On-node processing
algorithms for activity classification and monitoring in wireless sensor networks

Albu, D.C.

Award date: 2009
MASTER'S THESIS

On-node processing: Algorithms for Activity Classification and Monitoring in Wireless Sensor Networks

by

Ing Diana Albu

Supervisor:

Prof. Dr. Johan J. Lukkien

Dr. Ir. Richard P.H.F.M Verhoeven
Abstract

Wireless sensor nodes (WSNs) are networks of tiny computers that contribute to the creation of smart environments in industry, buildings, homes and hospitals. They are built from nodes equipped with different types of sensors that can gather information and signal certain events. The nodes are usually battery operated and have small memory and little processing capabilities. Transmitting all the information to a more powerful computer for processing is a common approach in dealing with the large amount of data. However, energy is a scarce resource in battery operated systems and the high amounts of radio communication lead to a fast depletion of the batteries. To solve this problem, algorithms for data processing can be developed and deployed directly on the nodes. Hence, optimizing performance of wireless sensor networks implies trade-offs between local computation and communication. With the usage of on-node processing, the lifetime of the sensor nodes can be extended as rather than sending all the raw data to be logged and processed by an external device this can be done online. However, standard algorithms are too complicated to be handled by the processors on the nodes, so new simpler and more efficient algorithms are needed.

The purpose of this research is to investigate algorithms for activity classification, discuss their applicability for wireless sensor networks and implement them using the Open Service Architecture for Sensors (OSAS). OSAS is an event-based programming system for sensor networks designed by the System Architecture and Networking (SAN) group in TU/e which can be used for programming heterogeneous WSNs on a high abstraction level. OSAS was partially developed inside the Wireless Accessible Sensor Populations (WASP) project.

Wearable personal sensors for monitoring and health care are setting a trend in the home care domain. By allowing for continuous monitoring of patients they should improve the quality of care and increase the chances of early diagnosis of high risk problems (for example, heart attacks). Sensors can be used for sensing, logging, transmitting and analyzing a number of physiological signals of the patient. Some conditions are hard to detect during a short doctor visit therefore devices can log the information over longer periods of time for later analysis by the doctor. To provide a faster diagnostic, classification algorithms can be directly deployed on the devices so that logging is not needed anymore. Such algorithms can immediately detect whether a dangerous situation has been reached and inform the patient about it.

This research uses three use cases for activity classification with WSNs: an electrocardiogram (ECG) sensor is used to gather data for heart monitoring and an acceleration sensor is used for movement classification for human monitoring and herd control.

**First part** of the research deals with on-node processing of ECG signals to signal abnormal heart rate conditions. An algorithm was developed that is significantly simpler than traditional approaches but has similar quality and that can run on sensors with very limited capacities. The
algorithm is tested on the nodes and also applied to reference sequences. The second part of
the research is concerned with processing data obtained from an acceleration sensor for several
purposes. As a first use case, a classification of the activities in which a person is involved is done.
This is based on the information obtained from an acceleration sensor positioned at the hip of
the monitored patient. The second use case corresponds to the analysis and implementation of
a herd control application based on the requirements defined in WASP project deliverable 6.3.
The mode in which a certain cow is at every time (laying, standing, sitting, walking) is detected;
the classification also includes step detection and step counting. As a special application, the
ECG processing is fused with a basic classification of a person’s activity level derived from the
accelerometer; this fusing provides a more accurate assessment of the health status. Possible
adaptations based on the current situation of the patient are also discussed.

To prove the usefulness of algorithm classification on the nodes, a comparison was done between
the data transmitted during external logging and online processing in the ECG case; this showed
that 14 times less data needs to be transmitted for on-node processing. Energy measurements
in the two cases also showed that less transmission leads to less energy consumption even if
processing is involved. The measurements revealed that less energy needs to be used for the on
node processing case as compared to the logging case for our given setup.
Foreword

As a final step of my studies in Embedded Systems at the Technical University Eindhoven, I have started the work for my final thesis. The past year I have grown a strong interest in the research of wireless sensor networks so it was only a logical choice to join the Systems Architecture and Networking group at TU/e. Out of the available projects that Prof. Dr. Johan Lukkien introduced to me, I have particularly liked the investigation of activity classification and monitoring. I am generally a result oriented person and I like to see things working so a project at the application level suited me very much.

I am grateful to my supervisor Prof. Dr. Johan Lukkien for offering me the possibility to do a very exciting master thesis and for his comments, feedback and advice throughout the project. I also want to give very special thanks to Dr. Ir. Richard Verhoeven for his constant support, assistance and cooperation every day. Furthermore, I would like to thank Dr.Ir. Marc Geilen for taking part in my examination committee and Dr. Rudolf Mak for taking the time to review my thesis.

Diana Cristina Albu
Embedded Systems
Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
# Contents

1 **Introduction** 19
   1.1 Motivation of Activity Classification with Sensor Nodes 20
   1.2 Problem Description 21
   1.3 Thesis Structure 22

2 **Brief Review of Used Concepts** 23
   2.1 Analog signal 23
   2.2 Digital signal 23
   2.3 Sampling theorem 23
   2.4 Analog to Digital Conversion 24
   2.5 Filters 24
      2.5.1 Analog filters 25
      2.5.2 Digital filters 25
   2.6 Sensors 25
      2.6.1 Sensor Networks 25
   2.7 Hardware 27
   2.8 Open Service Architecture for Sensors Framework 28

3 **Background Information and State of the Art** 33
   3.1 ECG Sensor Application 33
      3.1.1 About the ECG Sensor 33
      3.1.2 QRS detection in ECG 34
      3.1.3 Sources of Noise in ECG 35
      3.1.4 Related Work on Activity Classification of ECG Signals 35
   3.2 Acceleration Sensor Applications 37
      3.2.1 About the 3D Acceleration Sensor 37
      3.2.2 Activity Classification of Humans with Acceleration Information 37
      3.2.3 Herd Control 40

4 **Research and Implementation** 43
   4.1 ECG Data Analysis 43
      4.1.1 More information about the ECG signal acquisition. Pathologies of the heart 44
      4.1.2 New ECG algorithm 44
      4.1.3 Event Condition Action (ECA) model for ECG Signal Classification and OSAS implementation 51
## CONTENTS

4.2 Acceleration Data Analysis ........................................ 53
  4.2.1 Human movement classification with 3D acceleration data ........... 54
  4.2.2 Herd Monitoring ........................................... 73

5 Findings and implications ........................................... 93
  5.1 Feasibility of on-node processing for activity classification .............. 93
    5.1.1 Quality of the ECG Algorithm .................................. 93
    5.1.2 Data transmission: logging and on node processing ................. 94
    5.1.3 Energy measurements .......................................... 97
  5.2 Research Contributions .......................................... 98

6 Conclusions ....................................................... 101
  6.1 Future work ................................................... 101
  6.2 Conclusion ..................................................... 101
List of Figures

1.1 Classification System Components. One (or more) sensor is used to obtain sensory information from the environment as raw data. A feature extractor takes the raw data as input, makes computations and extracts certain features. A classifier then uses the results of the computations and matches them to pre-defined knowledge to determine what event has happened. .................................................. 20

2.1 A filter takes a raw signal as input and outputs the filtered signal .................... 24
2.2 A sensor node is powered by a small battery. The sensor samples the environment. The microprocessor can access the sampled values through the ADC. The radio is used for transmission and receiving of data. A small RAM is usually present as well as flash memory. ................................................................. 26
2.3 A typical sensor network consists of a multitude of sensors. The network can be homogeneous or heterogeneous. Each sensor communicates wirelessly with a few other local nodes within its radio communication range. For a large network there are nodes that play the role of the routing nodes for communication. Depending on the topology these nodes can change. A general purpose computer can also be part of the network. It usually has the role of the end host and takes care of processing the information. ............................................................... 26
2.4 BSN development kit v3 ...................................................... 27
2.5 BSN Node v3 ................................................................. 28
2.6 ECG node and a single use electrode ........................................... 29
2.7 The compiler translates network programs into configuration byte code and maintains a symbol table for the network containing the mapping of names to efficient IDs. The byte code loader broadcasts configuration messages (using UDP). This messages are processed by both virtual (simulated) nodes and real nodes in the network (through a transparent bridge). [4] ......................................................... 30
3.1 Placement of the electrodes for proper ECG signal recording [55]. The corners of the triangles correspond to the following limbs: right arm, left arm, left leg. ....... 34
3.2 Idealized electrocardiogram. The ECG signal consists of several waves: the P-wave, the QRS-complex and the T-wave. These waves represents the sequence of depolarization and repolarization of the atria and ventricles. ................. 34
3.3 Triaxial acceleration signal during cow movement. The three axis can clearly be distinguished. ................................................................. 37
4.1 Our ECG signal classification mapped on the three separate components of the Classification System presented in Figure 1.1. There are two parts of the classification. At the first level, each initial detected peak is compared against predefined amplitude and time thresholds. This is needed to get rid of unwanted false positives. At the second level, the classification of the beat as a normal beat or an arrhythmic beat is done. ................................................................. 45

4.2 Filtering steps and final results of the algorithm ........................................... 49

4.3 If only the ECG sensor is deployed, the heart rate values are the only ones used to detect tachy/bradycardia. While the algorithm gives good results on the nodes and on reference data (MIT Arrhythmia Database) raising an alarm only after 1 high/low HR value will probably cause many false alarms (every time strong noise is recorded for eg). That is why we require at least 4 consecutive values of low/high HR to raise an alarm. ................................................................. 50

4.4 Classification of movement ................................................................. 54

4.5 Accelerometer positioning for Activity Classification ................................. 55

4.6 Acceleration data set I, different activities: sit (initial), stand up, stand up, sit down, lay back, sit, stand up, sit down (end) ...................................................... 56

4.7 Acceleration data set II, different activities: down(initial), stand up, 4 steps, turn left, 5 steps, turn 180 degrees right, 5 steps, turn right, 4 steps, turn 180 degrees left, sit down, turn left while down (end) ...................................................... 56

4.8 Acceleration data set III, different activities: stand (initial), 6 steps, open door, 3 steps, 8 stairs up, turn 180 degrees left, 8 stair down, 3 steps, close door, 6 steps, stand (end). During the transmission there was high packet loss. ...................................................... 57

4.9 Acceleration data set IV, different activities: stand (initial), open door, 3 steps, 8 stairs up, turn left 180 degrees, 8 stairs down, 3 steps, stand (end) ...................... 57

4.10 Acceleration data set V, different activities: stand(initial) , 3 steps, 8 stairs up, 3 stairs down, fall down the stairs, walk ...................................................... 58

4.11 Acceleration data set I, different activities: sit (initial), stand up, sit down, stand up, sit down, lay back, sit, stand up, sit down (end). Level I Classification: static or dynamic. During dynamic activities the result signal has amplitude 1000. ........ 58

4.12 Acceleration data set II, different activities: down(initial), stand up, 4 steps, turn left, 5 steps, turn 180 degrees right, 5 steps, turn right, 4 steps, turn 180 degrees left, sit down, turn left while down (end). Level I Classification: static or dynamic. During dynamic activities the result signal has amplitude 1000. ........ 59

4.13 Acceleration data set III, different activities: stand (initial), 6 steps, open door, 3 steps, 8 stairs up, turn 180 degrees left, 8 stair down, 3 steps, close door, 6 steps, stand (end). Level I Classification: static or dynamic. During dynamic activities the result signal has amplitude 1000. ........ 59

4.14 Acceleration data set IV, different activities: stand (initial), open door, 3 steps, 8 stairs up, turn left 180 degrees, 8 stairs down, 3 steps, stand (end). Level I Classification: static or dynamic. During dynamic activities the result signal has amplitude 1000. ........ 60

4.15 Acceleration data set V, different activities: stand(initial) , 3 steps, 8 stairs up, 3 stairs down, fall down the stairs, walk. Level I Classification: static or dynamic. During dynamic activities the result signal has amplitude 1000. ........ 60
4.16 The mode is detected using the distance between the vectors that represent the acceleration values. ................................................................. 62

4.17 Acceleration data set I, different activities: sit (initial), stand up, sit down, stand up, sit down, lay back, sit, stand up, sit down (end). Level II static: determine sit (amplitude 500), stand (amplitude 700) or lay (amplitude 1000) ............................... 63

4.18 Acceleration data set II, different activities: down(initial), stand up, 4 steps, turn left, 5 steps, turn 180 degrees right, 5 steps, turn right, 4 steps, turn 180 degrees left, sit down, turn left while down (end). Level II static: determine sit (amplitude 500), stand (amplitude 700) or lay (amplitude 1000) .............................................. 63

4.19 Acceleration data set III, different activities: stand (initial), 6 steps, open door, 3 steps, 8 stairs up, turn 180 degrees left, 8 stair down, 3 steps, close door, 6 steps, stand (end). Level II static: determine sit (amplitude 500), stand (amplitude 700) or lay (amplitude 1000) ...................................................... 64

4.20 Acceleration data set IV, different activities: stand (initial), open door, 3 steps, 8 stairs up, turn left 180 degrees, 8 stairs down, 3 steps, stand (end). Level II static: determine sit (amplitude 500), stand (amplitude 700) or lay (amplitude 1000) ...................................................... 64

4.21 Acceleration data set V, different activities: stand(initial), 3 steps, 8 stairs up, 3 stairs down, fall down the stairs, walk. Level II static: determine sit (amplitude 500), stand (amplitude 700) or lay (amplitude 1000) ...................................................... 65

4.22 Acceleration data set VI, different activities: stand(initial), 3 steps, 8 stairs up, 3 stairs down, fall down the stairs, walk ...................................................... 65

4.23 Acceleration data set IV, different activities: stand (initial), open door, 3 steps, 8 stairs up, turn left 180 degrees, 8 stairs down, 3 steps, stand (end). Level II dynamic: determine step type, walking and climbing up the stairs (amplitude -100), climbing down the stairs (amplitude -200) ...................................................... 66

4.24 Acceleration data set V, different activities: stand(initial), 3 steps, 8 stairs up, 3 stairs down, fall down the stairs, walk. Level II dynamic: determine step type, walking and climbing up the stairs (amplitude -100), climbing down the stairs (amplitude -200) ...................................................... 67

4.25 Acceleration data set I, different activities: sit (initial), stand up, sit down, stand up, sit down, lay back, sit, stand up, sit down (end). Level III: transition classification. Standing-to-sitting (amplitude 1000), sitting-to-standing transition (amplitude 700), sitting-to-laying (amplitude 400), laying-to-sitting transition (amplitude 100) ...................................................... 68

4.26 Acceleration data set II, different activities: down(initial), stand up, 4 steps, turn left, 5 steps, turn 180 degrees right, 5 steps, turn right, 4 steps, turn 180 degrees left, sit down, turn left while down (end). Level III: transition classification. Standing-to-sitting (amplitude 1000), sitting-to-standing transition (amplitude 700), sitting-to-laying (amplitude 400), laying-to-sitting transition (amplitude 100) ...................................................... 68

4.27 Acceleration data set III, different activities: stand (initial), 6 steps, open door, 3 steps, 8 stairs up, turn 180 degrees left, 8 stair down, 3 steps, close door, 6 steps, stand (end). Level III: transition classification. Standing-to-sitting (amplitude 1000), sitting-to-standing transition (amplitude 700), sitting-to-laying (amplitude 400), laying-to-sitting transition (amplitude 100) ...................................................... 69
4.28 Acceleration data set IV, different activities: stand (initial), open door, 3 steps, 8 stairs up, turn left 180 degrees, 8 stairs down, 3 steps, stand (end). Level III: transition classification. Standing-to-sitting (amplitude 1000), sitting-to-standing transition (amplitude 700), sitting-to-lying (amplitude 400), laying-to-sitting transition (amplitude 100) 69

4.29 Acceleration data set V, different activities: stand (initial), 3 steps, 8 stairs up, 3 stairs down, fall down the stairs, walk. Level III: transition classification. Standing-to-sitting (amplitude 1000), sitting-to-standing transition (amplitude 700), sitting-to-laying (amplitude 400), laying-to-sitting transition (amplitude 100) 70

4.30 Positioning of sensor node on the cow's leg. 74

4.31 The farm setup. Positioning of the gateways in the test bed 74

4.32 The software structure for the on node processing has been defined in WASP D6.3 Deliverable. The measurement module gathers 3D acceleration data at 10Hz for 1 second every time it is called by the mode selector. The locomotion module is triggered by the activation of ModeIV from the mode selector. An internal timer module is used for timing the application. 75

4.33 Operation modes for the herd monitoring software. Mode0 is used for initialization only. During initialization the program is installed on the nodes and the data for calibration is obtained. The calibration data is obtained during a period when the cow is standing. ModeI is defined as the period when the cow is laying down. The cow is in ModeII when it is standing but not moving. ModeIII corresponds to the period when a cow starts walking. ModeIV is triggered by ModeIII in some situations (Figure 4.34) 76

4.34 When ModeIII is detected, if the step morphology was not analyzed in the last hour ModeIV starts. 76

4.35 Example of how mode detection actually works. The moments when the cow gets up can be easily identified with the degree calculations. 77

4.36 Method II for determining the status of the cow. The mode detection is based on variance and direction of acceleration. For modes with little variance, a vector is used to specify that mode. When the acceleration reading is close to that vector, that mode is selected. The vector calculations are done related to a base vector, which indicates the center of the acceleration readings when no motion occurs. These accelerations readings describe a circle if the node rotates around the leg of the cow. 80

4.37 Step detection flow chart 83

4.38 Acceleration signal as used for step detection. 84

4.39 Result of the step detection algorithm on the signal in Figure 4.38 84

4.40 Step detection flow chart 86

4.41 Using multiple sensor input for activity recognition 88

4.42 If only the ECG sensor is deployed together with the acceleration, the classifier takes into consideration both aspects when making a decision. While the algorithm gives good results on the nodes and on reference data (MIT Arrhythmia Database) raising an alarm only after 1 high/low HR value will probably cause many false alarms (every time strong noise is recorded for eg). That is why we require at least 4 consecutive values of low/high HR to raise an alarm. 90
LIST OF FIGURES

4.43 Sample communication to base station during logging and on-node processing . . . 91
LIST OF FIGURES
List of Tables

5.1 ECG Algorithm Results on Signals from MIT Arrhythmia Database . . . . . . . . 95
5.2 Comparison of the sensitivity and predictivity of the ECG algorithm on the MIT-
Arrhythmia database with results from other papers . . . . . . . . . . . . . . . . . . 96
5.3 Energy measurements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 98
Abbreviations

**WSN**  Wireless Sensor Networks

**OSAS**  Open Sensor Architecture for Networks

**ECG**  Electrocardiogram

**ECA**  Event Condition Action

**ADC**  Analog-to-Digital Conversion

**DAC**  Digital-to-Analog Conversion

**FIR**  Finite-Impulse-Response

**IIR**  Infinite-Impulse-Response

**CBA**  Content Based Addressing

**PDA**  Personal Digital Assistant

**ANN**  Artificial Neural Networks

**HMM**  Hidden Markov Model

**SVM**  Support Vector Machines

**HRV**  Heart Rate Variability
Chapter 1

Introduction

Wireless sensor networks represent an emerging technology with a wide range of potential applications. Sensors can be used for environment monitoring, smart homes and medical systems, robotic exploration and intelligent transportation systems among others (see [44]). They combine sensing, processing, communication and actuating in a small device which can be used for many operations from different domains.

