Cow behaviour monitoring with motion sensor

JOHAN SCHAGERSTRÖM

KTH Industrial Engineering and Management

Master of Science Thesis

Stockholm, Sweden 2014

Detektering av kobeteende med rörelsesensor

Johan Schagerström

Examensarbete MMK 2014:99 MDA 483
KTH Industriell teknik och management
Maskinkonstruktion
SE-100 44 STOCKHOLM

Cow behaviour monitoring with motion sensor

Johan Schagerström

Master of Science Thesis MMK 2014:99 MDA 483
KTH Industrial Engineering and Management
Machine Design
SE-100 44 STOCKHOLM

Examensarbete MMK 2014:99 MDA 483

KTH Industriell teknik och management

Detektering av kobeteende med rörelsesensor

Johan Schagerström

Godkänt 2015-01-25	Examinator Martin Törngren	Handledare Sagar Behere
	Uppdragsgivare DeLaval International AB	Kontaktperson Bohao Liao

Sammanfattning

Det är viktigt att övervaka status för mjölkkor från ett djurskydds- och mjölkbondens ekonomiska perspektiv, speciellt för storskaliga lantbruk och automatiska mjölkgårdar. Kobeteende är en indikator på kons välfärd och djurhälsa. I den här rapporten har data samlats in från kor i en mjölkgårdsmiljö och aktivitetsigenkänning har tillämpats för att automatiskt mäta daglig idisslingstid och upptäcka brunst i uppbundna kor. Målet är en produkt som ska uppfylla kraven på noggrannhet, batteritid, robusthet och detaljnivå. Därför har flera systemaspekter beaktats beträffande val av hårdvara, mjukvara och kommunikation.

I metoden ingår en två veckors period i en mjölkgård med sensordatainsamling och observation följt av en fem veckors period i ytterligare en mjölkgård med sensordatainsamling och observation med övervakningskamera. Flera algoritmer och signalegenskaper testades för idisslingsövervakning och utvärderas mot varandra med avseende på sensitivitet, specificitet och beräkningsbelastning.

Slutsatsen är att de signaler som krävs för att upptäcka idissling är tillgängliga med hjälp av en accelerometer på halsbandet och det är möjligt att nå 96 % sensitivitet och 94 % specificitet med hjälp av dessa sensordata, men de bästa algoritmerna misstänks dra för mycket energi för att klara kravet om 10 års batteritid i en produkt. Det visade sig att en accelerometer på kragen inte gör det möjligt att upptäcka läggningar och ställningar som ett mått på brunst i uppbundna kor.

Den externa validiteten behöver testas för detektering av idissling. Nya metoder behöver utforskas för att detektera brunst hos uppbundna kor. Algoritmer borde testas på en den tilltänkta mikrokontrollern så att noggrannare analyser kan göras för batteritid och beräkningskapacitet.

Master of Science Thesis MMK 2014:99 MDA 483

KTH Industrial Engineering and Management

Cow behaviour monitoring with motion sensor

Johan Schagerström

Approved 2015-01-25	Examiner Martin Törngren	Supervisor Sagar Behere
	Commissioner DeLaval International AB	Contact person Bohao Liao

Abstract

It is important to monitor status of dairy cows from an animal welfare and dairy farmer economic point of view, especially for large-scale farming and automatic milking dairies. Cow behavior is an indicator of cow welfare and cow health. In this thesis, data has been gathered from cows in a dairy farm environment and activity recognition has been implemented in order to automatically measure daily rumination time and detect heat in tied-up cows. The goal is a product that must fulfill requirements on accuracy, battery life, robustness and level of detail. Therefore, multiple system aspects have been considered concerning choice of hardware, software and communication.

The method included a 2 week period in a dairy farm with sensor data gathering and observation, physically in front of the cows, followed by another 5 week period in another dairy farm with sensor data gathering and observation using surveillance camera. Multiple algorithms and signal features were tested for rumination monitoring and evaluated against each other with respect to sensitivity, specificity and computational load.

The conclusion is that the signals required for detecting rumination are available using an accelerometer on the collar and it is possible to reach 96 % sensitivity and 94 % specificity using this sensor data, but the best algorithms are suspected to draw too much energy in order to comply with the requirement of 10 years of battery life in a product. It turned out that an accelerometer on the collar is not feasible for detecting lie-downs and stand-ups as a measure of heat in tied-up cows.

The external validity needs to be tested for rumination detection. New methods need to be explored for detecting heat in tied-up cows. Algorithms should be tested on the intended microcontroller so that more thorough analysis can be made with respect to battery life and computational capacity.

Acknowledgements

My supervisor at DeLaval, Bohao Liao, a system specialist at research and innovation, has provided me with great help with installation, signal analysis and strategies and also provided feedback throughout the project.

I would also like to thank Farm Management Automation specialist Göran Eriksson for the travels to Hölö and for helping with installation and uninstallation of the activity system.

There are also others at DeLaval that have been contributing to this project, among them Lars Hansen, project owner, Anders Cardell and Rickard Mellgren, for helping with setting up software in the system controller and establishing connection, Anna Strömgren and Martin Fredholm. In cooperation between DeLaval and Tritech, I would like to express my gratitude for the help and knowledge about the technical details of the system.

Lastly, I would like to thank my academic supervisor Sagar Behere, for providing excellent feedback and pointing me in the right direction.

Table of Contents

1. Terminology

ANN	Artificial Neural Network
DWT	Discrete Wavelet Transform, any wavelet transform for which the wavelets are discretely sampled. Captures both frequency and location in time.
FFT	Fast Fourier Transform, an algorithm to compute the discrete Fourier Transform and its inverse, in frequency analysis.
Heat	The periodic state of sexual excitement in the female of most mammals, excluding humans, that immediately precedes ovulation and during which the female is most receptive to mating. Also known as estrus or oestrus.
HMM	Hidden Markov Model, a statistical Markov model known for pattern recognition.
k-NN	k-Nearest Neighbors algorithm.
NM	Nearest Mean, a clustering algorithm.
SC	System Controller, a central component in DeLaval's system.
Tag	Mobile sensing module mounted on cow including sensor, processing device and wireless communication. The market name used by DeLaval for this unit is "meter" as in "activity meter".
MCU	Microcontroller, a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals.
RTC	Real-time clock, a computer clock that keeps track of current time with low power consumption and enables time-critical tasks. In many cases it runs on 32768 Hz.

2. Introduction

This report describes a Master Thesis project about activity recognition systems for cows, that was carried out in cooperation with DeLaval International AB in 2014. Activity recognition systems means a system that attempts to map sensor data to an activity that is likely to have caused the data, and it consists of a low-level sensing module, a feature processing module and a classification module [1].

2.1. Background

It is important to monitor status of dairy cows from an animal welfare and dairy farmer economic point of view. It is even more important for large-scale farming and automatic milking dairies. Cow behavior is an indicator of cow welfare and cow health. An on-cow motion sensor may be possible to use to observe motion characteristics of a cow, like walking, eating etc. In this thesis, cow behavior monitoring refers to detection of when the cow is ruminating[1] and additionally it may be of interest to detect when a tied-up cow is standing or lying which is an indicator of heat, also known as estrus.

There are a handful of activity recognition systems for cows available today, as documented in section 4.1, and the system parameters vary a lot. Designing such systems requires evaluation of different system aspects, such as the sensor sampling rate, mounting of sensor and number of sensor axes. This leads to a discussion of system requirements, choice of hardware, software and communication. Whilst doing this, it is valuable to set up possible system solutions and compare these. Activity recognition systems usually require a lot of computation. At the same time, the measurement system is preferred to be mobile and battery driven and has weak computation performance. The communication is usually wireless, but the transmission distance and data package size differ greatly.

2.2. Goals

The first goal in the project is a **feasibility study** of measuring **rumination time** with accelerometer in an activity tag and **standing/lying** detection with help of accelerometer. This includes:

- Algorithm development
- Performance evaluation with data collected in farms
- Implementation aspects

The second goal of the project is to develop a **recommendation of system solution** for cow behavior monitoring based on an on-cow motion sensor.

[1] Rumination is the activity when the cow chews repeatedly for an extended period as part of the digestive process among ruminants.

A reconfigured version of DeLaval’s Activity Meter system programmed to transmit raw sensor data was given to the project. This is instead of building a new sensor module which would log sensor data on a memory, just for the project. This puts some constraints on the study, limiting the sensor data to acceleration on the collar. The original version of DeLaval’s Activity Meter system reports a measure of cow activity to a database and a computer program visualizes this information and helps the farmer to detect if a cow is in heat. This measure is based on basic accelerometer signal processing.

2.3. Research hypothesis and problem

The first hypothesis is that it is possible to detect rumination activity in cows using a motion sensor. The second hypothesis is that it is possible to detect lie-downs and/or stand-ups using a motion sensor.

2.3.1. Questions that are needed to test the hypothesis

To test the hypothesis, some questions need to be answered. How should the algorithm and system for the cow behavior monitoring sensor be designed, to achieve high performance and to maximize energy efficiency? Is it feasible to detect rumination with a neck-mounted motion sensor? Can a motion sensor sense the rumination? Is it possible to distinguish rumination from other cow daily activities? Will the performance of rumination detection be good enough for the applications?

2.3.2. Engineering work needed

Other than the main algorithm, system aspects such as the sensor sampling rate, mounting of sensor and number of sensor axes have to be optimized. All these have a heavy impact on the performance on the algorithm and energy consumption of the sensor, for instance radio traffic. This will lead to requirements on the system. The new system setup also has to be tested in field and anomalies like reoccurring data due to radio transmission redundancy have to be dealt with. The electronic structure of the sensor must be investigated to see what can be reprogrammed and what functions that could be implemented on different parts. The performance will be evaluated with statistical analysis based on data collected from cows in a dairy farm condition.

2.4. Methodology

In order to develop software that successfully detects rumination as well as other activities, a set of activities had to be carried out. First, a pre-study about human activity recognition systems and cow activity recognition systems has to be done. What sensors, sampling rates, features, algorithms and so on are used today? The project is provided with a reprogrammed version of DeLaval’s current activity meter system that has increased sampling frequency and transmits all raw data from the sensors. Testing has to be done when the prototype hardware that is provided to the project is programmed. The testing focuses on anomalies and reoccurring data, due to radio transmission redundancy, after the modification of DeLaval’s activity meter system. Then, the modified hardware has to be installed in a dairy farm. After this, the evaluation of four different ways of mounting on cow collar is carried out. Only one cow will be observed at a time, and activities like start and stop of rumination, eating, drinking, stand-up, lie-down, head-shaking, walking will be manually logged, even if only rumination and standing/lying is of interest for the classifier, since having more information when post-analyzing the data helps. After this, the observation will continue with one sensor per cow, with a limit of 3-4 cows, to build up a training set and testing set that are used to build and verify a classifier. Data files will be manually extracted from a "system controller", a central control module in DeLaval's system setup, for use in MATLAB. For heat detection, a video camera will be used to gather observation data, since the observation takes up a lot more time and also because the cows stay at the same place.

In Figure 2-1, the dataflow from cow to computer in the modified DeLaval Activity Meter system, which was given to the project, is illustrated. Acceleration data is regularly sent from a sensor to a microcontroller in the tag on the cow's collar. The microcontroller, for this project, sends all data wirelessly to a stationary receiver using a radio protocol. The microcontroller in the receiver buffers the data and sends it on a two-wire bus network. A system controller collects data transmitted on the bus from all receivers and saves it on a memory card. The system controller is connected to a network through Ethernet and a computer can access the data in the memory card. The computer is configured to move data from the memory card to the internal hard drive daily to avoid filling the memory card. The boundary of the "system" is between the system controller and the network interface.

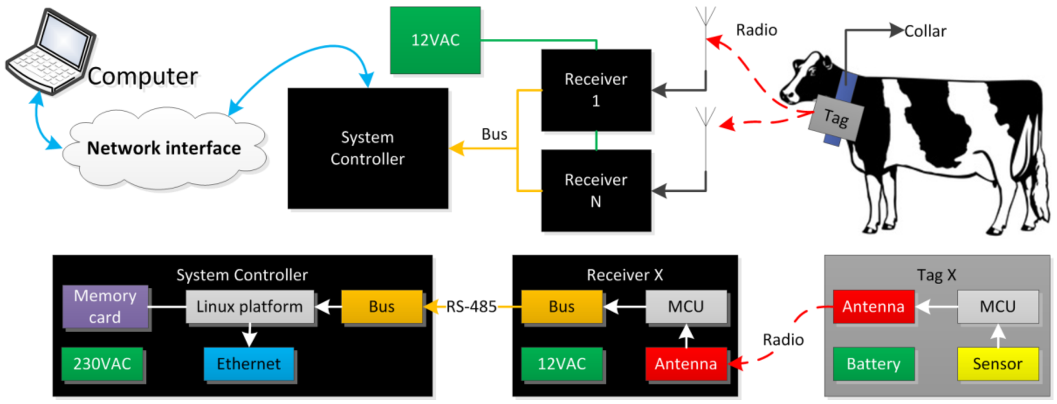


Figure 2-1. Simplified diagram of the system setup.

The necessary procedure to develop the optimal algorithm requires scripts for raw data gathering, preprocessing, segmentation, feature extraction and classification. Features or preprocessing methods to evaluate are variance, mean, derivative, fast Fourier transform (FFT), discrete wavelet transform (DWT). Activity classification methods to evaluate are threshold based, decision tree, k-nearest neighbor (k-NN), nearest mean (NM), Naïve Bayes and Fuzzy Logic. Feature sequence classification methods like hidden Markov model (HMM) are also of interest to be used. Combination of classifiers might yield higher accuracy.

To develop the system solution, a pre-study of activity recognition systems for cows had to be done. The pre-study also included interviewing designers of DeLaval's current activity meter system about memory usage, central processing unit (CPU) utilization, modes of operation and other system aspects. The last step is choosing an algorithm and optimizing the code for an embedded platform and estimating memory requirements and CPU utilization.

2.5. Delimitations

Due to the limited time of the project, the following delimitations had to be done:

- The project will not include designing any electronics in the sensor; instead, the focus lies on investigating main components and partitioning of functionality.
- No software changes will be implemented or tested on the system hardware after the initial prototype configuration, but ideas of modes of operation and algorithms will be evaluated and tested in MATLAB.
- The provided hardware will not be changed during the project.
- Different hardware will not be bought or evaluated. The project rather aims at using a modified version of DeLaval’s activity meter system provided to the project.
- The algorithm in this thesis should detect physical motions, but not draw biological conclusions such as health.
- The algorithm in this thesis will not take into account if the cow is near calving or illness.
- Sensor positions to evaluate is limited to the collar, other placements will not be tested.

3. State of the art

This chapter will discuss what activity systems exist today, how they work, what they are able to do and also what kind of activity classifier methods that are used, giving a clearer perspective on designing a system for activity recognition.

3.1. Activity systems

Several systems have been made for activity recognition but most reports focus on monitoring human motions. Human activity recognition (HAR) systems are usually realized with a smartphone app, because it’s easy and cheap. Activity systems for herds have been around for several years, but until recently these haven’t been able to recognize activities.

There is an activity system for cows called Qwes-HR or Heatime® HR LD or RuminAct™ that is said to be able to recognize rumination and daily activity. The sensor consists of an accelerometer and a microphone and transmits data with IR when the cow is at an automatic milking station. The sensor is able to hold 2 hours of data. The sensor is mounted on the upper left part of the collar and the position is ensured by a bound weight on the collar. The rumination time is computed directly in the sensor and the daily activity and rumination is analyzed by a computer that can compare and match day-to-day results with estrus behavior. The overall system also includes a weight scale at the milking robot, to give better detection of estrus. The wireless transmission is only done in the proximity of a receiving station, for example at the milking point or at a passage. [2]

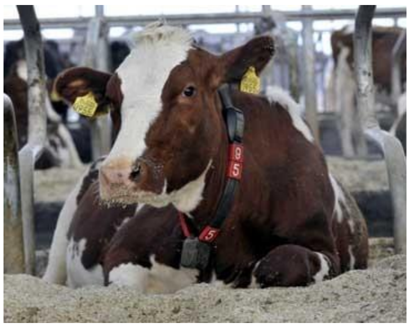
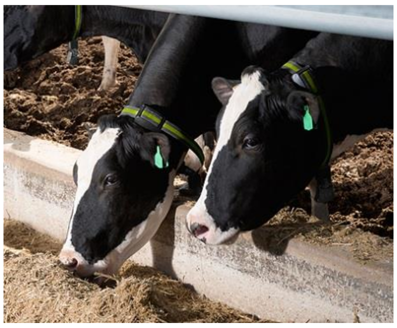
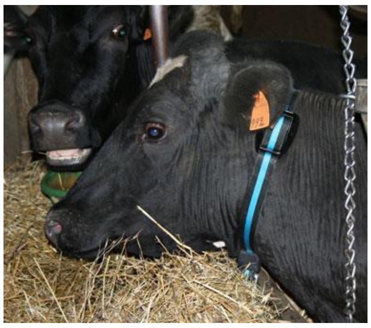




Figure 3-1. The Qwes-HR tag (lely.com). Figure 3-2. The Heatime® HR LD tag (scrdairy.com). Figure 3-3. The Heatime-RuminAct® tag (milkline.com).

