the SpO_2 sensor is continuously on. The top signal shows the raw signal, as indicated in Figure 16, the bottom signal shows the signal after the second Op-Amp. The scale is 0.2 Volt per division with the signal produced by the red signal, the scale for the infrared signal is 0.5 Volt per division. As Figure 17 clearly shows, the heart rate can clearly be deducted from this signal, especially considering with the infra red light.



(a) (b)

Figure 17: Results of the SpO_2 measurement circuit 17a: the result when only the red light is on, 17b: the result when only the infrared light is on

4 Influences of motion activity on sensor accuracy

The sensors will be worn by the patient during the day. Therefore the influences of motion on the respiration sensor has been investigated. Motion influences the accuracy of the sensor. First the motivation for the sensor choice will be given, followed by the breath detection algorithm and finally the accuracy of the breath detection algorithm is described.

4.1 Motivation respiration band

Two physiological sensors were considered in this thesis, namely the respiration and the SpO_2. Much research has been conducted on the influences and the reduction of motion artifacts of SpO_2 sensors.

Yan et al. [19] describes the influence of different types of motion on the sensors signal. Vertical movement of the arm causes blood pressure in the hand to change, which influences the signal quality. However composite movement had the most influence on the signal quality. Raghuram et al. [13] described an algorithm to reduce motion artifacts on signal level. Raghuram used different low and high pass filtering techniques to filter out the high and low frequencies of the signal, which could be caused by motion. Fourier series analysis has also been successfully used to reduce the motion artifacts on SpO_2 signals. In some cases the impact of motion artifacts was reduced from 37% to 3% [14].

Much research has been conducted on the SpO_2 sensor, however not much literature has been found on the influences and reduction of motion on the respiration signal. Cohen et al. [4] reported that the signal of the respiration band is influenced by movement of the body. Even waving arms influences the respiration band signal quality. The position of the respiration band seemed not to be of influence on the motion artifacts of the respiration signal [4]. Because not much literature has been found on the influences of motion activity on the respiration sensor, the motion influence on respiration sensor will be investigated.

4.2 Sensor data

Existing measurements of 9 healthy test subjects were given. The age of the test subjects was 38.3±18.6 years and with a Body Mass Index of 23.1±5.5. Measurements were taken with 3 accelerometers, a respiration band and Electro Cardiogram (ECG) electrodes. The sample rate of these sensors were 125 Hz for the accelerometer, 62.5 Hz for the respiration band and the ECG electrodes were sampled at 250 Hz. Data of the ECG electrodes were not used.

The placement of the sensors is shown in Figure 18. The accelerometers are placed on locations where eventually sensor nodes will be placed, near other sensors. The accelerometer in the pocket is added to make a distinction between sitting and standing. The accelerometer positioned on the chest was placed loosely on the chest. The accelerometer on the waist was put in the pocket of the test subject. The one on the wrist was strapped around the wrist. The respiration belt was placed on top of the clothing for convenience of the test subject. The ECG pads were placed underneath the clothing.

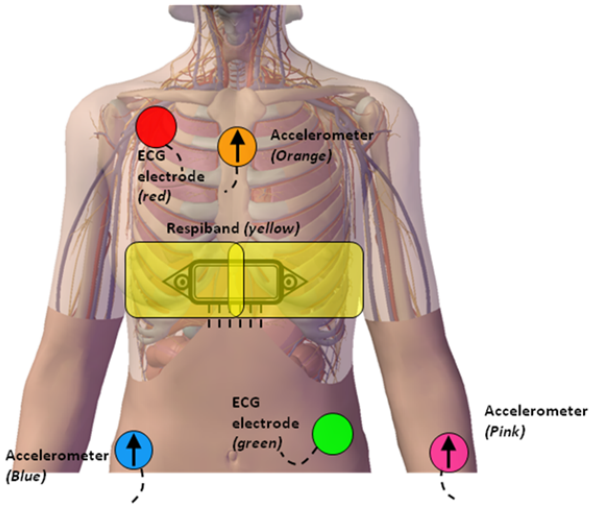


Figure 18: Placement of sensors for data gathering.

4.2.1 Protocol

Data was gathered while the test subjects performed the following protocol:

- Fill in questionnaire (baseline measurement)
- Slow walk for 5 minutes
- Fill in questionnaire
- Normal walk for 5 minutes
- Fill in questionnaire
- 6 Minutes Walking Distance Test (6MWDT)
- Fill in questionnaire
- Lie down for 2 minutes
- Stand for 2 minutes
- Sit for 2 minutes

- Fill in questionnaire (baseline measurement)
- Walk on the treadmills for 5 minutes (incline: 15, speed: 5)
- Fill in questionnaire
- Wait for return to normal breathing
- Fill in questionnaire

Slow walk was introduced to imitate the slow movement of COPD patients. COPD patients move significantly slower [12]. The protocol phases could be directly derived from the raw accelerometer sensor data. Figure 19 gives an example of how the sensor data looks like.

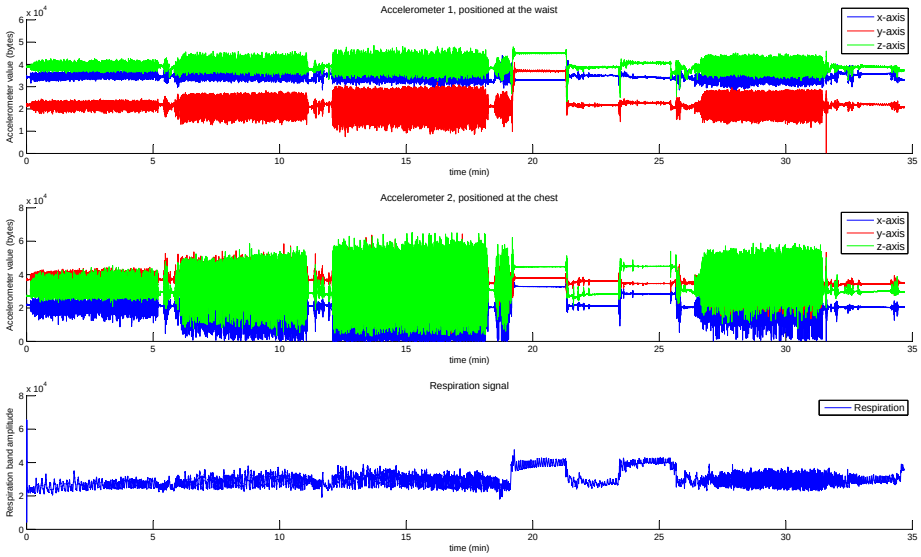


Figure 19: In the accelerometer graph the x-axis represents time in minutes, the y-axis shows the ADC values of the accelerometer data. The respiration graph displays a measurement, which is related to the thorax expansion.

In Figure 19 we can clearly see the protocol phases. The slow walk phase is shown between minute 1 and minute 5. After the slow walk phase the filling in of the questionnaire can be seen. Normal walk can be seen from minute 6 until minute 11. The 6MWDT starts from minute 12 until minute 18. Between minute 19 and 22, lying down can be seen, followed by standing (minute 22 - 24) and sitting down (minute 24 - 26). Finally we can see the treadmill walk between minute 27 and 32. As Figure 19 also shows, the respiration signal is influenced by the body position. The respiration signal has clear distinctive peaks between minute 19-22 and minute 24-26. This can be explained that the respiration band is more expanded in these phases.

4.3 Accuracy respiration data

To determine the accuracy of the respiration signal, the respiration cycles needed to be determined first. A respiration cycle consists of an inhale and exhale stage. In the data, a respiration cycle was defined as the time between two subsequent peaks.

4.3.1 Breath detection algorithm

Before the respiration cycles were determined, the respiration signal was filtered with a low pass filter, which took the moving average over a period of 1 second. A period of 1 second was chosen because it was expected that the breathing frequency is much lower and the low pass filter, with this period, removes most high frequency noise. To determine the start and end of a breathing cycle a hill climbing algorithm was used. The breath detection algorithm consisted out of a hill climbing algorithm, where the time between peaks is the breathing cycle. Two different hill climbing algorithms were considered: a reference hill climbing algorithm and the used hill climbing algorithm.

Reference hill climbing algorithm

The reference hill climbing algorithm was provided and works as follows. First the slope, the difference between two subsequent samples, is determined. As long as the slope is positive, the reference hill climbing algorithm continues to determine the slope of the next sample. When the slope becomes non-positive, which indicates a (local) maximum, the current position and height is stored as a candidate peak. The candidate peak is only stored when the height of this peak is higher than the previous stored candidate peak. Only when the current sample becomes lower than the candidate peak minus a threshold, the candidate peak is considered to be high enough to become a definite peak. When the threshold-limit is reached the candidate peak is reset. Requiring a minimum height of the peak eliminates small local maxima.

When all samples are analyzed, the set of peaks is filtered on the width of the peaks. This last part of the Reference hill climbing algorithm removes peaks, which are wider than a set width. First a window is placed over the hill, with the center of the window located on the peak. Then it is determined, if on both sides of the peak, there is a value which is less than the peak - the set threshold. When, on each side, there exists such a point, the peak is added to the final set of peaks.

Used hill climbing algorithm

We assumed that the patient breath frequently enough such that filtering on the width was not required. Filtering on the width requires extra memory. The amount of samples that need to be stored is equal to the required maximum width of the hills. This could require

a lot of memory. The width filtering also requires extra computational steps, which we wanted to keep to a minimum for when the algorithm must run on the DSP of the sensor node.

The used hill climbing algorithm can be found at [2]. In essence the used hill climbing algorithm works similar to the reference hill climbing algorithm. However the used hill climbing algorithm also determines the valleys of the signal. This is done in a similar fashion as determining the peaks. The used hill climbing algorithm also does not perform the last check, filtered on the width of the peaks. Implementation wise there is also a small difference. The main loops are differently defined, however due to a check in the reference hill climbing algorithm the loop bounds are the same. The used hill climbing algorithm was easier to understand than the reference hill climbing algorithm.

The output of both hill climbing algorithm is shown in Figure 20. The top graph of Figure 20 shows the found maxima and minima of the used hill climbing algorithm. The bottom graph shows the found peaks of the reference hill climbing algorithm.

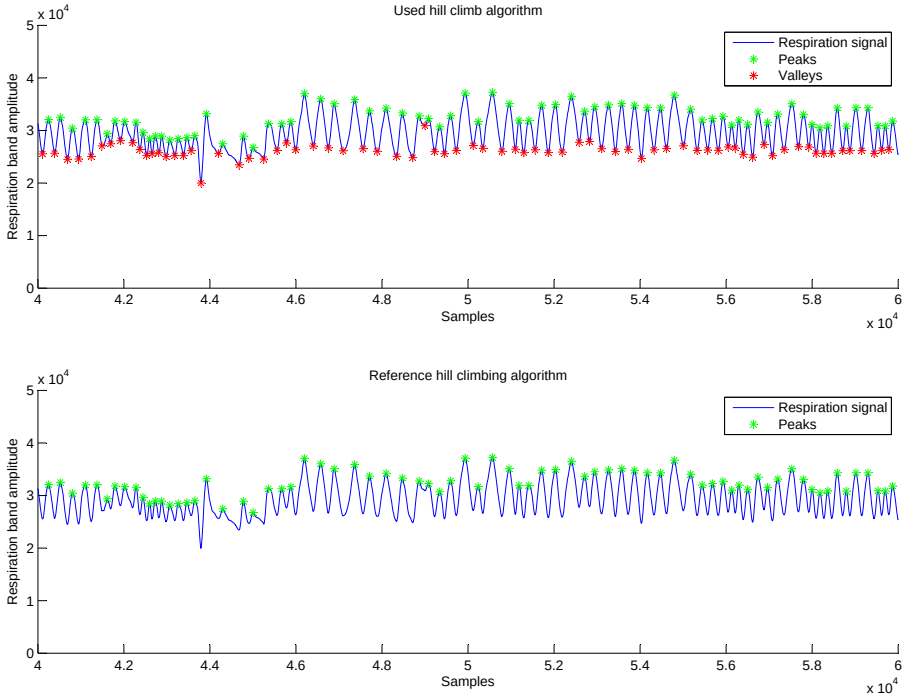


Figure 20: Comparison output hill climbing algorithm, top graph shows the output of the used hill climbing algorithm, bottom graph shows the output of the reference hill climbing algorithm, peaks are denoted with a green asterisk(∗) and valleys with a red asterisk(∗)

4.3.2 Accuracy respiration data

The accuracy of the respiration signal was determined manually with the help of the Marker tool. The Marker tool is a MatLab application where data sections can be annotated. The breathing cycles were labeled automatically with the breath detection algorithm. The phases where the breath detection algorithm misses a breathing cycle or phases where the breath detection algorithm determines a breathing cycle, when there was no breathing cycle, were annotated as inaccurate.