Sensor nodes typically contain a micro-controller and a radio transceiver, digital and analog interfaces for connecting sensors and actuators, batteries, storage memories and sometimes communication ports. Based on their radio capabilities, sensors have a certain coverage area. Networking and cooperation are essential in achieving the desired functionality of the WSN.

In general, the WSNs focus on monitoring their environment and transmitting some important data to base stations and/or gateways. Based on the information obtained from the sensor, different events can be classified. However, the early classification of events can be helpful in most situations. Therefore, it is desired that nodes process some of the information locally and act accordingly instead of reporting the data and waiting for events classification details from the base station.

With the advancement in wireless sensor technologies and wearable computing and the increase in the costs of health care, the use of sensor networks in real time monitoring of daily activities and physiological parameters became practical. Wearable personal sensors are setting a trend in the home care domain. By allowing for continuous monitoring of patients, they can improve the quality of care and increase the chances of early diagnosis of high risk problems. Sensors can be used for sensing, logging, transmitting and analyzing a number of physiological signals of the patient over extended periods of times. Frequent and regular health monitoring is particularly important for the elderly because their health suffers from fast changes. Wireless sensors, however, can be also successfully used in other domains such as in the industrial environment to monitor the activity of the employees and to help at the improvement of their work. They can also be used in agriculture in supervising crops or controlling the health of the herds by keeping statistics of the cows activity levels.

This chapter gives a definition of activity classification and mentions the possible uses of sensor networks in different domains. The thesis structure and an overview of the research problem are also presented.
CHAPTER 1. INTRODUCTION

1.1 Motivation of Activity Classification with Sensor Nodes

Classification is typically considered to be the placing of similar objects into similar groups. An activity is any specific deed, action, behavior or movement of an object. Activity classification is the process of distributing activities in their specific group. When applied to wireless sensor networks, activity classification means transforming the raw data obtained from sampling and assigning it to predefined activity groups. The data is classified based on a priori knowledge about the events or statistical information. WSNs can be used for classifying the events that happen in the environment and act accordingly. Figure 1.1 shows the components of a typical classification system. The raw data is read by the sensor, a feature extractor computes observations from this raw data, and the classifier analyses the features and determines what event happened.

Figure 1.1: Classification System Components. One (or more) sensor is used to obtain sensory information from the environment as raw data. A feature extractor takes the raw data as input, makes computations and extracts certain features. A classifier then uses the results of the computations and matches them to pre-defined knowledge to determine what event has happened.

There are more fields in which WSNs can be used for activity classification.

In an office environment a wireless sensor network can be deployed for activity classification for different purposes. The most obvious usage is for detecting the position and the availability of an employee. For example, by detecting the presence of a person at their desk the telephone calls can be put through or directed to some other location (or their mobile phone). Also, knowledge about the number and the position of people in a room can lead to better climate and light control and more efficient usage of resources in emergency situations.

Industry can benefit from activity classification. The productivity of workers can be improved if the sensors offer them relevant information in due time. Also, the safety-critical steps in the production chain can be better supervised to increase the reliability. The new workers can have an accelerated learning curve by basing their activity on the information received from the sensors [40]. Even more, information about the position of employees in a factory could lead to improved safety and security. In high risk environments, such as chemical and nuclear plants, the personnel is only allowed to be inside for short periods of times while sensors can do sampling and classification continuously without danger of contamination.

Sensor networks can be used in the homes for a variety of reasons. The idea of smart homes has been already in use for a couple of years. For home automation, technology and services are
1.2. Problem Description

The focus of this master thesis is on investigating algorithms for activity classification for patient monitoring and herd control. The effect that on-node activity classification has on energy consumption is also analyzed. The purpose of the thesis can be summarized in three objectives.
• Analyzing the possibility of using WSNs in the context of activity classification.

• Developing methods for activity classification for the following use cases. Mapping the methods on the Event Condition Action (ECA) model of the WASP system. Implementing them in OSAS.

1. Detecting arrhythmias in real time by applying a classification algorithm on the ECG signal obtained from a sensor node. Combining the heart rate details with information about the movement magnitude obtained from an acceleration sensor to do a better classification of the heart condition.

2. Determining a predefined set of human movements (walking, running, climbing stairs up or down, falling) and postures (standing, sitting, laying).

3. Detecting the activity level of cows by analyzing walking patterns from acceleration. Detecting different movement patterns: standing/laying, walking, climbing stairs, number of steps.

• Analyzing and measuring energy savings brought by activity classification on the sensor node as compared to the data logging approach in the ECG case. Analyzing amounts of data transmission in the same situation.

1.3 Thesis Structure

This thesis presents background information about electrocardiogram and acceleration sensors and describes the algorithms for activity classification that were implemented in OSAS. The thesis is structured in six chapters.

• Chapter 2 presents a number of background concepts that are used for the research.

• Chapter 3 gives an overview of the existing research in the topic of health care monitoring with the ECG, presents the existing literature about herd control applications and discusses the research that was already carried out in the field of activity classification with the help of wireless sensor networks.

• Chapter 4 shows the approach taken for the research and explains the algorithms.

• Chapter 5 presents the findings of research, the energy consumption analysis and the discussions about the feasibility of on-node processing.

• Chapter 6 offers an overview of the research, discusses the future work and concludes the thesis.
Chapter 2

Brief Review of Used Concepts

This chapter briefly describes some of the theoretical concepts of signal processing that are used in the rest of the thesis. It also introduces the sensor nodes, the programming environment and the hardware used.

2.1 Analog signal

The signals that can be encountered in engineering are mainly continuous: voltage varies over time, light intensity changes with distance. An analog signal is any continuous signal for which the time varying variable is a representation of some time varying quantity.

2.2 Digital signal

An analog signal must be turned into a digital signal before it can be processed. An analog-to-digital converter (ADC) is used for this purpose. A digital signal is discrete in time and is represented by a sequence of quantized voltage levels. The levels correspond to discrete intervals of time.

This digitization brings along greater functionality, more flexibility and convenient storage. The frequency with which the digital signal is sampled can be controlled. An analog signal can have a range of frequencies so the sampling has to be carefully performed. For example, if the analog signal has low frequency, then it should be sampled at larger time intervals and vice versa. Digitization introduces noise or high-frequency components in the power spectra of the signal.

2.3 Sampling theorem

The continuous signal can be represented by its instantaneous amplitude values taken at periodic points in time. The original signal can be perfectly reconstructed with just these sampled points.

The Sampling Theorem mentions when the original signal can be reconstructed from its samples without any loss of information [27]. It states that, for a continuous band limited signal that contains no frequency components higher than \( f_c \), the original signal can be completely recovered without distortion if it is sampled at a rate of at least \( 2 \times f_c \) samples/s. A sampling frequency
$f_s$ of twice the highest frequency present in a signal is called the Nyquist frequency. If Nyquist theorem is not respected then aliasing can appear.

Though theoretically we could easily establish the sampling rate based on the knowledge of the highest frequency in the signal, this is not always the case because of the noise which might contain higher frequencies than the signal itself. This can be solved by placing a low-pass filter at the input of the sampler. If the hardware imposes limitations on storage and energy consumption, a trade-off can be made between acceptable error and required accuracy [56].

### 2.4 Analog to Digital Conversion

To be able to process a signal, it first needs to be captured with sensors. An electrical signal such as an ECG can be captured directly with the help of an electrode which allows the current to pass from the body to the signal conversion system. For other types of signals such as an acceleration signal, the sensor needs to convert the captured data into an electrical signal.

Analog-to-digital conversion allows digital computers to interact with the real-life signals. As mentioned before, a digital signal differs from a continuous signal in two important aspects: it is sampled, and it is quantized. Therefore, a digital signal does not contain the same amount of information as a continuous signal but a number of points that were selected to represent the continuous signal digitally.

An ADC converts analog voltages to digital information which can be used by a computer. The A/D converter usually needs higher input values than the ones of the original signal so this is amplified as close to the source as possible to avoid degradations. A low-pass filter is used to minimize aliasing.

A sensor captures the original signal, amplifies it and feeds it to an ADC. The A/D converter changes the analog signal into a digital one which will either be stored in the sensor’s memory for further use or processed in real time.

### 2.5 Filters

A filter $h(n)$ represents a computation which takes a sequence of data as input $x(n)$ and produces a new sequence of data as output $y(n)$ (Figure 2.1). The function of a filter is to remove unwanted parts of the signal, such as (random) noise, to extract useful parts of the signal, such as the components lying within a certain frequency range, or to enhance portions of the signal, such as video signal enhancement.

![Figure 2.1: A filter takes a raw signal as input and outputs the filtered signal](image)

There are two broad classes of filters: analog filters for continuous signals and digital filters, which filter discrete signals.
2.5.1 Analog filters
An analog filter is any analog electronic circuit built with the help of resistors, capacitors and inductors to produce a desired filtering effect. Analog filters are built in hardware so they cannot be easily changed when needed.

2.5.2 Digital filters
A digital filter uses a digital processor to perform calculations on numerical values of a digital signal. The signal is passed through an ADC and the resulting successive sampled values are processed usually by additions and multiplications with different coefficients. Digital filters are programmable so they can easily be changed without touching the hardware. There are two type of filters: recursive and non-recursive.

2.5.2.1 Finite input response filters (FIR)
The non-recursive filters have a polynomial as a transfer function which only depends on a finite number of elements of the input signal.

2.5.2.2 Infinite input response filters (IIR)
The transfer function of the recursive filters is a ratio of two polynomials and shows the dependence of the output signal on an infinite number of elements of the input signal.

Integer filters are digital filters with the coefficients integer numbers. They are primarily deployed in environments requiring fast online processing. In general filtering requires many multiplications which are expensive operations by themselves. If floating point coefficients with accuracy of five digits are used then the computations are a lot more expensive than when small integer coefficients are used. Integers are efficient for use in real-time applications because they allow for faster computations by using binary shifts instead of floating point multiplications.

2.6 Sensors
Sensor nodes are small wireless devices that can perform sensing, processing and transmission of information. They usually have several kilobytes of flash and RAM, are powered by small batteries and have a small radio range. Radio transmission is error-prone. The operating systems installed on these nodes provide basic functionality through system calls. Figure 2.2 shows the main components of a sensor node.

2.6.1 Sensor Networks
Nodes are usually not used by themselves but are interconnected to other nodes with similar or completely different functionalities to build a wireless sensor network (Figure 2.3). A heterogeneous network is created by combining different types of sensors to make observations about the environment. For example, an accelerometer detects patterns in the movement of the body, a light sensor detects the sources of light, a rotation sensor detects body movements, a compass detects the orientation of a body, a skin temperature sensor detects the health state of the body,
CHAPTER 2. BRIEF REVIEW OF USED CONCEPTS

Figure 2.2: A sensor node is powered by a small battery. The sensor samples the environment. The microprocessor can access the sampled values through the ADC. The radio is used for transmission and receiving of data. A small RAM is usually present as well as flash memory.

A humidity sensor detects the physical activity level, a sensor measuring the electrical potential between probes detects the activity of the heart while a pressure sensor positioned on a chair, bed or sofa detects where the person is located. Every one of the sensors can provide information to a classifier. However, combining information from different types of sensors can lead to a better classification of activities and to a better exclusion of false detections.

Figure 2.3: A typical sensor network consists of a multitude of sensors. The network can be homogeneous or heterogeneous. Each sensor communicates wirelessly with a few other local nodes within its radio communication range. For a large network there are nodes that play the role of the routing nodes for communication. Depending on the topology these nodes can change. A general purpose computer can also be part of the network. It usually has the role of the end host and takes care of processing the information.

Unlike a system that is centralized, a sensor network is restricted by a set of resource constraints such as finite on board battery and limited communication bandwidth which is error-prone.
2.7. HARDWARE

It is known that communicating one bit over the wireless medium at short ranges consumes far more energy than processing that bit. For some nodes the ratio of energy consumption for communication and computation is between 1000 to 10000 [15]. So, minimizing the amount and the range of communication as much as possible will considerably prolong the life of a sensor network.

In-network processing or on-node processing is a style of processing in which the data is processed and combined near or at the source where it is generated. By employing such a mechanism, the life of the sensor network can be increased.

2.7 Hardware

The hardware used in this thesis for data acquisition, processing and measurements was developed at Imperial College London. The BSN development kit v3 consists of a USB programming board, a sensor board, a battery board, a prototype board and a pair of the new BSN v3 nodes (Figure 2.4) [36].

![Figure 2.4: BSN development kit v3](image)

- USB programmer board: designed for interfacing and programming the BSN node via the USB port and serves as the battery charger for the battery board.
- Sensor board: consists of a 3D accelerometer and a temperature sensor
- Battery board: the power supply for the BSN node, and a Li-polymer battery with 55mAh is attached as the power source.
- Prototype board: designed for prototyping different sensors and power supplies with the BSN node.
• BSN nodes: have the following specifications (Figure 2.5)
  – 19mm x 30mm
  – 16-bit ultra low power RISC processor [330μA (Active), 1.1μA (Standby), 0.2 μA (off mode)]
  – 48KB flash/ 10KB RAM
  – 8 channels 12-bits ADC
  – 2 channels DAC
  – 2 USART
  – TI CC2420 radio transceiver
  – IEEE 802.15.4 (2.4GHz DSSS)
  – Hardware MAC 128 bit AES encryption (Advanced Encryption Standard)
  – Range 50m (indoors) 125m (outdoors)
  – Fitted with a miniaturized chip antenna
  – 4MB external EEPROM
  – TinyOS with corresponding MAC

For the ECG acquisition a sensor board with three attached electrodes is used (Figure 2.6(a), 2.6(b)).

Besides TinyOS, MantisOS is also available with its default MAC protocol. On top of these platforms, the OSAS framework [9] is used. For energy measurements, the OSAS OS (developed at TU/e) with a variant of TRAWMAC are adopted.

2.8 Open Service Architecture for Sensors Framework

OSAS is an event-based programming system for networked devices that is built to cope with the limited resources available on sensor nodes. Moreover, it allows their reprogramming over the network. OSAS is partly developed inside the WASP project at TU/e [23].
Nodes follow a push and pull communication mode. The base of the communication is defined by services. Each node can have one or more services and other nodes can call these services by subscribing to them. Data consumers are connected to data producers through subscriptions.

Subscription can have different parameters among which also the handler from which the subscriber wants a response. OSAS follows an event condition action model. When an event occurs the node executes the service it was requested to execute. Subscriptions also specify non-functional behavior in terms of timing, priority and reliability [51]. Nodes are addressable by means of content-based addressing (CBA). This means that services are installed on nodes based on capabilities rather than IDs, which is a huge advantage when the exact number and identities of nodes are not known in advance. Also content-based addressing is extensively used in the subscription mechanism where a third party, typically the gateway, initiates a many-to-many subscription specified by relationships (i.e., the content-based addresses) between subscriber and provider.

OSAS defines the following four components: a compiler, a loader, a runtime system and a simulator (Figure 2.7).

The compiler is responsible for taking a program written in the domain specific programming language (.wsp files) and transforming it into configuration messages for the nodes (.wbc files) which are later interpreted by the run-time system on the devices. This translation consists of two parts:

- Generation of byte code for content-based addresses, event generator and event handler bodies.
- Embed the byte code into the proper handlers for transporting them to the correct nodes in the network.

Content-based addresses are verified by invocation of the content-based address handler. Transporting the byte code to nodes satisfying such an address, requires the inclusion of the byte code in a configuration handler embedded in the CBA handler. The compiler supports besides simple computation also dealing with specific operations.
Figure 2.7: The compiler translates network programs into configuration byte code and maintains a symbol table for the network containing the mapping of names to efficient IDs. These messages are processed by both virtual (simulated) nodes and real nodes in the network. The byte code loader broadcasts configuration messages (using UDP). This messages are processed by both virtual and real nodes in the network (through a transparent bridge).
The **loader** sends the CBA definitions and the compiled services into the sensor network, leading to a dynamic installation. It has to ensure the correct configuration of nodes.

The **simulator** implements a virtual machine similar to the one deployed on the nodes. It can generate an initial network state description (the .wnc files) and simulate one or more nodes. These virtual nodes can provide significantly richer functionality than the real nodes can provide. This allows the simulator to be deployed as a powerful gateway node, hub or just as a testing platform for sensor applications. Through a bridge application, a transparent broadcast domain is constructed between real sensor nodes and simulated nodes. The loader uses this broadcast domain to install the program and it is subsequently used by the connected services, irrespective of whether a service runs on a real or on a simulated node [51].
Chapter 3

Background Information and State of the Art

This chapter offers an insight into the specifics of electrocardiogram and acceleration processing for activity classification with wireless sensors. It also presents some information about the specific sensors used (ECG sensor and accelerometer) and a review of the state of the art.

3.1 ECG Sensor Application

The electrocardiogram has been an important instrument for heart diagnosis since 1920 [7] and is still a common means of detecting heart problems in patients. The ECG is one of the most relevant biological signals: the heart beat rate (HR) is extracted from the ECG and the continuous evolution of the ECG waveform is used to diagnose several cardiac disorders.

3.1.1 About the ECG Sensor

When the heart depolarizes and repolarizes, electrical current does not only spread within the heart but also to the adjacent tissue. A small portion of this current reaches the body surface. The electrocardiogram is the recording of the electrical activity measured by a number of non-invasive electrodes placed on the body skin. A lead is a view of the electrical activity of the heart from a particular angle across the body, obtained by using different combinations of these wires. ECG sensors are used to measure the time-varying magnitude of the electric fields emanating from the heart [25].

There are more types of ECG measuring devices, the most common used in medicine having 12 leads, however, sensors with as low as 3 electrodes exist. The placement of the electrodes is very important for diagnostic and depends on their number. By changing the position of either of the electrodes the angle at which the activity of the heart is viewed is altered. Depending on the number of wires different views at the heart can be obtained.

For this research a 3-wire ECG sensor is used. The electrodes need to be placed in a triangle on the patients chest as in Figure 3.1. Each corner of the triangle corresponds to one of the limbs: right arm, left arm, left leg. For the convenience of the patient and for decreasing the noise that comes from moving the limbs, the electrodes are attached to the corresponding shoulder and the
correct lower abdomen side with little influence on the result. A 3 wire cable (red, yellow, green) only gives a choice of one view of the heart.

![Placement of the electrodes for proper ECG signal recording](image)

**Figure 3.1**: Placement of the electrodes for proper ECG signal recording [55]. The corners of the triangles correspond to the following limbs: right arm, left arm, left leg.

The ECG is represented by an analog signal which is continuous in time. The sensor reads the analog signal and the ADC transforms it into a digital signal.

### 3.1.2 QRS detection in ECG

A representation of an ideal ECG signal is presented in Figure 3.2. Parts of the signal correspond to a phase in the cardiac cycle: P wave is caused by the depolarization of the atria, QRS complex by the depolarization of the ventricles and T wave by the repolarization of the ventricles. The repolarization of the atria is concealed in the ECG by the QRS complex [56].