Another system is the MSR-ART rumination sensor or RumiWatch that records rumination, eating and drinking, and can be supplemented with a pedometer to also record lying, standing and walking. The noseband sensor (NBS) consists of on oil-filled tube, pressure sensor, accelerometer and a wireless transmitter. The microcontroller samples at 10 Hz and does the calculations in real-time. The pedometer consists of an accelerometer and a wireless transmitter and the microcontroller operates the same way as in the NBS. The software algorithm to detect jaw movements is described as a fuzzy logic where the programmer has used intuition and reasoning to differentiate between activities. The algorithm consists of a set of rules, including time, local maxima, median, margins. To reduce classification computation, only one point during a chewing period is examined, with respect to variation in amplitude of previous and subsequent jaw movements, the variability of the pressure pattern over time and also the regularity of chewing frequency. The sensor nodes transmit data using ANT wireless sensor technology to a USB-dongle connected to a computer. [3]

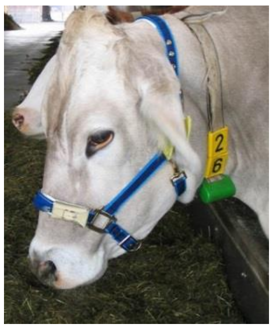
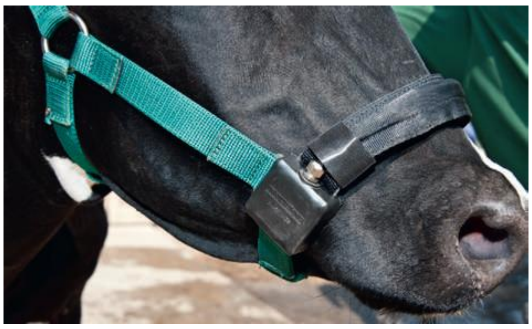
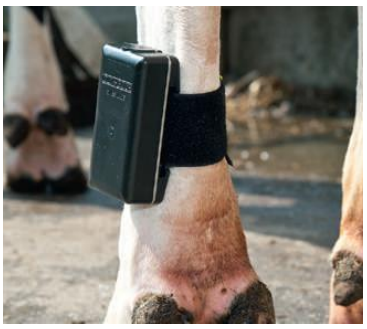




Figure 3-4. The MSR-ART rumination sensor (msr.ch). Figure 3-5. The RumiWatch NBS (rumiwatch.ch). Figure 3-6. The RumiWatch pedometer (rumiwatch.ch).

CowManager SensOor system consists of wireless sensor nodes that transmit data to the routers or the coordinator. The routers repeat the transmissions so that the information spreads to the coordinator station, to maximize the reach of the system. The coordinator is connected to a computer, and that computer can then transmit the data over internet to a distant computer as an optional workspace. The sensor is a chip that is clicked into the ear tag of the cow. The eartag has a reach of 100 meters and can store the information for two days. The activity that is monitored by the system is fertility, rumination and health. This is done by using an accelerometer and a thermometer. [4]

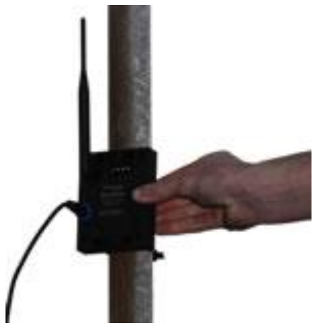
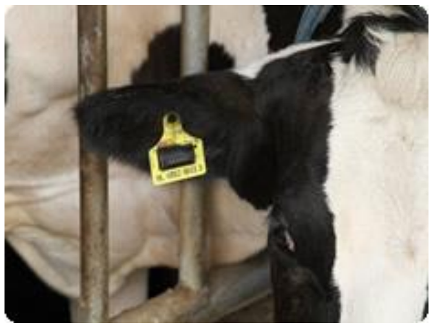



Figure 3-7. The SensOor system coordinator (cowmanager.com). Figure 3-8. The SensOor ear tag chip (cowmanager.com).

A system from Afimilk called AfiTag Pedometer contains an accelerometer and is strapped to one of the cow’s legs. The AfiTag Pedometer detects high motion activity, stationary angle of the leg and also works as an identification tag. The angle of the leg is monitored because the leg is not vertical when the cow is lying down, and that information is important for detecting estrus in tied-up cows [5].

Figure 3-9. AfiTag Pedometer Plus (afimilk.com).

A research system for a study was built in the Netherlands that consisted of GPS-collars and accelerometer. 9 cows were equipped with Lotek EGNOS-enabled GPS-collars and 6 geese were equipped with a platform from BioTrack containing GPS, 3D-accelerometer, microcontroller and memory card. The activities that were monitored were foraging, ruminating, standing and walking. The two sensors were never combined on cows during the test, but the researchers believe that the combination of the detection method on cows and geese would yield good detection results on cows. The system had 60% sensitivity and 52% specificity for rumination. [6]

Another study used a two-axis accelerometer (MPR2400 Micaz sensor motes from Crossbow with a MTS310 sensor module) together with a Chipcon CC2420 radio which used a 2.4 GHz IEEE 802.15.4/ZigBee RF transceiver. The accelerometer sampled at 1 Hz and data was transmitted directly. The study also tested a 3D-accelerometer (ADXL345) without RF transceiver. The accelerometer was positioned on top of the neck. The purpose of the system was to monitor grazing time and the classification was done with a threshold that was manually set by visual inspection of the time-series data of one axis. The system had 74% sensitivity and 82% specificity. [7]

There are many studies with activity recognition systems for humans. These systems are not mass-produced, have short battery life and are seldom analyzed in real-time. A publication in ISSN 1424-8220 had summarized 36 full-paper studies with human activity recognition systems. Almost all systems used at least one accelerometer with at least 2 axes. Almost all systems consisted of more than one sensor module. The majority of the systems also had a gyroscope. A third of the systems had a tri-axial magnetic sensor. Almost all systems used a sampling frequency of 100 Hz or above. Three out of four systems was meant for detection of walking or more activities. [8]

What is missing in the above systems and what problems can be solved with a new activity recognition system? By placing the sensor on the collar, it causes no damage to the cow as leg-mounted sensors. If the sensor were to be mounted in the ear, then there would be more motion disturbances than on the neck, which puts limitations on the detection performance. By using a microphone in the sensor on the collar, the collar is required to be tight, which causes some discomfort for the cow, but without it, the collar could be loosened. Also, none of the other systems, except the commercial ones, has a battery life in the 3-15 year range.

3.2. Activity classifier methods

There are three main challenges with activity recognition systems. The first challenge is that the same activity can be performed differently by different individuals or from time to time, called Intra-Class Variability. The second challenge is that different activities may show very similar characteristics in the sensor data, called Inter-Class Similarity. The third challenge is that the activities of interest can easily be confused with activities that have similar patterns but that are irrelevant to the application in question, called The NULL Class Problem. [9]

There are different methods of classifying activities depending on the circumstances and requirements. Classification can either be done real-time or offline. Real-time classification is done directly by the sensing node (in this case the tag) and offline classification would be done in either the receivers or the system controller. Offline classification is feasible when the computation is too heavy for a battery-driven processing unit and when the feedback isn't required immediately. The activity that is classified can also be in the form of a gesture or something that continues for longer periods, such as behavior or trends. Cow activities like rumination, eating, sleeping are behaviors or trends. A lie-down or stand-up motion is a gesture that only lasts for seconds.

Multiple different classifiers have been tested and evaluated in different studies with varying results. A publication in ISSN 1424-8220 [10] summarized different human activity recognition systems with focus on the classifiers. This publication listed what features, what classifiers and what sensors that were used and their accuracy. Common classifiers are threshold-based, binary decision, ANN and k-NN. But GMM, SVM, Naïve Bayesian, Template Matching and Markov chains are also used. Typical features used for classification are DC component, standard deviation, FFT, wavelet coefficients and correlation coefficients. The accuracy usually reaches above 90%. No classifier method or feature choice excels the other. [10]

3.2.1. General method

The general method to classify activities is through the activity recognition chain (ARC) [9]. The raw signal data is first preprocessed with filters and functions. Then, to reduce the computation during classification, the signals are segmented into a much smaller vector length[2]. Features are then extracted from the segmented data. The feature vectors are then classified as different activities.

3.2.1.1. Preprocessing

Depending on the character of the signal that is being classified, the preprocessing can look very differently. The preprocessing can include steps like reparation of corrupted data, artifact handling, unit conversion, frequency filtering and computation of signal variability.

[2] The vector length is all the data points in one timeseries variable for a given time interval.

3.2.1.2. Segmentation

Depending on if the activity is a gesture or behavior, the signal is either segmented into sections of interest that are likely to contain the activity or gesture [9] or it can be just a decimation method to minimize further computation.

3.2.1.3. Feature extraction

Feature extraction is the final stage before classification where features of the signal properties are calculated. For gesture detection, this could be properties within each segment. For behavior monitoring, this could be the extraction of a certain frequency spectrum combined with a certain signal strength.

3.2.1.4. Classification

Depending on what classifier that has been chosen, the steps in this stage may vary. For some classifiers this stage consists of going through every frame with a set of rules for classification. Other classifiers do matrix multiplication on segments and end up with a probability number. Some classifiers can output multiple classes, depending on the mix of signal features.

3.2.2. Threshold based, binary classifier, decision tree

For continuous activities or trends, it is suitable to use a single-frame classification algorithm. There are three types of classifiers that are very similar. Threshold based, binary classifier and decision tree can all be carefully handcrafted. Thresholds based on energy-related features, or data variance is an easy way to discriminate between motions and stationary activities [10]. The disadvantage is that the method is sensitive to individual variations and sensor placement, which requires extra care when setting the parameters in order to achieve a good generalized classifier [10].

The threshold based method can be continued in several different steps where different strategies or features are examined. The steps in which the activity features are examined and/or calculated build up a decision tree. As the classification goes deeper down the branches, the result becomes more refined. The decision tree can either reach a binary decision, choosing between two different states, or it can result in multiple classes. Since it is possible to reach complex decision boundaries with this method, it goes under non-linear classifier category. In Figure 3-10, the above thresholds and decision tree are illustrated.

Trees usually have low prediction accuracy, but they can be fitted fast, they classify fast, use low memory and are easy to interpret.

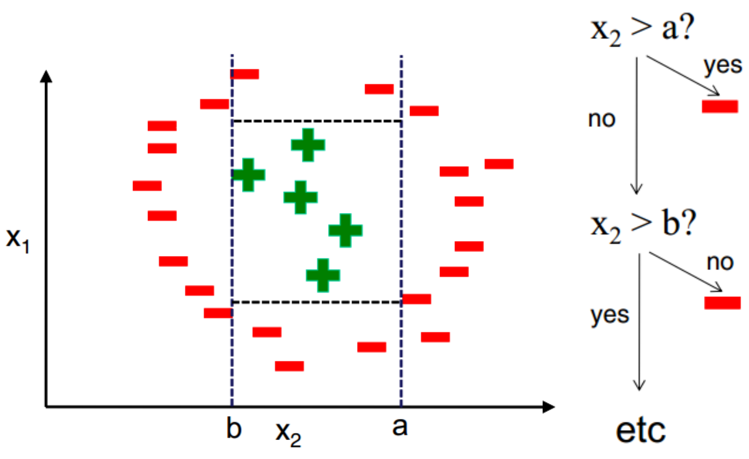


Figure 3-10. Left: Decision boundary between two classes, decided by values a and b. Right: The decision tree of such boundary. (Source: Quaid Morris, Non-linear classification)

3.2.3. Support Vector Machine

An algorithm that automatically sets up a decision boundary between classes is called an SVM, which is one of the most popular algorithms in modern machine learning [11]. The SVM is mainly a two-class classifier, but it can be modified into an N-class classifier. SVMs maximize the margin of hyperplanes that separate the data into classes, using critical data points called the support vectors, see Figure 3-11. The implementation of the learning algorithm is difficult and does not work well for large data sets, since it produces more support vectors, which in turn increases computation time. The algorithm is an optimization problem, and the classification method depends on what kernel function that is used to make the data linearly separable. A linear kernel function will generate a linear classifier, making the implementation easier. More complex decision boundaries are also possible, using either a polynomial function kernel, a sigmoid function kernel or a radial basis function kernel [11]. The prediction speed and memory usage of SVM's are good if there are few support vectors, but can be poor if there are many support vectors. When a kernel function is used, it can be difficult to interpret how the SVM classifies data, though the default linear scheme is easy to interpret [12].

There is a built-in tool in MATLAB to train and test a SVM model. This enables quick evaluation of the classification method and also gives a graphical approach to discriminating data. The `svmtrain` and `svmclassify` are part of the Statistics Toolbox, earlier Bioinformatics Toolbox, but there are alternatives like LibSVM [13].

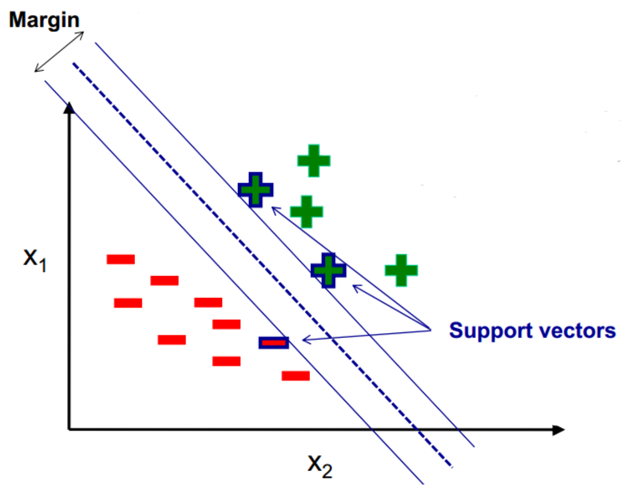


Figure 3-11. Decision boundary between two classes, decided by support vectors.
(Source: Quaid Morris, Non-linear classification)

3.2.4. Hidden Markov Model

A sequential approach to classifying motions is with HMM. The motions of the body can be broken down into a number of primitives, a chain of states. The number of states is finite and a model of transitions between states can be created [10]. The parameters of the model are what states there are, what observable features there are, and the transition probabilities between each state, the observation probabilities of different features. These parameters need to be estimated. Choosing which states and observable features there should be is the hardest part. The rest is completed by following the HMM forward algorithm, the HMM Viterbi algorithm or the HMM Baum-Welch algorithm. The forward algorithm computes the probabilities of different states at different times. The Viterbi algorithm computes the most likely sequence of states given the model parameters. The Baum-Welch algorithm estimates the model parameters automatically given training data [11]. The Baum-Welch algorithm can also be used on-the-fly to adapt for unexpected changes to the transition probability matrix [10]. Classification is performed by choosing the model which produces the largest log likelihood given a stream of feature data from the test set [14].

Just as with SVM, there are built-in tools in MATLAB to train and test a HMM model. This enables quick evaluation of the classification method and can also give a visualization of the classification information. The built-in tools are both part of the Statistics Toolbox and Bioinformatics Toolbox, but there are alternatives like Kevin Murphy’s HMM toolbox [15].

3.2.5. Algorithms selected for evaluation

The main algorithms of interest, that are also easy to implement, are decision tree and SVM. A decision tree can be handcrafted by intuition from looking at the signal characteristics from the motion sensor. It does however take some time to increase the performance of the decision tree. Therefore, it would be interesting to use SVM as a comparison and possibly get the best performance from the selected signal features. HMM has lower priority since it risks taking too long time to implement, so it would only be attempted if there was good data to back it up.

4. System aspects

In order to fully understand the context in this chapter, it is important to understand how a cow motion sensor system works. Each cow wears a tag containing a motion sensor, a processing device and a wireless transmitter. The wireless signals are collected by receivers that communicate the information to a base unit. The base unit is the center where logged data is stored and where information is analyzed. It is here called system controller and it can be seen in Figure 2-1. For more system details, visit chapter 5.1. on page 22. This chapter will first introduce the requirements of the system, describing what the system should do. Then a thorough discussion follows regarding various alternatives with pros and cons, divided into hardware, software and communication. The chapter is concluded by what was chosen as system solution.

4.1. Requirements

The requirements for the new system were derived from discussions with the project steering group, appointed by the project owner at DeLaval International. The steering group consisted of:

- a business manager in farm management automation
- a specialist in farm management automation
- the industrial supervisor and specialist in R&I
- the equipment supplier and project leader for DeLaval's Activity Meter system

The underlying factor for the requirements is that the product must satisfy the customer needs. Another factor that affected the requirements slightly is that fewer hardware changes or additions would be preferable. All requirements for the original Activity Meter system also apply to the new system, some of which are listed in this report.

Every requirement has a unique number, starting from 1 for the first requirement. The requirement priorities range from 1 to 3 where 1 is the highest priority. Priority 1 requirements must be met for the project to be considered successful. Priority 2 and 3 issues are desirable but not necessary for the successful completion of the project.

The requirements could be divided into hardware requirements, receiving side requirements, software requirements, performance requirements, and so on, but since the hardware is split into subsystems, and since the detail and depth rapidly can increase, all requirements can just as well be listed as system requirements. Only requirements that are related to this project are listed. The system requirements are divided into functional and non-functional requirements. The functional requirements describe how the system must behave, as a black box. The non-functional requirements describe the quality and performance level of the system.

Below only most critical requirements are listed. The functional requirements are in Table 3-1 and the non-functional requirements are in Table 3-2. A list of the full requirements can be found in appendix A.

Table 4-1. Functional requirements, the most critical.

No#	Description	Priority
1	Cow rumination and activity information must be transmitted from the tag to the base unit least once every 15 minutes.	1
2	The system must be able to detect rumination activity and calculate rumination time.	1

Table 4-2. Non-functional requirements, the most critical.

No#	Description	Priority	Domain
3	The system must be scalable, using any number of tags.	1	Scalability
4	Wireless communication must not interfere with other equipment or regulations.	1	Regulatory
5	The tags must withstand a dairy farm environment.	1	Environmental
6	The system must not cause damage or discomfort for the cows.	1	Environmental
7	The tags must be completely portable. No wires are permitted.	1	Portability

4.2. Possible solutions

There are many possible solutions to build a system upon. In this subchapter the different choices of hardware, software and communication are presented and how they relate to performance, power consumption and requirements.