The signal was divided in the protocol phases as described in Section 4.2.1. Filling in the questionnaires are annotated as transition phases in Figure 21. The accuracy was determined by Equation 5, where N_{Total} represents the total number of samples and $N_{inaccurate}$, the total number of inaccurate samples.

$$Accuracy = \frac{N_{Total} - N_{Inaccurate}}{N_{Total}} \tag{5}$$

4.4 Results

In shown Figure 21 the average accuracy of the breathing cycle algorithm compared to manual detection of breathing cycles. The $\perp$ sign indicates the minima of the recognition of all test subjects in this phase, the $\top$ the maxima. The periods where the algorithm was the least accurate was during the transition phases. This could be explained by the change in body posture in these phases. The change of body posture could have caused changes in the cross section of the respiration band, which were detected as a breathing cycle. The accuracy in the transition phases varied between 58% and 100%, with an average of 88%. In non transition phases the accuracy is around 97%.

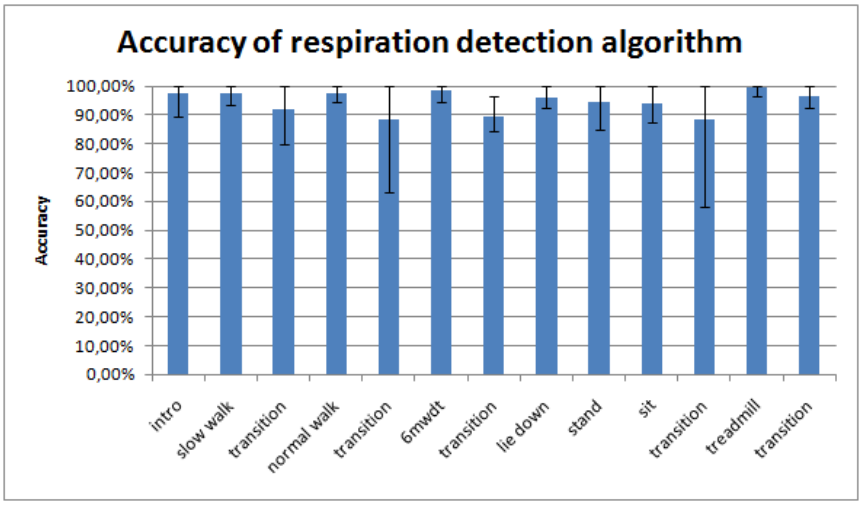


Figure 21: Average accuracy of the breathing cycle algorithm among all participants in %, compared to manual detection. $\perp$ indicates the minimum of the accuracy, $\top$ the maximum accuracy of one participant.

5 Energy reduction with activity dependent sampling algorithms

More investigation was conducted on the energy reduction of the respiration band, because the respiration band consumed much energy. It was expected that the respiration would be used throughout the day. It is also expected that at the end of the research, the SpO_2 sensor only for spot measurement. Therefore reducing the energy consumption of the respiration band would have a greater impact on the battery lifetime than the energy saving of the SpO_2 sensor. The respiration sensor draws 35 mA of current, compared to an estimated 40 mA for the SpO_2 sensor. The average current drawn by the generic sensor board is estimated to 1.85 mA, see Appendix D.

5.1 Related work

Before the selective sampling algorithm could be implemented, the activity needed to be recognized. Much of research has been conducted on using tri-axial accelerometers to detect activity. Tzu-Ping Kao et al. [8] used tri-axial accelerometer data with a fuzzy-basis-function-based classifier to recognize different activities. In their work the used features were: the mean, correlation between pair of axes and mean of the absolute deviation.

Others used the Fourier Transform to determine features like energy and power spectral density. A set of one-class classifiers has been used to recognize activity as done by Yang et al [20].

5.2 Algorithm constraints

To develop a classifier which can run on the sensor node the following constraints need to be considered.

1. Memory is limited to the range of kByte.
2. Processing time is limited to the range of MHz.

The limitation on the amount of data that could be stored in memory implies that no algorithm could run which requires large buffers, which is larger than a few samples. For instance comparing the signals of different cyclic activities requires a larger buffer.

The limited processing time effected the choice of available algorithms. Processing algorithms, which are computational intensive should be avoided. For example the calculation of a correlation between signals was considered to be too intensive to perform, since it requires to compare the measurement data with the reference data for a set window.

5.3 Activity classification

The aim of the activity classification algorithm was to determine the activity level of the patient. The activity includes the determination if the patient was sitting, standing, laying and different types of walking. A Naive-Bayes classifier was used to classify the protocol phases. A Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem (from Bayesian statistics) with strong (naive) independence assumptions. An advantage of the naive Bayes classifier is that it only requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification [17].

5.3.1 Feature extraction

The different types of walking were characterized by the amplitude and time between two peaks of the accelerometer signal. The accelerometer on the chest was chosen because the respiration band would also be located at the chest. To determine the difference between sitting and standing the orientation of the accelerometer placed on the leg was required.

The amplitude was determined with the same hill climbing algorithm, which was used to determine the breathing points. The amplitude was determined by the difference between the hill and the valleys, always in combination with the previous hill/valley. The time between two peaks was determined by the distance between the current sample and the last peak. To increase the robustness of the recognition the previous and its predecessor of the amplitude difference and time between peaks were also chosen as features.

To be able to make a distinction between standing and sitting the orientation of the accelerometer, which was placed on the leg, was required. The orientation of this accelerometer was used as a feature. The amplitude was subtracted from a baseline value, which consisted of the average of the first 60 samples.

5.3.2 Results

To determine the accuracy of the classifier the s-fold cross validation method was used. In essence this method consists of training the classifier with all datasets except the dataset, which you want to classify and repeat this until all data sets are classified.

A confusion matrix has been created to display the accuracy. The confusion matrix shown in Figure 22 is the sum of all confusion matrices, of the all tested cross validation fold. The generated confusion matrix excludes the datasets of subject 4, 8 and 9, because the data of these persons could not be trusted (e.g. from the data it could not be concluded if person 9 actually sat down). The classifier mismatched the standing class with an activity class, this can be explained because there was too much movement in these phases. In

standing phases it was assumed that the test subject filled in the questionnaires standing. The confusion in the walking part could be explained, due to walking in these classes were quite similar. Every person walks differently, the training data of the other persons could be similar to other types of walking for another person. With the treadmill test the speed was set on 5 km/h, during the 6MWDT it is expected that the test subject get similar result.

actual		lie	sit	stand	slow walk	normal walk	MWDT	treadmill
	lie	96,6%	0,7%	0,1%	0,4%	0,0%	1,5%	0,7%
	sit	0,0%	98,0%	0,4%	0,1%	0,1%	0,0%	1,4%
	stand	0,4%	0,9%	64,2%	15,6%	5,3%	4,0%	9,6%
	slow walk	0,0%	0,0%	0,7%	65,6%	12,4%	2,0%	19,3%
	normal walk	0,0%	0,0%	0,1%	17,8%	31,9%	27,2%	22,9%
	MWDT	0,0%	0,1%	0,1%	4,1%	17,9%	54,8%	23,0%
	treadmill	0,0%	0,2%	1,5%	4,4%	7,0%	38,9%	47,9%
		lie	sit	stand	slow walk	normal walk	MWDT	treadmill
					predicted			

Figure 22: Activity recognition confusion matrix

When normal walk, 6MWDT and treadmill test were combined, the following confusion matrix was obtained. This 3 classes were combined in the high activity class (high act.). As can be seen in Figure 23 there is still confusion between standing and walking, and slow walk and high act. walking. The same reasoning as above holds here.

actual		lie	sit	stand	slow walk	high act.
	lie	96,1%	0,7%	0,1%	0,4%	2,7%
	sit	0,0%	98,0%	0,4%	0,1%	1,6%
	stand	0,4%	1,0%	64,5%	15,7%	18,5%
	slow walk	0,0%	0,0%	0,7%	62,4%	36,9%
	high act.	0,0%	0,1%	0,6%	9,6%	89,7%
		lie	sit	stand	slow walk	high act.
				predicted		

Figure 23: Activity recognition confusion matrix, normal walk, 6MWDT and treadmill walking are merged in high activity.

5.4 Activity levels

Each protocol phase was labeled with a value indicating the type of activity. The activity level values vary from 1 till 7, based on the activity intensity. Lying down was set to 1 and the most intensive physical activity to 7.

By assigning values to the activity classes according to the actual type physical activity, the classifier predicts level of activity. The predicted level of activity could be low pass filtered, to filter out the noise caused by mismatched predictions. The result of the low pass filtered signal is shown in Figure 24. The top graph of Figure 24 shows the 3 axis of an accelerometer of a test subject. The bottom graph shows the protocol level and the classified activity level after applying the low pass filter.

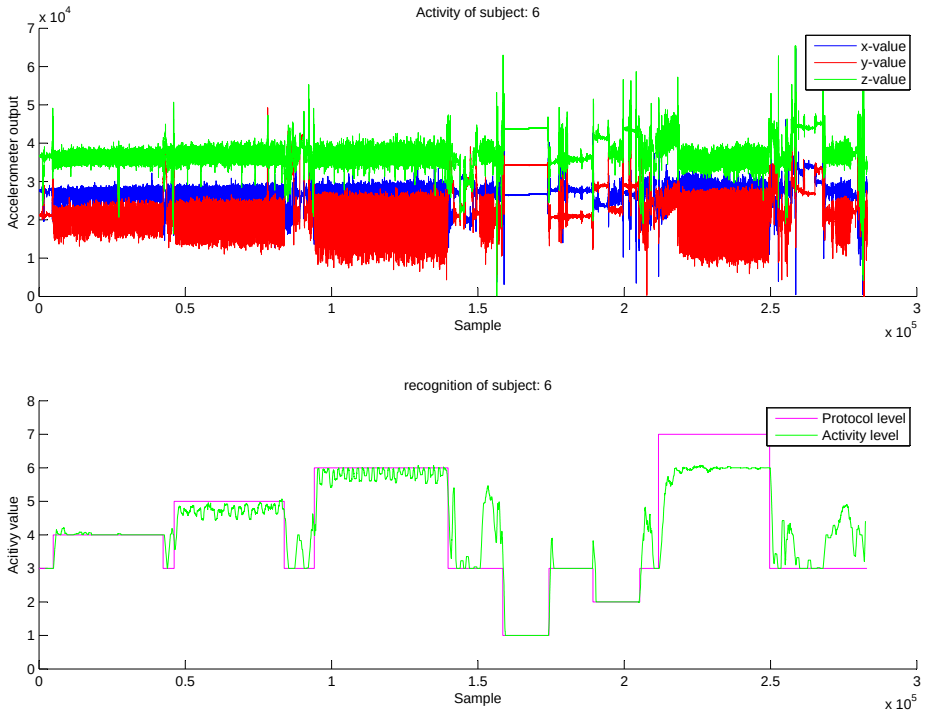


Figure 24: Activity level classification, the x-axis shows the sample number of the data, top graph shows the 3 axis of an accelerometer on the test subject, bottom graph shows the protocol type and classified activity type.

As can be concluded from Figure 24 the classifier recognized activity level. In this case the predicted activity levels approaches the levels of the protocol phases, with exception of the treadmill walking, which was recognized as the 6MWDT. The confusion matrix in Figure 22 shows that the classifier mismatched treadmill walking with the MWDT class, therefore we expected this to occur.

5.5 Respiration frequency

To determine the respiration frequency, 3 subsequent breathing cycles were taken and the interval time was determined by taking the begin time of the first breath and the end time of the last breath. By dividing the number of breaths by the time required for these breaths, the breathing frequency could be determined.

5.6 Sub sampling

To reduce the energy consumption of the sensor boards, a selective-sampling scheme, based on physical activity, was constructed. This section describes two ways of sub sampling, namely duty cycling, which is not based on physical activity and dynamic sampling algorithm, which is based on physical activity.

5.6.1 Duty cycling

The simplest of all sub-sampling algorithms is to duty cycle the measurement at a regular interval. First the sensor would be sampled for an amount of time, followed by a moment where it would not be sampled. This continues indefinitely. Figure 25 shows an example where the duty cycle is set to an interval of 60 seconds and a duty cycle of 50%.