The activity of the heart is reflected in the QRS complexes of the signal. The analysis of the QRS complexes can be used for automatic detection of heart rate values. In general, the heart beat rate is measured by detecting the R peaks in the QRS complex. The interval between two consecutive R peaks is the periodicity of the beat. Once the duration of RR interval is known, the instantaneous heart beat can be calculated by

\[
HR[\text{bpm}] = \frac{60 \text{[s/min]}}{RR[\text{s/beat}]}.
\]

![Idealized electrocardiogram](image)

**Figure 3.2**: Idealized electrocardiogram. The ECG signal consists of several waves: the P-wave, the QRS-complex and the T-wave. These waves represents the sequence of depolarization and repolarization of the atria and ventricles.
3.1.3 Sources of Noise in ECG

The ECG signal can be affected by noise which has an influence on the interpretation of the data. The noise can be caused by the following [16]:

1. Interference from power lines adding 50 or 60 Hz power-line frequency.
2. Muscle contraction and muscle activity generates high-frequency electromyography (EMG) noise.
3. Motion artifacts such as movement of the electrode over the skin surface.
4. Impedance changes at the skin/electrode interface due to temporary loss of contact or loose electrodes.
5. Baseline drift due to respiration.
6. Noise introduced due to instrumentation or electronic devices.

Before using the signal for calculating the heart rate it needs to be filtered to get rid of the unwanted noise and to increase the signal-to-noise ratio.

3.1.4 Related Work on Activity Classification of ECG Signals

Activity of the heart is a good indicator of the health status of a patient. Activity classification has been a topic of interest for a long time and with the advances in wireless sensor networks it was only a matter of time before researchers developed algorithms that can run directly on the nodes.

ECG signal analysis is a common means of detecting heart problems in patients. In the past 30 years many algorithms and methods have been developed for high quality analysis of the ECG signals. However, due to limitations in the processing power and memory that sensor nodes have, porting algorithms that were constructed for more advanced computers to sensor nodes is not feasible.

There are a number of products that offer the possibility of home monitoring for short periods of times. The most common type, the Holter monitor [25], has been developed for continuous recording of the hearth’s rhythms for 1 or 2 days. The data is stored on a flash card and transferred to the doctors computer for analysis at a later time. Other similar products do not store the data to an attached flash card but transmit it over the radio channel to a central gateway (a PDA or a personal computer) for storing, processing and feature extraction ([29], [58], [46], [48]). [21] presents a prototype wearable wellness monitoring system capable of recording, transmitting and analyzing continuous ECG data. The hardware introduced in [21] allows data to be transmitted wirelessly from on-body sensors to a handheld device using Bluetooth which is then further transmitted using either a wireless Internet connection, if available, or a cellular phone service to a back-end server for analysis.

Besides the inconvenience of carrying around a PDA which contradicts the idea of unobtrusive monitoring that the use of sensor nodes advocates, such approaches waste unnecessary energy by using the radio channel to transmit all the raw data to the central gateway. Also, the processing capabilities of the sensors go unused if all the processing is done elsewhere.
Extensive post-processing of the ECG signal is done on a PDA or on a separate computer. There are more pathologies that can be deduced from the detection of some or all the waves in the ECG (described in Section 4.1). Many of the algorithms that exist in literature are able to detect besides the QRS complexes also the P and T waves. To do so, sampling should be at high frequencies which is not feasible for sensor nodes (above 200Hz). General methods for detection of ECG features are based on filtering or adaptive thresholding, wavelet transform, multiscale transform or statistical analysis of the signal. There are a number of algorithms for detecting the beats. Accurate detection of beats is important as the heart rate is determined by measuring the length of the interval between two consecutive beats [49]. The most common algorithms suggest calculating the zero crossing of the first derivative of the signal, or calculating for the second derivative the points where this is above a certain threshold [31].

An extensively used approach for detecting the R-peaks is the Pan-Tompkins algorithm. This algorithm distinguishes 4 stages: a low pass filter and a high pass filter (combined to offer a bandpass that filters out noise such as the 50Hz power line noise), a differentiator that gives the QRS slope information, a squaring function that makes points positive and amplifies the output in the higher frequencies, and a moving window integrator that extracts information to detect the QRS by averaging a number of samples per window. The Pan-Tompkins algorithm has been implemented also in hardware [18]. Other approaches build nonlinear models of the electrocardiogram and do fitting of the parameters of the models using nonlinear optimization [17] or calculate heart rates by using curve length transformations combined with some decision rules [61]. The discrete wavelet transform [50] is also used to determine the position of the R-peaks and so are neural networks [45].

Algorithms proposed in literature are too complex to be implemented on the wireless nodes. They require high sampling frequencies, intense computations and storing. Few solutions have been proposed for systems that analyze ECG signals online and only transmit the relevant features to the central gateway ([33]) however they also use higher sampling frequencies. [52] presents an approach similar to the one presented in this research for doing on-node processing but on a node with larger memory and using a more complicated algorithm that requires more processing power and hence, also consumes more energy.

In the medical field (electrocardiogram ECG, electro-encefalogram EEG, and electromyography EMG), features are associated to pathologies, state of the body or motion patterns. Because one feature of the signal can be associated with multiple health problems and in some situations even with healthy responses of the body, research is looking for methods of obtaining better understanding of data on the subject of interest [7].

An analysis of the existing common feature extraction methods is important to illustrate the importance of selecting the correct features for detection and diagnosis. Features that can be extracted from the signal are not always useful in describing fully the pathology of the patient. An automatic diagnosis system for detecting heart beat irregularities with the help of ECG waveforms uses processed ECG signals as inputs and is trained to make connections between this information and a possible pathology. It will then be able to recognize the correct problem when faced with new input. Such tasks can be accomplished by using computational intelligent techniques such as Artificial Neural Networks (ANN), Support Vector Machines (SVM), Hidden Markov Model (HMM), fuzzy logics, and hybrid systems constructed from a combination of these techniques [7]. [35] presents a method to detect atrial fibrillation for an ambulatory monitoring system based on the variance of R-R intervals. It initially uses the algorithm in [61] for detecting the R-R intervals,
it normalizes the intervals according to a specified equation and calculates the variance of the
normalized value over a sliding window.

The Master thesis [57] proposes a method to detect epileptic seizures in real-time using pattern
recognition on the heart rate signal. It describes a model for the seizure related patterns that
may occur in the heart rate signal. The method is based on analysis of a linear acceleration,
a possible plateau and an exponential deceleration. The detection of these linear accelerations
and exponential decelerations is implemented as separate functions that continuously use linear
regression on the latest samples of the heart rate signal. However, the algorithm was not ported
on the sensor nodes but only implemented in Java.

3.2 Acceleration Sensor Applications

Acceleration sensors can be used for classifying more activities: day-to-day movement of humans
and animals, movement of different parts of a processing chain in a factory, movement of employees
on an assembly line, etc.

This research discusses two applications of wireless sensor nodes: classification of human move-
ments and classification of herd movement patterns.

3.2.1 About the 3D Acceleration Sensor

An accelerometer measures magnitude and direction of the acceleration forces as a vector quantity.
The acceleration forces may be static, like the constant force of gravity, or dynamic, caused by
moving or vibrating the accelerometer.

A 3D acceleration sensor records the acceleration in three orthogonal axes (x, y, z). An axis
of the acceleration sensors used in this research can measure values in the range of $\pm3g$. Figure
3.3 shows a representation of the acceleration on three axis when a cow makes some steps.

![Triaxial acceleration signal during cow movement. The three axis can clearly be distinguished.](image)

Figure 3.3: Triaxial acceleration signal during cow movement. The three axis can clearly be
distinguished.

3.2.2 Activity Classification of Humans with Acceleration Information

Advances in the sensor technology have resulted in interest for monitoring persons over long period
of times. Sensors that can detect changes in gravitational acceleration have been used to measure
Automatic classification of human movement is a feature that is desired for a multitude of applications. The small size of wireless sensors can bring along many advantages in the supervision of workers in factories, sportsmen during exercise and patients during recovery periods. It is especially important to be able to detect daily living activities for patients recovering after injury or elderly people that need constant monitoring.

Monitoring systems are usually complex and contain one or more sensors for each physiological parameter that needs to be analyzed. Body worn acceleration sensors can be used to classify activities and postures. Different approaches have been proposed to doing classification (see Section 3.2.2.1).

Most of the methods and algorithms have been described and applied in an offline mode but there are almost no implementations of algorithms that work in real time on a node. It is more advantageous in terms of energy consumption (Section 5.1.2) to be able to do processing on the node without transmitting the acceleration data over the wireless channel to an external device. Moreover, sampling rate and processing can be adapted immediately once a certain event is detected based on the current activity level and predefined rules.

### 3.2.2.1 Related work on Acceleration Signals Classification

[11] introduces a set of features for the classification of the following human activities: standing/sitting, lying, running, walking flat, walking upstairs and downstairs. A binary classification tree is used for ambulatory activity classification tasks which require low complexity processing. The set of features is extracted from a single vertical axis accelerometer placed on the thorax of the subject. If a rhythmic activity is detected, the system performs the activity classification in a three dimensional feature space. If no rhythmical movement is detected, the system successfully discriminates between standing/sitting and lying, by estimating the projection of the gravitational vector. The classification algorithm is based on the description of the kinematic of human walking which says that the mechanics of walking is defined as a controlled falling where the center of gravity oscillates over the supporting limb following an inverted pendulum movement.

[10] presents a rapid shake detection algorithm that consists of two major conditions that are required to trigger a fall alarm: first criterion is that the movement should be of a certain magnitude and the second condition computes the amount of time during which the elevated activity level happens. This second condition is used to eliminate false alarms caused by brief violent movements such as a jump, or the repetitive acceleration peaks caused by walking the stairs.

[28] detects four types of subjects motion by analyzing waveform changes of the accelerometer data. For analysis a principal component analysis and a support vector machine method for clustering the first and second principal components are used and for classification a supervised learning method for segmentation algorithm is used.

[60] describes a smart shirt that can measure the ECG and acceleration signals. All the data measured by the shirt is sent to a PC for remote monitoring.

[37] introduces the idea of a context-aware cardiovascular long-term monitoring system to enable continuous patient-friendly measurements of blood-pressure and electrocardiogram (ECG). The paper analyses the possibilities of extracting context information from an acceleration sensor.
and claim it as being an important factor in the diagnosis process. From the acceleration signals, different methods of movement detection and classification such as Nearest Neighbor rule, K-Nearest Neighbor rule and Neuro-fuzzy classification are investigated. Using the adaptive neuro-fuzzy inference method, an online activity recognition system has been implemented on a Personal Digital Assistant (PDA).

[19] presents three algorithms for detecting steps from an accelerometer signal. The Pan-Tompkins method for heart rate detection is adapted but it proves to result in false peak searching intervals. The template matching method detects steps adaptively and generates the representative templates according to the current step signal if the first template is correct. However, as the first template is estimated the method may not be appropriate. The paper also presents a peak detection based on combined dual-axial acceleration signals which seems to be the fastest and the easiest.

[54] classifies walking on level ground from walking on a stairway using a waist acceleration signal. The data is sampled at 256 Hz. Eleven healthy, elderly subjects are asked to walk through a corridor and up and down a stairway as a single sequence, without any instruction. The data is analyzed using a discrete wavelet transform. Walking patterns are classified using two parameters: the ratio between the power of wavelet coefficients which were corresponded to locomotion and total power in the antero-posterior direction (RPA) and the ratio between root mean square of wavelet coefficients at the antero-posterior direction and that at the vertical direction (RAV). Walking up stairs is distinguished by the smallest value in RPA from other walking patterns. Walking down stairs is discriminated from level walking using RAV. The paper showed it is possible to classify the walking pattern using acceleration signals in elderly people.

[26] presents a minimum complexity algorithm that performs human motion activity classification in 6 different classes: walking, running, sitting, lying and walking upstairs/downstairs. The algorithm was successfully implemented on a LPC2106 platform. The classification is done based on a decision tree on the signal obtained from a single acceleration channel corresponding to the subject’s vertical axis.

[38] introduces a generic framework for the automated classification of human movements using an accelerometer based monitoring system. The classification is done based on a binary decision tree in which movements are divided into classes and subclasses at different hierarchical levels. General distinctions between movements are applied in the top levels, and successively more detailed subclassifications are made in the lower levels of the tree. This framework is used to develop a classifier to identify basic movements from the signals obtained from a single, waist-mounted triaxial accelerometer.

[3] proposes an algorithm that has the ability to discriminate between falls and activities of daily living using tri-axial accelerometer sensors, mounted on the trunk and thigh. Data analysis is performed using MATLAB to determine the peak accelerations recorded during eight different types of falls. These included: forward falls, backward falls and lateral falls left and right, performed with legs straight and flexed. Falls detection algorithms are devised using thresholding techniques. Falls are distinguished from ADL for a total data set from 480 movements through a single threshold determined by the fall-event data-set, applied to the resultant-magnitude acceleration signal from a tri-axial accelerometer located at the trunk.
3.2.3 Herd Control

Due to the increasing scale of farming, technological advances in the domain of efficient monitoring of farming equipment and goods are needed. Having the possibility to closely monitor big groups of animals in real time in their actual environment, diseases (claw diseases, mastitis) can be detected faster and actions can be taken sooner. The help of wireless sensor networks can bring a major impact to animal welfare and health.

Normal scenarios for cow monitoring refer to detecting the location of the animals, creating a health status from locomotion patterns (for example, claw problems can be detected by step morphology) as well as doing health statistics and history for each animal in part.

The location of the cow can be determined with the signal strength with respect to the gateways spread in the barn or pasture. The movement acceleration patterns of the feet in the forward, side and vertical direction can be constantly monitored to detect the current locomotion status and movement morphology. From the acceleration data, activity and behavior can also be determined. The accelerometers can be used to also detect whether the cow is standing or sitting at any certain time. With this information health statistics are created and the farmer can be informed about a possible dangerous situation in due time.

One of the biggest problems in dairy farming nowadays is lameness. Usually lame cows have lower milk production so they have to be discharged early. If lameness is detected early enough and is due to claw health it can easily be treated. The number of cows has increased due to the technological advancements in feeding and milking systems, but the check for claws health is still done manually. This is becoming increasingly hard so the need for an automated detection of cow behavior is growing. Cows can be equipped with wireless acceleration sensors which do measurements on their movement patterns [32].

3.2.3.1 Related Work on Herd Monitoring

Though it is a promising area of research, there is not much work done in the field of herd control with wireless sensor networks. Researchers from Wageningen University [32] have investigated whether it is possible to analyze foot movement of dairy cows with an acceleration sensor. They have equipped cows at Proefboerderij Waiboerhoeve, Lelystad with three 3-axis acceleration sensors (one on each of the rear legs and one attached to the neck). The data obtained from the sensors is preprocessed to remove noise and then an algorithm for step detection is applied. Morphological parameters also calculated from the obtained data. The research showed that step detection with acceleration sensors is certainly possible.

A similar research is also done in the WASP project by [53]. They suggest an algorithm for filtering the data by means of a median and a moving average filter followed by the step detection on the obtained data.

[30] presents a wireless acceleration measurement system that was applied to free-moving cows and horses. Sensors were attached at the collar. Results were transmitted simultaneously by radio or stored in an 8-MB internal memory. By means of frequency distributions with standard deviations, spectral analyses, and fractal analyses basic behavior patterns (standing, grazing, walking, ruminating, drinking, and hay uptake) could be identified in cows. The paper claims that lameness could be detected in cows and horses by means of a sensor attached to the leg. Their approach however uses very high sampling rates (10, 100 or 1000 Hz). The results are based on standard deviation calculation.
3.2. ACCELERATION SENSOR APPLICATIONS

The presented methods show that using acceleration sensors for detecting lameness in cows can be done, however the researchers have not implemented the algorithms on real nodes. Doing an actual implementation is certainly important because the environment on the sensor nodes differs significantly from the one from a general purpose computer.
Chapter 4

Research and Implementation

This chapter presents the approach taken to solve the research problem exposed in Section 1.2. The first part of the chapter presents the new algorithm for heart rate detection and the classification of the health status of the patient determined from the calculated HR value. The second part of the chapter discusses the applications of classification with the help of an accelerometer for two separate situations: patient activity monitoring (ECG and acceleration monitoring) and herd monitoring. As mentioned in previous chapters, different sensor nodes can be networked together to produce better results for a classification. As an extra application to the accelerometer data processing and in network communication, the combination of an acceleration sensor and an ECG sensor is also studied.

4.1 ECG Data Analysis

The electrocardiogram was introduced in Section 3.1. It is a common means of supervising the health status of a patient during a recovery period or of a sportsman during intense exercise. In the context of body sensor networks that monitor health parameters such as heart rate, it is better to communicate relevant information immediately to the patient and doctor when a bad condition is detected, rather than postponing this by transmitting the raw data to a central processing unit first. Moreover, as radio transmission is error prone, it is hard to guarantee that all the transmitted packages have arrived at the destination and hence important information might get lost.

A preferred alternative to the external transmission of data is the processing of the ECG data on the sensor node. Local processing could save a great part of the energy required for transmitting the raw data over the wireless link. It has been long talked of, but little implementations exist for ECG processing on the node.

At the sensor level, a "biocomputer" [59] is created which can extract important features from the sampled data directly. Many of the existing classification techniques for ECG are designed for offline analysis. They require extensive computation resources and using them for on-node processing is not feasible due to the low processing capabilities of the sensor nodes. This puts forward the requirement for new or modified approaches in order to obtain results in real time on the sensor nodes.

The first part of this chapter is related to the analysis of the ECG signals received from an ECG sensor. The goal is to investigate the possibility of detecting the heart rate through on-
node processing and to implement a R-R detection algorithm on OSAS toolchain (Section 2.8). Sensor nodes are devices with limited capabilities (Section 2.7). Moreover, OSAS imposes its own restrictions in terms of data types and data amounts, so the complexity of the algorithm is bound by the limitations of OSAS and the sensor nodes.

For finding a good algorithm that uses very little resources different approaches are tested in Matlab. These are either combinations of ideas from literature or methods proposed by us. Eventually one method (Section 4.1.2) is chosen and ported to OSAS.

4.1.1 More information about the ECG signal acquisition. Pathologies of the heart

To be able to analyze the ECG signal, general knowledge about its representation is required. The ECG is represented by an analog signal which is continuous in time. The sensors are reading the analog signal and the ADC transforms it into a digital signal. The sensor amplifies the signal and records the electrical activity that results in the waveform of the ECG.

Though usually the bandwidth used for recording the standard 12-lead ECG is 0.05-100 Hz, for monitoring applications (e.g., ambulatory patients) the bandwidth is restricted to 0.5-50 Hz because in these cases only rhythm disturbances (i.e., arrhythmias) are principally of interest and not all the subtle morphological changes in the waveforms. [56]

Typical frequency components of a QRS complex range from about 10 Hz to about 25 Hz which causes the need for a filter stage prior to the actual detection in order to attenuate other signal components and artifacts.