4.2.1. Hardware

4.2.1.1. Sensors

There are multiple sensors available today that can be easily interfaced with microcontrollers. Some types of sensors are usually in the form of micro electromechanical systems (MEMS), which are energy efficient and very small. The MEMS usually include a central unit that processes data and other components that interact with surroundings. Typical MEMS sensors are accelerometers, gyroscopes, microphones, pressure sensors and Bio-MEMS. It should be mentioned that there are also MEMS actuators. [16]

Accelerometers measure physical acceleration experienced due to inertial forces or due to mechanical excitation. The accelerometer practically works as a damped mass on springs, where the displacement is measured. The mechanical motion is turned into an electrical signal using piezoelectric, piezoresistive and capacitive techniques. Depending on the application in terms of frequency range, different techniques are preferred. In a MEMS the accelerometer is realized by a seismic-mass on a cantilever beam in a microscopic area [17]. When the accelerometer rests in the earth’s gravitational field it will indicate an acceleration of 1 g caused by the weight force distributed over the axes depending on the orientation. MEMS accelerometers require 0.005 to 0.7 mA in operation, based on available product listings among the most common manufacturers.

Gyroscopes measure the physical angular velocity. There are several techniques that can be used as a gyroscope, including Draper tuning fork, piezoelectric plate, laser ring and micro-laser. In MEMS, the most usual techniques are tuning fork and vibrating ring but piezoelectric plates are also becoming popular [18]. Gyroscopes require significantly more current to operate than accelerometers, ranging between 4 to 7 mA in operation, based on available product listings among the most common manufacturers.

Pressure sensors require an extra medium in order to sense motions and are therefore rare in motion sensing. Pressure sensors require 3 to 7 mA in operation, based on available product listings among the most common manufacturers. There are also pressure sensors meant for measuring the atmospheric pressure which require significantly less power.

Microphones do not capture motions, but they do give the system ears. The output is only useful if it processed into either a frequency spectrum or sound intensity. MEMS microphones typically have a current consumption of 0.15 to 0.6 mA, based on available product listings among the most common manufacturers.

4.2.1.2. Processing

Today there are many MCU (microcontroller), DSP (digital signal processor) and DSC (digital signal controller) manufacturers, each having many devices in different product families. Selection of the processing element in an embedded device depends on peripheral needs, required resolution and precision, energy consumption and cost among other influences [19]. Different processing units have different strengths. In the industry, selection of hardware platforms for individual applications is a common task, and the decision is made on the basis of experiences from former projects and the knowledge of domain experts [20]. The attributes of different solutions can be divided into different categories and subcategories [20]:

- Dependability
    - Robustness
    - Reliability
    - Security
- Modifiability
    - Maintainability
    - Adaptability
    - Scalability
    - Configurability
- Reusability
- Testability
- Performance
- Functional range
- Marketability
    - Time-to-market
    - Cost
- System qualities
    - Mounting space
    - Power consumption

In this case, for the system to be successful, the key attributes of the tag are functionality, power consumption and cost, while on the receiving end of the system the key attributes are performance, functionality and robustness. In 2002, Texas Instruments gathered a set of real-time signal processing options and what attributes they have in Table 4-3, making it easier to decide on a solution [21]. Since then, the focus lies mainly on MCU (microcontrollers), DSP (digital signal processors), FPGA (field programmable gate arrays) and ASIC (application-specific integrated circuits) [22].

Table 4-3. Decision table for designers of real-time applications (Table from www.ti.com/lit/pdf/spra879).

	Time to Market	Performance	Price	Development Ease	Power	Feature Flexibility	Summary
ASIC	Poor	Excellent	Excellent	Fair	Good	Poor	Fair
ASSP	Fair	Excellent	Good	Fair	Excellent	Poor	Good
Configurable	Poor	Excellent	Good	Poor	Good	Fair	Fair
DSP	Excellent	Excellent	Good	Excellent	Excellent	Excellent	Excellent
FPGA	Good	Excellent	Poor	Excellent	Poor	Good	Fair
MCU	Excellent	Fair	Excellent	Good	Fair	Excellent	Good
RISC	Good	Good	Fair	Good	Fair	Excellent	Good

As can be seen, DSP and MCU have the best attributes for being used in the tags. So what are MCU’s and DSP’s? Microcontrollers are general-purpose devices for information processing and control that can be adapted to a wide variety of applications by software [22]. The unit cost is very low and the time to market is quite good because software programmers outnumber hardware designers by a significant margin. Performance is dependent on compact code that makes the most efficient use of the MCU architecture. Power consumption is not that good in general, but there are dedicated ultra-low power 8- and 32-bit MCU’s available today, such as Si10XX series from Silicon Labs and MSP430 series from Texas Instruments, that makes battery life in years a reality. Digital signal processors hard-wire the basic functions of many signal-processing algorithms, such as multiplication. The DSP’s can do this more efficiently than MCU’s, but at the expense of flexibility. Many MCU’s actually include basic DSP operations in their instruction, to speed up for instance byte multiplication [22]. One DSP to consider is the C5000 Ultra Low Power series from Texas Instruments. There are also MEMS sensors with built-in DSP that can be configured for preprocessing data. It is however harder to estimate the average power consumption of a DSP during operation.

It should also be mentioned that many single-chip solutions are possible, such as implementing a soft core MCU on an FPGA and thus having a customizable system on a chip (cSoC). There are also MCU’s with integrated 802.15.4 transceivers that are well suited for the tag.

4.2.1.3. Power

A mobile sensing platform requires a power source that is independent of the AC mains. This calls for the use of a battery. In order to extend the life, it is possible to use energy harvesting. The most popular method of energy harvesting is solar cells, but there are also methods to harvest mechanical and thermal energy. Solar cells are capable of harvesting 15 mW/cm^2. If a mobile sensing platform consumes 20µA continuously and daylight conditions are assumed for 5 hours per day, then a daylight charging current of 76µA is required for perpetual operation. [23]

There exist several kinds of batteries, and selecting the right one is the key to maximum operational lifetime of the mobile sensing platform. Conventional batteries that have been used in embedded systems that require long life spans are coin cells, AA lithium batteries and lithium-thionyl-chloride batteries [23]. However, lithium-thionyl-chloride batteries release a gas during discharge and may explode if shorted. A CR2032 coin cell battery is usually sufficient for wireless sensors because of the size, low self-discharge and unit cost, but they are not rechargeable. An alternative is the thin film rechargeable lithium battery, which can be recharged, have high energy density and can be very thin. These are very suitable for solar cell storage devices, smart cards, RFID tags, implantable medical devices and wireless sensors [24]. It is possible for a small thin film battery to have a total lifetime energy storage capacity of more than thirty CR2032 coin cells, thus weighing up the higher manufacturing costs [23].

Another suggestion is to improve product lifetime by only swapping battery when it is almost discharged. This however causes the same burden for the farmer as swapping the whole tag when the battery is discharged. This also puts special requirements on the battery lid, since it must handle all fluids thinkable in a dairy farm and also handle the weight of a cow if it would be stepped on. This is why DeLaval currently fills the tag with hot plastic which then cools down and hardens, protecting the electronics from fluids and external force. To quote a farm management automation specialist: "*This is the only method we've seen to handle the tough environment*".

A general power source is the mains that are usually available in stationary installations. This does however require that the powered device is not mobile, so it is only usable by receiver stations and the main system control unit. By only using a transformer down to 12VAC, the power cables can be connected interchangeably, which eases system installation.

4.2.1.4. Battery life estimation

The current battery discharge with the given modified DeLaval Activity Meter system hardware is shown in Figure 4-1. Since the tags mainly transmit messages over radio constantly, the main power consumption is in the antenna. The battery life was expected to be around 15 days, but it turned out that the batteries held for at least 38 days.

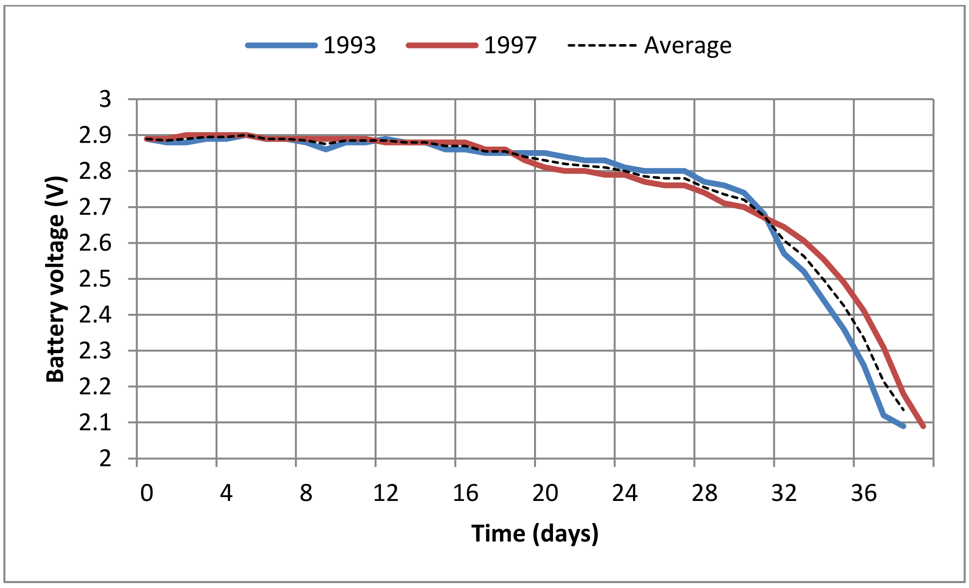


Figure 4-1. Battery discharge curve during testing for tag 1993 and 1997.

The following information is given. The current consumption I_{TX} while transmitting a message lies on 30 mA. A complete message takes 32 ms to complete. When the MCU is active, the current is 4.0 mA. There are three main components of the total current; the sensor, the microcontroller, and radio. From intuition, a model of battery life expectancy in hours can be simplified as

$$T = \frac{Q}{I} = \frac{N_b Q_b \left(1-k_b\right)^T}{I_{sensor} + I_{radio} + I_{\mu C}}, \tag{4.1}$$

where Q is the capacity of the energy source [Ah] and I is the total discharge current [A]. The energy source is determined by the number of batteries N_b, the capacity of each battery C_b [Ah] and the charge leakage percentage of the battery k_b [0…1]. The current of the sensor is considered constant in relation to the operation settings and can be found in the datasheet [25]. The utilization of the radio is described as the sum of the wake-up time from shutdown mode to TX mode, t_{start}, and time to transmit a number of bytes D, at the bitrate B, divided by the period time of the transmissions.

$$U_{radio} = \frac{t_{start} + \frac{D}{B}}{t_{TXperiod}} \tag{4.2}$$

The average current during a radio transmission can be described as the weighted average of the average current during wake-up I_{start} and the rated current for transmitting, I_{TX}.

$$I_{tx} = \frac{I_{start} t_{start} + I_{TX} \frac{D}{B}}{t_{start} + \frac{D}{B}} \tag{4.3}$$

Therefore, the total average current of the radio can be described as the weighted average of the average current during a radio transmission, I_{tx}, and the rated current in shutdown mode, I_{sd}. Put together, the average current of the radio becomes

$$\begin{aligned} I_{radio} &= U_{radio} I_{tx} + (1 - U_{radio}) I_{sd} \\ &= \frac{t_{start} + \frac{D}{B}}{t_{TXperiod}} \frac{I_{start} t_{start} + I_{TX} \frac{D}{B}}{t_{start} + \frac{D}{B}} + \left(1 - \frac{t_{start} + \frac{D}{B}}{t_{TXperiod}} \right) I_{sd} , \end{aligned} \tag{4.4}$$

where t_{start} is the wake-up time from shutdown mode to TX mode, D is the data length, B is the radio bitrate, $t_{TXperiod}$ is the time between radio transmissions, I_{start} is the average current during wake-up, I_{TX} is the current during transmission and I_{sd} is the current in shutdown mode. From intuition, the current of the microcontroller is calculated as

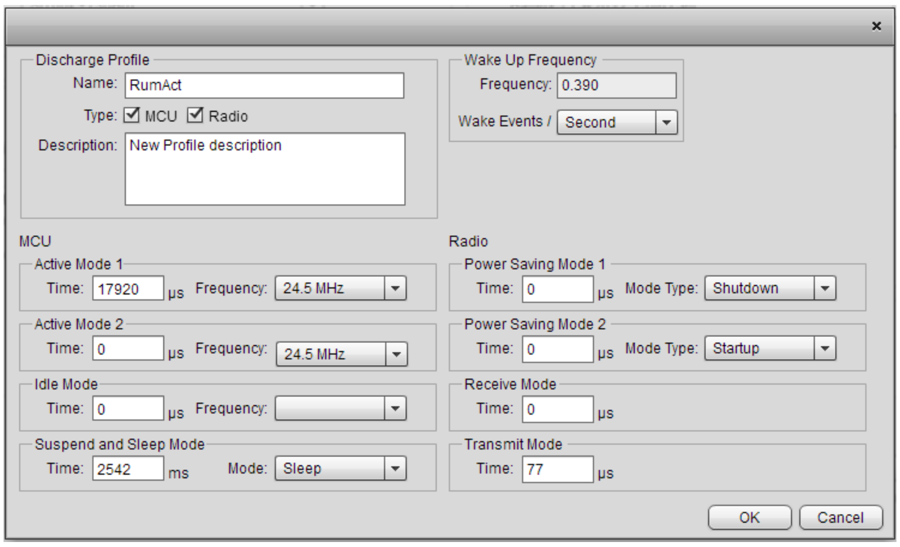
$$\begin{aligned} I_{\mu C} &= \left(U_{\mu C} + U_{other} \right) I_a + (1 - U_{\mu C} - U_{other}) I_s \\ &= \left(\frac{k_o}{k_t} \frac{F_{Intel}}{F_{\mu C}} T_{Intel24} + U_{other} \right) I_a + \left(1 - \frac{k_o}{k_t} \frac{F_{Intel}}{F_{\mu C}} T_{Intel24} - U_{other} \right) I_s , \end{aligned} \tag{4.5}$$

where I_a is the current in active mode, I_s is the current in sleep mode and the rest of the parameters are introduced in 4.2.2.1. U_{other} is the utilization for which the MCU does other things. $U_{\mu C}$ is explained in more detail by (4.6).

The model (4.1) proved to give an accurate estimation for the prototype configuration, giving an estimated battery life of 40 days (compare to Figure 4-1). However, in the final product, the $t_{TXperiod}$ will be very long, resulting in that the main current is heavily dependent on the CPU utilization $U_{\mu C}$, which in turn depends heavily on the estimation of the optimization factor k_o (further mentioned in (4.6)). It should however be pointed out that when writing the C-code for the microcontroller, several optimizations are made, using mostly addition, subtraction and bit-shifts, reducing the expectedly high k_o above to about 1.5. The battery life expectancy also depends heavily on the expected charge leakage k_b when the time is several years.

If an algorithm takes 1.88 s to compute in MATLAB, and the optimization factor is expected to be between 1.2 and 10 and the charge leakage is 3% per year, then a battery life of 2.8 to 7.7 years is realistic. To know for sure, a tag must be programmed and the current must be measured.

There is a tool on Silicon Labs website called Battery Life Estimator [26] that can be used to calculate the battery life and average power consumption of their 8051-microcontrollers as well as their Si10XX single-chip solution microcontrollers with built-in RF transceiver. The calculator requires only knowledge about time spent in different modes, as seen in Figure 4-2. It is also required to select which MCU that is used and what battery that is used. The input parameters are based on that the sensor buffers 32 samples and then wakes the MCU up to do calculations. In the case of a 12.5 Hz sampling frequency, this translates to a wake up frequency of 0.39 Hz. It is estimated that the computation time for preprocessing is 7.9 ms and other activities take up 10 ms at every wakeup. The actual radio transmission takes 28.9 ms every 900 seconds, which equates to 77 µs every 2.56 seconds.



Discharge Profile
Name: RumAct
Type: ☑ MCU ☑ Radio
Description: New Profile description

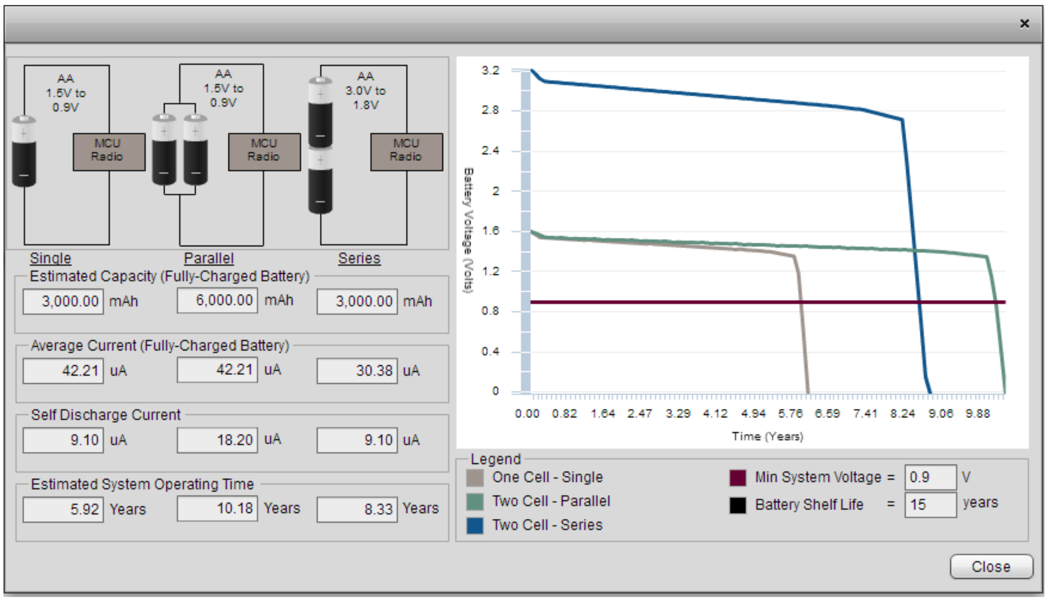
Wake Up Frequency
Frequency: 0.390
Wake Events / Second

MCU
Active Mode 1 — Time: 17920 µs Frequency: 24.5 MHz
Active Mode 2 — Time: 0 µs Frequency: 24.5 MHz
Idle Mode — Time: 0 µs Frequency:
Suspend and Sleep Mode — Time: 2542 ms Mode: Sleep

Radio
Power Saving Mode 1 — Time: 0 µs Mode Type: Shutdown
Power Saving Mode 2 — Time: 0 µs Mode Type: Startup
Receive Mode — Time: 0 µs
Transmit Mode — Time: 77 µs

Figure 4-2. Parameters in Silicon Labs Battery Life Estimator.