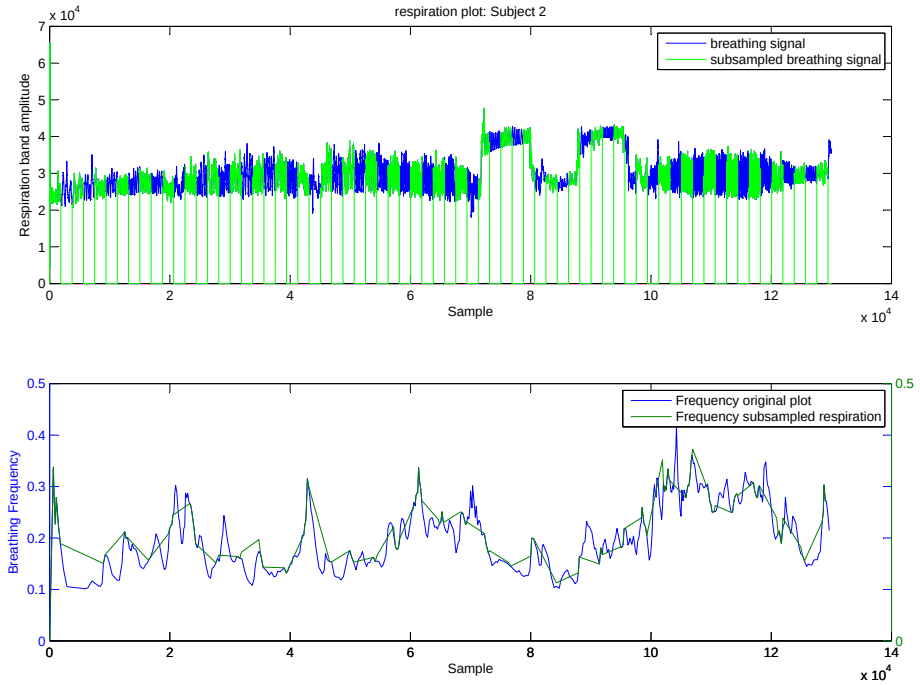


Figure 25: Respiration frequency determined from the sub-sampled data. The top graph shows the original signal and the sub sampled signal. The bottom graph shows the corresponding frequency graphs based on the sampled data and of the original data.

5.6.2 Dynamic sampling

The dynamic sampling algorithm was based on the interesting points to measure. In case of the COPD patients the interesting moments are defined by the breathing recovery time after, and the breathing frequency increase during, an activity. The increase of breathing frequency at the beginning of an activity was also considered to be interesting.

The dynamic sampling algorithm was based on the state diagram shown in Figure 26. As Figure 26 shows, the dynamic sampling algorithm has 4 states, namely steady_not_active, transition_active, steady_active and transition_not_active. These four states represent the 4 possible stages of activity namely idle, starting an activity, active and stopping an activity. The sampling method which was performed in each state are also shown in Figure 26 as effect of the transition. The duty cycles in the states steady_not_active and steady_active are variable.

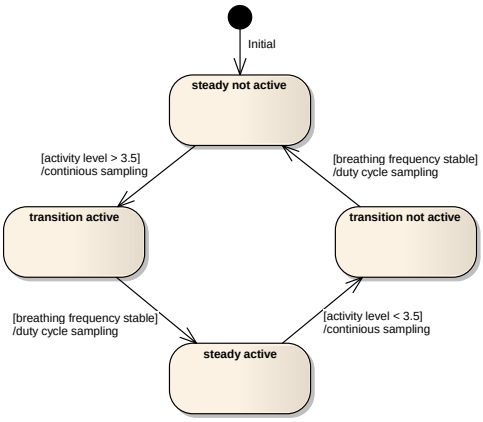


Figure 26: Dynamic sampling state diagram.

The initial state was steady_not_active. When the classified activity type becomes higher than a set threshold, the state changes to the transition_active state. This threshold was the boundary to determine when the person is starting an activity. In this algorithm the threshold is set to 3.5, which was the activity level between standing and slow walking. In the transition states, the respiration signal was always sampled until the respiration frequency remains constant, or the activity level changed again.

An example of the dynamic sampling algorithm is shown in Figure 27. The top graph in Figure 27 shows the respiration data (blue) and the sampled respiration data with the dynamic sampling method (green). The activity plot depicts the protocol activity level and classifier activity level. In this example the duty cycle in steady_active was set on 60% and in steady_not_active state on 40%. As can be seen from Figure 27 the sampling takes place right after the classified activity state is greater than the threshold of 3.5. As can be seen in Figure 27, after the end of an activity (min 7) it can be seen that the respiration is determined for a longer period.

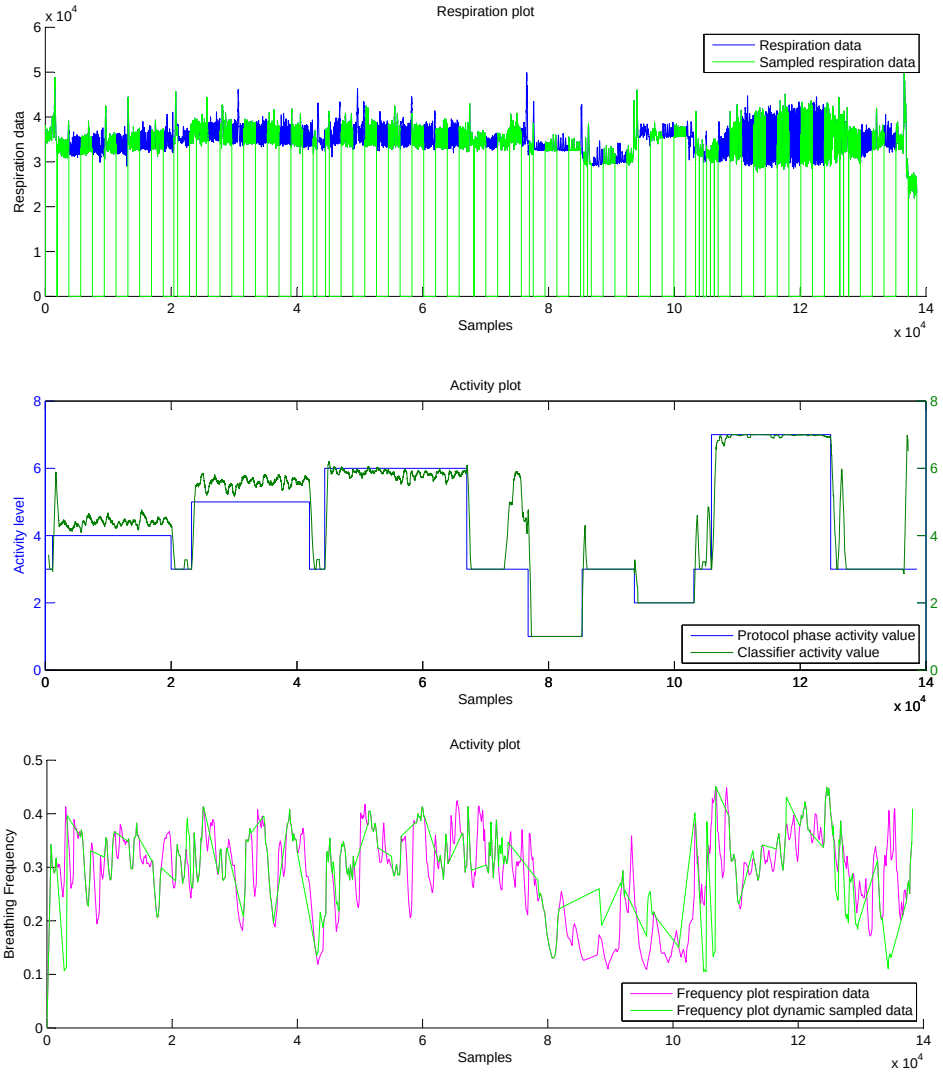


Figure 27: In the top graph: the respiration original respiration data, with the original data in blue, green represents the sampled data. The activity plot shows the determined activity level and the activity level of the protocol phase. The bottom graph shows the frequency plot of the original data (magenta) and the dynamic sampled frequency (green)

The corresponding respiration frequency plot is also shown in Figure 27. The frequency plot of the original respiration data is also shown in magenta and the result of the sub sampling algorithm is shown in green. The respiration frequency is determined from the sampled data.

5.7 Accuracy sub-sampling methods

The interesting measurement was the respiration frequency. To determine the accuracy, the frequency of the sub-sampled data was compared with the frequency of the original respiration data. Where the sub-sampled data missed the information to calculate the frequency, the breathing frequency was linear interpolated. To determine the accuracy of the dynamic sampling method a small deviation of 0.02 Hz was allowed. The accuracy was determined by the ratio of inaccurate number of points and the total sum of all points of the frequency plot.

The accuracy of the duty cycled algorithm is shown in Table 5. The accuracy of the dynamic sampling algorithm is shown in Figure 28. In Figure 28 both duty cycles were varied from 0 to 100% with steps of 10%, and the accuracy is shown in % of time which the duty cycled respiration frequency was accurate enough. At the origin the accuracy is 17%.

Table 5: Accuracy duty cycle

duty cycle	**accuracy**
20%	17,9%
30%	43,7%
40%	53,0%
50%	61,0%
60%	67,4%
70%	61,9%
80%	66,2%
90%	77,0%
100%	100,0%

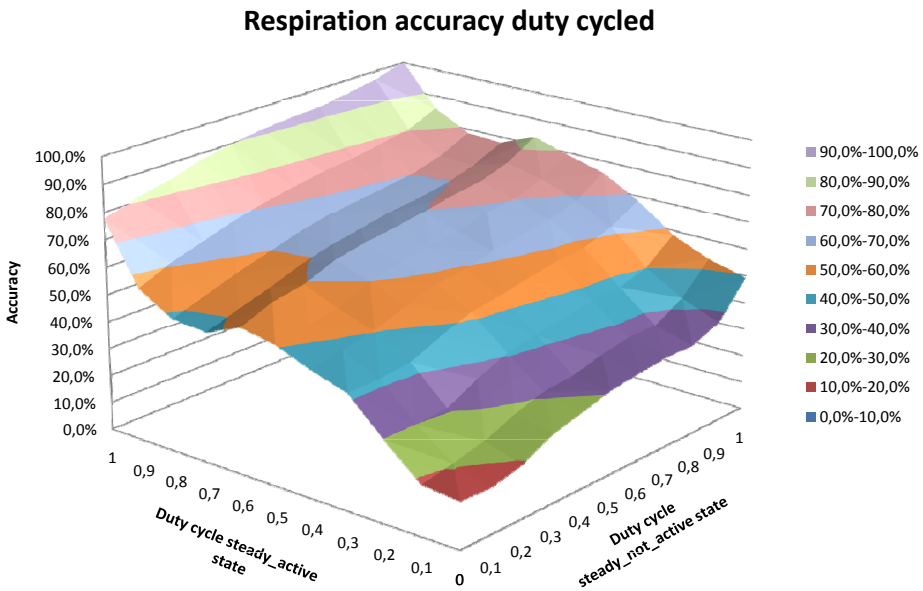


Figure 28: Accuracy of the dynamic sampled algorithm with the duty cycles varied from 0 to 100%, with steps of 10%.

5.8 Energy saving

When the sampling period is much greater than the startup period of the respiration band, the relative energy consumption of the duty cycled sub-sampling algorithm approaches the duty cycle. To determine the energy saving of the dynamic sampling algorithm, assumptions about the patients daily routine needed to be made.

In a study from F. Pitta et al. [12] the physical activities in daily life of COPD patients was compared with healthy elderly subjects. The average daily activity of the patents is graphically shown in Figure 29. For two days, 12 hour per day measurements were taken from patients and healthy test subjects.

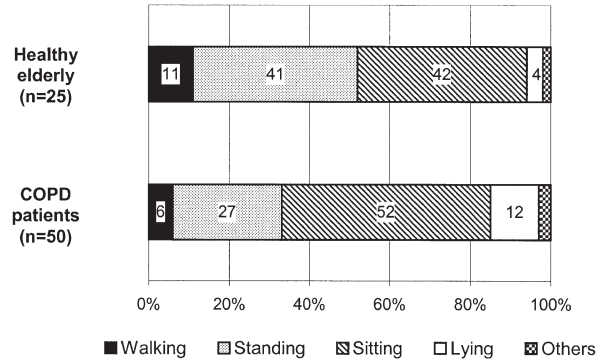


Figure 29: Percentages of time spent in each of the activities or body positions in healthy subjects and patients with COPD during the day. Others = cycling or undetermined activity (2% in healthy elderly subjects and 3% in COPD patients [12].

The study of F. Pitta et al. [12] gives insights about the total physical activities of COPD patients, however it does not give information about the average duration of a single activity. No literature was found, which describes the average duration of a single activity. Therefore assumptions about this have been made. It was assumed that the average single activity duration was approximately 5 minutes. The duration of the transition phases was assumed to be 3 minutes.

To determine the energy consumption, the amount activity periods needed to be calculated. This was done by dividing the activity time, with the average single activity duration time. During activity, energy saving would only take place during the last 2 minutes of every activity, since it was assumed that the transition phase lasted 3 minutes. The remainder of the time, when the person is not active, the sensor will be duty cycled and energy will be saved. The first 3 minutes after an activity was also set to be a transition phase and the sensor node is not yet in the steady_not_active.

Assuming the average COPD patient is active for 9 % of the day, walking and other activities in Figure 29, results in an active time of 130 minutes per day. By dividing this by 5 minutes this would result in 26 activity periods. When the time of each state would be calculated we would get the results as shown in Table 6.