There are numerous pathologies that can be deduced from anomalies of the ECG signal but to detect most of them high sampling rates are needed. Due to current limitations of the sensor nodes in terms of memory and power this research is only concerned with detecting arrhythmias, like brachycardia and tachycardia based on heart rate variations. Cardiac arrhythmias are disturbances to the normal cardiac rhythm. The normal healthy heart beats between 60 to 100 beats/min and variations in this rhythm can reflect several disorders. Conditions where the heart beats slower than 60 beats/min are known as bradycardia. Conditions where the heart beats faster than 100 beats/min are known as tachycardia. [22]

4.1.2 New ECG algorithm

The method used for classifying the ECG data is summarized in Figure 4.1. ECG signal classification mapped on the three separate components of the Classification System presented in Figure 1.1.

There are two types of the classification. At the first level, each initial detected peak is compared against predefined amplitude and time thresholds. This is needed to get rid of unwanted false positives. At the second level, the classification of the beat as a normal beat or an arrhythmic beat is done.

Saving energy and consequently, increasing the the lifetime of the sensor is important. This research is focused on obtaining a reliable heart rate estimation from the ECG data while keeping the algorithm as simple as possible. Experiments were done with processing the data on the nodes with our framework and transmitting only required information over the wireless link.

As can be seen in Section 5.1.2, the difference in data transmission between a logging application
Figure 4.1: Our ECG signal classification mapped on the three separate components of the Classification System presented in Figure 1.1. There are two parts of the classification. At the first level, each initial detected peak is compared against predefined amplitude and time thresholds. This is needed to get rid of unwanted false positives. At the second level, the classification of the beat as a normal beat or an arrhythmic beat is done.
and on-node processing is quite large and is also reflected in the energy measurements from Section 5.1.3. There is a trade off between processing and communication: less energy is consumed for communication but more energy needs to be consumed for processing. However, processing of a sample consumes orders of magnitude less than transmitting the same sample. Also, depending on whether the heart rates need to be constantly sent to a base station or not and the latency permitted for the transmission, the communication level can be reduced up to no communication at all. For the processing part, a very simple algorithm will consume less energy than a more complicated one.

It is sufficient to detect the presence of the QRS complexes to obtain the heart rate. A first resource saving is obtained by lowering the sampling rate. While for detecting all the waves of the ECG signal as shown in Figure 3.2 higher sampling rates are needed due to the high frequency content of the signal, finding the QRS complexes is done in this case with a sampling rate of just 100Hz. We refer to the sampling rate by $sr$.

Noise sources that can appear during ECG sampling include 50Hz interference from power lines and baseline wander caused by movement and respiration. Filtering and preprocessing needs to be done to attenuate other wave components and artifacts and to ensure an accurate detection of the R–peaks. Due to limitation of the platform (Section 2.7), integer filters are used as they provide faster processing by doing integer arithmetic operations. The approach presented here is derived from the Pan-Tompkins algorithm.

The first step of the classification process is obviously the sampling. This is done at $sr = 100Hz$.

The next step of the classification process involves processing the data and extracting relevant features. The data is passed through a number of processing steps. Thresholds are also calculated.

The filtering consists of several steps. P, T waves as well as the baseline drift should be attenuated. The digital ECG signal $x(n)$ (Figure 4.2(a)) is initially passed through a 3–tap filter. The $n^{th}$ value of the signal is calculated from the $n^{th}$, $(n-1)^{st}$ and $(n-2)^{nd}$ samples of the unfiltered signal, in the following way:

$$y(n) = x(n) - 2 \times x(n-1) + x(n-2) \tag{4.1}$$

The 3-tap filter from Equation 4.1 completely eliminates the baseline drift (Section 3.1.3 from the raw signal, enhances the R-peaks and diminishes the background noise and the P and T waves as can be seen in Figure 4.2(b).

After the filtering, the first and the second derivative of the signal are calculated to determine the characteristics of the signal. The first derivative represents the rate of change of the initial signal. It will identify the high frequency portions of the signal and therefore enhance the steep changes. The steepest changes are at the R-peaks of every heart beat. The second derivative will be zero when the first derivative is constant and will be negative/positive at the bottoms/peaks of the first derivative so they will be minimum/maximum at the R-peaks. Using the sum of first and second derivative will then enhance the QRS complexes of the ECG signal.

Due to the nature of the data, the first derivative is approximated by the five-point equation with integer coefficients described by the following equation. This form is chosen over the more regular consecutive differences approximation as the latter, in the presence of noise, tends to amplify it. Pan-Tompkins has shown that this equation approximates an ideal derivative between dc and 30 Hz so it is a good choice for an ECG application. \footnote{In this equation the $\frac{1}{8}$ scaling factor is ignored due to the need of having integer coefficients for the calculations.} This five-point derivative prevents...
4.1. ECG DATA ANALYSIS

high-frequency noise amplification:

\[
fd(n) = 2 \cdot y(n) + y(n - 2) - y(n - 3) - 2 \cdot y(n - 4) \tag{4.2}
\]

The second derivative is approximated by a five-point equation. This is obtained by applying the center difference formula for first derivative twice consecutively.

\[
sd(n - 2) = y(n) - 2 \cdot y(n - 2) + y(n - 4) \tag{4.3}
\]

The results of the two equations are added and the absolute value of the sum is taken (Figure 4.2(c)).

\[
z(n) = \text{abs}(fd(n) + sd(n - 2)) \tag{4.4}
\]

A QRS complex usually takes 0.06 to 0.1 seconds so for a 100Hz sampling rate it will occupy 6 to 10 samples. It is then safe to apply a 6-tap moving average which will create a smoothing of the signal and will diminish random peaks that are not part of the QRS complexes but will not completely remove the real peaks (Figure 4.2(d)).

\[
\text{final}(n) = \text{Average}(z(n - 5 : n)) \tag{4.5}
\]

The amplitude threshold \( Th \) is calculated. For this half the maximum value of \( \text{final} \) calculated during the previous period is used. The rationale behind this is that, while the amplitude of the signal can differ from one beat to the other, the maximum does not change that much. The variance over the last 6 values of \( \text{final} \) is also calculated.

The last step of the classification, detection of QRS complexes, can now start. The granularity \( G \) of the heart rate calculation is a parameter and can be set to the desired value (e.g. \( G = 5 \) seconds or \( 5 \cdot sr \) samples). Output will be generated every \( G \) seconds.

In order to find the positions of the R–peaks every value in \( \text{final} \) is compared against the threshold \( Th \). If the value is larger than \( Th \) and the variance of the previous 6 values is larger than 1000 (which is also a parameter) the sample is considered to be an initial R–peak.

The algorithm is calibrated for the data obtained from the sensor described in Section 2.7 (i.e., using the value of 1000 and choosing half the maximum as threshold). For different sensors, new calibration might be needed. However, for most of the test sequences in the MIT-Arrhythmia database these numbers gave good results without recalibration (Section 5.1.1).

For the reduction of false positive detections a number of extra decision rules are fed into the classifier:

- Samples with amplitude smaller than 1000 (int16 values on our system – represented on two bytes) are discarded because they correspond to periods when the battery level is low and do not provide accurate representation of the ECG signal. After reading samples below 1000 for more than 10 seconds a battery low alarm is activated.

- If the difference to the previous peak is less than 15 samples then the new peak is discarded. This ensures a detection of up to 400 bpm which is large enough to accommodate even the higher heart rates that a patient suffering of tachycardia can get.

- If the difference between two consecutive detected peaks is larger than 300 samples the old peak is discarded and the measurements restart from the current peak. This ensures a heart rate up to 20 beats/sec. If this number exceeds 300 samples then an Alarm is raised to check the connection of the leads to the skin or the battery level.

The scaling factor of the next equation, 4.3, is also ignored.
The last step is the calculation of the heart rate every $G$ seconds by taking into consideration the number of beats and the average number of samples between them (according to 3.1.2). A sample signal and the results obtained on the nodes for $G = 2$ seconds can be seen in Figure 4.2(e).

At a second level of the classifier, the classification of the beat type into either normal beat or arrhythmic beat is done. In case arrhythmic beats are detected, alarms can be raised according to predefined rules. A sample classification of arrhythmic beats is presented in Section 4.1.2.1.

4.1.2.1 Classification of Heart Beats

Increase and Decrease in the Heart Rate  The classification of the ECG signal is done based on a number of rules. Variations of the heart rate values from normal are used to detect tachy/bradycardia. While the algorithm gives good results on the nodes and on reference data (MIT Arrhythmia Database) raising an alarm only after 1 high/low HR value will probably cause many false alarms (every time strong noise is recorded for example). That is why we require at least 4 consecutive values of low/high HR to raise an alarm. Figure 4.3 shows the way the implementation works. Three different states are defined: s0 – normal heart rate, s1 – high heart rate, s2 – low heart rate. Getting from one state to another is done through a pseudostate. When the entry conditions are fulfilled, a message is transmitted to the subscribers with the required information (in the figure that is represented by the current heart rate and the state code). When the exit conditions of a state are fulfilled, the transition to a pseudostate happens. Here some required variables are updated and the entry conditions of all the states are evaluated.

Variations in the heart rate  Heart rate variability (HRV) is a measure of variation in heart rate and refers to the variation in the beat-to-beat interval. It is an important predictor of mortality after myocardial infarction. Reduced HRV appears to be a marker of fatal ventricular arrhythmia. There are over 26 different types of arithmetic manipulations of R-R intervals used in literature to represent HRV such as [13]:

- the standard deviations of the normal mean R-R interval obtained from successive 5-minute periods over 24-hour Holter recordings (called the SDANN index);
- the number of instances per hour in which two consecutive R-R intervals differ by more than 50 msec over 24-hours (called the pNN50 index);
- the root-mean square of the difference of successive R-R intervals (the rMSSD index);
- the difference between the shortest R-R interval during inspiration and the longest during expiration (called the MAX-MIN, or peak-valley quantification of HRV);
- the base of the triangular area under the main peak of the R-R interval frequency distribution diagram obtained from 24-hour recording etc.

Literature [13] shows that there is no evident distinction between the many different manipulations. The pNN50 index can easily be calculated so it was chosen for the classification. The value of pNN50 can be used to detect a number of pathologies but mentioning the exact values that cause problems is outside the scope of this research.
4.1. ECG DATA ANALYSIS

(a) Real sampled ECG Signal
(b) Noise removal ECG Signal
(c) Linear combination of first and second derivative
(d) Moving average
(e) Initial signal, detected QRS complexes and Heart Rate

Figure 4.2: Filtering steps and final results of the algorithm
Figure 4.3: If only the ECG sensor is deployed, the heart rate values are the only ones used to detect tachy/bradycardia. While the algorithm gives good results on the nodes and on reference data (MIT Arrhythmia Database) raising an alarm only after 1 high/low HR value will probably cause many false alarms (every time strong noise is recorded for eg). That is why we require at least 4 consecutive values of low/high HR to raise an alarm.
4.1. ECG DATA ANALYSIS

4.1.3 Event Condition Action (ECA) model for ECG Signal Classification and OSAS implementation

The algorithm presented in Section 4.1.2 was mapped on the ECA model and implemented in OSAS. A few concepts are used in the execution model: services containing event generators and handlers (also called actions), and subscriptions. Nodes expose services on the network that represent a handler call interface and/or an event interface. This section analyzes the service running on the ECG node and the subscription needed to subscribe to the results. Because the ECA model is intuitive, the exact behavior of the events is explained only for this application in detail. For the other applications (herd monitoring and person activity monitoring) only a listing of the services is done.

The definition of the filterECG service takes a handler. A node that receives a message and knows the handler calls it with the arguments supplied. The CBA information is defined as well at this point. In this case the node on which the ECG sensor resides has NodeID() = 22. All the variables used for processing need to be defined at the beginning of the service.

```
service filterECG ($handler)
    for [Network | NodeID() == 22]
    define
        enoughvalues := 0  # counts the samples up to G=5 seconds
        sumx := 0  # sum of number of samples between two beats
        prev := 0  # previous position of beat
        nR_beat := 0  # number of beats
        maximold := 0  # previous maximum peak value
        maximnew := 0  # current maximum peak value
        fin[6] := 0  # squared result
        ecg[10] := 0  # last 10 ecg readings
        res[5] := 0  # result of smoothing_filter
        final[4] := 0  # absolute value of sum of fd and sd
        variance := 0
        count := 0
        condition := 0
        rate := 0
        position := 0
        current := 0

on event sampling when True do
    current := AccelZ();
    if current > 1000 then
        ecg[9] := current;
        count := count + 1;
        condition := 1
    else
        condition := 0
    fi
```

The event sampling is always triggered (as it answers to the condition True). Its action is to sample the ECG signal and discard the values smaller than 1000. These values are either outliers or correspond to a low battery status. If the sample is a good sample the value of condition = 1

```
    The event calculations is triggered when condition == 1, that is whenever a valid sample was read. Its actions process the data and calculate the variance.
```
on event calculations when condition==1 do
  # smooth data
  enoughvalues := enoughvalues+1;
  current := Time( )−prev;
  # first and second derivative
  # result squaring
  if final[3]<0 then
  fi;
  fin[5]:=Avg(final,4);
  variance := AvgSqr(fin, 6)

The event maxi is triggered when the new sample is valid and the last calculated amplitude value is larger than the current maximum. Its action is to update the maximum of the current period.

on event maxi when condition==1 && maximew < fin[5] do
  maximew := fin[5]

The event newR is triggered when the temporal, amplitude and variance conditions are fulfilled and the newest sample is considered to be the R–peak. Its action is to increase the value of the number of beats for the current period and update the sum of RR interval sizes. This event is part of the first level classification of sample as a R–peak, hence a beat.

on event newR when current>15 && 2∗fin[5]>maximold && variance> 1000 do
  nr_beat := nr_beat+1;
  sumx := sumx+current;
  prev := Time( )

The event calculaterate calculates the heart rate when the G period has passed and there was more than a beat during that period.

on event calculaterate when (nr_beat>0)&&(enoughvalues==500) do
  rate := (600∗nr_beat)/(sumx/10)

The event sending is used to transmit the ECG data for logging to the subscriber node and a virtual simulated node can subscribe to the event printing to get the value of the heart rate and print it.

on event sending when (count==10) do
  SendToSubscribers($handler, *ecg);
  count := 0
on event printing when (enoughvalues%100==0) do
  SendToSubscribers(print, rate)

The events creset and shiftall are used for updating the variables.
4.2. ACCELERATION DATA ANALYSIS

```plaintext
on event ereset when enough values == 500 do
  maxinold := maxinnew;
  maxinnew := 0;
  nr_beat := 0;
  enough values := 0;
  sumx := 0

on event shiftall when condition == 1 do
  ShiftLeft(res, 5, 1);
  ShiftLeft(ecg, 10, 1);
  ShiftLeft(final, 4, 1);
  ShiftLeft(fin, 6, 1)
```

A StoreService can be defined for running on a simulated node and receiving the information transmitted from the ECG node.

```plaintext
service StoreService($Handler)

for [Network | HasSysCall(Store)]
  define
    fileRaw := 0
    fileRate := 0

action StoreValue(rate) do
  if (fileRate == 0) then
    fileRate := OpenFile(100, 3)
  fi;
  Store(fileRate, rate);
  SyncFile(fileRate)

action StoreRawSamples() do
  if (fileRaw == 0) then
    fileRaw := OpenFile(1, 12)
  fi;
  i := self.handler_start + 1;
  while (i < self.handler_end - 1) do
    Store(fileRaw, self.message[i]);
    i := i + 1
  od;
  SyncFile(fileRaw)
```

A subscription Hrate to the StoreRawSamples action defined in the StoreService service will trigger the filterECG service.

```plaintext
subscription Hrate
  for [Network | HasSysCall(Store)]
  to filterECG($handler=StoreRawSamples)
  on [Network | NodeID() == 22]
  with (period=10ms, deadline=20s, send="Normal", exec="Normal")
```

4.2 Acceleration Data Analysis

Accelerometers provide instantaneous measurement of the acceleration (caused by movements) that is currently acting on the device. It varies an output voltage directly related to the magnitude of acceleration in a given direction. The changes in acceleration are given by movements so
the accelerometer provides information about the movements it makes. By data processing and analysis, the movements can be classified in different predefined classes. 3D accelerometers can detect dynamic changes of acceleration in all directions by using independent X, Y, and Z axes. The accelerometer generates an analog voltage for each axis that is relative to the acceleration force (in g units, where g is the Earth’s gravitational acceleration) parallel to that axis. The sensors used for this project have a range of ±3g on X, Y and Z directions.

There are multiple applications for which accelerometers can be used, as whenever an object is picked up or put down, moved or thrown away the acceleration involved can be detected and classified in an action. This research is interested in the usage of acceleration sensors for human movement classification and herd monitoring. Data obtained from acceleration sensors is classified into movements or positions of human and cows. The two applications are presented in the next sections.

4.2.1 Human movement classification with 3D acceleration data

Supervision of human movements can be applied in many fields, from the employees that take part in a production and assembly chain as in [40] or athletes doing exercise or recovering. Knowing the activity of patients and/or elderly persons is also imperative in supervising their quality of life and providing accurate and to-the-point feedback to improve their condition.

This research is concerned with human activity classification in the home environment. The following activity classes are defined: sitting, standing, laying, walking, running, falling, going up the stairs, going down the stairs.

We define a hierarchical tree for the classification (Figure 4.4). At the top level a distinction is made between the static and dynamic state and down the leaves the classification gets more detailed. Figure 4.4 gives an overview of the 3-level classification. Depending on the application for which the classification is used, it is as detailed as the third level or only determines whether the person is moving or not (as is the case when an ECG sensor is networked with an accelerometer to improve quality of heart supervision, see Section 4.2.2.3).
4.2. ACCELERATION DATA ANALYSIS

4.2.1.1 Sensor positioning

Figure 4.5: Accelerometer positioning for Activity Classification

The sensor is positioned on the hip, always in the same position (right side of the body, antenna up, see Figure 4.5). This is needed so that the thresholding is applied to the correct acceleration component. It can be overcome by checking for the position at startup every time. However, for a custom built sensor it is not hard to enforce such a rule. For measurements and testing the sensor is attached to a waist belt with Velcro. The sampling frequency is 50Hz.

4.2.1.2 Data Sets

The algorithms were tested on the following data sets: Figure 4.6, 4.7, 4.8, 4.9, 4.10.

4.2.1.3 Level I: Static and Dynamic

At the first level of the classification the distinction between the static and the dynamic state needs to be done. To do so, the values of the acceleration over 1.5 seconds are analyzed. Every 15 samples the variance of the acceleration is calculated on every axis. The average variance over the 1.5 seconds is compared against a predefined threshold $Th = 1000$. If the resulting value is larger than the threshold $Th$ in any of the three directions then the signal is classified as dynamic, otherwise it is classified as static.

The results of the Level I classification on the data set defined above (Section 4.2.1.2) can be seen in Figure 4.11, 4.12, 4.13, 4.14, 4.15. If the blue line is at 1000 then the supervised person moves, if it is at 0 the person is static.