The output is in the form of average current of the chip and estimated system operating time and a time-graph showing voltage and limits. Note that the average current of the sensor has to be added, which with the current MEMS sensor would be 8 µA. The results from this calculator differ with 1.7% from the calculation model (4.1) using an optimization factor k_o of 1.5. The comparison verifies that the battery life calculation model is accurate.



	Single	Parallel	Series
Estimated Capacity (Fully-Charged Battery)	3,000.00 mAh	6,000.00 mAh	3,000.00 mAh
Average Current (Fully-Charged Battery)	42.21 uA	42.21 uA	30.38 uA
Self Discharge Current	9.10 uA	18.20 uA	9.10 uA
Estimated System Operating Time	5.92 Years	10.18 Years	8.33 Years

Legend
🟫 One Cell - Single
🟩 Two Cell - Parallel
🟦 Two Cell - Series
🟥 Min System Voltage = 0.9 V
⬛ Battery Shelf Life = 15 years

Figure 4-3. Results in Silicon Labs Battery Life Estimator.

4.2.2. Software

Independently of how the hardware is chosen, there are many options on how to partition the functionality with regards to signal processing and classification. And within a certain partitioning, there are choices for modes of operation. The main components in the system were introduced in Figure 2-1. As seen, there are possibilities to do computations in the tag sensor, the tag microcontroller, the receiver microcontroller, the system controller, or even on a remote computer. Some of these options can easily be ruled out when looking back at Figure 4-1, where the tag transmits large amounts of data. Since the radio traffic must be kept to a minimum, then this rules out doing preprocessing on any other device than on the tag. However, if the preprocessed data can be formatted to use little data for long time spans, then that data could be transmitted and any other device could do advanced classification. But since the preprocessing, segmentation and feature extraction of the signals require at least half of the computations, there is probably not much energy left to save, since transmitting processed data might as well draw more power than classifying it on the tag. It would also add complexity and more dependencies to the system if other devices would carry out processing.

The mode of operation has a slight impact on the power consumption. The goal is to have the processing unit in the tag in sleep as much as possible. The signal processing could be done in these ways:

1. The processing unit does the signal processing directly after receiving sensor data.
2. The sensor uses an internal DSP to do as much calculation as possible and sends preprocessed data.

The activity classification could be done in the following ways, but depending on classification method, there might only be one option.

1. The classification is done directly after signal processing.
2. Processed signals are accumulated in the memory and are then classified.
3. Processed signals are accumulated and transmitted and are then classified by a stationary device.

Wireless data transmission could also be done in different ways:

1. Data is transmitted regularly but infrequently.
2. Data transmission is triggered by an activity or a position, such as passing a transponder, or reaching above a certain accumulated rumination time.

Other factors that have an impact on power consumption are data types and computation simplification. Data types of 8 or 16 bit integers require little memory, especially when using arrays, and compute much faster than floats and doubles on an 8-bit MCU. Some mathematical computations can be simplified with bit operations, such as bit-shifting. A multiplication of 64 can be simplified as shifting the integer 6 steps. Division by 64 can be simplified as shifting back 6 steps. Division and multiplication by other numbers than 2^n (powers of two) is not recommended, but is also possible to optimize by combining bit-shifting with addition and subtraction.

4.2.2.1. Computation time analysis

Different methods and algorithms require different amounts of computation. Four of the most interesting algorithms/methods have been profiled in MATLAB using an Intel® Core™ i5-520M Processor (3M Cache, 2.40 GHz). Execution was limited to one thread of the CPU. The timed execution classified 24 hours of data. In the profiler it is possible to see what functions that take longer time and how many times they are called. It is also possible to see how much time each line of code takes up.

A function that is very computation heavy that has been used is the segmentation which utilizes the `mean` command for each segment. Doing frequency analysis also requires heavy computation. Many functions in MATLAB are optimized, and replacing built-in functions with multiplication, division, addition and subtraction increases the computation time significantly. Many filters used in the development are very sophisticated and have zero phase shift. But if they are to be used in a low-power microcontroller, then simplifications have to be made.

The utilization of the microcontroller to perform a certain algorithm is estimated from the profiler information in MATLAB. If the optimization factor k_o of the Intel 64-bit processor is estimated as 1.5 times more operations per second than an 8-bit microcontroller, then

$$U_{\mu C} = \frac{k_o}{k_t}\frac{F_{Intel}}{F_{\mu C}}T_{Intel24} = \frac{1.5}{24 \cdot 60 \cdot 60}\frac{2.4 \cdot 10^9}{25 \cdot 10^6}1.88 = 0.0031, \tag{4.6}$$

where k_t is a time scaling factor, F_{Intel} and $F_{\mu C}$ are clock frequencies and $T_{Intel24}$ is the time it takes to compute 24 hours of data in MATLAB. In the example above, the computation time of a rewritten version of an example algorithm is used. The rewritten version is specially designed for implementation, using unsigned 8-bit integers and no hidden functions.

4.2.3. Communication

The communication in the system can be split into wireless and wired communication.

4.2.3.1. Wireless communication

There are 4 main modes of wireless communication, namely radio, free-space optical, sonic and electromagnetic induction. Radio is capable of long-range line-of-sight communication as well as short-range. Free-space optical is usually in the form of infrared (IR) light and is popular in consumer electronics. Sonic is in the form of ultrasonic sound, which makes it short range. Electromagnetic induction is usually used in short-range RFID tags.

Since the application requires above 10 m in range, this rules out electromagnetic induction and sonic communication. In order to decide on a solution within radio and free-space optical communication, a comparison chart, Table 4-4, is introduced with some properties of interest for different communication protocols.

Table 4-4. Comparison of different wireless communication protocols. A summary from [27], [28].

	ZigBee 802.15.4	**WiFi 802.11g**	**Bluetooth 4.0**	**UWB**	**IR Wireless**
Data rate	20-250 kb/s	11-54 Mb/s	1 Mb/s	100-500 Mb/s	20 kb/s – 16 Mb/s
Range	10-100 m	50-100 m	10 m	<10 m	<10 m
Topology	Ad-hoc, P2P, star, mesh	Point to hub	Ad-hoc, small networks	Point to point	Point to point
Power consumption	Very low	High	Medium	Low	Low

4.2.3.2. Wired communication

There are many ways to do the topology of the wired communication and depending on choice of technology the topology varies. The main components of the wired communication are receiver stations and a system controller. No matter how the receivers are connected, the received information must reach the system controller. This could either be done using, for example, a bus topology, a star topology or a linear topology. In the bus the different nodes use identification in the beginning of each message. Each node has a unique ID-number and the number of nodes is limited by the set of ID's and the bandwidth of the bus. In a star topology every node is directly connected to the system controller. This limits the number of possible nodes to the number of available communication ports on the system controller. In a linear topology, every node is only connected to its neighbors. The nodes repeat the information until the information reaches the system controller. The number of nodes is limited to the bandwidth and utilization of the nodes.

There are a wide variety of fieldbus standards, including CAN, but there's also special protocol developed by DeLaval/AlfaLaval that works well in a dairy farm environment, called the Alcom bus. Alcom is a 50 kbps token based RS-485 bus.

4.3. Conclusion

This subchapter explains what system solution that was chosen as a recommendation for a future product and why. It also elaborates with the chosen solution, giving a small discussion about the choices. Many choices were determined by the fact that the system was already built and tweaked for this specific project. However, as this would result in a system recommendation, some parameters are chosen to fit the end user product rather than a high performance data collection system.

- Hardware
  - An accelerometer was chosen as sensor, since it has very low power consumption and since it is already used in the DeLaval Activity meter system.
  - An 8-bit MCU was chosen as processing device in both the tags and the receivers, since it is already used in the DeLaval Activity meter system. The processing power on the tag should be partitioned between the MCU and a DSP in the accelerometer.

    The receivers could, for ease of development, use the same MCU as the tags. It is also recommended to use MCU’s with integrated wireless communication.
    - 12 VAC was chosen as a power source for the receivers, since it eases the installation of the system and since it is already used in the DeLaval Activity meter system. A battery of manganese dioxide lithium type with 3000 mAh was chosen for the tag since it has more than 10 times the capacity of a CR2032 and since it is already used in the DeLaval activity meter system.
    - Aside from what is discussed in this chapter, the tags should have the same kind of plastic filled housing in order to resist tough environments, but the interface with the collar should be changed so that it locks onto the collar instead of sliding freely on it.
- Software
    - The DSP in the sensor should do as much preprocessing as possible and buffer samples as much as possible to be sent to the microcontroller, giving the microcontroller more sleep time.
    - Depending on if the classifier is a frame classifier or if a longer time series is required, the classification should be done directly after receiving data, reducing memory usage, or regularly with accumulated processed signals allowing time aspects to be taken into account.
    - The data transmission from the tags should be time triggered, regularly but infrequently, keeping the power consumption down. This is also how it is already being done in the DeLaval Activity meter system.
    - Rumination and activity data should also be sent twice for redundancy. Each transmission should contain the newest data and the previous data. This is also how it is already being done in the DeLaval Activity meter system.
    - Optimizations of computations, in the form of data types and bit shifting operations, are done wherever possible to decrease power consumption.
- Communication
    - For wireless communication between tags and receivers, ZigBee 802.15.4 was chosen because of the long range combined with very low power consumption and also since it is already used in the DeLaval Activity meter system. For testing, the carrier wave frequency is tweaked down to 418 MHz instead of 433 MHZ to not disturb other activity meter systems from DeLaval present at the test farm.
    - For wired communication between receivers and the system controller, Alcom was chosen because of its robustness in tough environments and long distances, its ease of installation and also because it is used today in the DeLaval Activity meter system.

Some customizable system parameters are left undecided, like classification time resolution, receiver information buffer, radio transmission data package content, number of sensor axes, since these parameters give trade-offs with no clear superior setting. The mounting of the tag can be optimized with regards to statistical classification performance and also, if affected, the radio transmission quality. A discussion of these conclusions and how the choices comply with the requirements follows in chapter 8.7. Requirements.

5. Implementation

This chapter explains the details about how the main work during the project was carried out.

5.1. System setup

To get started with the project, a system has to be installed so that data can be collected. The system is a modified version of the original DeLaval activity meter system. The system consists of a system controller, receivers and the tags. A full schematic of the system can be seen in Figure 5-1. The different parts are explained below.

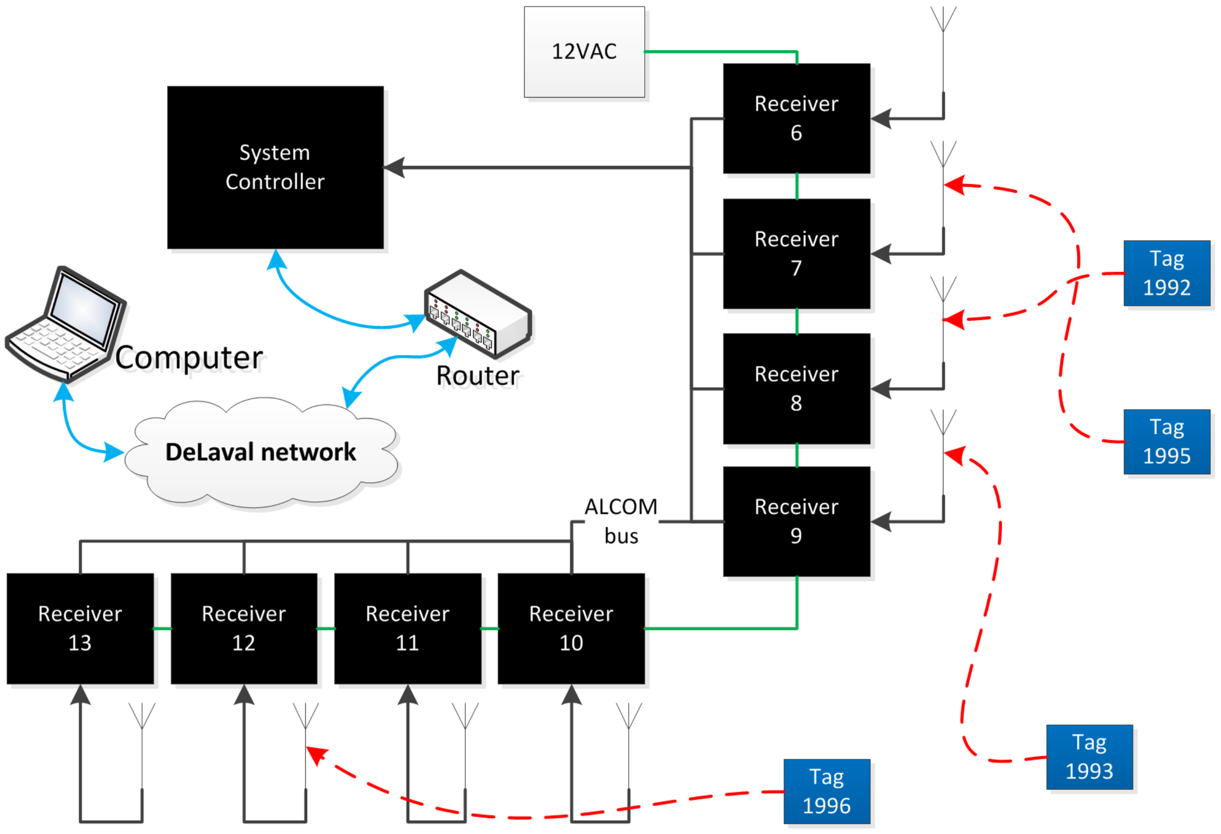


Figure 5-1. System setup at Hamra for initial testing and data collection.

5.1.1. Tags

The tags consist of a 3-axis 14-bit/8-bit digital accelerometer and an ultra-low power MCU with integrated 240–960 MHz transceiver. The tag is powered by a 3000mAh 3V battery. The systematic layout of the tags can be seen in Figure 5-2. The tags are configured to transmit data at 418 MHz instead of the usual 433 MHz carrier frequency to not disturb other equipment at the test facility. 4 channels are used for the radio transmissions. Tags with even ID-numbers transmit at even channels and tags with uneven ID-numbers transmit at uneven channels. The tags only use one channel at a time, but sends on each every other time. The wireless protocol is DeLaval-specific. The tags are activated with a transponder such as the one at any milking station. The 3-axis accelerometer samples data at 12.5 Hz and the data is sent in packages of 4 samples of each sensor axis meaning an average of 3.125 radio transmissions per second. Moreover, the accelerometer can be configured for three different sensitivities, +/- 2, 4 or 8g with 14 bit digital output, meaning 1024 counts per g at 8g sensitivity. The accelerometer also has a built-in DSP with configurable functions. The communication between sensor and MCU is I^2C. The accelerometer consumes 6 to 165 µA depending on configuration. The MCU consumes 4.0 mA in active mode and 0.61 to 0.76 µA in sleep mode with real-time clock (RTC) running.

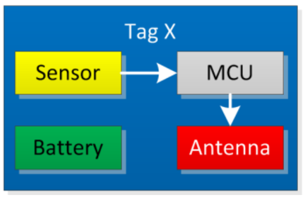


Figure 5-2. Systematic description of a tag.

In Figure 5-16 on page 36 it is further explained how the sensor axis directions relate to the cow.

5.1.2. Receivers

The receivers consist of an antenna, a microcontroller, bus communication and an AC-DC converter. The systematic layout of the receivers can be seen in Figure 5-3. The microcontroller is of the same family as the microcontroller in the tags. The receiver communicates with a bus called Alcom, developed by DeLaval. This protocol is similar to CAN, but only uses 2 wires that can be connected interchangeably. The receivers are powered with 12VAC, so the supply voltage wires can also be connected interchangeably. The address of receivers is set with 5 jumper pins. The address becomes 35-N, where N is the decimal value of the binary jumper pins. The lowest address becomes 4. The address decides which radio channels the receivers will use, as mentioned in 5.1.1. The receivers can be reprogrammed from the system controller via the Alcom bus.

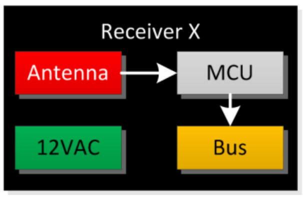


Figure 5-3. Systematic description of a receiver.

5.1.3. System controller

The system controller consists of a computer, running embedded Linux, and a big IO-board called SEBA. The system controller has both Alcom bus and CAN bus connection. The system controller is interfaced through an Ethernet cable connected either directly to a computer or through a network.

The system controller can only be accessed either with an IP-address or a hostname. A program called AlproBrowser can be used to see system status, configure settings and also run updates. A tool called ConfigTool is used to install necessary software onto the system controller. During the project, WinSCP and PuTTY were used to extract logging files and to adjust system time.

5.1.4. Installation

The system was installed at Hamra Gård in Tumba. Figure 5-4 explains the location of the different system components. Four receivers are dedicated to the milking area whilst the rest of the receivers are meant for the general area. The system controller and network router is located in the office.