Table 6: Time spent in each state

State	**time (min)**
steady_not_active	1232
steady_active	52
transition_active	78
transition_not_active	78

With the time spent in each state, the impact of each energy saving can be calculated. First the relative impact of each state is calculated with Equation 6. Where Time_{state} is the time spent in each state and Time_{total} the total time.

$$Rel_{impact} = \frac{Time_{state}}{Time_{total}} \tag{6}$$

The total energy consumption can be calculated by taking the sum of all states, as shown in Equation 7.

$$Energy_{relative} = \sum Rel_{impact} \cdot D_{state} \tag{7}$$

To calculate the relative energy consumption only the two duty cycled states have to be taken into account, because the impact of the transition phases are constant. To calculate the energy consumption Equation 8 can be used. In Equation 8, $\text{D}_{steady\_not\_active}$ represents the duty cycle of the not active state and D_{steady_active} the duty cycle of the active state.

$$Energy_{relative} = 0.856 \cdot D_{steady\_not\_active} + 0.036 \cdot D_{steady\_active} + 2 \cdot 0.054 \cdot 1 \tag{8}$$

5.9 Pareto graph

The accuracy of the dynamic sampling algorithm and its corresponding energy consumption are shown in the pareto graph of Figure 30. Data of all examined settings were used to construct this graph. The accuracy of the sub-sampling method, together with its corresponding relative energy consumption is shown in this graph. This graph only shows the dominating points of energy efficiency and accuracy. The pareto points of the dynamic sampled algorithm and the pareto points of the duty cycled algorithm are shown in this pareto graph. As can be seen in Figure 30 the dynamic sampled algorithm dominates the duty cycled algorithm.

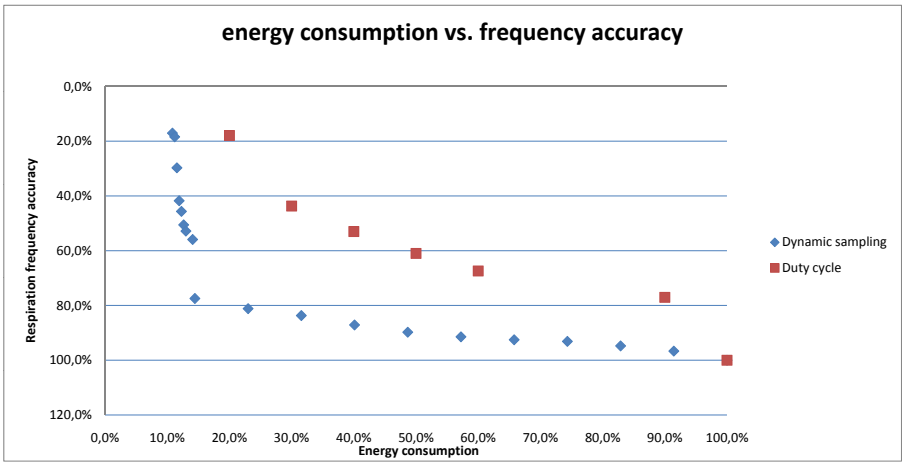


Figure 30: Pareto graphs of energy efficiency and sampling accuracy. The x-axis shows the relative energy consumption, the y-axis represents the accuracy.

5.10 Results

When we look at the pareto graph in Figure 30, energy reduction of 14.4% of the original can be achieved, with the respiration frequency being not accurate 22.5% of the time. This would increase the battery lifetime from 9.4 hours to 48.6 hours. The dynamic sampled algorithm provided a better accuracy, with comparable energy consumption, than the duty cycled algorithm. If the goal is set to reduce the energy consumption to 14.4%, the respiration frequency is not accurate 22.5% of the time. When it is required that the respiration frequency is determined correctly 90% of the time, only 42.8% energy could be saved with this dynamic sampling algorithm.

6 Conclusions and further work

In this thesis the focus was on the development of an architecture and implementation of a sensor node. On the algorithmic side the focus was on a dynamic sampling scheduling algorithm based on accelerometer data. During the thesis a hardware design was developed and lab tested. A software framework for the DSP was implemented and a dynamic scheduling algorithm to sub-sample the respiration band was developed and evaluated.

A global architectural design is proposed which could be used for after the iCare4COPD project. The architecture consists of the sensor node, a gateway and a server.

The sensor node design consisted of a hardware design for the generic sensor board and two specific sensor boards, namely for the respiration band and an SpO_2 sensor. The circuit for the respiration band and the measurement circuit of the SpO_2 sensor has been successfully tested in the lab. Due to a small mistake, the DSP of the generic sensor board could not be programmed without hardware modifications. The RTC and the Flash on the generic sensor board were not fully qualified, this because no the software was available to test it with.

Analysis of the motion artifact of the respiration band in the given data turned out to be more accurate than expected. The accuracy was approximately 90 %.

Different postures could be distinguished by the classifier, sitting and laying could be accurately recognized with an 97% and 98% accuracy. The classifier mismatched some movement classes with each other. The introduced activity level gave an accurate representation of physical exertion.

With the developed sub sampling algorithm, energy consumption could be decreased to 15% of the original, with a frequency inaccuracy of approximately 22.5% of the time, when a deviation of 0.02 Hz was allowed. The dynamic sampling algorithm performed better than the duty cycled sub-sampling algorithm, on the energy-accuracy tradeoff.

6.1 Contribution

The contributions of this work are:

- A global architecture was presented, which is intended to be used for the iCare4COPD project.
- A sensor node has been developed to provide first feedback about the patient monitoring system.
- The accuracy of the respiration band was evaluated with test data.
- A dynamic sampling algorithm was developed and evaluated.

6.2 Further work

When SpO_2 is measured continuously it might be interesting to develop a sub-sampling method for the SpO_2 sensor. From literature we know that the SpO_2 sensor has different properties regarding movement artifacts. The sensor produces more error when moving vertically than horizontally, this can be taken into account while developing a sub-sampling algorithm.

More types of classes might be interesting to be measured e.g. cycling. The data measurements were also too short, i.e. sometimes the breathing frequency was not yet recovered after an activity. Also a daily routine would be interesting to see. The current data set was based on healthy persons. Measurements with real COPD patients would be desired to see the impact of the disease on the sensors measurements and on the accuracy of the dynamic sampling algorithm. With the daily routine the impact of the dynamic sampling algorithm can be determined more accurately. Measuring the daily routine of COPD patient would be a great opportunity to gain more insights in the daily activity of the patients. This could provide feedback on how to increase the accuracy of the algorithms.

However the energy is saved due to selective sampling of the respiration band, the classifier requires the combination of two sensors to determine the activity level. An adjustment of the feature extraction, where the communication between the sensor nodes should be kept to a minimum, should be explored. With the current implementation of the classifier, the orientation of the accelerometer on the leg has been taken. It would be sufficient to look at one axis of the accelerometer.

The current version of the dynamic sampling algorithm contains only one threshold, with only two distinct sampling duty cycles. A version with multiple thresholds and sampling frequencies might result in a higher accuracy. The dynamic sampling algorithm should also be implemented on the sensor node, this to see the actual performance increase.

The proposed dynamic sampling algorithm takes a duty cycle over a large period (1 minute). To decrease the power consumption, the impact of a duty cycled sub algorithm with a small period (order of hundreds of seconds) could be explored. This sub sampling algorithm could work in combination with the proposed dynamic sampling algorithm.

References

[1] AME. Website ame, 2010. URL `http://www.ame.nu/`.

[2] Eli Billauer. peakdet: Peak detection using matlab. http://billauer.co.il/peakdet.html, 2011. URL `http://billauer.co.il/peakdet.html`.

[3] Min Chen, Sergio Gonzalez, Athanasios Vasilakos, Huasong Cao, and Victor Leung. Body area networks: A survey. *Mobile Networks and Applications*, 16:171–193, 2011. ISSN 1383-469X. URL `http://dx.doi.org/10.1007/s11036-010-0260-8`. 10.1007/s11036-010-0260-8.

[4] K P Cohen, D Panescu, J H Booske, J G Webster, and W J Tompkins. Design of an inductive plethysmograph for ventilation measurement. *Physiological Measurement*, 15(2):217, 1994. URL `http://stacks.iop.org/0967-3334/15/i=2/a=009`.

[5] DestinyQx. Cardiac cycle left ventricle, 2007. URL `http://en.wikipedia.org/wiki/File:Cardiac_Cycle_Left_Ventricle.PNG`.

[6] National heart lung and blood institute (US). What is copd?, June 2010. URL `http://www.nhlbi.nih.gov/health/dci/Diseases/Copd/Copd_WhatIs.html`.

[7] Texas Instruments. Msp430x2xx family users guide (rev. e) (slau144e). Technical report, Texas Instruments, 2008.

[8] Tzu-Ping Kao, Che-Wei Lin, and Jeen-Shing Wang. Development of a portable activity detector for daily activity recognition. In *Industrial Electronics, 2009. ISIE 2009. IEEE International Symposium on*, pages 115 –120, july 2009. doi: 10.1109/ISIE.2009.5222001.

[9] Benny Lo and Guang-Zhong Yang. Architecture for body sensor networks. In *Perspectives in Pervasive Computing*, pages 23 –28, oct. 2005.

[10] G. Trster M. Kusserow, O. Amft. Bodyant: miniature wireless sensors for naturalistic monitoring of daily acitivity. 2009.

[11] Vishal Markandey. Pulse oximeter implementation on the tms320c5515 dsp medical development kit (mdk). Technical report, TI, June 2010.

[12] Fabio Pitta, Thierry Troosters, Martijn A. Spruit, Vanessa S. Probst, Marc Decramer, and Rik Gosselink. Characteristics of physical activities in daily life in chronic obstructive pulmonary disease. *Am. J. Respir. Crit. Care Med.*, 171(9):972–977, 2005. doi: 10.1164/rccm.200407-855OC. URL `http://ajrccm.atsjournals.org/cgi/content/abstract/171/9/972`.

[13] M. Raghuram, K.V. Madhav, E.H. Krishna, and K.A. Reddy. Evaluation of wavelets for reduction of motion artifacts in photoplethysmographic signals. In *Information Sciences Signal Processing and their Applications (ISSPA), 2010 10th International Conference on*, pages 460 –463, may 2010. doi: 10.1109/ISSPA.2010.5605443.

[14] K.A. Reddy, B. George, and V.J. Kumar. Use of fourier series analysis for motion artifact reduction and data compression of photoplethysmographic signals. *Instrumentation and Measurement, IEEE Transactions on*, 58(5):1706 –1711, may 2009. ISSN 0018-9456. doi: 10.1109/TIM.2008.2009136.

[15] C. Schreiner, P. Catherwood, J. Anderson, and J. McLaughlin. Blood oxygen level measurement with a chest-based pulse oximetry prototype system. In *Computers in Cardiology, 2010*, pages 537 –540, sept. 2010.

[16] A. Sinton and R. Suntheralingam. Respiratory inductance plethysmography with an electrical impedance plethysmograph. *Medical and Biological Engineering and Computing*, 26:213–217, 1988. ISSN 0140-0118. URL `http://dx.doi.org/10.1007/BF02442267`. 10.1007/BF02442267.

[17] Wikipedia. Naive bayes classifier. http://en.wikipedia.org/wiki/Naive_Bayes_classifier, 2011. URL `http://en.wikipedia.org/wiki/Naive_Bayes_classifier`.

[18] wikipedia. Eindhoven university of technology, June 2011. URL `http://en.wikipedia.org/wiki/Eindhoven_University_of_Technology`.

[19] Y.S. Yan and Y.T. Zhang. A model-based artifact reduction method for the non-invasive estimation of blood oxygen saturation. In *Computer Architectures for Machine Perception, 2003 IEEE International Workshop on*, pages 90 –92, 0-0 2004. doi: 10.1109/ISSMD.2004.1689569.

[20] Jie Yang, Shuangquan Wang, Ningjiang Chen, Xin Chen, and Pengfei Shi. Wearable accelerometer based extendable activity recognition system. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3641 –3647, may 2010. doi: 10.1109/ROBOT.2010.5509783.

7 Appendix

A Thesis description

Technische Universiteit Eindhoven University of Technology

Electrical Engineering
Signal Processing Systems (SPS)
ACTLab research group
Fall 2010

MASTER THESIS
(Graduation project)

for

Paul Bakker

Ubiquitous remote patient monitoring system

Begin:	'01-11-2010
Targeted graduation symposium:	'
Advisor AME:	Micha Stalpers
Advisor TUE:	Dr. Oliver Amft, amft@tue.nl

Introduction

Physicians currently miss information when they have to diagnose a patient from observations made during the examination periods at the clinic. This is a particular challenge when patients do not stay at the hospital, thus being at a remote location. Ideally, the physician could review a patient's status and disease trends from data captured during the patient's everyday life, not limited to the examination.