ECA for Level I of Human Movement Detection

Level I can be easily transcribed in OSAS. A service $MovementMagnitude$ is defined together with the variables that are needed for processing. The event $getdata$ samples the three acceleration axes as long as less than 75 samples (1.5 seconds at 50 Hz) have been acquired. The event $calculate$
Figure 4.6: Acceleration data set I, different activities: sit (initial), stand up, sit down, stand up, sit down, lay back, sit, stand up, sit down (end)

Figure 4.7: Acceleration data set II, different activities: down(initial), stand up, 4 steps, turn left, 5 steps, turn 180 degrees right, 5 steps, turn right, 4 steps, turn 180 degrees left, sit down, turn left while down (end)
4.2. ACCELERATION DATA ANALYSIS

Figure 4.8: Acceleration data set III, different activities: stand (initial), 6 steps, open door, 3 steps, 8 stairs up, turn 180 degrees left, 8 stair down, 3 steps, close door, 6 steps, stand (end). During the transmission there was high packet loss.

Figure 4.9: Acceleration data set IV, different activities: stand (initial), open door, 3 steps, 8 stairs up, turn left 180 degrees, 8 stairs down, 3 steps, stand (end)
Figure 4.10: Acceleration data set V, different activities: stand(initial), 3 steps, 8 stairs up, 3 stairs down, fall down the stairs, walk.

Figure 4.11: Acceleration data set I, different activities: sit (initial), stand up, sit down, stand up, sit down, lay back, sit, stand up, sit down (end). Level I Classification: static or dynamic. During dynamic activities the result signal has amplitude 1000.
4.2. ACCELERATION DATA ANALYSIS

Figure 4.12: Acceleration data set II, different activities: down (initial), stand up, 4 steps, turn left, 5 steps, turn 180 degrees right, 5 steps, turn right, 4 steps, turn 180 degrees left, sit down, turn left while down (end). Level I Classification: static or dynamic. During dynamic activities the result signal has amplitude 1000.

Figure 4.13: Acceleration data set III, different activities: stand (initial), 6 steps, open door, 3 steps, 8 stairs up, turn 180 degrees left, 8 stair down, 3 steps, close door, 6 steps, stand (end). Level I Classification: static or dynamic. During dynamic activities the result signal has amplitude 1000.
CHAPTER 4. RESEARCH AND IMPLEMENTATION

Figure 4.14: Acceleration data set IV, different activities: stand (initial), open door, 3 steps, 8 stairs up, turn left 180 degrees, 8 stairs down, 3 steps, stand (end). Level I Classification: static or dynamic. During dynamic activities the result signal has amplitude 1000.

Figure 4.15: Acceleration data set V, different activities: stand (initial), 3 steps, 8 stairs up, 3 stairs down, fall down the stairs, walk. Level I Classification: static or dynamic. During dynamic activities the result signal has amplitude 1000.
calculates the variance on every axis every 15 samples and the event `meancalculate` calculates the average of these variance values every 1.5 seconds.

```plaintext
service MovementMagnitude ($handler, $complete)
for [Network | NodeID() == 21]
define
  logx[15] := []
  logy[15] := []
  logz[15] := []
  avx := 0
  avy := 0
  avz := 0
  avxarray[5] := 0
  avyarray[5] := 0
  avzarray[5] := 0

on event getdata when $complete < 75 do
  logx[14] := AccelX();
  logy[14] := AccelY();
  logz[14] := AccelZ();
  ShiftLeft(logx, 15, 1);
  ShiftLeft(logy, 15, 1);
  ShiftLeft(logz, 15, 1);
  $complete := $complete + 1

on event calculate when $complete%15==0 do
  avxarray[4] := AvgSqr(logx, 15);
  avyarray[4] := AvgSqr(logy, 15);
  avzarray[4] := AvgSqr(logz, 15);
  ShiftLeft(avxarray, 5, 1);
  ShiftLeft(avyarray, 5, 1);
  ShiftLeft(avzarray, 5, 1)

on event meancalculate when $complete==75 do
  avx := Avg(avxarray, 5);
  avy := Avg(avyarray, 5);
  avz := Avg(avzarray, 5);
  SendToSubscribers($handler, (avx > 1000 || avy > 1000 || avz > 1000));
  $complete := 76
```

**Level II: Activity classification**

At the second level the classification gets more detailed. Once the static state is detected, the person can be in one of the three sub-states: sitting, standing or laying. To distinguish between the three states we need to know the exact meaning of each of the acceleration components.

If the dynamic state is detected, then the person can be in one of the following states: running, walking, other. The distinction between running and walking can be done based on the variance level.

**Determining the current Static State**

Level I of the activity classification makes a distinction between the static case and dynamic case. If Level I determines that the patient is static, calculations need to be carried out to determine whether he is sitting, standing or laying. The calculations are based on a-priori knowledge of normal values for the three positions. The accelerometer measures only the gravitational acceleration when it is stationary. However, an accelerometer is not equally sensitive on all axes so a zero
acceleration does not necessarily give a zero sensor output. The standard values for (x,y,z) in the sitting, standing and laying case can be obtained at the deployment of the sensor by observing the patient and sending predefined subscriptions when he is in each of the three states. These subscriptions will retrieve the corresponding (x,y,z) values.

The (x,y,z) values that an accelerometer reads represent a vector in the 3D space. To show how the distinction between the laying and sitting case can be done let the calibration values for these positions be \((x_1, y_1, z_1)\) and \((x_2, y_2, z_2)\) respectively. For each set of new samples, the program calculates the distance between the new vector \((x,y,z)\) and the known calibration data. If the distance between \((x,y,z)\) and \((x_1, y_1, z_1)\) is smaller than the one between \((x,y,z)\) and \((x_2, y_2, z_2)\) then the person is laying. If the situation is the other way around the person is sitting (Figure 4.16). This reasoning is valid for the laying-standing and sitting-standing situations as well. If the person is in the laying state then the distance between the laying vector and the calibration vector is smaller than both the one between the standing and sitting vectors and their respective calibration vectors.

The distance to the laying vector is calculated as
\[
d_l = \sqrt{(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2}
\]
and the one to the sitting position as
\[
d_s = \sqrt{(x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2}.
\]

It should be mentioned that as the acceleration is not a proper kinematic acceleration, the distance also does not refer to the actual spatial distance but is rather a more abstract concept. Moreover, in OSAS the square root is not calculated but only the sum of the squared values is taken to avoid floating point calculations.

The results of the Level II classification of static positions on the data set defined above (Section 4.2.1.2) can be seen in Figure 4.17, 4.18, 4.19, 4.20, 4.21. If the blue line is at 500 then the supervised person is sitting, if it is at 700 the person is standing and if it is at 1000 the person is laying. Moreover, to eliminate the cases when the vertical acceleration is naturally lower after a step and might be considered as a standing position, the mean of the past five values for acceleration is calculated and introduced in the formula for calculating the distance instead of just using the last value.
Figure 4.17: Acceleration data set I, different activities: sit (initial), stand up, sit down, stand up, sit down, lay back, sit, stand up, sit down (end). Level II static: determine sit (amplitude 500), stand (amplitude 700) or lay (amplitude 1000)

Figure 4.18: Acceleration data set II, different activities: down (initial), stand up, 4 steps, turn left, 5 steps, turn 180 degrees right, 5 steps, turn right, 4 steps, turn 180 degrees left, sit down, turn left while down (end). Level II static: determine sit (amplitude 500), stand (amplitude 700) or lay (amplitude 1000)
CHAPTER 4. RESEARCH AND IMPLEMENTATION

Figure 4.19: Acceleration data set III, different activities: stand (initial), 6 steps, open door, 3 steps, 8 stairs up, turn 180 degrees left, 8 stair down, 3 steps, close door, 6 steps, stand (end). Level II static: determine sit (amplitude 500), stand (amplitude 700) or lay (amplitude 1000)

Figure 4.20: Acceleration data set IV, different activities: stand (initial), open door, 3 steps, 8 stairs up, turn left 180 degrees, 8 stairs down, 3 steps, stand (end). Level II static: determine sit (amplitude 500), stand (amplitude 700) or lay (amplitude 1000)
4.2. ACCELERATION DATA ANALYSIS

Figure 4.21: Acceleration data set V, different activities: stand(initial), 3 steps, 8 stairs up, 3 stairs down, fall down the stairs, walk. Level II static: determine sit (amplitude 500), stand (amplitude 700) or lay (amplitude 1000)

Determining the current Dynamic State

The activities that the person does while moving can be classified as well. The following actions are proposed: walking, falling, going up and going down the stairs. The test for determining any of these actions starts after it was determined that the person started moving.

The first distinction is done between normal movement and a fall. A fall is characterized by a very large change in acceleration. To test for this change it is enough to calculate the variance of the signal for every 5 samples and apply thresholding to these values. A sample result can be seen in Figure 4.22.

Figure 4.22: Acceleration data set VI, different activities: stand(initial), 3 steps, 8 stairs up, 3 stairs down, fall down the stairs, walk
CHAPTER 4. RESEARCH AND IMPLEMENTATION

After checking for a possible fall, the classification of the other activities can start. The algorithm can determine when a step has happened. The proposed classifier, however, cannot make a distinction between normal walking and climbing up the stairs. With the positioning of the accelerometer on the hip there is no perceptible difference between the two cases if only 7 samples are stored. To have a classification of walking down the stairs storing one second of data would be needed. This greatly increases the state of the program and was not implemented at this step.

For the climbing down the stairs case, the acceleration during the downward movement of each step is visibly affected so the distinction is possible.

The differentiation between walking and climbing up the stairs is done similar to determining the static state. Two vectors are defined, \( \text{walk} (x_{\text{walk}}, y_{\text{walk}}, z_{\text{walk}}) \) and \( \text{up} (x_{\text{up}}, y_{\text{up}}, z_{\text{up}}) \). If the distance from the current point to the \( \text{walk} \) vector, as defined in Section 4.2.1.3, is smaller than the one to the \( \text{up} \) vector then that peak is considered as being a normal walking step. Otherwise it is a climbing up the stairs step.

Before applying the distance calculation the signal is transformed in the following way.

Because the walking is done while the person is standing the standing vector is subtracted from the current measurement \( x_{\text{new}}(i) = x(i) - x_{\text{stand}}, y_{\text{new}}(i) = y(i) - y_{\text{stand}}, z_{\text{new}}(i) = z(i) - z_{\text{stand}} \) (\( x_{\text{stand}}, y_{\text{stand}}, z_{\text{stand}} \) defined before). Then, the standard deviation of each consecutive 7 samples is calculated. If the results are smaller than a threshold (\( Th = 40 \)) they are discarded and made zero. This eliminates the fine values that the sensor reads because of short trembles in the body. The next step is to determine the actual steps and climbing down the stairs. This is done by comparing the values to a threshold and detecting the peaks and then calculating the distance to the vectors \( \text{walk} \) and \( \text{up} \). The results for the two most relevant data sets can be seen in Figure 4.23 and 4.24.

![Figure 4.23](image)

**Figure 4.23:** Acceleration data set IV, different activities: stand (initial), open door, 3 steps, 8 stairs up, turn left 180 degrees, 8 stairs down, 3 steps, stand (end). Level II dynamic: determine step type, walking and climbing up the stairs (amplitude -100), climbing down the stairs (amplitude -200)

**Level III: Transitions classification** To get from a static state to another state the monitored person has to undergo some transition states. These can be sitting-to-standing, standing-to-
4.2. ACCELERATION DATA ANALYSIS

Figure 4.24: Acceleration data set V, different activities: stand(initial), 3 steps, 8 stairs up, 3 stairs down, fall down the stairs, walk. Level II dynamic: determine step type, walking and climbing up the stairs (amplitude -100), climbing down the stairs (amplitude -200)

sitting, laying-to-sitting and sitting-to-laying. They are based on the changes in the corresponding acceleration component. The algorithm can determine the moment when the person moves from one position to the other by checking the mode of the new position (sitting, standing, laying) and comparing it with the previous known mode before the movement. In Figures 4.25, 4.26, 4.27, 4.28, 4.29 the following codes are used for illustrating the results:

- 1000 for a standing-to-sitting movement
- 700 for a sitting-to-standing transition
- 400 for a sitting-to-laying transition
- 100 for a laying-to-sitting transition

As it can be seen in the figures, the detection of the transitions works well. The only mismatch is in Figure 4.28 where a few consecutive transitions are detected during a turn at the top of the stairs. This corresponds with the mismatch in the mode detection (sitting or standing) that could be seen previously. The discrepancy is probably caused due to the change in angle and an unexpected type of movement from the turn.

ECA for Human Movement

In this section only parts of the services will be mentioned. The decision instruments are packed inside functions. We define the following variables:
Figure 4.25: Acceleration data set I, different activities: sit (initial), stand up, sit down, stand up, sit down, lay back, sit, stand up, sit down (end). Level III: transition classification. Standing-to-sitting (amplitude 1000), sitting-to-standing transition (amplitude 700), sitting-to-laying (amplitude 400), laying-to-sitting transition (amplitude 100)

Figure 4.26: Acceleration data set II, different activities: down (initial), stand up, 4 steps, turn left, 5 steps, turn 180 degrees right, 5 steps, turn right, 4 steps, turn 180 degrees left, sit down, turn left while down (end). Level III: transition classification. Standing-to-sitting (amplitude 1000), sitting-to-standing transition (amplitude 700), sitting-to-laying (amplitude 400), laying-to-sitting transition (amplitude 100)
4.2. ACCELERATION DATA ANALYSIS

Figure 4.27: Acceleration data set III, different activities: stand (initial), 6 steps, open door, 3 steps, 8 stairs up, turn 180 degrees left, 8 stair down, 3 steps, close door, 6 steps, stand (end). Level III: transition classification. Standing-to-sitting (amplitude 1000), sitting-to-standing transition (amplitude 700), sitting-to-laying (amplitude 400), laying-to-sitting transition (amplitude 100)

Figure 4.28: Acceleration data set IV, different activities: stand (initial), open door, 3 steps, 8 stairs up, turn left 180 degrees, 8 stairs down, 3 steps, stand (end). Level III: transition classification. Standing-to-sitting (amplitude 1000), sitting-to-standing transition (amplitude 700), sitting-to-laying (amplitude 400), laying-to-sitting transition (amplitude 100)
Figure 4.29: Acceleration data set V, different activities: stand(initial), 3 steps, 8 stairs up, 3 stairs down, fall down the stairs, walk. Level III: transition classification. Standing-to-sitting (amplitude 1000), sitting-to-standing transition (amplitude 700), sitting-to-laying (amplitude 400), laying-to-sitting transition (amplitude 100)

The functions below can be used to detect each of the actions defined in theory. The function `transitionType()` detects the standing-to-sitting, sitting-to-standing, sitting-to-laying or laying-to-sitting transitions. The function `calculateLay()`, `calculateSit()`, `calculateStand()`, `calculateUp()` and `calculateDown()` determine the distance from the sit, stand, lay, up and down vectors to the current measured vector. The function `staticMode()` determines the static state in which the
4.2. ACCELERATION DATA ANALYSIS

person is while falling and determineStepType() check whether the person has fallen and the type of step, respectively.

```plaintext
function determineStepType() do
  if oldmode==7 &
  & nd[1]==5 then
    tr:=20 #standing to sitting
  else
    if oldmode==5 &
    & nd[1]==7 then
      tr:=30 #sitting to standing
    else
      if oldmode==5 &
      & nd[1]==10 then
        tr:=40 #sitting to laying
      else
        if oldmode==10 &
        & nd[1]==5 then
          tr:=50 #laying to sitting
        fi
      fi
    fi
  fi

function calculatesUp() do

function calculatesDown() do

function determineStepType() do
  ddown[1]:= calculatesDown();
  dup[1]:= calculatesUp();
  if Avg(ddown,2)<Avg(dup,2) &
  & x[6]<0 &
  & y[6]<0 &
  & z[6]<0 &
  & current-lastt>25
```
a := 2;
lasti := current
else
  a := 1;
lasti := current
fi
f1;
return a

on event init when sit[0] == 0 do
  sit[0] := 2299;
sit[1] := 2169;
stand[0] := 2367;
stand[1] := 2211;
lay[0] := 1871;
lay[1] := 2287

on event getData when True do
  x[6] := AccelX();
z[6] := AccelZ();
counting := counting + 1;
current := current + 1;
x4[6] := AvgSqr(x, 7);
y4[6] := AvgSqr(y, 7);
z4[6] := AvgSqr(z, 7);
  if counting == 100 then
    counting := 0;
  if moving == 0 then
    md[1] := staticMode();
    a := transitionType();
oldmode := md[1];
  if a != 0 then
    SendToSubscribers($handler, a)
  f1
  SendToSubscribers($handler, md[1])
f1
fi

on event processData when moving do
  meanx := Avg(x4, 7);
  meany := Avg(y4, 7);
  meanz := Avg(z4, 7);
  if meanx > 3200 && meany > 600 && meanz > 600 && current−lasti−last1 > 30 && oldmode == 7 then
    SendToSubscribers($handler, 1);
  lasti := current
else
  if (current%100 == 0) then
    SendToSubscribers($handler, 0)
  f1
fi

on event ShiftData when True do
  ShiftLeft(x, 7, 1);
  ShiftLeft(y, 7, 1);
  ShiftLeft(z, 7, 1);
  ShiftLeft(x4, 7, 1);
  ShiftLeft(y4, 7, 1);
  ShiftLeft(z4, 7, 1);
  ShiftLeft(md, 2, 1)
4.2. ACCELERATION DATA ANALYSIS

```latex
\begin{verbatim}
service StoreStatus($Handler)
    define
    fileId2 := 0

action StoreData(val1) do
    if (fileId2 == 0) then
        fileId2 := OpenFile(50,10)
    fi;
    Store(fileId2, val1);
    SyncFile(fileId2)

subscription Movement
    to DetectNode($Handler=StoreData)
    with (period=20ms, deadline=20s, send="Critical", exec="Critical")
\end{verbatim}
```

4.2.2 Herd Monitoring

Herd monitoring is another possible application of the acceleration sensor nodes. Though it is a very promising field, there is little research in the area. The objective of this part is to implement an algorithm that can determine in which of the five modes (defined by Deliverable6.3 of WASP and shown in Figure 4.33 and Figure 4.34) a certain cow is, as well as implementing an algorithm that counts the number of steps the cow performs during a fixed period of time.

The herd monitoring research as described in this thesis is carried out following the details given in WASP D6.3 Deliverable (also explained below). The research applies to herds of dairy cows that need to be monitored closely for possible claw problems. Lameness and other locomotion problems in dairy cows often occur in current dairy systems and should be given more attention from the animal welfare point of view. If a dairy cow develops mastitis which is not discovered in time, it has to be discharged earlier. This leads to important losses for the farm. If however the problems are discovered early they can be treated with a high success rate. Cows with claw problems have a changed walking pattern and tend to move less than normal. Therefore, observing the movement pattern and making statistics with the data could help at detecting the diseases in time.

The herd monitoring test bed is defined on issues that refer to the mobility of the cows. Nodes are attached to the legs and the heads of the cows and information about the locomotion, behavior and activity of the animals is logged. If any problems or deviations occur these can be signaled immediately so that the farmer can take measures. Acceleration sensors can be used to detect the activity level of a cow. By analyzing the patterns of the movement conclusions can be drawn about the health status of the animal: little movement throughout the day usually means claw problems. That is why it is important to check for movement at set interval times and make statistics with this data. A back-end system is used for such statistics but it is outside the scope of this research.