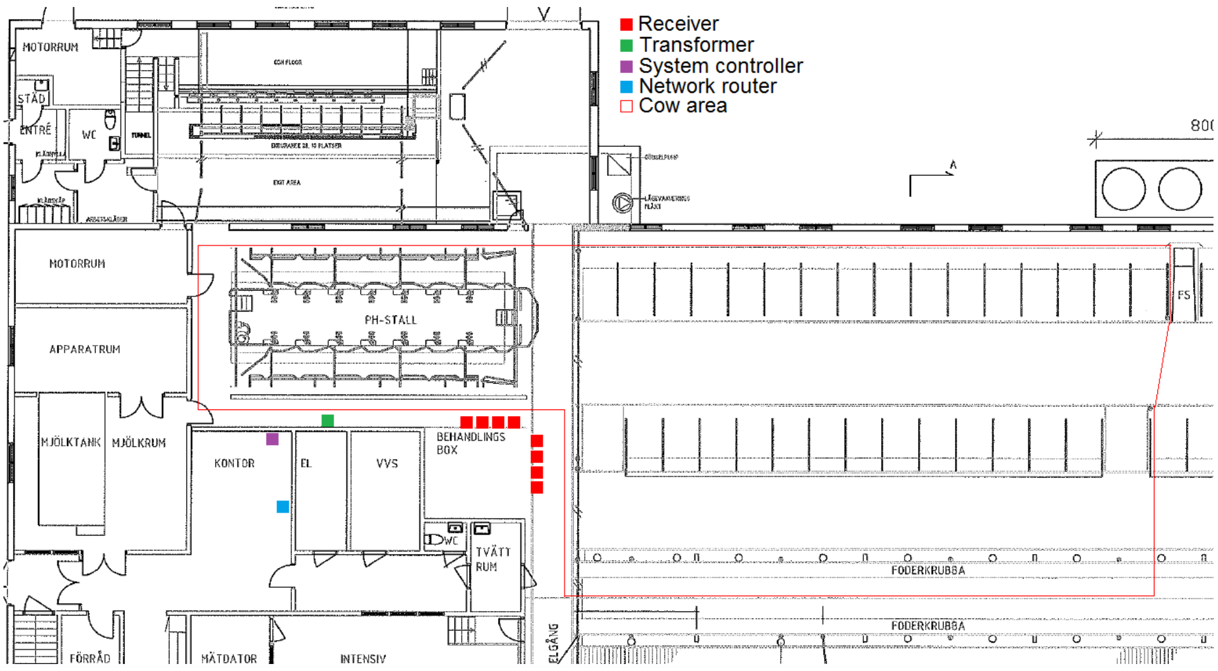


Figure 5-4. Locations of system components at Hamra.

5.2. Observation method

5.2.1. Rumination observation

The observations were done in two stages. In the first stage mounting position had to be optimized, so all sensors were placed on the same cow and observations was done for two days. More about that can be found in chapter 5.5 followed by results in chapter 6.1. After selecting the optimal mounting position on the collar, up to three cows were observed simultaneously. The time spent on observation can be seen in Table 5-1. As seen, the number of cows observed went down from 3 to 2. This is because one of the tags malfunctioned between 2014-02-28 and 2014-03-03. The tag was replaced 2014-03-06.

Table 5-1. Accumulated observation time.

Date	**Observation hours**	**Cows**	**Cow hours**
2014-02-26	1.0	1	1.0
2014-02-27	2.0	3	6.0
2014-02-28	1.2	3	3.6
2014-03-03	3.3	2	6.6
2014-03-04	2.1	2	4.2
2014-03-05	3.0	2	6.0
2014-03-06	3.1	3	9.3
2014-03-07	2.8	3	8.4
Sum	18.5		45.1

During an observation session, both continuous activities and events were registered. Continuous activities were lying, lying rumination, standing, standing rumination, eating silage, eating from feed station, drinking and salt licking. Events were transitions between standing and lying and also the bolus event in rumination. The bolus[3] event was only registered a few times since it takes a lot of concentration. The most important thing was to get the rumination logging correct.

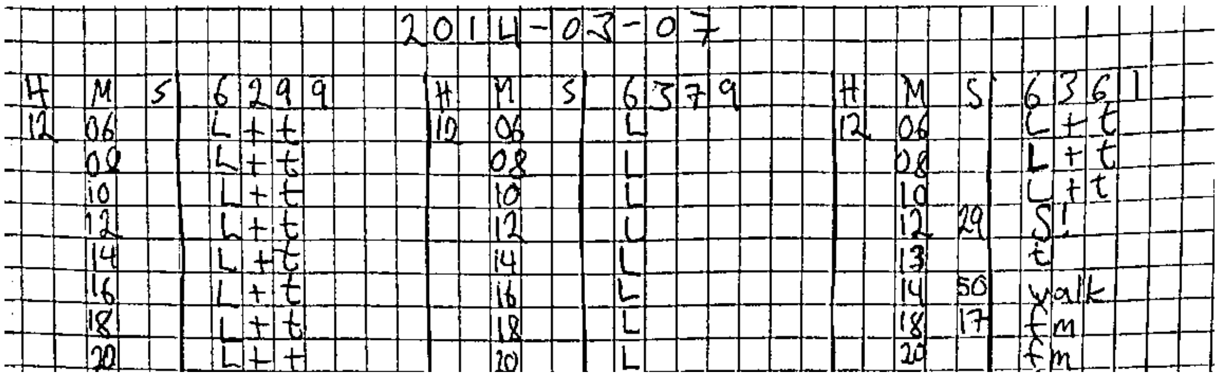
The activity of rumination can be explained as a repeated act of regurgitation[4] of boluses, which are then chewed continuously for almost a minute and then swallowed again. The cow pauses the chewing to regurgitate a bolus for 3 to 10 seconds. The whole period of the activity is usually 60 seconds. The animal might be either standing or lying during rumination, and the activity may take place at any given time. A cow typically ruminates for 450 minutes accumulated time per day. Rumination typically spans over 5 to 60 minutes per occasion.

A cutout of the observation documentation is depicted in Figure 5-5 and an explanation of the activities can be found in Table 5-2. For each cow/tag there is a time column and an activity column. As can be seen, the continuous activities are only registered every other minute, but events are registered immediately with a time stamp including seconds. For most continuous

[3] A bolus (Latin word for ball) is a mass of food that has been chewed at the point of swallowing.

[4] Regurgitation is the expulsion of material from the pharynx, or esophagus, usually characterized by the presence of undigested food.

activities, the start and end time is registered with seconds. Although walking, social activities and other anomalies were registered on paper, this information was not used later.


2014-03-07

H	M	S	6299	H	M	S	6379	H	M	S	6361
12	06		L+t	10	06		L	12	06		L+t
	08		L+t		08		L		08		L+t
	10		L+t		10		L		10		L+t
	12		L+t		12		L		12	29	S!
	14		L+t		14		L		13		t
	16		L+t		16		L		14	50	walk
	18		L+t		18		L		18	17	fm
	20		L+t		20		L		20		fm

Figure 5-5. Documentation of time and activity.

Table 5-2. Description and numbering of different activities.

Notation	Number	Description	Minutes
L	1	Lying	285
L+t	2	Lying and chewing (rumination)	637
L+s	3	Lying and bolus (rumination)	
Stand	4	Standing	129
T	5	Standing and chewing (rumination)	193
S	6	Standing and bolus (rumination)	
E	7	Eating silage	375
D	8	Drinking	54
Fm	9	Eating from feed station	104
ss/salt	10	Licking salt	28
S!		Transition from lying to standing	
L!		Transition from standing to lying	

In order to make use of the observations, it had to be digitalized into a format that was readable by MATLAB. Typical cell documents that are readable by MATLAB are csv and xlsx. Creating a timeline with one row per timestamp, the start and stop times for each activity and each tag was registered. It was also suitable to use an activity numbering so that plotting the activities over time would be easier. In Table 5-3 an extract from the digital observation file can be seen. Only continuous activities were documented in this file.

Table 5-3. Digital format of the observation data.

Tag	Date	H	M	S	H	M	S	Activity	Activity	Minutes
1997	20140307	12	6	0	12	24	0	2	L+t	18,0
1997	20140307	12	26	0	12	31	0	1	L	5,0
1997	20140307	12	33	0	12	35	0	4	stand	2,0
1997	20140307	12	43	0	12	46	15	9	fm	3,3
1997	20140307	12	47	0	12	51	0	8	d	4,0
1997	20140307	13	2	0	13	4	0	4	stand	2,0
1996	20140307	12	6	0	12	12	20	2	L+t	6,3
1996	20140307	12	12	40	12	14	20	5	t	1,7
1996	20140307	12	18	17	12	22	0	9	fm	3,7
1996	20140307	12	22	30	12	25	0	8	d	2,5
1996	20140307	12	25	30	12	30	0	7	e	4,5
1996	20140307	12	32	0	12	44	0	7	e	12,0

Observation was done at between 1-10 meters of distance, depending on if all cows were close together or spread out. “Wall time” used for the timestamps was from a wrist watch which was calibrated 2014-02-21 and has a ±30 s drift off per month.

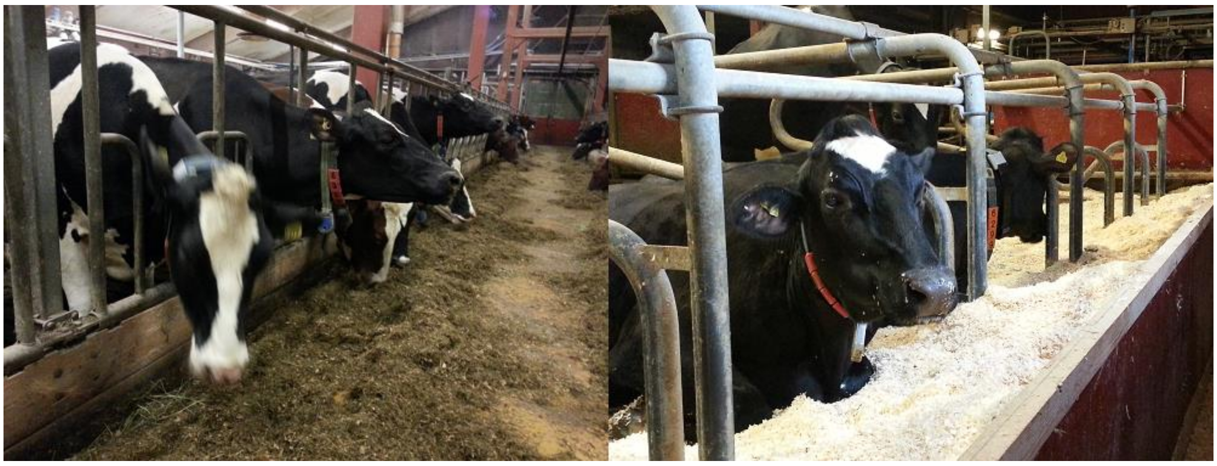


Figure 5-6. Cows under observation during activity 7 (silage eating) and 2 (lying and ruminating).

5.2.2. Tied-up observation

Observation in the tied-up dairy farm was done using a security camera mounted high up on a wall and a similar setup as in the first test farm. This test aimed at using lying/standing to detect heat, with a hypothesis that a cow on heat stand more often than normal. Using this setup, it was tried to use the sensor to detect transition events between lying and standing. The camera recorded 4 cows from 2014-03-26 to 2014-04-29. The camera is equipped with IR light, so that objects are visible up to 20 meters from the camera at night. The view of the camera during day time and at night can be seen in Figure 5-8. The field of view of the camera is explained by Figure 5-7. The security camera system records the video files onto an external hard drive with 5 minutes per video clip.

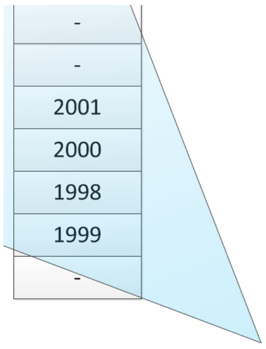


Figure 5-7. Camera field of view (FOV) and cow tag numbers.

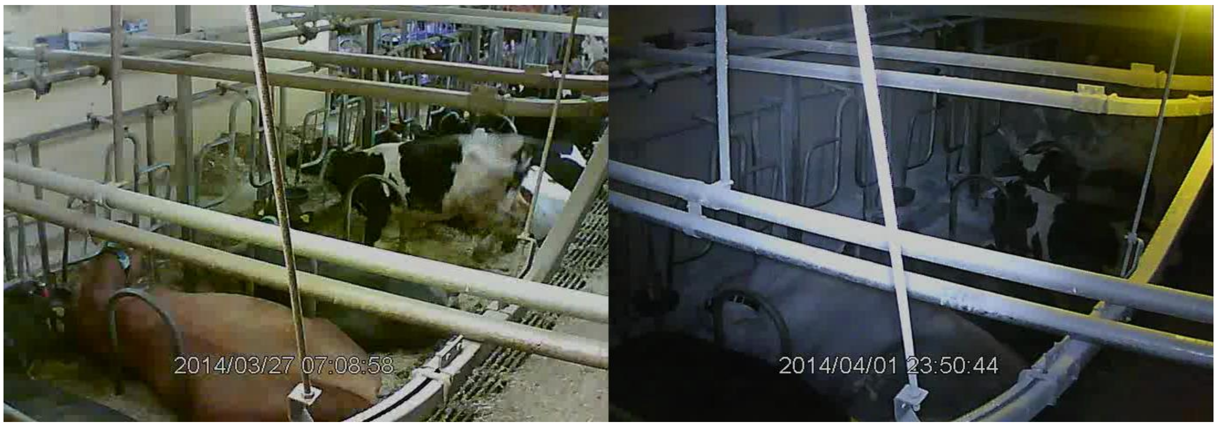


Figure 5-8. Screenshot from the security camera during daytime and at night.

As can be seen, the video is watermarked with continuous time. These timestamps were used to register all lie-down or stand-up events. Since this work was time consuming in relation to benefit, only 12 hours of observation were registered manually, making a total of 48 observation hours. An extract from the lie-down/stand-up documentation is listed in Table 5-4. The first event in Figure 5-8 can also be found in Table 5-4.

Table 5-4. Extract from the lie-down/stand-up documentation.

Cow	**Date**	**Time**	**Event**
1999	20140327	150903	L
1999	20140327	151201	S
1999	20140327	172343	L
2000	20140327	061013	L
2000	20140327	061612	S
2000	20140327	063417	L
2000	20140327	070858	S
2000	20140327	071605	L
2000	20140327	074123	S

5.3. Data acquisition and preprocessing

The sensor data is stored in a folder on the system controller. A new csv-file is created every minute for each sensor, and each file is usually 50kB is size. This means that the files need to be extracted regularly to avoid filling the system storage of 1GB. A script was set up to automatically extract all files that contain yesterday's date in the name via FTP into a folder with yesterday's date. The extraction triggers at one hour after midnight every night. The only bug is that yesterday's date is calculated as an integer of today's date minus one. So 20140331 becomes 20140400. The files can also be extracted manually over FTP.

The files are read in MATLAB with the `dlmread` command. The information in the file can be seen in Table 5-5. The device-specific sampling frequency is calculated using the RTC clock frequency and the median RTC tick differential (rtcDiff) that is transmitted in each transmission. Each transmission contains four samples (newNdata) and the previous four samples (oldNdata). It also contains battery voltage (batU).

Table 5-5. Raw data structure in each csv-file in the system controller.

ReceivedTm	tmDiff	rtcTick	rtcDiff	batU	Sample-Index	newXdata	newYdata	newZdata	oldXdata	oldYdata	oldZdata
1393546114.222	1.222	1847798266	10631	2.902	0	16	-933	-464	17	-933	-471
1393546114.222	1.222	1847798266	10631	2.902	1	23	-922	-474	20	-918	-475
1393546114.222	1.222	1847798266	10631	2.902	2	9	-907	-478	12	-907	-479
1393546114.222	1.222	1847798266	10631	2.902	3	20	-924	-474	15	-918	-468
1393546114.464	0.464	1847808897	10631	2.902	0	15	-924	-472	16	-933	-464
1393546114.464	0.464	1847808897	10631	2.902	1	16	-916	-475	23	-922	-474
1393546114.464	0.464	1847808897	10631	2.902	2	20	-921	-468	9	-907	-478
1393546114.464	0.464	1847808897	10631	2.902	3	15	-922	-469	20	-924	-474

The first step is to go through the data and repair. If the RTC clock overflows, then the rtcDiff will be almost negative infinity. This is fixed by just assuming the median rtcDiff. If an old transmission is received again, the rtcDiff is negative, and is therefore discarded. If a transmission is delayed, there are three options:

1. Shift time towards previous transmission. This method eliminates all anomalies in the time-domain, but will however in the long-term increase the time offset.
2. Create pseudo-data for the time when no transmission was received. Median rtcDiff is assumed between transmissions. The pseudo-data is initiated as SHRT_MIN (the minimum value of a short integer) since this means that the data is corrupt. This will be handled later.
3. Leave it as it is. In the time-domain this will show up as a gap in time where no data points are available.

After that, the output time axis is initiated with correct time-stamps. The rest of the output is then filled up by stepping one transmission at a time. The normal case is that newNdata is copied, but a failure handling procedure is required.

If the data in a transmission is recognized as SHRT_MIN, then the next transmission might contain uncorrupt oldNdata and repair the damage. If oldNdata is also corrupt or missing, then:

1. A sequence copy can be made, replicating all data before the missing data. This is a good solution to not get any interruptions or anomalies in the output. If the accelerometer signal character is very stable when the failure happens, this solution will make it look like there never was a failure.
2. The last known newNdata can be copied and applied to all failed data. This will turn the accelerometer signal into a sudden flat line. This can misguide the representation in later use.
3. If 1 and 2 is not possible, then 1 and 2 is repeated but using future data instead of previous data.

Lastly, the axis data is converted to G-scale, so that -1024 means -1g and 1024 means 1g. After importing data from a csv-file and handling the faults, the data matrix is appended to the rest of the data from other csv-files. The time axis is also adjusted to continue from the last time value in the previous csv-file.