Novel ubiquitous systems could continuously monitor and support patients during daily life when discharged from hospitals. Ubiquitous systems comprise miniaturised electronics, sensing and processing, and interaction functions that are embedded in daily life for the benefit of their user. The systems could help to monitor a patient's status (vital functions) and provide insight into daily activities that is needed to interpret the patients' vital data and disease trends. Furthermore, patients with a chronic condition that are continuously at risk for a worsening condition would benefit from technology that can regularly provide

coaching support and objective feedback to healthcare providers. By using such personal monitoring solutions health risk could be reduced and the patient's comfort be increased.

This project shall develop a ubiquitous system for remote patient monitoring by capturing data from distributed sensors and processing sensor data to extract information on physiological and activity state. The sensors will be mainly worn on-body, but could also be embedded in objects and the environment. Essential work steps of the project are (1) investigating a ubiquitous system architecture that addresses constraints of mobile distributed systems, (2) implementing a prototype monitoring system, including hardware and software, (3) evaluating the system in tests that simulate the later use in remote patient monitoring, (4) developing a small demonstrator that shows the system's operation.
The system has to cope with the dynamically changing conditions during the patients daily life, in particular: erroneous sensor readings, sensor/communication failures, processing and power constraints. In addition, the system needs to be comfortable to wear and use for the patient.
The architecture and implemented solution needs to incorporate different multi-modal sensors that are commercially available, in order to quickly adapt the system for specific use cases. Relevant sensor modalities include, Electrocardiogram (ECG), heart rate (HR), chest motion (respiration rate and depth), motion (acceleration and body/limb orientation), body sound (respiration, HR), . Distributed sensors may incorporate on-line processing functions to pre-process data and extract information (features) that can be further processed in other entities of the architecture. Processing needs to be continuously performed, as sensor data is continuously captured. Information between system nodes need to be transferred wirelessly using power-efficient communication protocols, primarily ZigBee and ANT. The architecture design shall conceptually incorporate all functional elements (sensing, processing, feedback and communication). The system topology should be elaborated in the project.

Project task description and procedure

This project targets to derive design of the ubiquitous system, implement a prototype, validate the prototype, and finally show the implementation's operation in a demonstration.

The following individual tasks are foreseen:

1. **Familiarisation with the application of remote patient monitoring, system architecture concepts and technologies, and relevant sensor technologies:** An analysis of user requirements shall be performed to identify essential functions of the system (system requirements). A review of technologies shall be performed with regard to available sensors, data processing, system topology, and expected use (pertained to patient comfort, acceptance, and compliance). A project demonstration shall be defined, which shall be further pursued during the

project and presented at project conclusion.

2. **Selecting a system architecture and implementing a prototype:** Based on the reviewed technologies, sensors, processing options, and constraints, a decision on the system architecture shall be prepared. Constraints shall be analysed for different architectural options, including scalability (sensors, processing, communication bandwidth), energy-efficiency (data storage vs. communication tradeoff), and extensibility (commercial sensors, interaction options).
   Based on the selected architecture (decided in collaboration with the advisers), the implementation is approached. A prototype solution of the system architecture is realised which shall be used with different sensors, and subsequently evaluated.
3. **Situation-aware resource usage operation:** Options shall be investigated to optimise the architecture's resource use based on its operation situations, such as weak performance of individual sensors, specific context of the user, low battery state, etc. Selected resource optimisation strategies shall be simulated in Matlab and implemented in algorithms for the prototype solution. Performance of the algorithms shall be evaluated.
4. **Validation and feedback:** The developed solution will be evaluated in tests where daily life situations (in which the system will be later used) are elucidated. The solution's performance and options for optimising the solution shall be analysed, and realised where feasible. Other refinement options shall be conceptually prepared and documented. A specific demonstration goal will be defined in the first phase of the project to practically illustrate the achieved results and show the system's operation in real-life.

Administrative aspects of the project

- **Prerequisites and preparation for the project:** A general understanding of the areas ubiquitous computing, pattern recognition, sensor modalities is required before commencing with the project. Literature for this preparation will be recommended. Further discussion with the advisor shall help to broaden the acquired knowledge.
- **Familiarisation with the task and start of the project:** Relevant literature on ubiquitous system architectures, patient monitoring, and dynamic resource optimisation shall be studied based on provided references, but also independently by extending the recommended reading list. It is strongly recommended to keep a literature database in JabRef and add comments on conclusions of individual papers.

- **Project plan:** After a startup phase, a project Gantt chart shall be prepared. This plan shall detail the work packages foreseen, including a brief listing of goals, prerequisites, milestones, and outcomes (deliverables), as well as a time effort estimations. This project schedule shall be continuously updated to match the project progress. It will be reviewed in meetings with the project advisers.
- **Adviser feedback and coaching:** Meetings with the project advisers are foreseen to discuss achieved work results, define next steps, and agree on changes to the project plan. The candidate is asked to thoroughly prepare for the meetings, in particular, to prepare a plan for the next work steps, project risks, and contingency resolution. For the master project, the meetings will be primarily scheduled on request of the candidate, at least once every two weeks.
- **Project documentation, presentations, and symposium paper:** It is recommended to keep an adequate record (e.g. in a draft report) of project results, findings, and decisions made. This information can be used to prepare symposium paper, final report, and the presentations (SPS student colloquia, graduation symposium, and graduation committee meeting). Upon project completion a final report (prepared using Latex, in English language) shall be presented as an extension of the symposium paper. The final report shall be accompanied by an electronic appendix (CD or DVD), containing all materials used, developed, or otherwise acquired during the project. This will at least include: software source codes, binaries, report and presentation source codes. A brief documentation shall be added to navigate in the electronic appendix. An extended abstract (detailing goals and achievements of the project) shall be prepared as a webpage (html format, length: 0.5 pages text, 1-2 graphs of the project, one picture of the candidate) to be released on the ACTLab group website after the project completion.
- **Confidentiality of the project and used materials:** To not inhibit further research and deployment of results, the candidate is asked to not disseminate detailed project reports, presentations, task descriptions, and other intermediate or final result or data artefacts. Exceptions must be individually granted by the project advisers.
- **Workplace:** The work will be carried out at Applied Micro Electronics "AME" BV, Esp 100, 5633 AA EINDHOVEN.
- **Project evaluation:** Presentations and symposium paper are evaluated by the graduation committee. In addition, daily work during the project will be considered in the adviser's evaluation. The following evaluation categories will be considered to determine the candidate's performance: (1) scientific approach and systematic working style, (2) work effort and independence, (3) results (quality vs. difficulty level), (4) scientific and formal quality of the symposium paper and final report, (5) presentation performance.

Recommended reading list

- Christopher M. Bishop, Pattern Recognition and Machine Learning, selected chapters.
- Lukowicz, P.; Amft, O.; Roggen, D. & Cheng, J.: On-Body Sensing: From Gesture-Based Input to Activity-Driven Interaction. Computer, 2010, 43(10), pp. 92-96.
- Martin Kusserow, Oliver Amft, and Gerhard Tröster. BodyANT: Miniature wireless sensors for naturalistic monitoring of daily activity. In Bodynets 2009: Proceedings of the 4th International Conference on Body Area Networks, ACM press, 2009.
- Harms, H.; Amft, O.; Winkler, R.; Schumm, J.; Kusserow, M. & Troester, G.: ETHOS: Miniature Orientation Sensor for Wearable Human Motion Analysis. Sensors 2010: Proceedings of IEEE Sensors conference, IEEE, 2010.
- Sensor technologies, e.g. using Wikipedia: accelerometers, gyroscopes, magnetic field, Electromyography (EMG), Force sensitive resistors (FSR), and other potential modalities.
- Linear classifiers and statistical classification : http://en.wikipedia.org/wiki/Statistical_classification

Supplemental courses

- TBD

B Data bandwidth estimation

Assumptions about sampling frequency have been made, in order to make estimations about the amount of data that the system will transmit. The assumed sampling frequencies can be seen in Table 7.

Table 7: Sampling frequency assumptions

Sensor	**Sampling frequency** (Hz)
Accelerometer	25
Respiration sensor	10
SpO_2 sensor	1/60th

It is assumed that both respiration and SpO_2 sensors produce each 2 bytes of data per sample and the accelerometer would produce 2 bytes per sample per axis. Additional information, described in Appendix G, also needed to be taken into account. The packet header is 7 byte large. Each packet contains sensor data of only one sensor and an additional byte per sample is required for the relative time. The estimated amount of data, produced by the sensors is shown in Table 8. When all sensors are connected to a single sensor node, the sensor node would produce 696 kB per hour.

Table 8: Data amount

Sensor	**per sec**	**per min**	**per hour**	**per day**
Accelerometer	150	9,000	540,000	12,960,000
Respiration	20	1,200	72,000	1,728,000
SpO_2	0	2	120	2,880
Auxiliary data	23	1,401	84,060	2,017,440
total	193	11,603	696,180	16,708,320

C Communication policies

Possible communication interfaces between MSP430 and EM357 are SPI and I^2C. The chosen interface is SPI. This is chosen because SPI has a higher maximum throughput. The maximum throughput of I^2C is 400 kbps. The maximum throughput for SPI is 3 Mbps, this is bounded to the hardware of the DSP.

C.1 Communication policy MSP430 - EM357

To ensure a reliable communication between the DSP and the transceiver, a transmission policy has been created. With the predecessor of the EM357, the EM250, it was required that the transceiver was master on the bus. After the creation of the communication policies, AME discussed the use of the EM357.

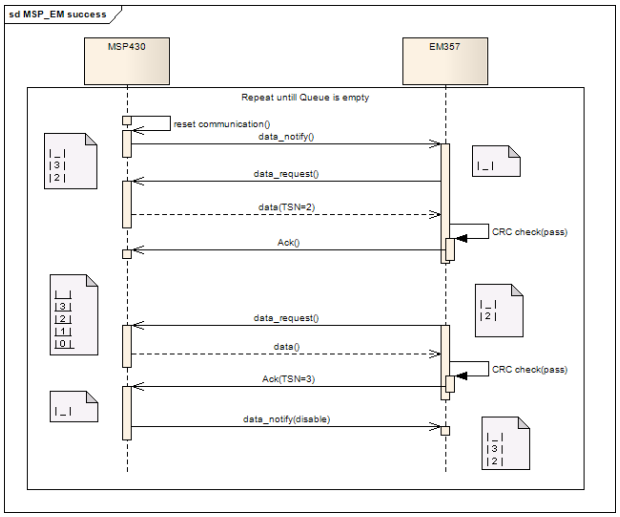


Figure 31: Communication policy DSP and EM357

The normal use of the communication policy is outlined in Figure 31. First the DSP resets its communication state, this ensures that any old state is erased. After the reset, the DSP will set the pin, that is connected to the transceiver high. This pin indicates that there is data for the transceiver to retrieve. Because the transceiver is master on the SPI bus it must request the data from the DSP. The DSP will return the data to the transceiver. The package contains a CRC. The transceiver calculates the CRC of the data and compares it with the CRC of the package. When it is the same the transceiver sends an Ack to the DSP. After receiving the Ack, the DSP removes the sent packet from memory.

As long as the pin is pulled high, the transceiver can continue requesting data from the DSP. When the queue is empty the DSP will disable the wake up. With this protocol the

transceiver is in charge of the communication and the DSP is in charge of when and which data is communicated to the transceiver.

When the transceiver is too slow in retrieving the data from the DSP, it will be noticeable at the USB-stick via the time stamp in each packet. Which can be seen in the packet structure, which is discussed in Appendix G. This protocol is not influenced by the wake up overhead of the transceiver, assuming that the buffer of the DSP is large enough and that the communication is fast enough to empty the buffers.

C.2 Recovery policy

When communication fails the following sequence is in order as shown in Figure 32. When the data is sent from the DSP to the transceiver and the CRC check fails the transceiver will not do anything. A timeout at the DSP will be generated. The DSP will pull the pin low and communication will start as normal.

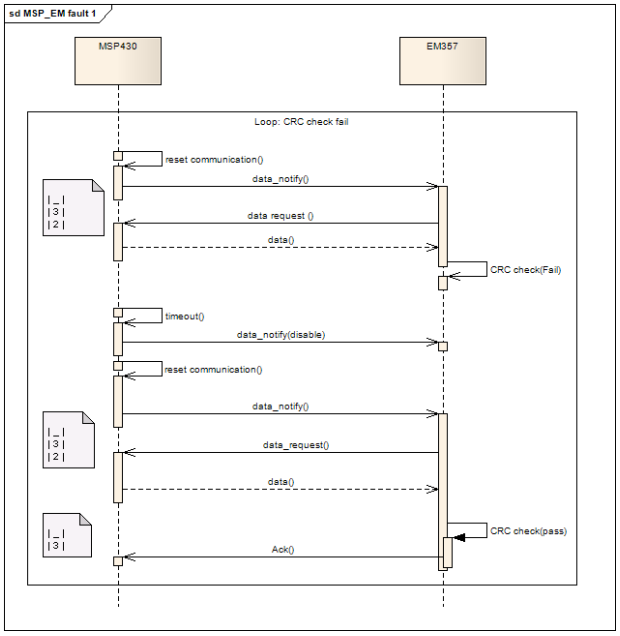


Figure 32: Recovery of communication policy DSP and EM357

C.2.1 Implementation

When implementing this protocol, it was decided to slightly change this protocol. The DSP will now pull down the pin after the packet of data was transmitted to the transceiver. This is chosen because this makes the implementation of the communication protocol at the transceiver easier. The transceiver now asks for data until the pin is pulled down and

can analyze the data later. With the outlined communication policy it was unknown when the package would end. Looking in the packet to see how large it was, (see Appendix G) is not an option since it must be assumed that data can become corrupted while sending. This changes the functionality of the pin from data ready to package not yet completely send.