Figure 4.30 shows how the nodes are positioned at the leg of the cows. One node is connected to each of the rear legs of the animals and one node is connected at the neck. For the purpose of this research only information obtained from one leg node is used.

The cows are currently living in a farm building. The setup for the herd monitoring system can be seen in Figure 4.31.

The software structure for the on node processing has also been defined in WASP D6.3 Deliverable (Figure 4.32). The measurement module gathers 3D acceleration data at 10Hz for 1 second
CHAPTER 4. RESEARCH AND IMPLEMENTATION

Figure 4.30: Positioning of sensor node on the cows leg.

Figure 4.31: The farm setup. Positioning of the gateways in the test bed
every time it is called by the mode selector (explained below). The locomotion module is triggered by the activation of ModeIV from the mode selector. An internal timer module is used for timing the application.

Figure 4.32: The software structure for the on node processing has been defined in WASP D6.3 Deliverable. The measurement module gathers 3D acceleration data at 10Hz for 1 second every time it is called by the mode selector. The locomotion module is triggered by the activation of ModeIV from the mode selector. An internal timer module is used for timing the application.

WASP Deliverable6.3 also defines the different operating modes for the software (Figure 4.33 and Figure 4.34).

- Mode0 is used for initiation of the software and getting the calibration data.
- A transition to ModeI happens when the cow is laying down. While the node senses that the cow is in the laying position, the sampling of data is done every 120 seconds for a period of 1 second at 10Hz.
- A transition to ModeII happens when the cow is detected to be standing. During standing periods the chances that the cow is changing its position is higher so sampling is done once every 30 seconds for a 1 second interval with a sampling frequency of 10Hz.
- A transition to ModeIII happens when the cow starts walking. ModeIII is running continuously for as long as the cow is moving once every 5 seconds for 1 second intervals with a frequency of 10Hz.
- A transition to ModeIV happens in some situations. If the cow is walking and the walking pattern of the cow was not calculated in the last hour then a step detection algorithm is started. The sampling is done with a frequency of 50Hz and is considered valid only if at
least 5 steps are detected in a 10 seconds window. If not, the process is restarted if the cow is still walking.

<table>
<thead>
<tr>
<th>MODE O</th>
<th>MODE I</th>
<th>MODE II</th>
<th>MODE III</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interval: start/stop</td>
<td>Interval: 1 / 120 s</td>
<td>Interval: 1 / 30 s</td>
<td>Interval: 1 / 5 s</td>
</tr>
<tr>
<td>Duration: continue</td>
<td>Duration: 1 s</td>
<td>Duration: 1 s</td>
<td>Duration: 1 s</td>
</tr>
<tr>
<td>Frequency: 10 Hz</td>
<td>Frequency: 10 Hz</td>
<td>Frequency: 10 Hz</td>
<td>Frequency: 10 Hz</td>
</tr>
</tbody>
</table>

Figure 4.33: Operation modes for the herd monitoring software. Mode0 is used for initialization only. During initialization the program is installed on the nodes and the data for calibration is obtained. The calibration data is obtained during a period when the cow is standing. ModeI is defined as the period when the cow is laying down. The cow is in ModeII when it is standing but not moving. ModeIII corresponds to the period when a cow starts walking. ModeIV is triggered by ModeIII in some situations (Figure 4.34)

Figure 4.34: When ModeIII is detected, if the step morphology was not analyzed in the last hour ModeIV starts.

4.2.2.1 Calibration and Mode Detection

Method I

To be able to switch between modes, some calculations are carried out on the acceleration data obtained from the 3-axis accelerometer. The first calibration and mode detection is similar to the
4.2. ACCELERATION DATA ANALYSIS

While for the human application it is easier to impose a fixed position for the sensor, it is almost impossible to attach the node into a fixed position and use the same calibration data in all cases with the cows; that is why calibration needs to be done online. When the node is attached to the cow and the cow is standing still, a subscription is sent to retrieve the values of $x, y, z$. These are then considered to be the values of the standing position and can be used for calculations. The same approach is applied for the laying situation.

So the conditions for selection of the modes are:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode I</td>
<td>$d_l &lt; d_s$</td>
</tr>
<tr>
<td>Mode II</td>
<td>$d_s &lt; d_l$</td>
</tr>
<tr>
<td>Mode III</td>
<td>$d_s &lt; d_l$  and variance $&gt; threshold$</td>
</tr>
</tbody>
</table>

Figure 4.35 shows how the mode detection works. The vectors $L = (1943, 2365, 2123)$ and $S = (2067, 1951, 2427)$ are the calibration vectors for the laying and standing positions. If two vectors are chosen from the recorded acceleration data $P_1 = (1643, 2207, 2094)$ and $P_2 = (2301, 1907, 2366)$ the distances between these and the calibration vectors can be calculated. Those are $L_{P1} = 340$, $L_{P2} = 630$, $S_{P1} = 596$ and $S_{P2} = 245$. From these values it can therefore be concluded that point $P_1$ corresponds to a laying position and point $P_2$ corresponds to a standing position, which can be confirmed from the manual annotations that were brought to the signal during the acquisition.

**Figure 4.35:** Example of how mode detection actually works. The moments when the cow gets up can be easily identified with the degree calculations.

ECA for Method I

The service `modeDetectionCow` is used for detecting the Mode (I, II, III or IV) in which the software should run. Once a decision is made about the node, an internal message is sent through `sendMessage` for the appropriate software piece to start.

```plaintext
service modeDetectionCow ($handler, $x1, $y1, $z1, $x2, $y2, $z2)
for [Network] HasSysCall(accelX)
```
define
ax[10] := 0
ay[10] := 0
az[10] := 0
vector[6] := 0
idx := 0
variance[3] := 0
node := 0
angle := 0
distancevert := 0
distancehor := 0
d[6] := 0
idle_condition := 0

on event setSubscriber when True do
  if (self.period > 0) then
    SetProperty("Subscriber", 0, "Global")
  else
    SetProperty("Subscriber", self.subscriber, "Global")
  fi
on eventgetData when idx <= 0 do
  ax[idx] := AccelX();
  ay[idx] := AccelY();
  az[idx] := AccelZ();
  if (idx == 0) then
    self.period := 100ms
  fi;
  idx := idx+1
on event variance when idx == 10 do
  variance[0] := AvgSqr(ax,10);
  variance[1] := AvgSqr(ay,10);
  variance[2] := AvgSqr(az,10)

on event idle_class when True do
  idle_condition := (idx==10 &&
    variance[0]<40 &&
    variance[1]<40 &&
    variance[2]<40)

on event distancecalc1 when idle_condition do
  d[0] := vector[0]−Avg(ax,10);
  d[1] := vector[1]−Avg(ay,10);

on event distancecalc2 when idle_condition do

on event distancecalc3 when idle_condition do

on event distancecalc4 when idle_condition do

on event decideMode12 when idle_condition do
  if distancevert > distancehor then
    mode := 2;
    self.period := 15s
  else
    mode := 1;
    self.period := 20s
  fi;
  # Cancel subscription to step detection service
### 4.2. Acceleration Data Analysis

The distance between \( \mathbf{a} \) and \( \mathbf{a}' \) is calculated as: 
\[
\sum (a_i - a_i')^2
\]

Pseudocode:

```java
SendMessage(self, (subscription stepDetect
to stepsCowMode4($nodeID=StoreSteps, $init=0)
with {period=0ms, deadline=0ms, send="Normal", exec="Normal"}))
```

on event decidenode3 when idx==10 do
  if (node < 3) then
    mode := 3;
    # Invoke subscription on step detection service
    # with subscriber of this service.
    SendMessage(self, (subscription stepDetect
to stepsCowMode4($nodeID=StoreSteps, $init=1)
with {period=20ms, deadline=0ms, send="Normal", exec="Normal"}))
    self.period := 4s
  fi

on event resedata when idx==10 do
  idx := 0;
  SendToSubscribers($nodeID, NodeID(),
GetProperty("Hour", "Clock"),
GetProperty("Minute", "Clock"),
GetProperty("Second", "Clock"), mode)

action SetVector(nodeid, smode) do
  if (nodeid==NodeID()) then
    1 := smode==3-3;
    vector[1] := AccelX();
    vector[1+1] := AccelY();
    vector[1+2] := AccelZ();
  fi

Method II

While Method I works well for most of the situations, it can happen that the node moves around the leg of the cow or the cow bends the leg in the opposite direction when laying down. A more general approach can be taken to determine the calibration data and hence the modes. This method is an improvement of Method I.

The mode detection is based on variance and direction of acceleration. For modes with little variance, a vector is used to specify that mode. When the acceleration reading is close to that vector, that mode is selected. The vector calculations are done related to a base vector, which indicates the center of the acceleration reading when no motion occurs. These accelerations readings describe a circle if the node rotates around the leg of the cow.

Let us define vectors \( \mathbf{b} \), \( \mathbf{s} \), \( \mathbf{m} \). The computation of the vectors \( \mathbf{b} \) and \( \mathbf{s} \) in Figure 4.36 is based on an offline analysis. Vector \( \mathbf{m} \) represents the current acceleration measurement.

- Base-center vector: \( \mathbf{b} = (x_b, y_b, z_b) \)
- Stand vector: \( \mathbf{s} = (x_s, y_s, z_s) \)
- Measurement vector: \( \mathbf{m} = (x_m, y_m, z_m) \)

Let \( a \) be the sensitivity of the mode detection. By changing the value of \( a \) down, the measured vector can be more distant to the stand vector and still be considered as standing. By changing the value of \( a \) up, the standing position is more restricted.

The distance between \( \mathbf{b} \) and \( \mathbf{s} \) is calculated as: 
\[
\sum (b_i - s_i)^2
\]

And the distance between \( \mathbf{b} \) and \( \mathbf{m} \) is 
\[
((1-a) * x_b + a * x_s - x_m)^2 + ((1-a) * y_b + a * y_s - y_m)^2 + ((1-a) * z_b + a * z_s - z_m)^2.
\]

If the coordinates are translated to use the base vector \( \mathbf{b} \) as origin, then the distance between \( \mathbf{s}' \) and \( \mathbf{m}' \) is 
\[
(x_s' - x_m')^2 + (y_s' - y_m')^2 + (z_s' - z_m')^2
\]

and the distance between \( a * \mathbf{s}' \) and \( \mathbf{m}' \) is
Figure 4.36: Method II for determining the status of the cow. The mode detection is based on variance and direction of acceleration. For modes with little variance, a vector is used to specify that mode. When the acceleration reading is close to that vector, that mode is selected. The vector calculations are done related to a base vector, which indicates the center of the acceleration readings when no motion occurs. These accelerations readings describe a circle if the node rotates around the leg of the cow.

\[(a * x'_s - x'_m)^2 + (a * y'_s - y'_m)^2 + (a * z'_s - z'_m)^2\]. These values are easier to compare as they only depend on \(s\) and \(m\). Several factors from the equations only depend on \(s'\), which is constant and can be pre-calculated from data obtained during experiments. The other factors might cause 16-bit overflows, so bit-shifting has to be used for compensating. By using the proper value of \(a\), the complexity of the calculation with respect to integer operations, divisions and multiplications can be reduced.

ECA for Method II

```plaintext
service NodeDetection($handler)
for [Network]*|NodeType()=='MobileNode']
define
  ax[10] := []
  ay[10] := []
  az[10] := []

  # Callibaration of 3-axis acceleration sensor
  # What is the average value in each axis.
  basevec[3] := []

  # Vector from base vector that indicates standing position.
  # Can have default values. Could be set over the network.
  standvec[3] := []

  # Preprocessed data for vector comparison.
  standcmp := 0
```


4.2. ACCELERATION DATA ANALYSIS

```plaintext
idx := 0
mode := 0
isstand := 0
modetime[4] := [2, 137, 137, 137] # 20ms, 2m, 19s, 4s
idle_condition := 0

# VectorNear determines whether vector (x, y, z) is close to the target vector. The closeness indicates the distance, which is independent of the length of the vector.

# function CompareStand(x, y, z) do
  # return ( (standcmp - ((x-basevec[0])/16) * (standvec[0]/16)
  #          - ((y-basevec[1])/16) * (standvec[1]/16)
  #          - ((z-basevec[2])/16) * (standvec[2]/16)) < 0)

on event initVectors when basevec[0] == 0 do
  basevec[0] := 2305;
  basevec[1] := 2284;
  standvec[0] := 2287 - basevec[0];
  standvec[1] := 1827 - basevec[1];
  i := 0;
  standcmp := 0;
  while (i < 3) do
    j := standvec[i];
    if (j < 0) then
      j := 0 - j;
    fi;
    j := (j shr 4) * (j shr 4);
    standcmp := standcmp - (j shr 2) + j; # + (j*3)/4;
    i := i + 1
  od

on event setSubscriber when True do
  if (self.period > 0) then
    SetProperty("Subscriber", self.subscriber, "Global");
    SetProperty("Handler", $handler, "Global")
  else
    SetProperty("Subscriber", 0, "Global")
  fi

on event getData when idx <= 9 do
  ax[idx] := AccelX();
  ay[idx] := AccelY();
  az[idx] := AccelZ();
  if (idx == 0 && self.period > 0) then
    # sample at 10Hz
    self.period := 100ms
  fi;
  idx := idx + 1

on event idle_class when True do
  idle_condition := (idx == 10);
  if (idle_condition) then
    idle_condition := (AvgSqr(ax, 10) < 400 &&
                      AvgSqr(ay, 10) < 400 &&
                      AvgSqr(az, 10) < 400)
  fi

on event distancecalc2 when idle_condition do
  isstand := ((standcmp - ((Avg(ax, 10) - basevec[0])/16) * (standvec[0]/16)
              - ((Avg(ay, 10) - basevec[1])/16) * (standvec[1]/16)
              - ((Avg(az, 10) - basevec[2])/16) * (standvec[2]/16)) < 0);
  if (isstand) then
    # Standing
```

---

# Acceleration Data Analysis

idx := 0
mode := 0
isstand := 0
modetime[4] := [2, 137, 137, 137] # 20ms, 2m, 19s, 4s
idle_condition := 0

# VectorNear determines whether vector (x, y, z) is close to the target vector. The closeness indicates the distance, which is independent of the length of the vector.

# function CompareStand(x, y, z) do
  # return ( (standcmp - ((x-basevec[0])/16) * (standvec[0]/16)
  #          - ((y-basevec[1])/16) * (standvec[1]/16)
  #          - ((z-basevec[2])/16) * (standvec[2]/16)) < 0)

on event initVectors when basevec[0] == 0 do
  basevec[0] := 2305;
  basevec[1] := 2284;
  standvec[0] := 2287 - basevec[0];
  standvec[1] := 1827 - basevec[1];
  i := 0;
  standcmp := 0;
  while (i < 3) do
    j := standvec[i];
    if (j < 0) then
      j := 0 - j;
    fi;
    j := (j shr 4) * (j shr 4);
    standcmp := standcmp - (j shr 2) + j; # + (j*3)/4;
    i := i + 1
  od

on event setSubscriber when True do
  if (self.period > 0) then
    SetProperty("Subscriber", self.subscriber, "Global");
    SetProperty("Handler", $handler, "Global")
  else
    SetProperty("Subscriber", 0, "Global")
  fi

on event getData when idx <= 9 do
  ax[idx] := AccelX();
  ay[idx] := AccelY();
  az[idx] := AccelZ();
  if (idx == 0 && self.period > 0) then
    # sample at 10Hz
    self.period := 100ms
  fi;
  idx := idx + 1

on event idle_class when True do
  idle_condition := (idx == 10);
  if (idle_condition) then
    idle_condition := (AvgSqr(ax, 10) < 400 &&
                      AvgSqr(ay, 10) < 400 &&
                      AvgSqr(az, 10) < 400)
  fi

on event distancecalc2 when idle_condition do
  isstand := ((standcmp - ((Avg(ax, 10) - basevec[0])/16) * (standvec[0]/16)
              - ((Avg(ay, 10) - basevec[1])/16) * (standvec[1]/16)
              - ((Avg(az, 10) - basevec[2])/16) * (standvec[2]/16)) < 0);
  if (isstand) then
    # Standing
```
node := 2;
self.period := nodetime[node]
else
    # Not standing
    node := 1;
    self.period := nodetime[node]
fi;
# No activity: stop the step detection service
SetProperty("Mode", node, "Global")
on event decideNode3 when idx==10 && !idle_condition do
    if (node < 3) then
        # Perhaps detect the laying to walking transition.
        # It might indicate unrestfull laying.
        node := 3;
        # Invoke subscription on step detection service
        # with subscriber of this service
        SendMessage(self, (subscription stepDetect
to StepDetection($handler=0, $init=1)
            with (period=20ms, deadline=0m, send="Normal", exec="Normal")))
    fi;
    # Walking/activity. Determine the end of the activity.
    self.period := nodetime[node]
on event resetData when idx==10 do
    idx := 0;
    SetProperty("Mode", node, "Global");
    SendToSubscribers($handler, NodeID());
    GetProperty("Hour", "Clock");
    GetProperty("Minute", "Clock");
    GetProperty("Second", "Clock");
# The encoding of time values makes this function difficult to use.
    action SetModeTime(laytime, standtime, walktime) do
        nodetime[1] := laytime;
        nodetime[2] := standtime;
        nodetime[3] := walktime

4.2.2.2 Step detection

Every hour ModeIV is started to analyze the morphology of the cows movement. This means counting the number of steps. If 5 steps are not detected during the 10 seconds interval then the analysis is repeated.

Because the algorithm for step detection should run on the nodes in real time it has to be simple in terms of computation power used and memory needed. Two algorithms are presented that have provided accurate results for detecting the cow steps.

Algorithm I

The first algorithm (Figure 4.37) is based on analyzing the acceleration on all three axis AccelX, AccelY and AccelZ. To reduce the memory used for storing data, the normal form of the acceleration data is taken, that is the values of the acceleration on the three axis are added:

\[ accel(i) = AccelX(i) + AccelY(i) + AccelZ(i) \]

The median of the last 4 acceleration samples is then calculated. This ensures that no peaks are caused by a misreading of the acceleration sensor:

\[ med(i) = \text{median}(accel(i-3 : i)) \]
4.2. ACCELERATION DATA ANALYSIS

After the preprocessing, the peak detection can start. The moving average and the variance of the last 4 processed samples of the median $med$ are calculated and then fixed thresholding is applied based on the two values:

$$mov(i) = \text{average}(med(i - 3 : i))$$
and

$$var(i) = \text{variance}(med(i - 3 : i))$$

The time between steps can be used as indication as to how fast the cow is moving.

Figure 4.37: Step detection flow chart

Figure 4.38 shows a part of the acceleration signal used for testing. In Figure 4.39 the results of the step detection algorithm can be seen.

ECA for Algorithm I of Cow Step Detection

The first algorithm for step detection is described above. A service $steps\_\text{CowMode4}$ is defined to determines the position and number of steps for a period of time.