5.4. Segmentation and feature extraction

When developing an algorithm to classify a motion pattern, different features of the raw signal are extracted to see what really characterizes different activities. In this case the features were extracted before segmenting. This is because the segmentation method here was decimation method that decreased the time resolution of signals in order to reduce computations on the classification. In this study, segmentation was applied to any data that did not need high time resolution for further processing or feature extraction. The segmentation method can however also take up a lot of computation depending on how it works. In this case, the segmented values x_s were calculated as

$$x_S\left[i\right]=\frac{\sum_{n=T_S f_s(\mathrm{i}-0.5)}^{N=T_S f_s(\mathrm{i}+0.5)} x\left[n\right]}{T_S f_s}\quad , \tag{5.1}$$

where T_S and f_s are the segmentation time and sampling frequency, respectively. The moving index span $n\ldots N$ is calculated so that half of the data points in the window are before the data point of which the time is the same as the output data point, and half of the data points are after. Simplified, it is a moving average filter with downsampling so that the period time of the output is the same as the window length. The extracted features were chosen by intuition of studying the signal and by inspiration from other studies.

5.4.1. DC-level

The first feature of interest is the “DC-level” of the signal, meaning the low-pass filtered raw data. The filter was a 4th order Butterworth with 0.5 Hz cutoff frequency. That filter is very effective at removing signal noise and retaining the general shape. In Figure 5-9 the DC-level or “mean” of the raw data is shown together with the raw data. This kind of filter isn’t realistic to use on a battery powered embedded system, but it does however give the best possible result. For implementation, a 1st order IIR is more suitable. The DC-level will henceforth be referred to as x_f.

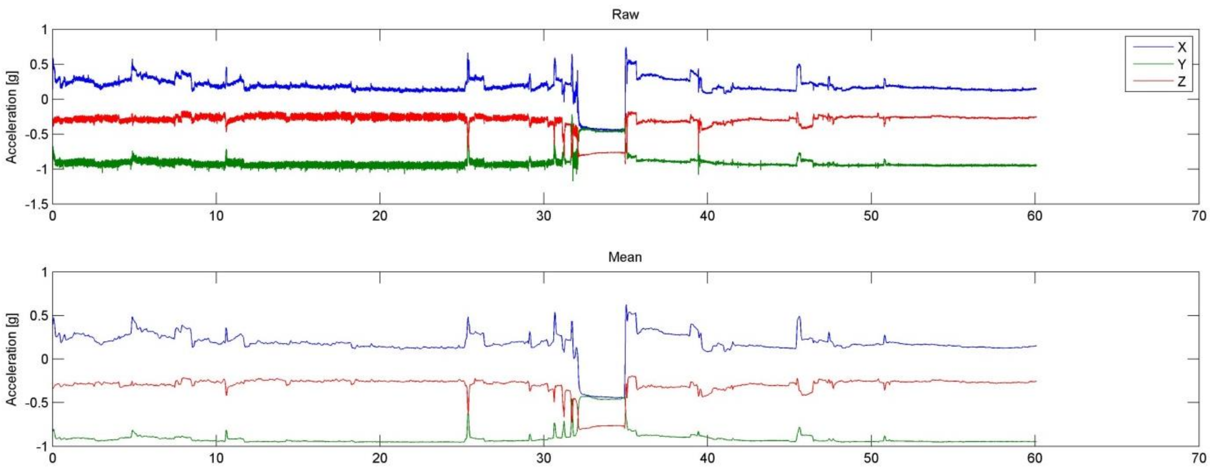


Figure 5-9. The low-pass filtered raw signal.

5.4.2. Standard deviation

The second feature of interest is the standard deviation of the signals. This indicates the intensity of different activities and makes it possible to distinguish between resting, rumination and eating more easily. During rumination it is possible to see all the pauses characterized by periodic dips in the signal for each bolus. The usual calculation of standard deviation is

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2} \quad \text{where} \quad \mu = \frac{1}{N}\sum_{i=1}^{N} x_i \,. \tag{5.2}$$

But in this case the mean μ, or x_f, is changing and thus it is more suitable with continuous computations using filters. If $f(x)$ represents a low-pass filtering of x, then the variable x_{var} can be calculated as

$$x_{\mathrm{var}} = f\left(\left|x - x_f\right|\right). \tag{5.3}$$

This is neither standard deviation nor variance since that requires the difference between x and x_f to be squared, instead of just removing the sign. But it does however represent the same signal characteristics at a lower cost.

5.4.3. G-vector

The three accelerometer axes can be combined into a single value, the vector length of true acceleration. This value is independent of sensor position. It is calculated as

$$v_L = \sqrt{x^2+y^2+z^2} - 1. \tag{5.4}$$

However, this is not very useful for a classifier, since it crosses zero all the time, but the standard deviation of it could be, since it explains the total “energy” of the activities. If $f(x)$ represents a low-pass filtering of x, then

$$v_{\mathrm{var}} = f\left(\left|v_L\right|\right). \tag{5.5}$$

5.4.4. Frequency character

A feature that is hard to implement, but that is very interesting is the frequency character of the signal. The FFT should only be applied to the high-pass filtered raw data (the AC-part) to avoid a high gain in the lower frequency coefficients. The output from this feature extraction is segmented to reduce the number of FFT computations. Before computing the frequency character over time, it has to be decided how long time series the FFT should run on, T_{FFT} and what segmentation time, T_S that should be used. The time T_{FFT} decides the resolution and stability of the output. Longer T_{FFT} increases stability and frequency resolution, but also smoothens the output over time. In this case, 5 seconds of data was used, meaning that if the segmentation time would be 2.5 s, then the FFT would be done with 50% overlap. In case of a segmentation time of 10 s, then only 50% of the data would be used for FFT. The FFT was done using the `fft` command in MATLAB.

There is a specific pattern that characterizes rumination. This frequency is generally 2.5 Hz but can vary between 2 to 3 Hz during rumination. This is most probably an overtone to the fundamental frequency of 1.25 Hz which is the chewing frequency.

Sometimes it isn’t enough to use a single sensor axis when doing a frequency analysis. The signal strength in either the y- or z-axis may suddenly become much weaker, thus a rumination period might be missed. This is solved by putting the frequency spectrums of different sensor axes on top of each other, as layers, and only using the strongest frequency coefficient in each layer. This takes a lot of computation, but the result is much more robust.

More about the computational chain will follow in 5.6 on page 39, but first it will be explained how the above mentioned signal features were enhanced by sensor mounting optimization.

5.5. Mounting optimization

In order to maximize the sensitivity and reduce the radio transmission energy consumption, the mounting position of the sensor was investigated. The mounting position was limited to the collar. The sensor can be positioned by using the neck strap ID sleeves or black plastic sleeves. A counterweight is added at the bottom of the collar to keep the collar from rotating. Four different sensor positions were investigated, as can be seen in Figure 5-15. The four sensors collected data simultaneously while paired with observation data. The same axis could then be compared among the sensors at different times of interest, such as when the cow was ruminating.

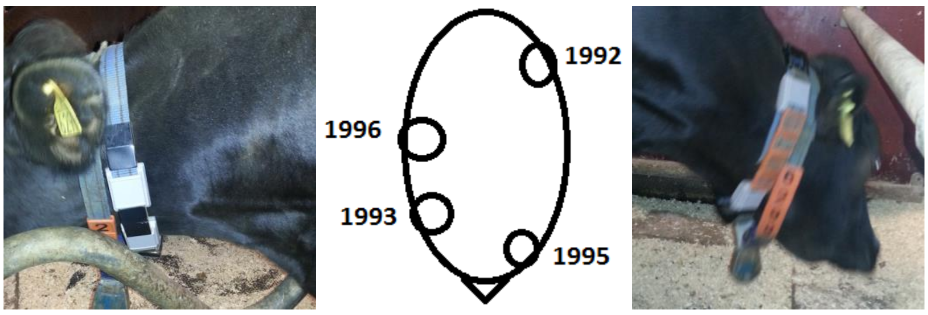


Figure 5-10. Sensor positions on collar during mounting optimization.

Regardless of sensor position, the sensor axes directions can be explained. Figure 5-16 helps with the explanation below. The x-axis is in the direction of the neck, pointing in the same direction as the head, resulting in a positive value when the head or neck is tilted upwards. The y-axis is in the direction of the collar, giving either positive or negative values depending on if the sensor is on the right or left side of the collar. The z-axis is perpendicular to the cow skin and will therefore give either a positive or negative value depending on if the sensor is on the upper or lower part of the collar.

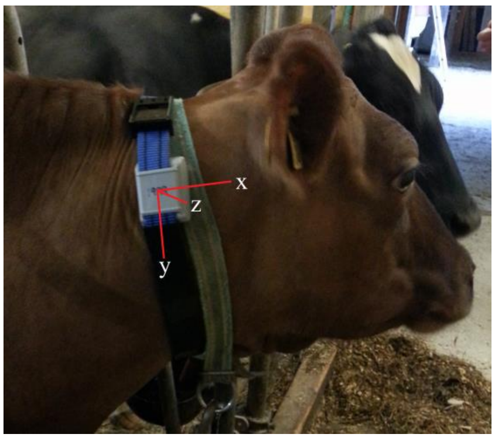


Figure 5-11. Sensor axis directions when the tag is mounted on the cow collar.

To decide the best position, the signal of each sensor was investigated during certain activities. A score from 1 to 3 was given for how distinguishable the signal character was in relation to other activities. A table describing this is found in chapter 6.1.1. The decision of best mounting position depends on the average distinguishability score across all activity types and also all axes.

Lastly, it had to be investigated if and how the mounting position would affect the radio transmissions from the tag to a receiver antenna. The performance was measured in number of transmitted sensor samples and number of transmissions with broken data during 24 hours. Theoretically, a sampling rate of 12.5 Hz should give 1.08 million samples per 24 hours. This is visualized in Figure 6-2 found in chapter 6.1.

Other than the position of the sensor, it still has to be decided if the collar should be tight or loose. No experiment was made for this, the general idea can be explained with a set of rules.

- It should not be so tight that the cow is disturbed by it.
- It should not be so tight that the counterweight doesn’t work.
- It should be tight enough so that the sensor moves together with the cow body.

For the above mentioned reasons, it seems like medium tight is optimal. This choice is also supported by the instructions in Figure 6-1 in chapter 6.1.

5.6. Algorithm development

5.6.1. Rumination

Multiple different approaches and algorithms were tested for rumination detection. Below, one of these approaches is described, which also yielded the highest performance.

5.6.1.1. Support Vector Machine

Instead of analyzing signal features and then putting up rules from intuition and testing them, the classification can be determined automatically and with more accuracy. If you consider a set of signal features upon which a certain activity is distinguishable, then a line or hyperplane separating the activities would be the classification. See Figure 5-17 for an example of how two activities, green and red, are characterized by two signal features on the axes. The black line would be the separating hyperplane. An SVM is trained by pairing signal features with "the truth", or supervised observations. Running the feature extraction on 4 observation periods[5] per cow and pairing the features with observation data, it is possible to train an SVM, but there will be too many training data points. To reduce the number of data points in the training set, only 1/10th of the data points that were correctly classified in a first iteration were saved. The last part is important in order to reduce the number of deviations in the training set. The deviations can also have occurred due to unsynchronized time axes of the system controller and observation timestamps.

The SVM was trained with the `svmtrain` command in Bioinformatics Toolbox in MATLAB. The inputs to the function are the feature arrays, the truth array and the kind of kernel function to use. The kernel chosen for this was the radial basis function (RBF) since it often gives the best accuracy. The RBF is based on testing the Euclidean distance between a support vector and a testing point. It is not easy to understand exactly how this is used to construct a hyperplane, see [29] for further reading. The function can also be extended with a specific scaling factor σ for the RBF kernel and a box constraint c for the soft margin, but these values were set as default, which is 1. In Figure 5-17 the trained classifier rumination is shown. A blue line is added to the figure, showing how a linear kernel function would have looked. A linear kernel function requires less computation. The SVM can only be visualized in 2D.

[5] An observation period is the time span during which a cow was observed at a certain day.

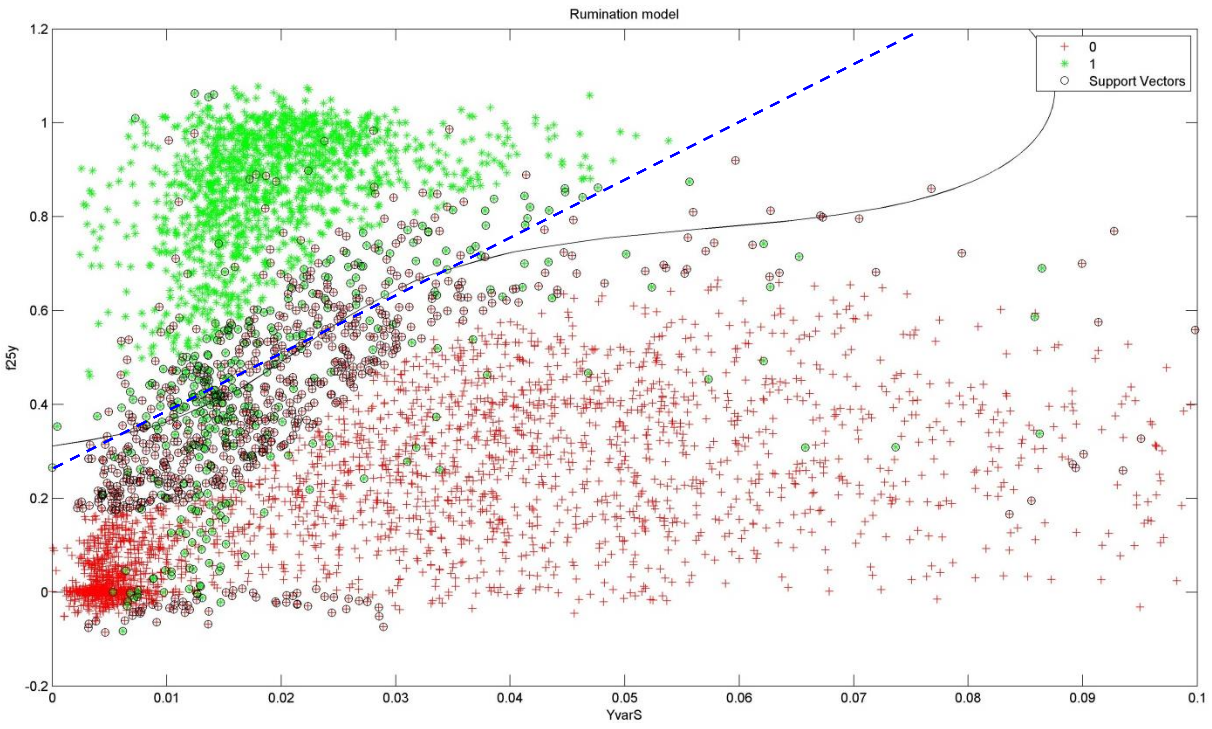


Figure 5-12. Support vectors and hyperplane for rumination. Green is rumination, red is all other activities. There is no unit for the signal features on the axes, but the x-axis corresponds to signal variability and the y-axis corresponds to a certain frequency characteristic.

Sometimes eating is accidentally classified as rumination. This can be solved by making a second SVM that has learned eating. The procedure is the same as for learning rumination. Depending on what features that are used, the accuracy of the classifier varies. Regardless of which features that were chosen for the eat model, the classification was very poor, since eating has a bit of chaotic motion pattern which causes Inter-Class Similarity, as seen in Figure 5-18. The non-linear classifier might be replaced by the same linear classifier used for rumination, but with an added boundary to reject idle activities, visualized as blue lines in the figure.

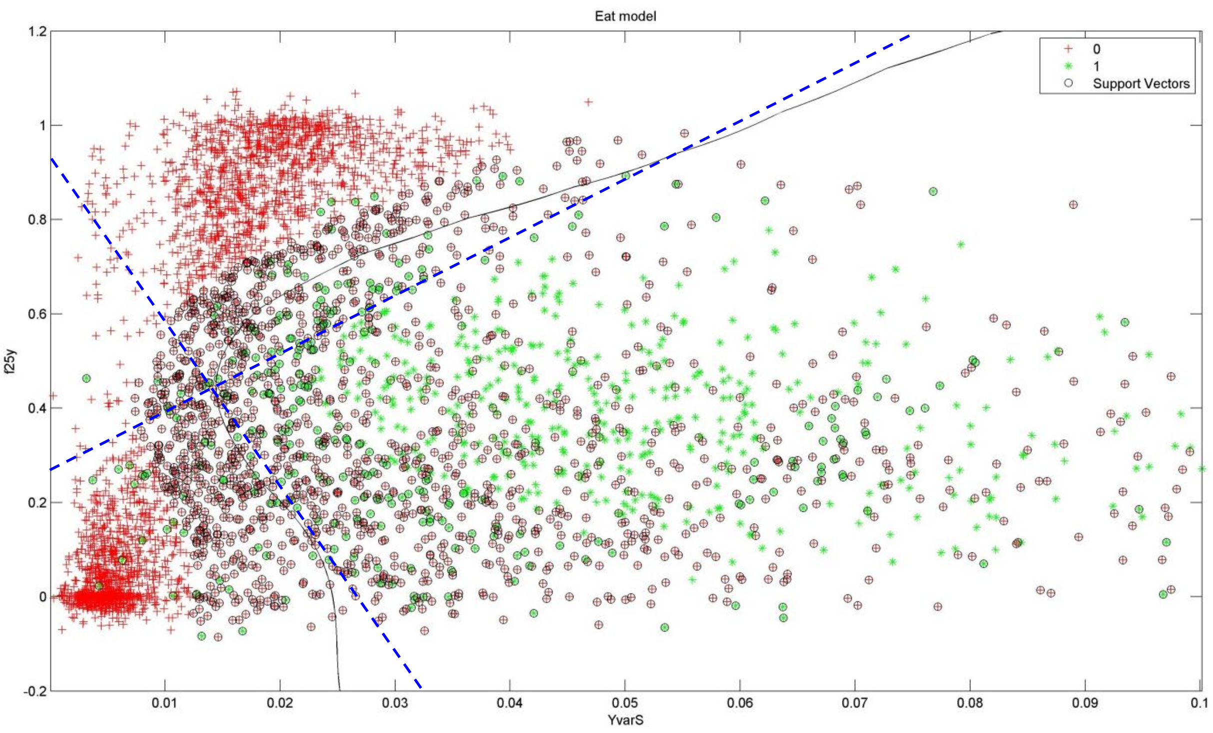


Figure 5-13. Support vectors and hyperplane for eating. Green is eating, red is all other activities. There is no unit for the signal features on the axes, but the x-axis corresponds to signal variability and the y-axis corresponds to a certain frequency characteristic.