C.3 Communication policy EM357 and Gateway

The sensor node communicates with the gateway via the ZigBee protocol. Each ZigBee packet contains sensor data of only one sensor. The sensor data is divided into packets with a maximum size of the size of a ZigBee packet.

On the ZigBee stack level the it is checked if data correctly received from the sender to receiver. For the wireless communication this was used to check if data is transmitted correctly or not. When data is transmitted correctly a conformation will be received by the initiator of the communication. To ensure data integrity and to meet throughput requirements an efficient, concurrent, transmission policy is required.

From measurement taken at AME, (Appendix H) it is shown that sending data strictly sequential was not sufficient to reach the required performance, a concurrent model is required. There are two basic concurrent transmission models examined, the concurrent push and concurrent pull model.

C.3.1 Concurrent push model

The concurrent push model is a data driven model where the sensor node initiates the datastream. The sequence diagram of the concurrent push model is shown in the Figure 33. As Figure 33 depicts, the transceiver transmits the next packet, before it received the acknowledgement from the previous message. The amount of pending messages is limited to a predefined "window size".

When the gateway does not receive a packet with the succeeding Transaction Sequence Number (TSN), as shown with TSN=3 (TSN=2 is not yet received by the gateway). The gateway will keep the data of packet with TSN =3 in memory. When the gateway does not receive the package with TSN =2, the transceiver will eventually generate a time out of TSN=2 and will resent the data packet with the successor TSN of its last received TSN-Ack (TSN=1). The transceiver keeps the data in local memory until it received the Ack.

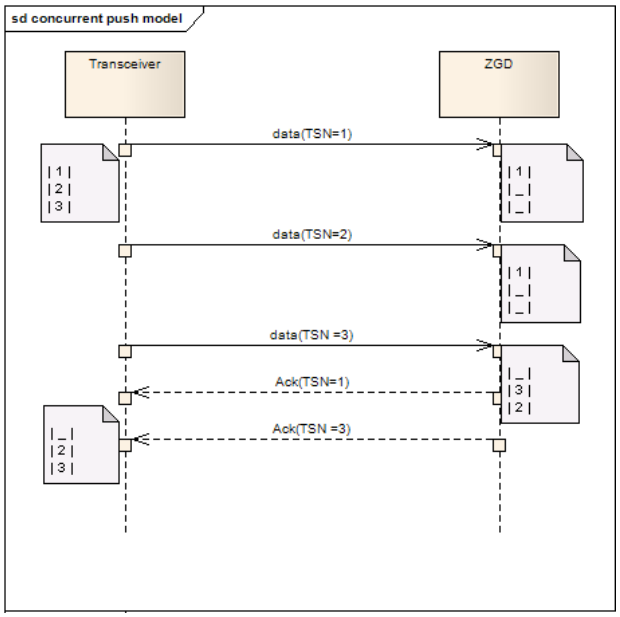


Figure 33: Sequence diagram of the concurrent push model

C.3.2 Concurrent pull model

With the concurrent pull model, the gateway requests data from the transceiver, as Figure 34 depicts. The gateway keeps track of the TSN numbers. The transceiver responds to the messages by sending the data which corresponds to the requested TSN. When the gateway does not receive the data, the gateway will request the data again.

When the transceiver does not have the data ready, it will ignore the gateway's request, which will cause a timeout at the gateway. When the gateway receives data of a different TSN as requested, the gateway will ignore that data and request the data of the succeeding TSN number of its last received datapacket.

With this protocol the gateway is the initiator of the datastream. This protocol ensures that there will be no congestion at the gateway. However this protocol does requires that the gateway continuously transmit requests, which can cause congestion at the node if the request interval is to fast.

Conclusion

The concurrent push model is the desired transmission model because this model is a data driven model. The gateway also does not send out unnecessary messages to check if there is data available or whether the node is in the vicinity of the gateway.

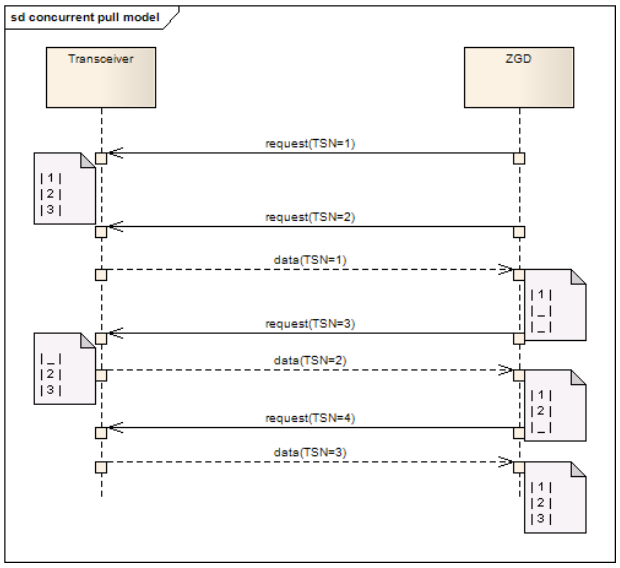


Figure 34: Sequence diagram of the concurrent pull model

Concurrent push model, recovering policy

Figure 35 depicts a sequence diagram showing the concurrent push model recovering policy. Whenever the transceiver receives a timeout, it will resend the datapacket with the succeeding TSN of its last subsequent received acknowledgement.

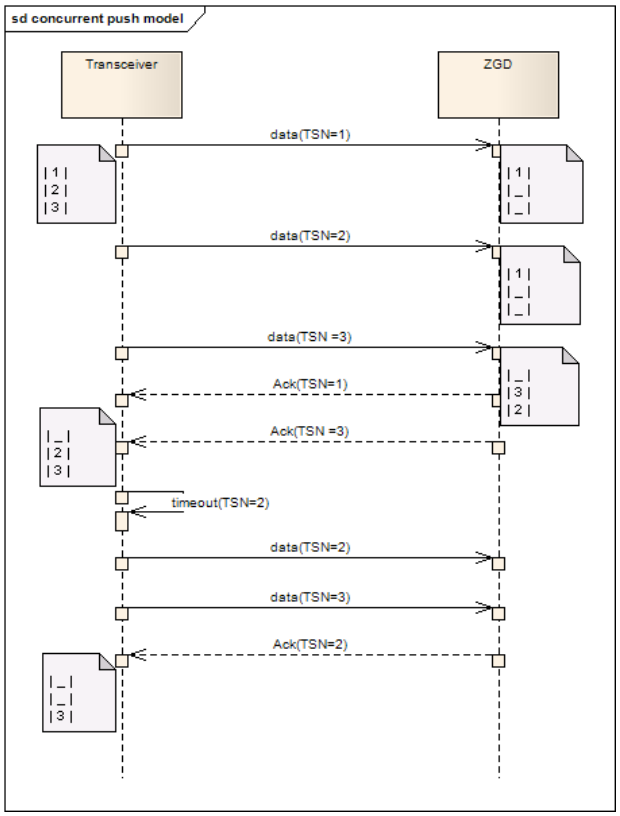


Figure 35: Sequence diagram of the concurrent push model recovery

Even though the Ack of TSN =3 is received by the transceiver, retransmission begins at TSN =2. Because TSN=2 causes the timeout it re-starts the transmission from TSN=2. The gateway already received the packets and should ignore its content, but will give an confirmation about receiving these datapackets. Such that the transceiver can continue its progress.

D Energy estimations

In this section describes the theoretical power consumption. First the energy consumption of the individual components will be given, followed by the theoretical power consumption calculations. Unfortunately the power consumption could not be verified because the source code of the transceiver was not containing sleep procedure. In the remainder of this section the power consumption will be given in the (average) current drawn by the components. To calculate the power consumption the (average) current has to be integrated over time.

D.1 DSP characteristics

Table 9 gives the characteristics of the DSP. The DSP will run at 1 MHz. The SPI active current was unknown, however for calculation it was assumed that SPI module of the DSP would consume as much as the SPI module of the transceiver. The datasheet does not say anything about a supply voltage of 2.7 Volt but only specifies at 3.0V and 2.2V, for further calculation the energy consumption of 3.0V is considered.

Table 9: DSP characteristics

Parameter	**Current**	**Unit**
MSP430 stand-by current	0.6	μA
CPU active current (at 1 MHz, 2.2V)	270	μA
CPU active current (at 1 MHz, 3.0V)	350	μA
CPU active current (at 12 MHz, 3.0V)	4.3	mA
CPU active current (at 16 MHz, 3.0V)	6.5	mA
SPI Active current	0.2	mA
ADC converter active current (supply + ref)	1.2	mA
Settling time	100	ns
Conversion time	3.51	μs
Sampling time	1220	ns
Total sampling time (settling time+ conversion time + Sampling time)	4.83	ms

D.2 EM357 characteristics

Table 10 shows measured and datasheet data of the ZigBee transceiver. The experimental results of the data throughput are described in Appendix H.

Table 10: EM357 characteristics

Parameter	Current	Unit
Measured throughput	7000	bytes/sec
Transmission current (datasheet)	31	mA
Serial controller current	0.2	mA
Average Idle Current	1.7	μA
CPU active current (at 12 MHz)	7.0	mA
CPU active current (at 24 MHz)	8.0	mA
Wake up time from deep sleep	100	μs

D.3 Memory characteristics

With the prototype, all raw sensor data must be sent to the gateway. This resulted in a large amount of data. It was expected that all this data needed to be stored locally on the sensor node. For this we needed to take flash memory into account. The characteristics of the used flash memory are shown in Table 11. Values were taken from the datasheet of the AT46DB641.

Table 11: External flash characteristics

Parameter	Current	Unit
Standby current	50	μA
Read current	9	mA
Page program current / erase current	18	mA
Startup time (40 cycles at 1MHz)	50	μs
Block erase time	400	ms
Page program time	1	ms
Read time (assumed half write time)	3.5	μs
Page size	256	Bytes
Blocksize	65536	Bytes

To be able to make an estimation of the energy consumption to store and retrieve data from memory we take this datasheet values. This does result in a pessimistic energy calculation.

To write one byte, first a whole block has to be erased, because this block is 65536 bytes large, we divided the time of erasing one block by the number of bytes and we needed to add the startup time.

$$Erasetime = (blockerasetime + startuptime) = 400ms + 50\mu s \approx 400.05ms \quad (9)$$

This is per 65536 bytes, therefore this value could be divided by 65536 to get the erase time per byte.

$$Erasetimeperbyte = \frac{Erasetime}{numberofbytes} = \frac{400.5}{65536} \approx 6\mu s \tag{10}$$

To write the one byte the following formula can be used to calculate the page program time.

$$Writetime = pageprogramtime + startuptime = 1ms + 50\mu s \approx 1.5ms \tag{11}$$

This is per page, (256 bytes) therefore the Write time can be divided by 256 to get the average write time per byte.

$$Writetimeperbyte = \frac{Writetime}{numberofbytes} = \frac{1.5}{256} \approx 5.8\mu s \tag{12}$$

The average write time is the combination between erase and write time and is 11.8 μs.

The total read time was calculated in a similar fashion. Here one byte at a time is read, which makes the calculation pessimistic.

$$Readtime = (Readtime + startuptime) = 3.5\mu s + 50\mu s \approx 53.5\mu s \tag{13}$$

D.4 Miscellaneous power consumption

It was assumed that the sensors had a constant energy consumption and that the only states possible were On and Off. In reality a sleeping state could also be considered. The energy consumption of the accelerometer is 40 μA. The power consumption of the oscillator, is measured to be 30 mA. The majority of the power for the SpO_2 circuit goes to the LED and the LEDs light intensity is dependent on the patients finger. Therefore an estimation of 44 mA was made.

The voltage regulator also consumes a small amount of power, which is 75 μA. The RTC consumes 450 nA. The battery discharge current was assumed to be 2 μA.

D.5 Duty cycle

From Appendix B it is known that the amount of generated data is 696.180 bytes per hour. The ZigBee transmitter transmits 7000 bytes per second, thus 99 seconds would be required to transmit 1 hour of Data. The transmission duty cycle is calculated in Equation 15.

$$Dutycycle = \frac{activetime}{totaltime} = \frac{99}{3600} \approx 0.02 \tag{14}$$

The time required to communicate the data over SPI at 100 kbps would be, $\frac{696.180}{100.000/8} = 55.6944$ sec. The SPI duty cycle is:

$$Dutycycle = \frac{activetime}{totaltime} = \frac{55.7}{3600} \approx 0.015 \tag{15}$$

It was assumed that the data would only be stored and retrieved only once. The time required to store the data is: $696.180 \cdot 11.8\ \mu s = 8.2$ s. This results in a duty cycle of approximately 0.002. Table 12 summarizes the duty cycle per component.