```
service steps\_\text{CowMode4} \{ $handler \}
for [\text{Network} * | \text{NodeID}() == 9]
define
ax := 0
ay := 0
az := 0
```
Figure 4.38: Acceleration signal as used for step detection.

Figure 4.39: Result of the step detection algorithm on the signal in Figure 4.38
4.2. ACCELERATION DATA ANALYSIS

Algorithm II

The second algorithm (Figure 4.40) is a lot simpler in terms of memory consumption and processing but was not extensively tested on a large dataset. The algorithm considers only the acceleration on the X axis and disregards the Y and Z axis. This means that the nodes have to be attached in a fixed position every time and it should be ensured that they do not move while functioning. An alternative would be to automatically detect which is the good axis for analysis every time the

```plaintext
prev := 0
enoughsamples := 0
stepindex := 0
step := 0
max := 0
variance := 0
cond := 0
sizebtwsteps[15] := 0

on event meanfilter when enoughsamples < 500 do
  ax := AccelX();
  ay := AccelY();
  az := AccelZ();
  initialsamples[3] := ax + ay + az;
  enoughsamples := enoughsamples + 1;
  medianarray[3] := Median(initialsamples, 4)

on event movingsverage when enoughsamples < 500 do
  meanarray[3] := Avg(medianarray, 4);
  variance := AvgSqr(meanarray, 4)

on event calculatemax when enoughsamples < 500 do
  if meanarray[3] > max then
    max := meanarray[3]
  fi

on event calculatecondition when True do
  cond := enoughsamples < 500 && 10 * meanarray[3] > 7 * max && (enoughsamples - prev >= 40) && variance > 6000

on event getsteps when cond > 0 do
  step := step + 1;
  prev := enoughsamples;
  stepindex := enoughsamples;
  sizebtwsteps[step - 1] := stepindex

on event finishedsamples when True do
  if enoughsamples <= 500 then
    SendToSubscribers($handler, stepindex, initialsamples[3], variance)
  fi

on event updatemean when enoughsamples < 500 do
  ShiftLeft(initialsamples, 4, 1);
  ShiftLeft(medianarray, 4, 1);
  ShiftLeft(meanarray, 4, 1)

on event refreshsamples when enoughsamples == 500 do
  enoughsamples := 0;
  step := 0;
  prev := prev - 500;
  max := meanarray[3]

on event updatemovingsverage when enoughsamples < 500 do
  stepindex := 0
```
step detection starts. This would require additional processing.

![Step detection flow chart]

The algorithm first calculates the sum of two consecutive samples of the AccelX signal. This will ensure and enhancement of the peaks.

\[ fd(i) = (AccelX(i) + AccelX(i-1)) \]

Then, the maximum of the last six values of \( fd \) is calculated:

\[ \text{max}(i) = \text{maximum}(fd(i-5:i)) \]

After this preprocessing, the peaks can be detected based on thresholding values calculated offline from signals read apriori. Although the thresholds are empirical, the results may be useful to classify other walking patterns as well. Thresholds can also be calculated online every time the sensor is deployed.

The thresholds refer to the time between steps, the amplitude of the maximum and the difference between two consecutive maximum values.

**ECA for Algorithm II of Cow Step Detection**

The service `stepsCow` provides a simpler alternative to the previous algorithm. It consists of only two processing steps and classification based on temporal and amplitude conditions.

```java
for [Network | NodeID() == 7]
```
4.2. ACCELERATION DATA ANALYSIS

```plaintext
sd[2] := []
lasti := 0
ax[2] := []
icurrent := 0

function maximum() do
  i := 1;
  m := fd[0];
  if i <= 5 then
    if fd[i] > m then
      m := fd[i];
      i := i + 1
    fi
  fi
  return (m)

on event readdata when icurrent < 10000 do
  ax[1] := AccelX();
  icurrent := icurrent + 1;
  sd[1] := maximum();
  if icurrent - lasti > 20 && sd[1] > 5800 && sd[1] - sd[0] > 300 then
    SendToSubscribers($handler, 1000, ax[1], sd[1], sd[1] - sd[0]);
    SendToSubscribers(print, 1000, ax[1], sd[1], sd[1] - sd[0]);
    lasti := icurrent
  else
    SendToSubscribers($handler, 0, ax[1], sd[1], sd[1] - sd[0]);
    SendToSubscribers(print, 0, ax[1], sd[1], sd[1] - sd[0]);
    ShiftLeft(fd, 6, 1);
    ShiftLeft(sd, 2, 1);
    ShiftLeft(ax, 2, 1)
  fi
end

on event finishedsamples when icurrent == 10000 do
  icurrent := 0;
  lasti := 10000 - lasti
end
```

4.2.2.3 Networking sensors: ECG and Acceleration Data Analysis

This section treats a special application, the networking of an ECG sensor with an accelerometer. The ECG signal alone cannot provide a full interpretation of the health condition of the patient, so along with the knowledge of heart rate values, it is also important to know the activity being performed by the monitored patient. If the patient is sitting and a sudden increase in the heart rate happens then most probably a tachycardia signal should be generated as the increase in the heart rate is medical. If however, the subject is doing exercise, the parameters of tachycardia diagnosis change. That is why it is important to provide the ECG activity classifier with information about the activity level of the person as well so an acceleration sensor is used for this purpose. The classification method presented here is based not only on information from the ECG sensors but also on input from an acceleration sensor.

The ECG sensor that calculates the heart rate is networked with an acceleration sensor for detecting the activity levels of the patient (Figure 4.41). The information about the level of physical activity that the person is performing (no or little activity and intense activity) can be used in multiple ways. For example, it can be used as a trigger for starting the ECG monitoring to make sure the heart rate stays in normal parameters when the patient is exercising or sleeping. It can also be used for filtering out some of the alarms raised by the ECG sensor as normal limits for the heart rate increase when moderate or intense movement is carried out without posing threat
CHAPTER 4. RESEARCH AND IMPLEMENTATION

Figure 4.41: Using multiple sensor input for activity recognition

to the health of the patient.

The values of 60 to 100 bmp for the heart rate are the normal values for an average person doing no or little physical activity. The normal parameters are not fixed but depended on the individual and on the activity that (s)he is doing. Along with the knowledge of heart rate values, the usage of an accelerometer gives information about the activity level of the monitored patient.

Classification Method

When an abnormal condition is discovered the ECG sensor requests information about the level of activity from the acceleration sensor. The acceleration sensor detects the level of movement by thresholding on the variance of the signal over the last 1.5 seconds; it determines whether the person is static, does light or strong movements and then transmits to the ECG sensor whether there is intense exercise associated to the heart rate. According to the reply the ECG sensor raises an alarm or not.

The following states for a system with an ECG and an acceleration sensor networked together are defined:

\[
\begin{align*}
  s_0 &= \text{normal heart rate} \quad 60 < \text{rate} < 100 \\
  s_1 &= \text{normal heart rate exercise} \quad 60 < \text{rate} < 120 \\
  s_2 &= \text{high heart rate exercise} \quad \text{rate} > 120 \\
  s_3 &= \text{high heart rate no exercise} \quad \text{rate} > 100 \\
  s_4 &= \text{low heart rate} \quad \text{rate} < 60
\end{align*}
\]

The acceleration sensor detects the level of movement (The approach is described in Section 4.2.1.3, Level I of the decision tree; as here we are interested in intense movement rather than just movement, the threshold for the variance is raised to 5000). It then transmits this level to the ECG sensor.

If the ECG sensor is deployed together with the acceleration, the classifier takes into consideration both aspects when making a decision. While the algorithm gives good results on the nodes and on reference data (MIT Arrhythmia Database) raising an alarm only after 1 high/low HR
4.2. ACCELERATION DATA ANALYSIS

value will probably cause many false alarms (every time strong noise is recorded for eg). That is why we require at least $nr = 4$ consecutive values of low/high HR to raise an alarm (Figure 4.42).

ECA for Network of ECG and Acceleration Sensor Node

The $MovementMagnitude$ service is similar to the one presented in 4.2.1.3 so is not repeated here. The filtering of the ECG and calculation of the heart rate was also presented before. Only the new part of the classification related to the communication between the two sensors is listed below.

There are four different states in which the system can be. These also correspond to the messages that get transmitted from the classifier on the ECG sensor node.

| # 0 = normal heart rate $60 < rate < 100$ |
| # 1 = normal heart rate exercise: $60 < rate < 120$ |
| # 2 = high heart rate exercise: $rate > 120$ |
| # 3 = high heart rate no exercise: $rate > 100$ |
| # 4 = low rate $rate < 60$ |

The $forwardprocessaccel$ does the transmission of the information about the level of danger as defined by the four different states in which the system can be.

```java
forward processaccel(value)

on event lowrate when (enoughvalues==500) && rate<60 do
  counthearthrates :=0;
  SendToSubscribers(print, rate, 4)

on event normalrate when (enoughvalues==500) && rate>60 && rate<100 do
  counthearthrates :=0;
  SendToSubscribers(print, rate, 0)

on event unsubscribe when (enoughvalues==500) && rate>60 && rate<100 do
  SendMessage([Network]+[NodeID()]==21], {subscription movement to MovementMagnitude($handler=processaccel$, $complete=0)
  with (period=0ms, deadline=0m, send="High", exec="High")})

on event e7 when True do
  if (nr_beat>0)&&enoughvalues==500&&sumx>0 then
    rate := (600+nr_beat)/(sumx/10)
  fi;
  if rate > 100 then
    counthearthrates:=counthearthrates+1
  fi

on event highevent when (enoughvalues==500) && rate>100 do && counthearthrates>=4
  SendMessage([Network]+[NodeID()]==21], {subscription movement to MovementMagnitude($handler=processaccel$, $complete=0)
  with (period=20ms, deadline=0m, send="High", exec="High")})