The classification was done with the `svmclassify` command in Bioinformatics Toolbox in MATLAB. The `svmclassify` function uses results from `svmtrain` to classify vectors x according to the following equation:

$$c = \sum_i \alpha_i k\left(\mathbf{s}_i, \mathbf{x}\right) + b \tag{4.7}$$

In this equation, α_i are weights, b is the bias, and k is a kernel function of the support vector $\mathbf{s}_i$ and $\mathbf{x}$ is the feature vector [30]. The index *i* is the index of each support vector. Using the RBF kernel function, equation (4.7) turns into

$$c = \sum_i \alpha_i e^{\gamma \|\mathbf{x}-\mathbf{s}_i\|^2} + b. \tag{4.8}$$

As understandable, when the quantity of support vectors are in the range of 500-1000 as in Figure 5-17 and Figure 5-18 the computation cost for classifying each segment becomes high. It also requires a lot of memory on an embedded system to store the weights and support vectors. There are ways of reducing the number of support vectors [31], or a linear classifier could be used instead. As long as there are only two features, it might be possible to handcraft a polynomial function with the same decision boundary as the support vectors.

After the non-linear classifier, some aftertreatment is required. If a signal is classified as both eating and rumination, then the result will be eating. For rumination, at least 15% of the segments 10 s before and after the current segment must correspond to rumination and this must also apply for at least 50% of a 3 minute window around the current segment.

5.6.2. Heat detection

For the heat detection of tied-up cows, it was decided that the motion pattern to detect was lie-downs and stand-ups. The motions almost always follow a certain pattern explained in Figure 5-25.

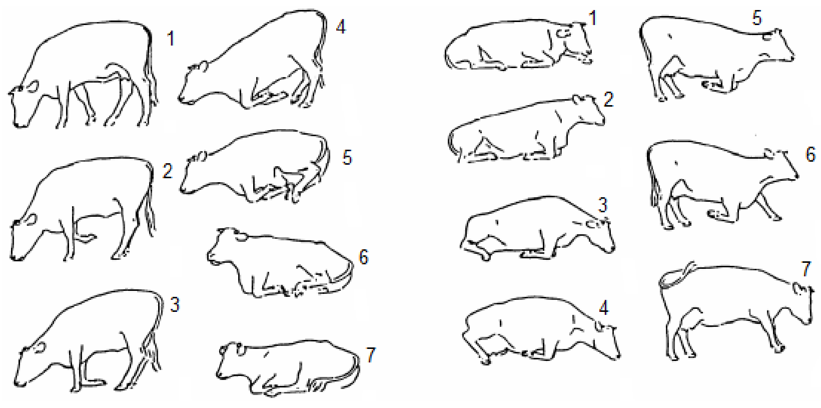


Figure 5-14. Lie-down and stand-up motion of cows (Schnitzer, U. 1971).

Having an accelerometer on one of the legs would make this easy, since it can measure the stationary angle of the leg. Only having an accelerometer on the neck makes it more difficult to distinguish. The raw data in x, y, and z-axis for 26 lie-downs and 28 stand-ups are plotted in Figure 5-26.

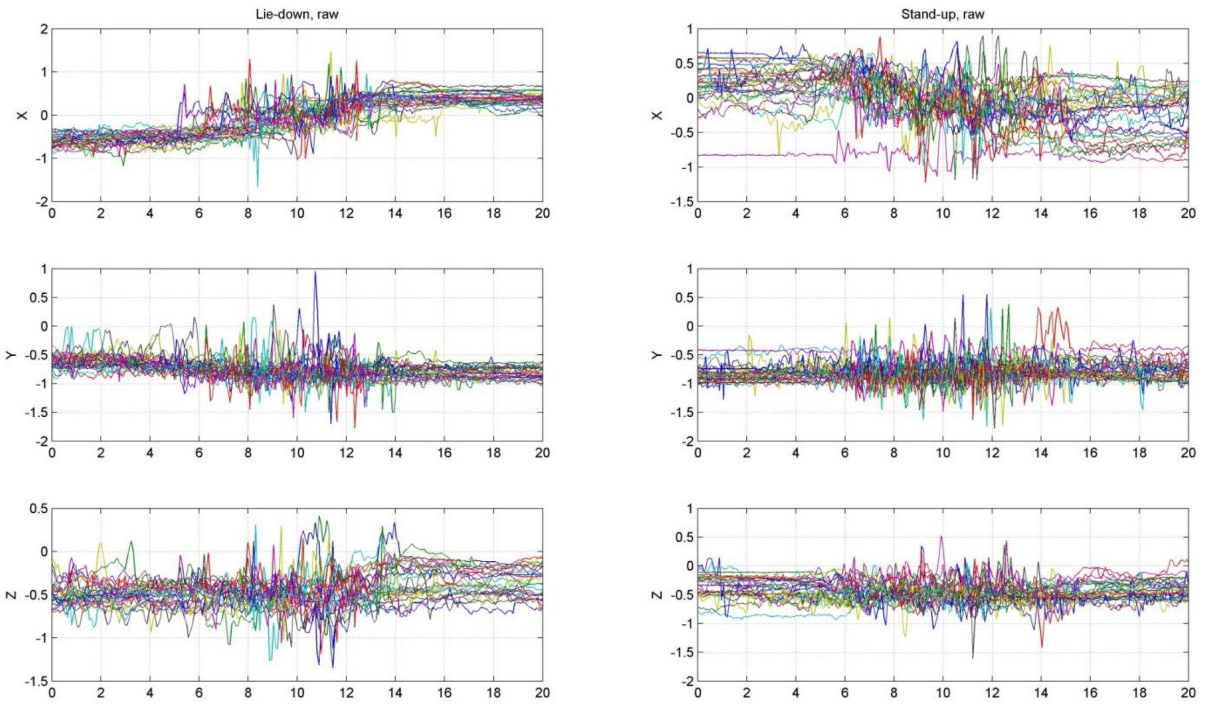


Figure 5-15. Lie-down and stand-up motions with raw data. Time is in seconds, acceleration scale is in g.

By applying a 4th order Butterworth low-pass filter with 0.5 Hz cutoff frequency, the noise is reduced and it is possible to see tendencies. In Figure 5-27 the low-pass filtered signals can be seen.

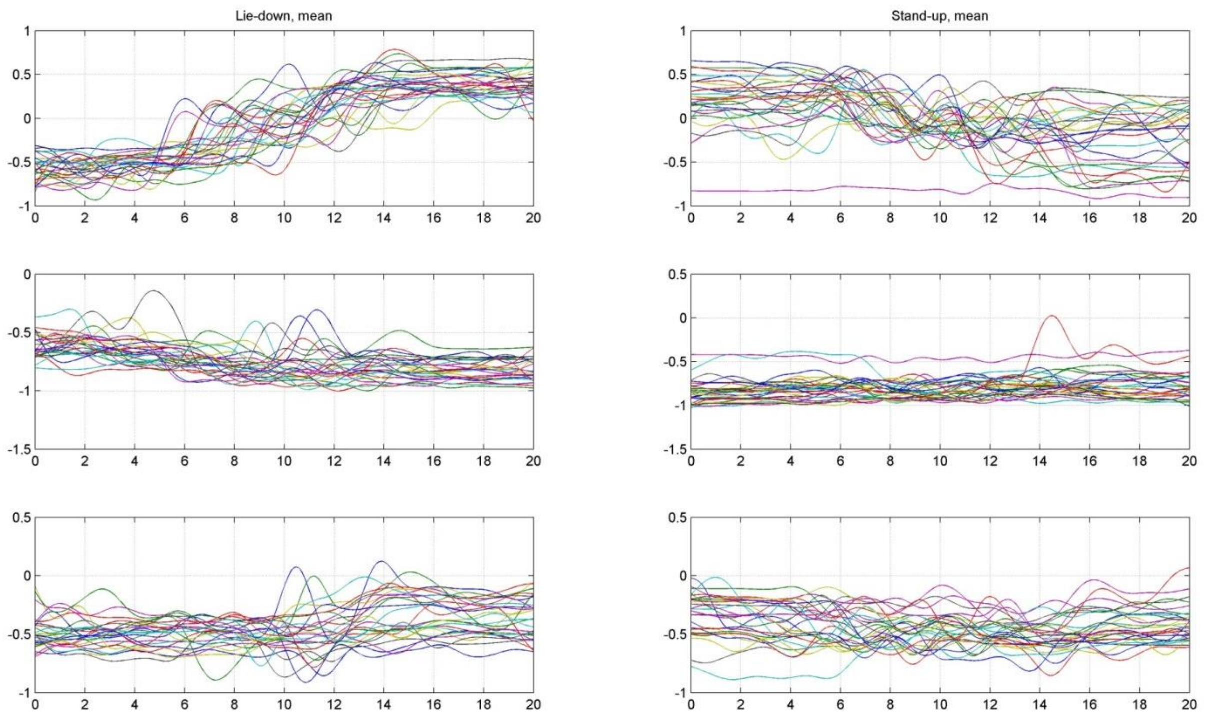


Figure 5-16. Lie-down and stand-up motions with low-pass filter applied. Time is in seconds, acceleration scale is in g.

The raw signal can then be subtracted by the LP-filtered signal to acquire the high-pass filtered signal. Then the RMS of the HP-filtered signal is calculated, followed by the same 4th order Butterworth low-pass filter with 0.5 Hz cutoff frequency. The signal represents the average noise level throughout the motion, and this can be seen in Figure 5-28.

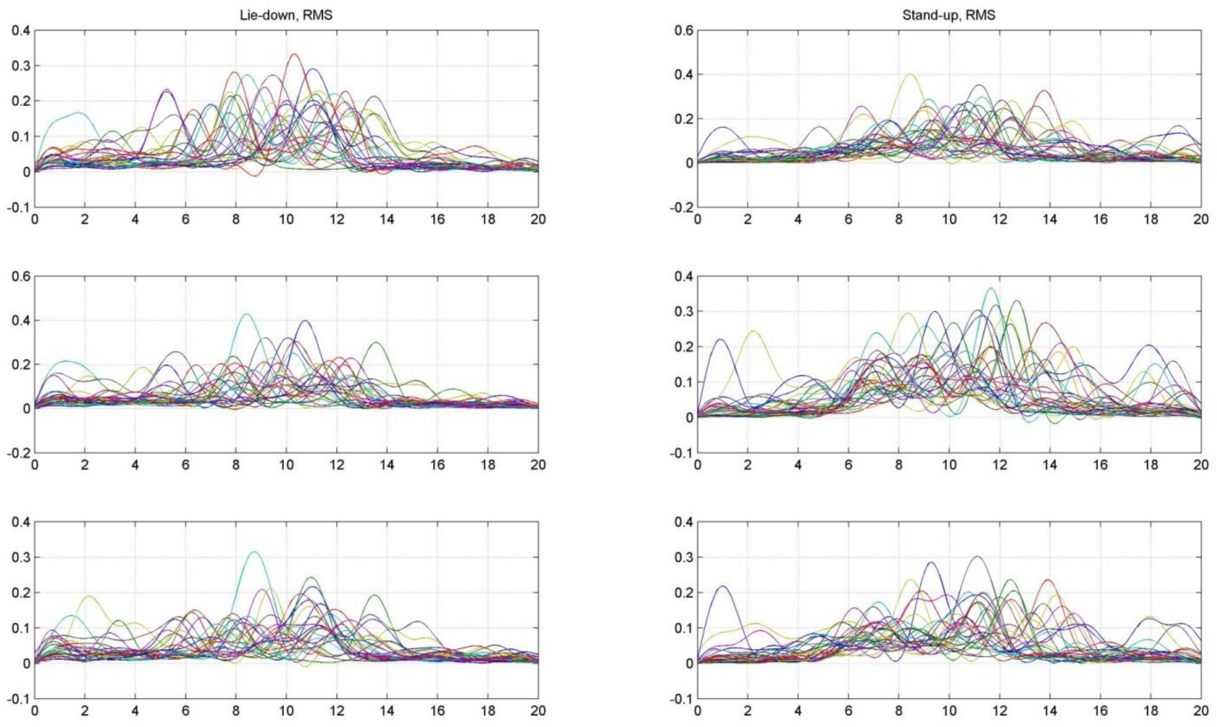


Figure 5-17. The noise level in lie-down and stand-up motions. Time is in seconds, acceleration scale is in g.

With this knowledge it can be said the motions are characterized by a shifting average in the x-axis with a bit of noise in the middle. This is because the x-axis indicates the tilt of the head, and as seen in Figure 5-25, the only detectable motion is the angle change of the head. The motion can be divided into 3 time groups, the first 4 seconds, the last 4 seconds, and the time in between. The features used here are:

- The difference in average x-value between time group 1 and 3 (“mean diff”).
- The highest RMS in the x-axis in time group 2 divided by the average RMS in the rest of the time series (“peak significance”).

The spread of the features are visualized in Figure 5-29. The stand-up motion differs so much from time-to-time that it was not worth the time to make a detection algorithm for that.

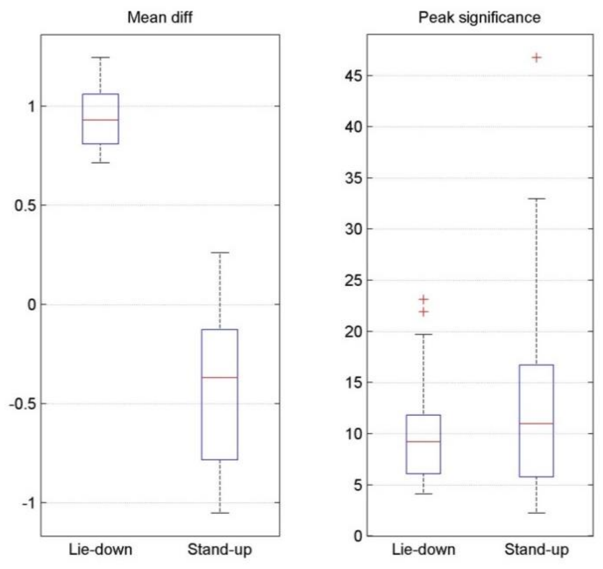


Figure 5-18. The spread of the features for stand-up and lie-down.

The detection of lie-downs can be built upon a set of rules:

- The peak must be in time group 3
- The “peak significance” must be above 3.5
- The RMS at the peak must be above 0.1 g
- The average RMS in time group 1 must be above 0.01 g
- The “mean diff” must be greater than 0.6 g
- The mean x-value at the peak must be above 0.0 g
- The mean x-value in time group 1 must be below 0.0 g

The algorithm is meant to operate with a resolution of 0.5 to 1.0 samples per second.

5.7. Statistical performance evaluation

As mentioned in [9], there are multiple ways of evaluating the performance of a classifier. Common metrics that are frequently used in activity recognition research are confusion matrix, ROC- and PR-curves, time-based evaluation, event-based evaluation and evaluation schemes. Confusion matrix is a table with rows for predicted classes and columns for actual classes. From this, it is easy to extract sensitivity and specificity. ROC is a curve illustrating sensitivity as a function of fallout (calculated as 1 - specificity) at various threshold settings in order to optimize classifiers. The ROC curve includes a line for random guess. A PR curve is almost the same thing, but it illustrates the precision as a function of recall. Time-based evaluation illustrates true classes and predicted classes over time. Event-based evaluation focuses only on moments where an event was actually happening or where an event was predicted. The statistical performance evaluation is both done for rumination detection and heat detection. Data sets are used to estimate how real world performance would look.

5.7.1. Data sets

For rumination detection, just under 10 % of the total data set[6] from Hamra was used as a training set. Since the training set was built up by 5000 data points[7] randomly distributed over the whole data set, the testing set also includes the training set. With this knowledge, it is expected that the performance will be a bit better than what is realistic. The testing set can be divided into each cow, in order to see if the performance is cow dependent. For heat detection, the training set is also the testing set. This is because it takes a lot of time to build up a data set for four cows and since the sensitivity and specificity was expected to be so low that a larger testing set was unnecessary. However, since data continued to be collected after the training set, the stability of the output can be examined and judged.

5.7.2. Rumination time

The main purpose of the rumination detection is to measure total rumination time per day. The main indication of performance will therefore be the error percentage from real observations regarding accumulated rumination time. Extra parameters to inspect are sensitivity and specificity. However, since the time axes of the system controller and wall time were not fully synched, a small timing error is introduced that affects the representation of sensitivity and specificity, but not the accumulated rumination time. The time-based evaluation can both be plotted over time where classification is compared with ground truth, and a confusion matrix can be added to extract sensitivity and specificity. Different algorithms, with different input parameters, different segmentation time and different sampling frequencies were tested on the whole testing set, but also subsets representing each cow. In addition to this, the rumination time is also tested for multiple days in a row, in order to see stability and trustworthiness of the detection algorithm.

[6] Total data set counts as all sensor samples from all sensors during which manual observations were registered.

[7] A data point counts as a feature vector with or without class information. The feature vector describes the sensor signal characteristics at a single point in time.