Table 12: Duty cycle per component and functionality

Parameter	**duty cycle (%)**
ZigBee communication	4
SPI communication	0.4
Mem write	0.1
Mem read	1.0
EM357 wakeup	0.05
Read Battery	<0.01
Maintenance	<0.01

D.6 Searching gateway

When the sensor node is out of reach from the gateway, the sensor node will try and find its gateway. This requires the radio of the transceiver to be active. We assumed that every search requires the radio to be active for 5 seconds. When the search interval would be set to 30 minutes this would result in a duty cycle of 5/1800 = 0.2%. On average the contribution of power consumption of searching for the gateway would be 86 μA .

D.7 Power consumption

For the calculation of the power consumption we took the duty cycle of each component and multiply that with corresponding current, as Equation 16 shows. In Equation 16, the duty cycle is denoted as D, active power consumption as Power_{Active} and idle power consumption as $\text{Power}_{inactive}$.

$$Energy = D \cdot Power_{Active} + (1 - D) \cdot Power_{Inactive} \tag{16}$$

In Appendix D.5 the duty cycles, the active and the inactive power consumption are given. Table 13 gives the energy consumption of each component.

Table 13: Energy consumption

Parameter	**Current**	**Unit**
Accelerometer	90	μA
Real time Clock	450	nA
Voltage regulator	75	μA
Battery leakage	2	μA
MSP SPI (to sensor)	2.13	μA
MSP SPI (to EM357)	2.13	μA
EM357 wake up overhead	4	μA
EM357 SPI comm MSP	31.7	μA
EM357 SPI transmit flash	31.7	μA
EM357 SPI receive flash	31.7	μA
EM357 transmission ZigBee	1.23	mA
EM357 searching gateway	86	μA
Flash (1 read+ 1 write)	0.2	mA
MSP idle	595	nA
EM357 idle	1	μA
Battery read	0.00483	nA
EM357 maintenance	179	nA
total sum	1.85	mA

Figure 36 shows the distribution of the contributing parts of the energy consumption as shown in Table 13.

D.8 Total battery lifetime

From the calculations in Appendix D.7 it is shown that the theoretical average power consumption could be 1.85 mA. The chosen battery has a capacity of 300mAh battery. The

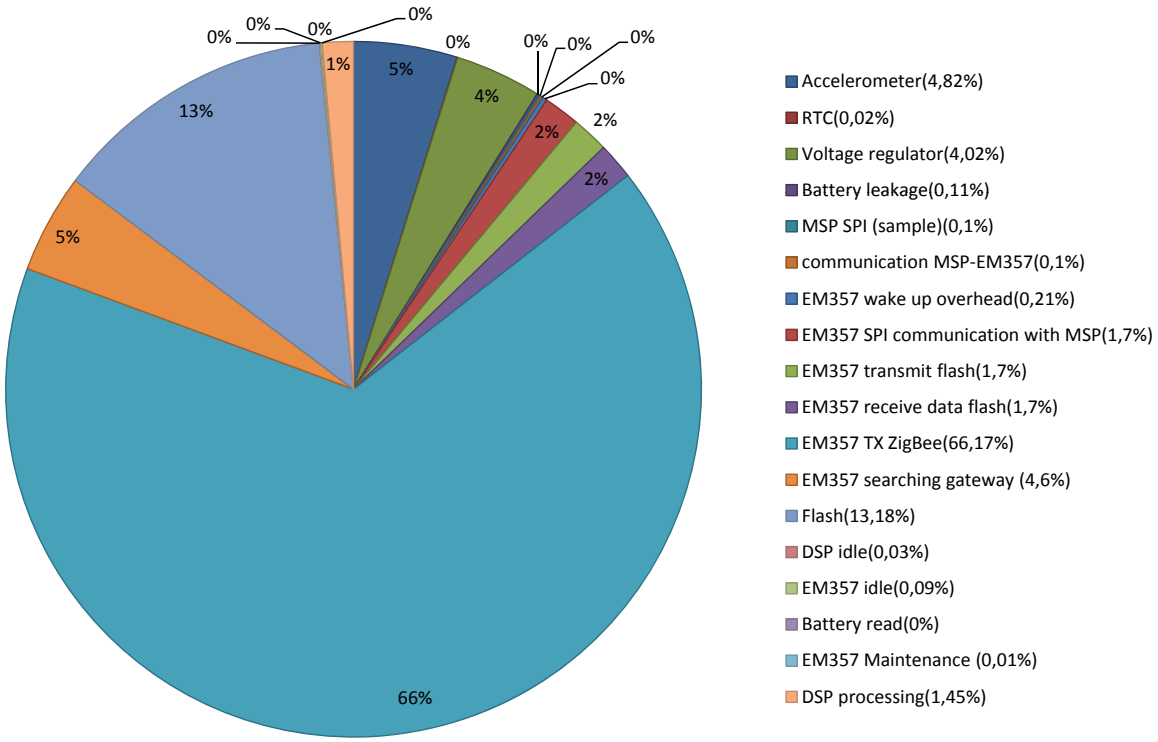


Figure 36: Distribution of power consumption.

battery lifetime is calculated in Equation 17. As can be seen in Equation 17 the estimated battery lifetime is 161 hours, which means the battery could last for approximately 1 week. This is under assumption that the battery capacity of 300 mAh can be used.

$$Batterylifetime = \frac{batterycapacity}{Powerconsumption} = 300/1.85 = 161hours \tag{17}$$

E Hardware

This section discusses the hardware, which were not discussed in the main part of the report.

E.1 Accelerometer

The chosen accelerometer is the ADXL346. Within the pre-study different accelerometers were considered. This particular accelerometer was chosen due to its low power characteristics and for its accuracy. This accelerometer does require that the supply voltage does not exceed the 2.75 Volt.

E.2 Voltage regulator

Because the battery voltage is 3.6 Volt, a voltage regulator should be used to archive the 2.7 Volt supply required by the accelerometer. The use of the voltage regulator also lowers the power consumption of other components, like the DSP and transceiver. However the lower voltage limits the computational power of the DSP to 12 MIPS.

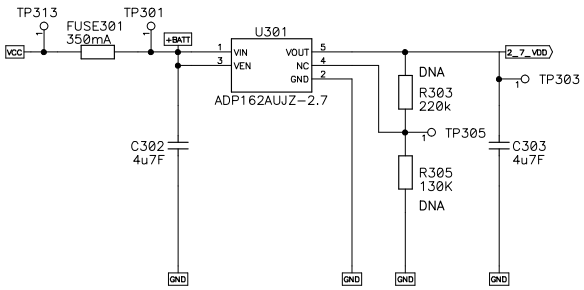


Figure 37: voltage regulator circuit

The supply nets of the EM357 and the DSP are both regulated using the same ADP162AUJZ-2.7 voltage regulator. This is an ultra-low power, low drop-out, voltage regulator, which creates a 2.7V supply voltage.

Figure 37 shows the circuit design of the voltage regulator. Resistors R303, R305 are added to the design so that there is a possibility to use adjustable regulators instead of the fixed voltage regulator. The output voltage of an adjustable voltage regulator such as the ADP161 or 163 is defined as equation 18 shows. By choosing the resistors, as shown in Figure 6, the output voltage would become approximately 2,7 Volt.

$$V_{Out} = 1 \cdot (1 + \frac{R303}{R305}) = 1 + \frac{220k}{130k} \approx 2.69V \tag{18}$$

E.3 Battery charge circuit

The MCP73832 is the main component of the battery charge circuit, which is a Li-Ion battery charger. The charge circuit is shown in Figure 38. To charge the battery, the sensor node must be connected to a PC with the mini-USB connector (J301). When the sensor node is connected to USB, the voltage of the USB will also power the circuit.

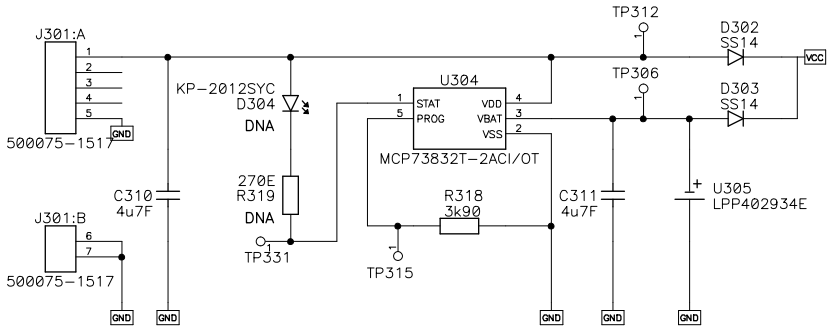


Figure 38: Battery charge circuit

E.4 Battery voltage measurement circuit

Since the voltage supply of the DSP and EM357 are regulated, it is not possible for the DSP or for the EM357 to directly measure the battery voltage from its own supply voltage. Therefore an additional measurement circuit is required to allow the battery to be measured. It is required to measure the battery voltage to prevent deep discharge of the LIPO battery. LIPO batteries do not cope well with deep discharge. Figure 39 shows this battery measurement circuit.

Since a minimization of power consumption is desired, the battery measurement circuit can, and should, be turned off when the battery is not measured. The circuit can be turned off by triggering the BATT_ADC_ENABLE signal. When BATT_ADC_ENABLE is high, current will go through resistor R305 and R308. The voltage can then be measured at BATT_ADC. When both resistors have a tolerance of 1%, the voltage accuracy is about 2%.

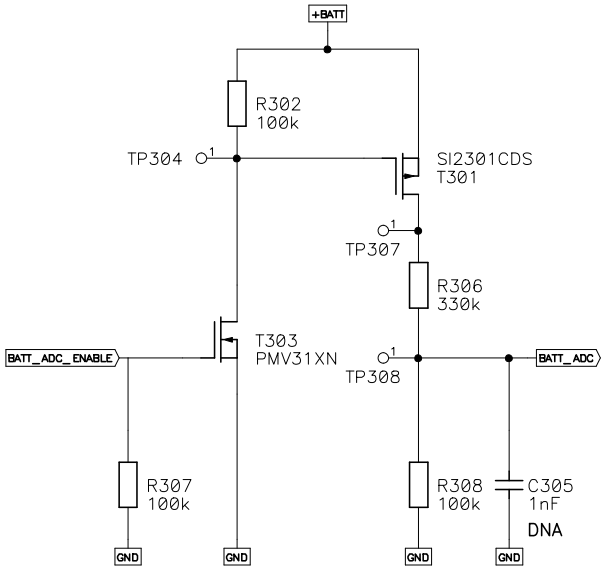


Figure 39: Battery measurement circuit

E.5 Buttons

There are two buttons on the sensor board. Both buttons are connected to the EM357. One button will act as a wake-up button. The other button will act as a reset button. The reset button is connected to a one-shot reset circuit.

A signal will pass a bandpass filter when the button is pressed. A transistor will pull the reset line of the EM357 low for a short period, causing the EM357 to reset. After the EM357 is reset, the EM357 is capable of determining whether it was a manual reset by checking the button state. The EM357 can take the appropriate action when a manual reset has occurred.

F Implementation difficulties

During the implementation of the project some implementation difficulties came up. In this section some of these difficulties are addressed.

F.1 Accelerometer

During changes made in implementation, the accelerometer stopped working. The accelerometer did respond on certain commands but did not store the given settings. Therefore the accelerometer would also not go into measurement mode. Because this happened during implementation changes it required some time to identify the source of the problem.

After identifying that the accelerometer was the cause of the problem, a new accelerometer was taken to test the code. Interchanging the accelerometer resolved this issue. The exact cause of breaking the accelerometer is unknown.

F.2 Flash

The flash was the component, which caused the most delay. Yet the functionality of the flash is not implemented. AME has working code of the a different flash-chip, but from the same supplier. This code is used in other products of AME. This code uses a header file among the whole project, ïncludes.h: After examining the code, three files should be responsible to use the flash. These three files were copied into a different project to see if the code would work in an isolated environment. The hardware configurations were set the same way as the AME-project. This approach did not bring the desired result.

After the above described method failed, the functions were copied one by one. Each time the compiler would give an error, whenever a function or configuration was missing. These functions and configurations were then copied from the original project into this project. This approach did not work either.

Finally the external flash was tried to being implemented from scratch. After not being able to communicate to the device, with knowing that the settings were as specified in the datasheet, any effort of making the flash work is halted. Unfortunately until now the flash is not implemented.