on event accelerating when (enoughvalues==500) do && counthearthrates>=4
  if (rate<120) && movementstate then
    SendToSubscribers(print, rate, 1)
  else if (rate>120) && movementstate then
    SendToSubscribers(print, rate, 2)
  else if (rate>100) && movementstate==0 then
    SendToSubscribers(print, rate, 3)
  fi
```
CHAPTER 4: RESEARCH AND IMPLEMENTATION

Figure 4.42: If only the ECG sensor is deployed together with the accelerometer, the classifier takes into consideration both aspects when making a decision. While the algorithm gives good results on the nodes and on reference data (MIT Arrhythmia Database) raising an alarm only after 1 high/low HR value will probably cause many false alarms (every time strong noise is recorded for eg). That is why we require at least 4 consecutive values of low/high HR to raise an alarm.
Adapting processing for patient monitoring

Prolonging the lifetime of the wireless devices is important. In practice, the wireless transmission of information is the activity that consumes the most energy and the processing of data is orders of magnitude less expensive. Wireless link usage can be reduced by using an efficient MAC protocol, doing cross-layer optimizations, or using on-node processing to decrease the amount of communicated data.

Figure 4.43: Sample communication to base station during logging and on-node processing

Depending on the current situation of the patient, different data needs to be communicated to the base station. In the logging case, every sensed sample needs to be communicated (Figure
4.43(a)). With on-node processing, the transmitted amount of data can be reduced by adapting the transmission based on the patient situation.

The traditional approach of transmitting the information to a gateway for extensive processing can sometimes be better than processing on the nodes due to the resource constraints of the sensors. A normal use of a ECG supervision would be after the patient has been subject to a surgery. During the first days more careful supervision should be employed so more or all data could be sent to the base station for accurate processing. For example, if the doctor needs to see the activity of the heart for the last 1 hour every time an alarm is raised or needs a high frequency sampled ECG signal (usually above 200 Hz) in order to detect particularities of the heart, then the processing capabilities of the node are not sufficient and the memory is insufficient to store all the raw data. So sending all the information through the wireless link and logging the data on an external computer is the best way to solve this problem.

The next day however, the doctor can decide that only the heart rate needs to be logged so the processing algorithm is deployed on the sensor node. The only communication to the base station is the estimation of the heart rate every $G$ seconds (Figure 4.43(b)). For an application that needs to calculate the heart rate of the patient at specific intervals a sampling rate of just 100 Hz is enough and for this only few data needs to be saved at a time (in terms of memory usage, the presented approach needs to store only 3 samples of the raw ECG signal and 11 samples of filtered data). Moreover, it is also possible to sample the data and calculate the HR periodically (for example, 20 seconds every half an hour) instead of continuously. The concerns related to the resource constraints of the nodes can then be partly eliminated; processing on the nodes on such a situation is a viable alternative.

Depending on the current situation of the patient or on rules determined by the doctor, different data needs to be processed and/or communicated to the base station. If the supervision gets prolonged, the doctor can decide that only the irregular beats need to be sent to the base station. When such irregular conditions are detected, the patient can also be immediately notified. If the ECG sensor is networked with an acceleration sensor as described in Section 4.2.2.3, for a perfectly healthy patient the energy consumption for communication can be reduced to almost zero (if no irregular beats ever happen no transmission happens). There is of course still the communication between the ECG sensor and the acceleration sensor, but this is negligible compared to the consumption of logging.
Chapter 5

Findings and implications

This chapter discusses the feasibility of using on-node processing for activity classification in Section 5.1. In the next sections an analysis of the ECG processing algorithm in terms of sensitivity, predictivity, data transmission and energy consumption is performed. Section 5.1.1 shows the results of the ECG algorithm with respect to a standard reference in literature, the MIT-Arrhythmia Database. Section 5.1.2 compares logging and on node processing in terms of the amount of data transmissions. Section 5.1.3 compares the energy improvements when doing on node ECG processing rather than transmitting the data for logging.

5.1 Feasibility of on-node processing for activity classification

One of the advantages of wireless sensors is the possibility of doing the monitoring without interfering with every day activities. The usage of battery operated wireless devices is unobtrusive as long as the battery does not need to be changed very often. Usually nodes are only used for sensing the environment and transmitting the information to an external device for doing the processing. The transmission of all the information through the wireless link is the main energy consumer and leads to a faster depletion of the batteries. Changing the battery every few hours is not desired so an alternative to the transmission of information should be found. Processing usually consumes orders of magnitude less energy than communication so it is a logical option to be considered for battery savings. Besides energy efficiency, on-node processing moves the classification closer to the monitored phenomena and allows a faster signaling of dangerous situations if it is the case.

Moving the processing from a more powerful external device to the sensor node requires creating a simple and efficient algorithm that does not exceed the capabilities of the node. This research has shown that it is possible to use and ECG and an acceleration sensor for monitoring patients and herds of cows. The analysis is not extended to the other use cases, however similar results are expected.

5.1.1 Quality of the ECG Algorithm

To determine the accuracy of the algorithm, it was tested on the MIT-Arrhythmia Database [2], a collection of 48 real ECG signals of 30 minutes each that have been annotated by hand by
CHAPTER 5. FINDINGS AND IMPLICATIONS

specialists. This database is a standard reference in literature and it is used here for testing purposes only. The signals in the database are named record\textsubscript{x}, where \(x\) is a number between 100 and 300 (not all numbers are represented). The signals in the MIT-Arrhythmia database are originally sampled at 360Hz but were downsampled to 90Hz.

Similar to literature, TP is defined as the number of true positive beats that were detected, FP the number of false positive or false beats and FN the number of false negatives or missed beats. The sensitivity is then \(Se = \frac{TP}{TP + FN}\) and the positive predictivity is \(P = \frac{TP}{TP + FP}\). The average on all signals was \(Se = 0.97993\) and \(P = 0.9845\). The list with all the records and the corresponding results can be seen in Table 5.1.

The sensitivity for 35 of the signals is above 0.99. For record 108 the sensitivity is \(Se = 0.8517\) and for record 207 it is \(Se = 0.7482\) which has an impact on the final result. The results can be explained by the irregularities in the records that behaved badly and the different thresholds that are needed for those signals. The morphologies of the signal 108 is atypical at parts with very short QRS complexes and through the downsampling to 90Hz some narrow peaks are skipped. During the acquisition of signal 207 the patient suffered of atrial fibrillations. While simple atrial fibrillations can be detected easily, in the stronger fibrillation cases the P waves are skipped completely and the beats can happen more often. The distinction between the different beats for such a period is hard to make hence the unsatisfactory results. Such type of results are common for this particular situation.

Though the MIT-Arrhythmia database is a common reference in literature, most of the published papers do not present their results on all the data set but only on selected signals. Therefore it is hard to make a realistic comparison of the different approaches. Even more, in literature all the signals are used in their original higher frequency state at 360Hz however we downsample our signals before testing to 90Hz so some loss in signal quality can happen. 90Hz was chosen rather than 100Hz as the downsampling could be easier done by ignoring samples of the signal without the need for interpolation. A better approach might be averaging consecutive values but this was not tested.

Table 5.2 shows performance numbers of other algorithms as presented in the cited papers. While some of the algorithms go deeper in classifying the beats as normal or premature contractions rather than just normal classification, the results are comparable.

5.1.2 Data transmission: logging and on node processing

The energy on a sensor node is mainly consumed by three elements: the sensor (sampling the sensor value), the microprocessor (processing) and the wireless link (communication). Energy consumption should be optimized at all levels, though the communication is usually an order of magnitude more expensive than computation. The case of remote processing is compared with local processing in terms of energy consumption and discussion is carried on about possible situations when trade-offs can be made.

The power required for sensing one sample depends on the power needed by the circuitry and the energy required by the sensor. Whether the processing is done on the node or the data is sent to an external device, the sensing power needed is the same for the same type of node and the same sampling frequency \(sr\). Because the algorithm needs a fixed sampling rate, any improvement that can be brought at this level in terms of energy consumption is by changing the sensor type.

The power consumed by the processor is directly related to its clock, code to be executed, du-
5.1. FEASIBILITY OF ON-NODE PROCESSING FOR ACTIVITY CLASSIFICATION

<table>
<thead>
<tr>
<th>Record number</th>
<th>Sensitivity</th>
<th>P</th>
<th>True positives</th>
<th>False negatives</th>
<th>False positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record: 100</td>
<td>1</td>
<td>0.999557</td>
<td>2255</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Record: 101</td>
<td>0.998379</td>
<td>0.988235</td>
<td>1848</td>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>Record: 102</td>
<td>0.999539</td>
<td>1</td>
<td>2169</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Record: 103</td>
<td>1</td>
<td>0.999517</td>
<td>2068</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Record: 104</td>
<td>0.973768</td>
<td>0.921267</td>
<td>2153</td>
<td>58</td>
<td>184</td>
</tr>
<tr>
<td>Record: 105</td>
<td>0.999217</td>
<td>0.960105</td>
<td>2551</td>
<td>2</td>
<td>106</td>
</tr>
<tr>
<td>Record: 106</td>
<td>0.956759</td>
<td>0.996377</td>
<td>1925</td>
<td>87</td>
<td>7</td>
</tr>
<tr>
<td>Record: 107</td>
<td>0.999528</td>
<td>0.994836</td>
<td>2119</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>Record: 108</td>
<td>0.851746</td>
<td>0.895846</td>
<td>1488</td>
<td>259</td>
<td>173</td>
</tr>
<tr>
<td>Record: 109</td>
<td>0.995619</td>
<td>0.993641</td>
<td>2500</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>Record: 111</td>
<td>0.999525</td>
<td>0.995274</td>
<td>2106</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Record: 112</td>
<td>1</td>
<td>0.993297</td>
<td>2519</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>Record: 113</td>
<td>0.993824</td>
<td>1</td>
<td>1770</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>Record: 114</td>
<td>1</td>
<td>0.993081</td>
<td>1866</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>Record: 115</td>
<td>0.999484</td>
<td>1</td>
<td>1937</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Record: 116</td>
<td>0.992481</td>
<td>0.9979</td>
<td>2376</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>Record: 117</td>
<td>0.905388</td>
<td>0.994946</td>
<td>1378</td>
<td>144</td>
<td>7</td>
</tr>
<tr>
<td>Record: 118</td>
<td>1</td>
<td>0.974569</td>
<td>2261</td>
<td>0</td>
<td>59</td>
</tr>
<tr>
<td>Record: 119</td>
<td>0.993912</td>
<td>0.9949</td>
<td>1959</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>Record: 121</td>
<td>0.962121</td>
<td>0.992741</td>
<td>1778</td>
<td>70</td>
<td>13</td>
</tr>
<tr>
<td>Record: 122</td>
<td>1</td>
<td>0.999593</td>
<td>2455</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Record: 123</td>
<td>0.958804</td>
<td>1</td>
<td>1443</td>
<td>62</td>
<td>0</td>
</tr>
<tr>
<td>Record: 124</td>
<td>0.959527</td>
<td>0.997411</td>
<td>1541</td>
<td>65</td>
<td>4</td>
</tr>
<tr>
<td>Record: 200</td>
<td>0.997675</td>
<td>0.910216</td>
<td>2575</td>
<td>6</td>
<td>254</td>
</tr>
<tr>
<td>Record: 201</td>
<td>0.955247</td>
<td>1</td>
<td>1857</td>
<td>87</td>
<td>0</td>
</tr>
<tr>
<td>Record: 202</td>
<td>0.968912</td>
<td>0.997092</td>
<td>2057</td>
<td>66</td>
<td>6</td>
</tr>
<tr>
<td>Record: 203</td>
<td>0.987817</td>
<td>0.907932</td>
<td>2919</td>
<td>36</td>
<td>296</td>
</tr>
<tr>
<td>Record: 205</td>
<td>0.998483</td>
<td>0.99962</td>
<td>2632</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Record: 207</td>
<td>0.748273</td>
<td>0.960111</td>
<td>1733</td>
<td>583</td>
<td>72</td>
</tr>
<tr>
<td>Record: 208</td>
<td>0.902489</td>
<td>0.988055</td>
<td>2647</td>
<td>286</td>
<td>32</td>
</tr>
<tr>
<td>Record: 209</td>
<td>1</td>
<td>0.989062</td>
<td>2984</td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td>Record: 210</td>
<td>0.990487</td>
<td>0.976735</td>
<td>2603</td>
<td>25</td>
<td>62</td>
</tr>
<tr>
<td>Record: 212</td>
<td>1</td>
<td>0.988048</td>
<td>2728</td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td>Record: 213</td>
<td>0.995971</td>
<td>0.998757</td>
<td>3214</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>Record: 214</td>
<td>0.993764</td>
<td>0.995904</td>
<td>2231</td>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>Record: 215</td>
<td>1</td>
<td>0.995231</td>
<td>3339</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>Record: 217</td>
<td>0.997718</td>
<td>0.996354</td>
<td>2186</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>Record: 219</td>
<td>0.999532</td>
<td>1</td>
<td>2135</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Record: 220</td>
<td>1</td>
<td>0.990452</td>
<td>2030</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Record: 221</td>
<td>0.990452</td>
<td>0.997075</td>
<td>2386</td>
<td>23</td>
<td>7</td>
</tr>
<tr>
<td>Record: 222</td>
<td>0.995538</td>
<td>0.991515</td>
<td>2454</td>
<td>11</td>
<td>21</td>
</tr>
<tr>
<td>Record: 223</td>
<td>0.996521</td>
<td>0.998838</td>
<td>2578</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>Record: 228</td>
<td>0.986248</td>
<td>0.902472</td>
<td>2008</td>
<td>28</td>
<td>217</td>
</tr>
<tr>
<td>Record: 230</td>
<td>1</td>
<td>0.996881</td>
<td>2237</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Record: 231</td>
<td>0.99357</td>
<td>1</td>
<td>1555</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Record: 232</td>
<td>0.997735</td>
<td>0.988222</td>
<td>1762</td>
<td>4</td>
<td>21</td>
</tr>
<tr>
<td>Record: 233</td>
<td>0.996073</td>
<td>0.991208</td>
<td>3044</td>
<td>12</td>
<td>27</td>
</tr>
<tr>
<td>Record: 234</td>
<td>0.998902</td>
<td>1</td>
<td>2729</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>0.970933646</td>
<td>0.984504188</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 5.2: Comparison of the sensitivity and predictivity of the ECG algorithm on the MIT-Arrhythmia database with results from other papers

<table>
<thead>
<tr>
<th>Paper</th>
<th>Se</th>
<th>P</th>
<th>Dataset and comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>[20]</td>
<td>99.69%</td>
<td>99.7%</td>
<td>ignored parts of 207 during atrial fibrillation; usage of 12 ECG samples and 79 data points for filtering at once</td>
</tr>
<tr>
<td>[42]</td>
<td>95.85%</td>
<td>-</td>
<td>entire dataset</td>
</tr>
<tr>
<td>[47]</td>
<td>98.55%</td>
<td>-</td>
<td>only selected certain parts of the signals</td>
</tr>
<tr>
<td>[12]</td>
<td>99.3%</td>
<td>-</td>
<td>only selected very short parts of 10 signals</td>
</tr>
<tr>
<td>[5]</td>
<td>90%</td>
<td>-</td>
<td>different algorithms, highest value</td>
</tr>
<tr>
<td>[6]</td>
<td>98.7%</td>
<td>-</td>
<td>selected segments of the database</td>
</tr>
<tr>
<td>[39]</td>
<td>98.5%</td>
<td>-</td>
<td>entire dataset</td>
</tr>
<tr>
<td>[1]</td>
<td>90%</td>
<td>-</td>
<td>entire dataset</td>
</tr>
<tr>
<td>[14]</td>
<td>99.05%</td>
<td>-</td>
<td>entire dataset</td>
</tr>
<tr>
<td>[24]</td>
<td>99.68%</td>
<td>99.59%</td>
<td>only on selected signals</td>
</tr>
<tr>
<td>own</td>
<td>97.993%</td>
<td>98.45%</td>
<td>entire dataset</td>
</tr>
</tbody>
</table>

ration of execution and any implementation of power saving techniques. In the remote processing case there is no processing done so there is no energy consumed on processing. When processing is done on the node, the energy consumed differs depending on the algorithm that is employed. A simpler algorithm, that requires few operations and little memory storage would use less energy than a more complicated one so the goal of this research was to implement a very simple algorithm (Section 4.1.2) that still provides good sensitivity for detecting the heart rate.

The wireless link power consumption depends on the energy required to run the electronics of the radio and the power required to transmit one bit towards the destination. The energy needed by the electronics is dependent on the radio itself and can be reduced by either changing the radio or employing power saving techniques at the MAC level (turning the radio on and off when no communication is needed, and adjusting the power depending on the distance to which the information needs to be transmitted).

Obviously, in the logging case, there is no processing done on the nodes so all sampled data needs to be transmitted towards the base station. This is very energy consuming. It is here that significant gain was expected of the algorithm as it reduces communication to just one packet every $G$ seconds. If a delay in transmitting this information is allowed the communication is decreased even further.

The amount of data that needs to be transmitted over the wireless link in the logging and on node processing cases can be estimated. The maximum packet size that can be transmitted is $M = Data + PO$ bytes, where $PO$ is the packet overhead. Given the 2 bytes ECG values, the sampling rate $sr$ and an interval of time $I$ seconds during the remote logging application a minimum of $nr(sr, I) = \left\lfloor \frac{2sr + I}{Data} \right\rfloor$ packets will be transmitted. For the on-node processing application this number depends on the interval size, granularity and the allowed latency for only so $nr(G, I) = \left\lfloor \frac{I}{G} \right\rfloor$ where $p$ is the allowed latency in seconds. For the present system $M = 76$ bytes and $PO = 16$ bytes. During $I = 10$s, an application that logs data sampled at $sr = 100Hz$, will transmit minimum $2 * 100 * 10/60 = 34$ packets so a total of $33 * M + 1 * (20 + PO)=2544$ bytes. For the online processing case and $G = 1s$ only 10 packets need to be sent with one heart
rate per packet, so $10 \times (2 + PO)$ bytes which totals to $10 \times 18 = 180$ bytes. If a delay of 10 seconds is allowed, the ten calculated heart rates can fit into a single packet so the amount of information that is transmitted is $20 + PO = 36$ bytes. This is 72 times smaller than the amount of data transmitted for the logging application.

Another important aspect that should be considered is the change in sensitivity and predictivity (as defined in Section 5.1.1) when the data is logged rather than processed on the node. Because radio transmission is usually error-prone and there is no guarantee that all the data packets will reach the destination, there will be a dramatic decrease the sensitivity and predictivity of any algorithm when employed on an external host machine. To ensure better results, the packets will need to be retransmitted. This will, however, cut back the battery lifetime even more.

5.1.3 Energy measurements

An important reason to advocate for on-node processing rather than transmitting data to an external device is the believed energy savings that would appear from the decrease in the amount of information that needs to be transmitted over the wireless link. While it can be speculated that there are energy savings when transmitting less information due to the usual high costs in transmission caused by the high amount of data that needs to be transmitted and the particularities of the electronics and protocols used (see Section 5.1.2 experiments were carried out to see the exact energy consumption for our test setup.

The power consumption is measured using a Coulomb counter [8]. The Coulomb counter is used for making energy measurements and can measure the total power flow by counting the amount of mili–Amps that are consumed while the node was on. To calculate the power from the obtained results the power law equation is used: \[ Power = Voltage \times Current \]

The tests were carried out continuously for 10 minutes. The values presented below are averages over this period of time. From the designed tests the following data was obtained:

To be able to draw relevant conclusions, the power consumption is measured in a number of situations. Sampling is done at 100 Hz in all cases.

1. Sampling (#1): The radio and the processor are in low power mode. This corresponds to the case when the data is only sampled and no processing or transmission happens.

2. Logging (#2): Transmitting 1 packet of 66 bytes every 250 ms. This corresponds to the case when the data is sampled and logged by an external device. All the data is buffered for 250 ms and transmitted to the base station.

3. HR often (#3): Transmitting one packet of 18 bytes every second. This corresponds to the case when the data is sampled and processing of the ECG is done. The heart rate is calculated and transmitted to the base station every second.

4. HR rare (#4): Transmitting one packet of 26 bytes every 5 second. This corresponds to the case when data is sampled and processing is done. The heart rate is calculated every second, buffered for 5 seconds and transmitted to the base station.

5. Processing (#5): Sampling and processing the information. One heart rate is transmitted every second (comparable with #3). By subtracting the value obtained for #3 from the value obtained at this step the amount of power used for processing can be isolated.
The results of the experiments can be seen in Table 5.3. The values are in mili-Amps. The battery powering the sensor nodes offers 3.3V.

As it can be seen from the results, the processing on the nodes of the ECG signal with our algorithm (#5) takes 3 times less energy than the transmission of all the data to an external gateway (#2). If a latency of 5 seconds is permitted in the transmission of the heart rate then there is 5 times less energy consumed (#4).

The sampling of one second worth of data consumes 5.34mW while the processing of the data takes just 14.6 - 14.1 = 0.5mW. For the given sampling rate, sensor board and algorithm these values are fixed and are hardly influenced by other elements.

As expected, the difference in energy consumption comes from the size of the transmitted packets, the interval between transmissions and the energy the MAC consumes. Transmission of the logging information consumes 35.83mW for 264 bytes/sec, sending one heart rate every second consumes 8.75 mW for 18 bytes/sec and transmitting five heart rates every five seconds consumes 2mW for 5 bytes/sec. The influence of the data amounts transmitted is obvious from the results, the more data needs to be transmitted per second, the higher the energy consumption. However, the increase in energy is not directly proportional to the increase in data transmission. This can be explained by the influence of the MAC on the total energy consumption. For case #4 the radio is ON only once in five seconds while for case #3 the radio turns ON and OFF five times. The increase in the energy consumption is hence not only due to the extra 18*5- 26=64 bytes transmitted during five seconds but also due to the MAC, synchronization costs and the extra times that the radio turns ON and OFF.

From the energy measurements, it can be concluded that important energy savings can be obtained by deploying an algorithm for on-node processing. More savings can be obtained by allowing latency in the transmission of the results.

5.2 Research Contributions

The research done for this project on a system composed of an ECG sensor brings the following contributions:

- A simple algorithm for filtering the ECG signal on the sensor nodes was proposed. It takes into consideration the hardware limitations presented in Section 2.7.

- An implementation on an actual node was provided. The testing was done on real persons. This has proved that processing on the nodes is possible and can bring good results.

- The discussion about the difference between ECG logging and processing in terms of data transmission and energy consumption shows the advantages of on node processing. Even so,
the benefits of logging are acknowledged for some situations and a possible way of trade-offing between the two is also explained.

- Following the energy measurements of the ECG application we can conclude that there are important energy savings brought along by the on node processing. When deploying a simple algorithm the communication through the wireless link can be reduced dramatically and with it the energy consumption of the node.

When adding an acceleration sensor to the system it is shown that:

- A simple algorithm can be implemented on the nodes to distinguish between different levels of activity of the person.

- Communication between nodes is easy to achieve. While this case is concerned with only networking of two nodes, this number can always be extended if needed. A personal area network can be created with the OSAS framework. Communication between nodes is possible and can improve the results of using just one node.

The herd control application was used in the tests done on cows at the WASP meetings in Lelystad.

- The mode detection successfully distinguishes between modes if the node is fixed.

- The step detection algorithm can detect the steps accurately. By comparing the step duration during the same interval from two legs problems with the health of the cows claws can be distinguished.

The activity classifier has several levels of classification.

- A basic classification between static and dynamic states can be done at the top level. This can be used in the case when only information about whether the person is static or not is needed. It was successfully used for the special networked application.

- At lower levels the following states can be detected: standing, sitting, laying, walking, climbing up the stairs.
Chapter 6

Conclusions

This chapter concludes the thesis and provides some suggestions for future work.

6.1 Future work

The thesis mentions the networking of the ECG sensor with an acceleration sensor for improved classification. For the future, a more complex body area network can be built with different types of sensors. It can be integrated in a smart-home environment.

Moreover, the algorithm for ECG processing only detects the R–peaks. An attempt can be made to improve this and be able to detect the onset of P and T waves in real time as well.

The research does not focus on classification of heart problems with the help of heart rate variation. However this can be extended and more problems rather than just tachycardia and bradycardia can be detected.

The human classifier only supports a limited number of activities. It can be improved to detect more activities (such as doing exercise, dancing, working on a laptop, washing dishes, walking the dog). Moreover, information about the step length and the velocity of the movement can be useful in both human and cow supervision.

6.2 Conclusion

Reducing the burden of treatments through prevention and early detection of diseases is both economical and beneficial on the social level and can be achieved by shifting the care from in hospital doctor-driven towards the more personal monitoring in the home environment. Just as beneficial is distributed monitoring of animals and crops for agriculture, human and machine monitoring in industry or in the office. The use of sensor networks in real time monitoring of daily activities and physiological parameters became practical. The research carried out for this Master Thesis is focused on algorithms for activity classification deployed in wireless sensor networks. The research started from the idea that doing processing at the node level rather than the gateway level will bring along optimizations in the energy consumptions due to the reduction in the amount of data that needs to be transmitted over the wireless link.

Simple algorithms were designed and programmed on the OSAS Framework (developed at TU/e) for two different applications: health monitoring and herd control. The research is part of
an international project, WASP (Wirelessly Accessible Sensor Populations).

Frequent and regular health monitoring is particularly important for the elderly because their health can suffer from fast changes but it is not restricted to them. In health care monitoring, one sensor can be used for sampling continuously a certain physiological parameter (such as physical activity level, heart rate) and more types of sensors can be networked together to obtain a better classification of the health status of the patient.

The first algorithm developed was used to calculate the heart rate of a patient from the ECG signal obtained from a wireless sensor. The algorithm consists of a number of processing steps, adaptive threshold calculations and heart beat detection. It was tested in real conditions and also on the MIT-BIH Arrhythmia Database which is the standard reference in literature with satisfactory results. A network between the ECG sensor and an acceleration sensor was also built. The acceleration node ran an algorithm to calculate the magnitude of the movement of the person. Based on the information from both sensors a classification can be done regarding the danger in which the monitored patient is in. In short, a high heart rate during intense exercise is considered less dangerous than a high heart rate during sleep.

Moreover, on node processing is used for two more use cases: classifying the activities of a person and herd control. The class of human activities is defined beforehand as: sitting, standing, laying, running, walking, climbing up or down the stairs, falling. The classification is done in a tree-like fashion on three layers. First layer makes a distinction between static and dynamic activities. The second layer determines whether the activity belongs to one of the categories mentioned while the third level determines the transitions between two static states.

The herd control application is defined in one of the deliverables of the WASP project. The cows behavior needs to be monitored during the entire day. An early detection of changes in the walking pattern of a cow can lead to the successful healing of the cow’s claw problems and hence improve the animal welfare. To do this an algorithm was defined for classifying the state in which the cow is at any time: standing, laying or walking. At predefined times, when the cow is moving, the acceleration information is processed so that the walking pattern (number of steps, time between steps etc.) can be extracted. This walking pattern is saved in a back-end system and used for statistics.

In all cases, the ongoing monitoring means gathering and analyzing data continuously. It implies high energy consumption, hence a fast depleting of the batteries. The need for changing the battery every few hours would diminish the advantages of home care. Also, changing the battery of the sensors attached to a patient or a cow a couple of times a day would be a great inconvenience. Prolonging the lifetime of the wireless devices is therefore a must. In practice, the wireless transmission of information is the task that consumes the most energy and the processing of data is orders of magnitude less expensive. This project has shown that on-node processing is feasible and can really help in saving the battery and consequently prolonging the lifetime of the sensor nodes.
Bibliography


[28] Eun Jong Cha Tae Soo Lee Joo Hyun Hong, Nam Jin Kim. Classification technique of human motion context based on wireless sensor network.

[30] Cora Gromann Klaus Manfred Scheibe. Application testing of a new three-dimensional acceleration measuring system with wireless data transfer (was) for behavior analysis.


[34] Lakhmi C. Jain; Shing Chiang Tan; Chee Peng Lim. *An Introduction to Computational Intelligence Paradigms*. 2008.


[48] Rahime (R); Karlik Bekir (B) Ozbay, Yksel (Y); Ceylan. A fuzzy clustering neural network architecture for classification of ecg arrhythmias.


[50] Ivo Provaznik, Jiri Kozumplik, Jana Bardonova, Marie Novakova, and Zuzana Novakova. Wavelet transform in ecg signal processing.


[57] M.J.P. van Bussel. Detection of epileptic seizures based on heart rate patterns. April, 2005; Kempenhaeghe Epilepsy Centre and Eindhoven University of Technology.