F.3 Join issue EMBER

The first chain (sensor-→ MSP-→ Ember-→ gateway) was set up using an EM250, with EmberZNet 3.5.0 stack. Requesting to join a network was done by the following function call: emberScanForJoinableNetwork. When this worked AME discussed that it would be better to use the EM357. However the EmberZNet 3.5.0 stack was not available for the EM357. Therefore the EmberZNet 4.3.0 was taken. The join algorithm, using the emberScanForJoinableNetwork did not work on this stack.

When emberScanForJoinableNetwork function was called with a PanID = 0 the application should search for the first join-able network, which complies with security settings and would request to join this network. When emberScanForJoinableNetwork is called with extendedpanid = 0, a variable, ignoreExtendedPanId, was set. Later a callback is given to emberJoinableNetworkFoundHandler, in this callback the information about the found network was given. In this handler the ignoreExtendedPanID was ignored. In this handler the actual joining takes place. Within this function a guard was preventing the joining action because the PanId's were not the same. The guard checks if the found network PanID was the same as the requested PanID. If this was not the case, than the rest of the function, including joining, is skipped.

By placing a guard in front of this call, which checks if the requested PanID contains all zero's we avoided that the joining was skipped. When this would not be the case, the check if PanId's are the same could take place. This guard fixed the joining problem.

This problem was also issued to EMBER and EMBER confirmed that this was a bug in EmberZNet 4.3.0.

F.4 Gateway memory full

The gateway was an existing prototype of AME, which received messages from the ZigBee-chip and forwards it via a SOAP interface. Within this implementation, messages are stored locally on the gateway itself. When the gateway runs out of memory it would terminate the application without any notification that it ran out of storage space.

An intermediate fix for this was done by removing the log files from memory before starting the application. Later this issue is fixed by not creating any log files at all.

F.5 Gateway wrongly implemented

After implementation of concurrent sending of messages via ZigBee, as described in Section C it was noticed that packages were not stored by the gateway. First the problem was

thought to be in the storage mechanism to flash. However, when storage were disabled, packages were still missing.

After examination of the code, it was concluded that the buffer, which contained the packets were retrieved pull based. While retrieving the packets from this buffer, a sleep function was called. Within this function the processor always slept for 100 μs. This function is changed in the following code.

```
while(lvCallback == NULL)
{
    lvCallback = lvEZSPWrapper->ezspFetchEvent();
    if (lvCallback== NULL)
    {
        usleep(10);
    }
}
```

Listing 3: Code on gateway

However this does not completely fixes the speed issue, it does help. At this moment the ZigBee transmitter still cannot sent at full transmission speed, however it can transmit faster than before the code change. To enable the gateway to be data-driven the architecture of the gateway must be altered, which is out of scope of this project.

F.6 SpO2

For the SpO_2 sensor only one part of the circuit has been tested, which is the measurement circuit. A 50 Hz noise was dominating in the taken measurements. The problem was resolved by placing a single inductor. This inductor was designed in the electronics for the SpO_2 sensor was forgotten in the measurement.

G Packet structure

This section describes the packet structure. The packets are the element of communication in Appendix C. There are two points of communication, namely between DSP and transceiver and between transceiver to gateway. The packets, which are transmitted from DSP to the transceiver is shown described in Appendix G.1. The packets designed for the ZigBee communication is shown in Appendix G.2.

G.1 SPI packet structure

Figure 40 depicts the packet structure, which is designed to transfer the data over SPI. The first byte indicates the size of the packet. This is can be used by the transceiver to determine where the CRC is located in the packet. After the packetsize, the relative creation time is transmitted, this is two bytes large. The creation time is followed by the sensor origin, data type and the number of data. The sensor origin indicates where the data is from. The data type indicates what size the data is. The number of data is used to indicated the number of data samples follows.

After the header the actual data follows. The actual data has the relative time of the sample, compared to the creation time of the packet and the actual data. The packet is closed with the CRC. The CRC is 2 bytes long. The CRC is needed to verify if the communication is completed successfully.

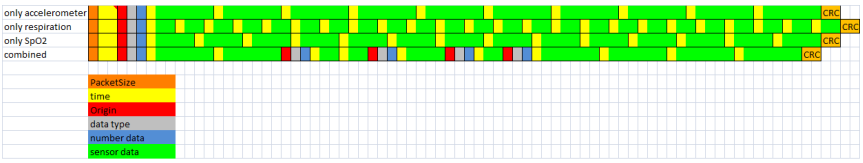


Figure 40: SPI packet structure

G.2 ZigBee payload structure

Figure 41 depicts the ZigBee payload structure, which is designed to transfer the data over ZigBee. The packet is inspired by Zigbee Cluster Library (ZCL), however it deviates a little from it, this because ZCL is setup in a generic fashion, which brings extra overhead with it.

The ZigBee packet has an ZCL header, this is required such that the gateway knows whats in the packet. After the ZCL header the actual time of the packet creation is shown. This actual time is indicated in four bytes. The time is based on the Unix time. Change in the

first byte happens only once in the 6 months. Therefore the first byte of the Unix time can be eliminated, since that logic can be shifted to the gateway. The sacrificed byte can be used to increase the resolution of the actual time to hundreds of second.

After the time, the same structure as the SPI communication is applied in the packet, but without the CRC. With the wireless communication CRC is not needed, because the EMBER stack checks if the packet is successfully transferred or not.

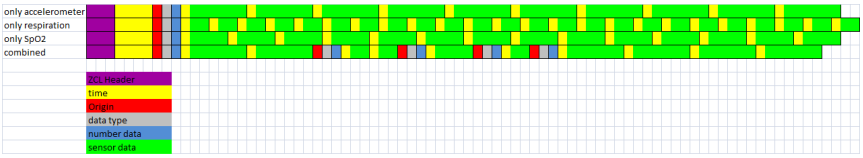


Figure 41: ZigBee payload packet structure

H ZigBee data throughput

Within the thesis project measurements were taken to determine the transmission speed of the ZigBee communication. This chapter describes this test. The tests were required to determine the energy consumption and to see if it would be a risk to use ZigBee to transmit the sensor data with. Initial tests conducted by AME indicated that the throughput of ZigBee communication was 287 Bytes per second. This test was conducted using strict sequential communication.

H.1 Experimental setup

The experiment was conducted using the EM357 development kit and the gateway. The setup is shown in Figure 42. The ZigBee transmitter is identified in Figure 42 with a red box. The green box indicate the serial communication connection from the EM357 to the PC. Finally the yellow box in Figure 42 indicate the gateway.

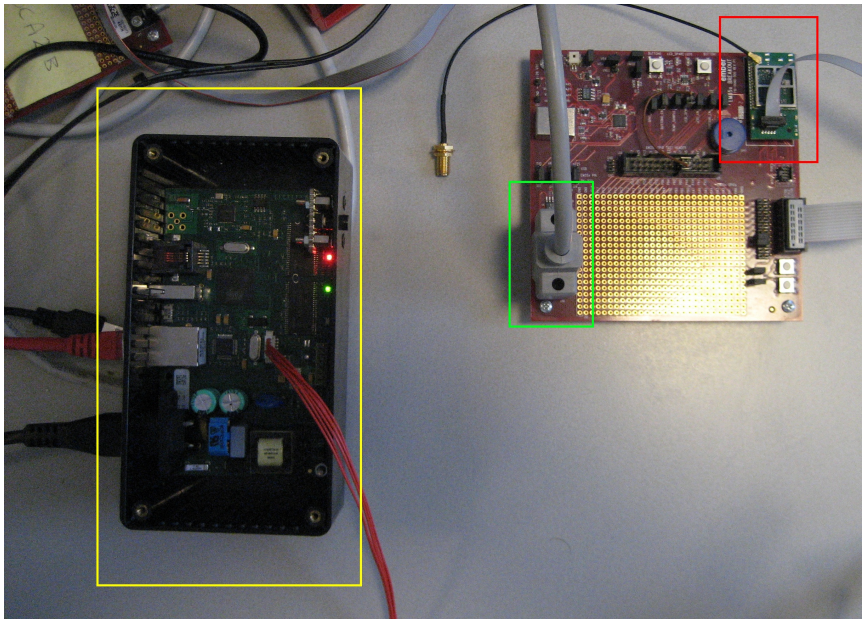


Figure 42: ZigBee data throughput test setup

For the measurements the EM357 would notify the PC whenever 256 packets were sent. This notification was done via the serial interface. The packets size of the ZigBee packets were 83 bytes, which is maximal for ZigBee packages. After the second notification was transmitted to the PC a stopwatch is enabled. After 50 notifications the stopwatch was stopped. It is chosen to use 50 packages to limit the influence of human error for starting and stopping the stopwatch.

Within this experiment the queuesize of the sliding window was varied. The amount of ticks required to transmit data was also varied. After an x amount of ticks the EM357 transmits data.

H.2 Measurements

The measurement results are shown in Table 14. The windowsize of these results was set to 5 packages and the buffersize was set on 20 packages. From this initial measurements the maximum throughput is 6 kB/s. It was observed that the amount of retransmits of the test with the throughput of 5kB/s was more than the amount of retransmits of the 6.2kB/s.

Table 14: Measurement results windowsize = 5 and buffersize =20

wait time (ticks)	**time (s)**	**Throughput (Bytes/s)**
75	169,1	6282,673
75	176,9	6005,653
75	208,0	5107,692
100	190,7	5571,054
100	177,9	5971,894
100	181,1	5866,372

The windowsize was increased to 10 packages and the buffersize to 40 packages. The results of this measurement is shown in Table 15. It was noticed that while running the measurement there were many retransmission actions during the test with the Tick set on 75 and 125 ticks. There was no environmental reason detected why these retransmits were required. Because the windowsize was set on 10 packages all the 10 packages are resent, which caused a large overhead.

Finally the processor of the gateway was disabled, but the ZigBee chip still enabled. The ZigBee chip of the gateway would still transmit the Acknowledgements required for this test. The following measurements was conducted measuring the time required to transmit 20 blocks (256 packets) of data. The results is shown in Table 16. It was observed that there were only a few retransmits required. When the antenna of the EM357 was touched, the packets should be retransmitted. This test shows that it could be concluded that the data was actually transmitted.

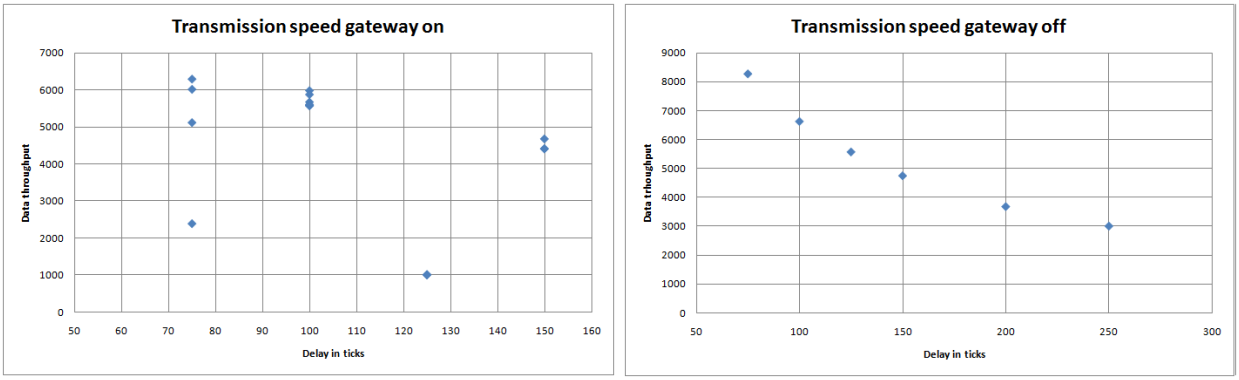
Finally the data is represented in Figure 43. Figure 43a shows the throughput while the gateway processor is online and Figure 43b while the gateway processor is off.

Table 15: Measurement results, with gateway processor on, windowsize= 10, buffersize =40

wait time (ticks)	**time (s)**	**Throughput (Bytes/s)**
75	446,7	2378,33
100	187,6	5663,113
100	191,0	5562,304
100	189,9	5594,523
125	211,6	1004,159
125	212,7	998,9657
125	214,1	992,4334
150	227,6	4667,838
150	241,2	4404,643
150	241,6	4397,351

Table 16: Gateway processor off, ZigBee receiver on, windowsize=10 buffersize=40

wait time (ticks)	**time (s)**	**Throughput (Bytes/s)**
75	51,4	8267,704
100	64,2	6619,315
125	76,4	5562,304
150	89,7	4737,57
200	115,8	3669,775
250	142,0	2992,676



(a) (b)

Figure 43: Transmission speeds, Figure 43a: Data throughput graph when the gateway processor is on. Figure 43b: Data throughput graph when the gateway processor is on

H.3 Conclusions

As can be compared from the throughput results, of when the gateway processor is enables and disabled, the processor of the gateway has a great impact on the data transmission speed. When the processor of the gateway is disabled, the throughput is 8kB/s compared to 6,2kB/s when the processor is enabled. The graph in Figure 43b shows that the amount of waiting ticks directly influences the data throughput, which was expected.