5.7.3. Heat detection

The heat detection algorithm is tested on the training set plus the rumination data set to extract sensitivity and precision. The statistical evaluation is carried out in an event-based evaluation manually by comparing function output with observations. Only one algorithm with one configuration of parameters was used during testing. In addition to the sensitivity and precision, the algorithm is tested continuously for 23 straight days and on four cows to see stability and trustworthiness of the detection algorithm.

6. Analysis/result

6.1. Mounting position on collar

The results of the mounting position optimization can be divided into activity distinguishability and radio transmission quality. Decisions or conclusions from the test results below can be found in chapter 7 on page 58.

6.1.1. Activity distinguishability

In Table 6-1 the scores for each activity, sensor and axis can be seen. 1 means impossible to distinguish, 2 means hard to distinguish and 3 means distinguishable. As it turned out, the upper positions on the collar had the most distinguishable signal characteristics. Tag positions are found in Figure 5-15.

Table 6-1. Distinguishability for each activity, sensor and axis.

Tag	Axis	Rumination	Eating	Drinking	Salt Licking	Stand-up	Lie-down	Axis avg	Tag avg
	x	2	3	3	1	3	3	2.5	
1992	y	2	2	2	3	2	1	2.0	2.4
	z	3	2	3	2	3	3	2.7	
	x	2	2	3	1	2	2	2.0	
1993	y	2	2	1	2	3	3	2.2	1.9
	z	1	2	1	2	2	2	1.7	
	x	3	2	2	2	2	2	2.2	
1995	y	1	2	2	3	3	2	2.2	2.2
	z	2	2	2	2	3	3	2.3	
	x	1	3	3	1	2	2	2.0	
1996	y	3	2	3	3	3	3	2.8	2.4
	z	3	3	3	3	2	1	2.5	
		2.1	2.3	2.3	2.1	2.5	2.3		

The validity of the result is strengthened by the mounting instructions of the Qwes-H/HR tag from Lely, see Figure 6-1. Note however that the Qwes-H/HR tag require this placement for the use of its microphone. The tag can be redesigned to "lock" onto the collar instead of sliding freely. Locking the sensor into a certain position also greatly helps with the classification, since if the vector or directions never have to be normalized, the specificity of the classifier increases since there is less confusion.

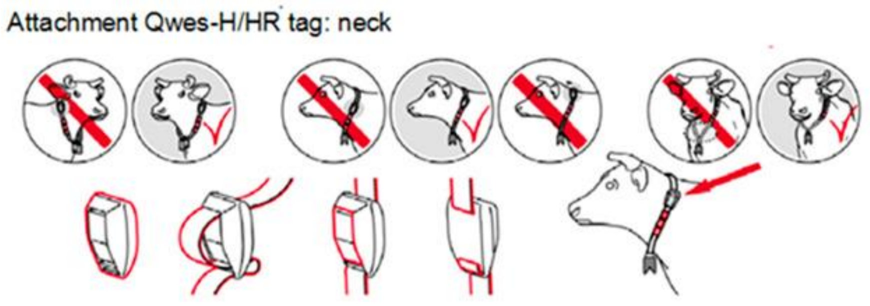


Figure 6-1. Mounting instructions for the Lely Qwes-H/HR tag (lely.com).

6.1.2. Radio transmission

As it turned out, the lower sensors had fewer failed transmissions, but the number of transmissions could not be said to be correlated to the sensor position. A small difference in clock speed inside the sensors can also affect the number of samples per day. Although this is true, tags 1992, 1995 and 1996 were having big time gaps in the data logging files. The most probable reason for the transmission quality variety is small differences in hardware connected to the antenna since that technology is very sensitive. The theory about hardware variations is supported by the fact the tag life length had a direct correlation to the number of transmissions registered daily. The tag life length is documented in Appendix B – Tag battery life. Individual variation in transmission quality is assumed to be far greater than the effects of mounting position.

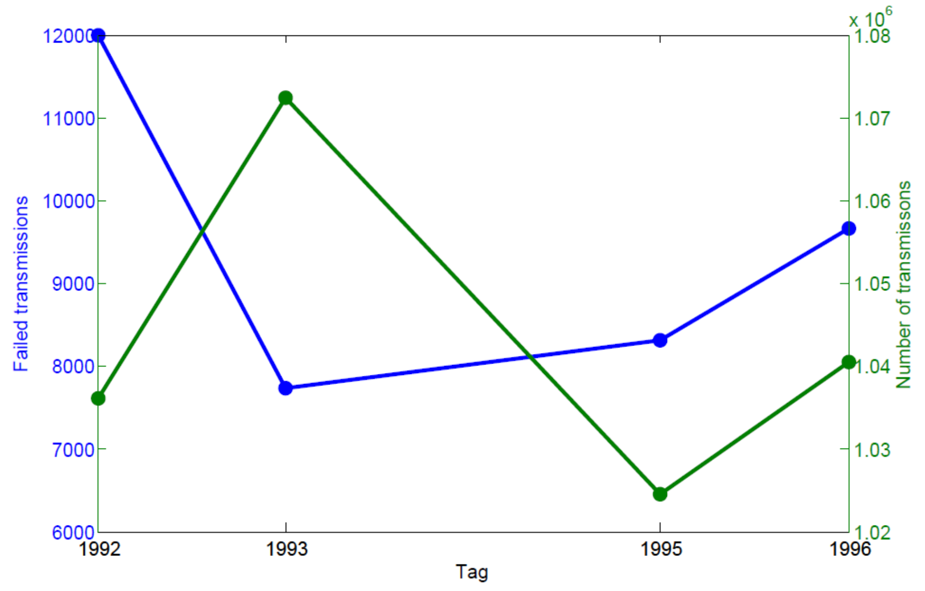


Figure 6-2. Radio transmission quality vs mounting position.

6.2. Statistics

6.2.1. Rumination detection

The SVM algorithm results are listed with different methods, where M stands for method, the number tells how many features that are used in the SVM, then which axes, b means that the variance was calculated with a Butterworth filter, m means that the axes was mixed/combined and NF means that no frequency analysis was computed. The complete information about each method can be found in Appendix C – SVM classification methods.

First, the rumination time error (RT-error) was registered for all methods. This can be seen in Table 6-2. The results on cow 6299 are misguiding for method M2.YY.b and M6.XYZ.b, since the methods missed several minutes of RT due to quiet frequency character in the y-axis, but at the same time classified salt licking as rumination. Cow 6299 is also the only cow that licked salt (which has a similar characteristics as rumination), which explains the high insertion[8] in method M2.YZ.b.m and M7.XYZV.b.NF. The reason why the results are better on cow 6379 is partially because it had the biggest data set, but also because cow 6379 never ruminated standing, which reduces the confusion with other activities. This is because standing rumination is closer to NULL-class behavior and contains more anomalies than lying rumination.

Table 6-2. Deviation in rumination time for the testing set and subsets using different algorithms.

Method	**All cows**	**Cow 6379**	**Cow 6361**	**Cow 6299**
M2.YY.b	-0.90%	-1.04%	-3.18%	0.27%
M2.YZ.b.m	4.07%	1.22%	5.43%	13.25%
M6.XYZ.b	-2.85%	0.54%	-6.32%	4.66%
M7.XYZV.b.NF	12.97%	8.02%	8.86%	35.87%

The algorithms were also tested day-by-day on two cows to see stability and trustworthiness. On cow 6379, the different methods showed the same character, even though the values differed.

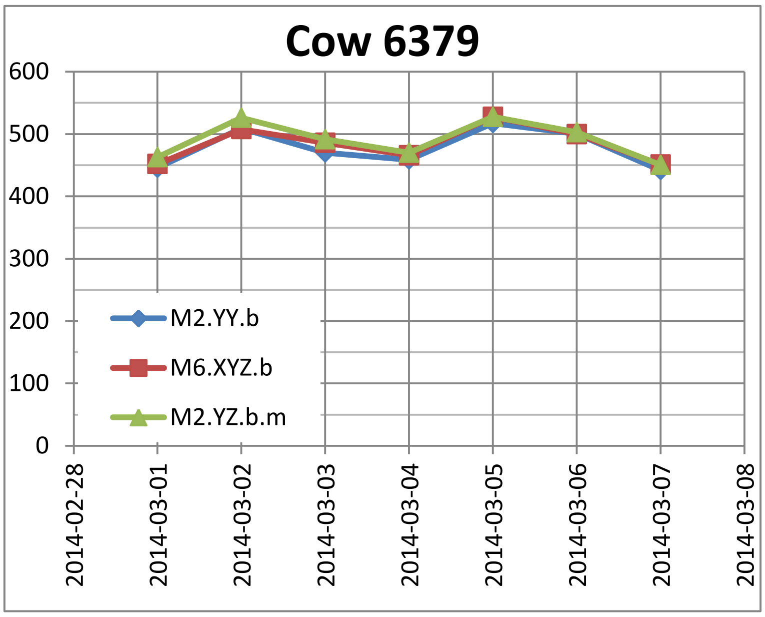


Figure 6-3. Rumination time per day (minutes) on cow 6379 using different algorithms.

[8] Insertion is the case when the classifier detects activity where there is none in the ground truth.

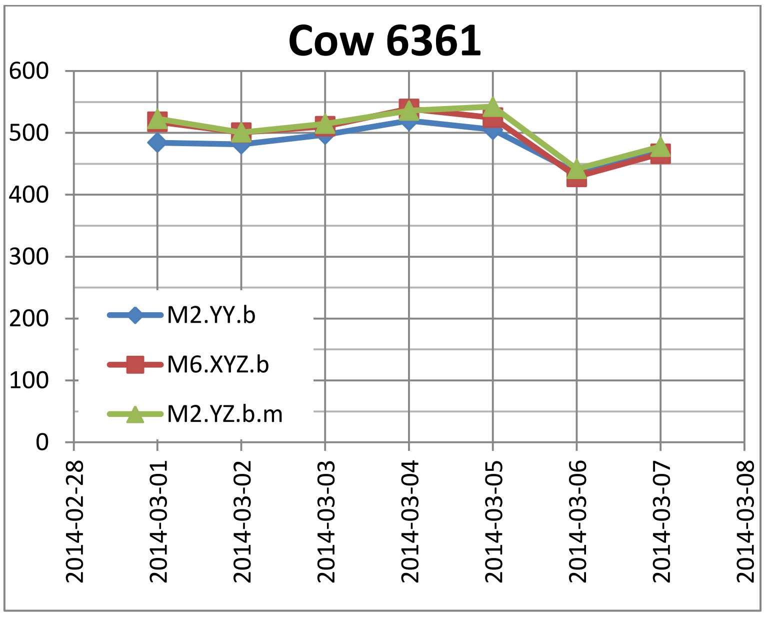


Figure 6-4. Rumination time per day (minutes) on cow 6361 using different algorithms.

All algorithms showed similar curvature. If the purpose of the RT data is to detect slopes or deviations, any method should be sufficient to detect this, since the curvature is similar regardless of method.

In Figure 6-5 the first 4 hours of observation data on cow 6299 is compared with classification results using M2.YZ.b.m with 8 s segmentation and 12.5 Hz sampling rate. Activity numbers are explained in Table 5-2, but the most important comparison is between the red and green line (classed rumination and true rumination). At 3.4 hours the activity of salt-licking is misclassified as rumination. It is also possible to see the timing error by comparing the edges of the green and red lines.

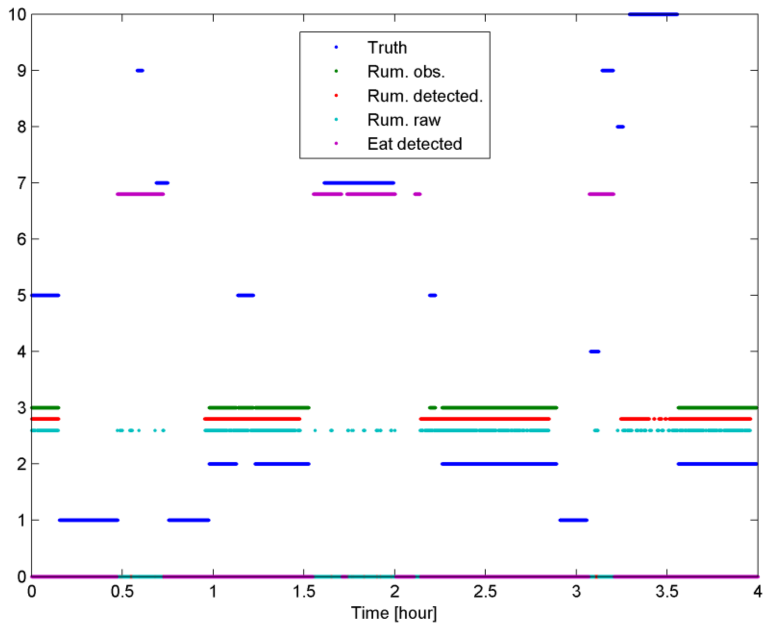


Figure 6-5. Example of ground truth compared with classifier output.

In Table 6-3 the sensitivity and specificity of each method of interest and test set is listed. As seen, the performance is not as good as it seemed in Table 6-2 with RT-errors. But, the true performance is actually better because the sensitivity and specificity numbers here include a timing error. The M6.XYZ.b algorithm shows very promising performance.

Table 6-3. Sensitivity and specificity for different methods and test sets.

fs	Tseg	Method	Test set	Sensitivity	Specificity
12,5	8	M2.YY.b	All cows	89.50%	90.30%
12.5	8	M2.YY.b	Cow 6379	92.78%	93.75%
12.5	8	M2.YY.b	Cow 6361	88.71%	91.62%
12.5	8	M2.YY.b	Cow 6299	78.41%	78.19%
12.5	8	M6.XYZ.b	All cows	94.56%	90.50%
12.5	8	M6.XYZ.b	Cow 6379	95.11%	93.23%
12.5	8	M6.XYZ.b	Cow 6361	95.92%	90.40%
12.5	8	M6.XYZ.b	Cow 6299	91.75%	84.18%
12.5	8	M2.YZ.b.m	All cows	94.03%	90.56%
12.5	8	M2.YZ.b.m	Cow 6379	95.15%	94.10%
12.5	8	M2.YZ.b.m	Cow 6361	96.73%	92.11%
12.5	8	M2.YZ.b.m	Cow 6299	90.36%	81.18%

6.2.2. Heat detection

The sensitivity and precision for the lie-down motion detection algorithm was 53% and 56% respectively. This means that the probability of detecting a lie-down is 53% and if the algorithm indicates a lie-down, it is 56% probable that the indication is true. It isn't comparable to random guess (50/50) since the occurrence, or prevalence, over time is low, where random guess in every sample would give a diagnostic odds ratio[9] of 1.

The result of testing the algorithm on longer periods is seen in Figure 6-6. There is one line per cow. The trustworthiness of the algorithm is judged as low, considering that a cow does not lie down 1-3 times average per day, as in the case of cow 1998. According to [5], the number of lie-downs can vary between 1 to 39, with an average of 14 lie-downs per cow and day.

[9] The probability of positive test outcome divided by the probability of negative test outcome. The balance between positive and negative test outcomes.

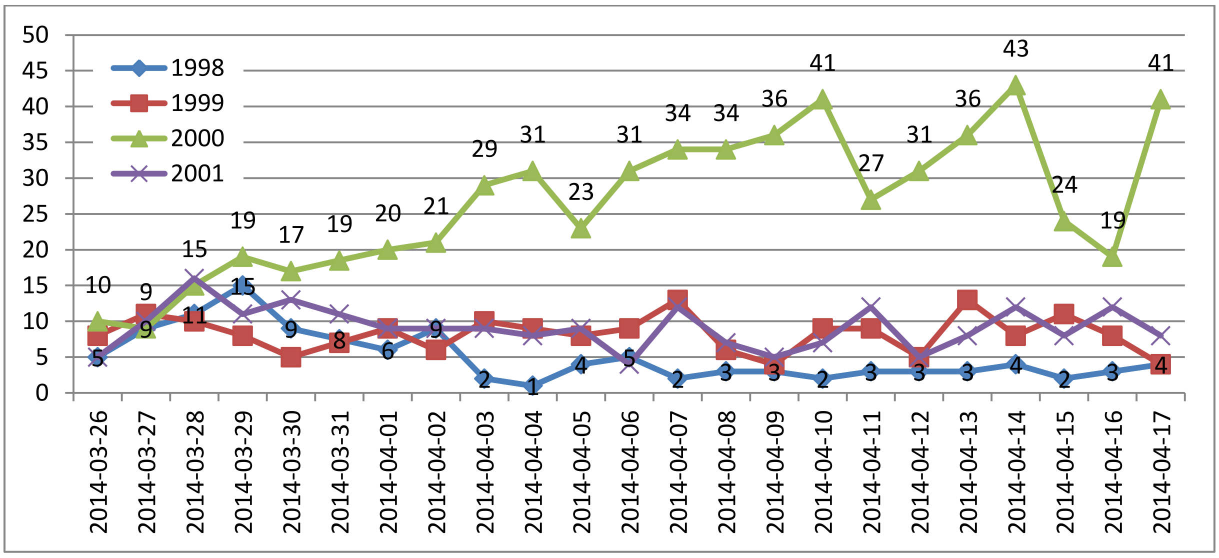


Figure 6-6. Number of lie-downs per day and cow detected by algorithm.

6.3. Computation aspects

6.3.1. Profiling

Different methods and algorithms require different amounts of computation. Six of the most interesting algorithms/methods have been profiled in MATLAB using an Intel® Core™ i5-520M Processor (3M Cache, 2.40 GHz). Execution was limited to one thread of the CPU. The results are shown in Table 6-4 and the execution time is specified for 24 hours of data.

Table 6-4. CPU time comparison between different methods/algorithms for rumination detection.

Method	**Time**
M2.YY.b	4.01 s
M2.YZ.b.m	4.97 s
M6.XYZ.b	5.75 s
M7.XYZV.b.NF	5.74 s

6.3.2. Sampling frequency

Reducing sampling frequency from 12.5 Hz does not lower the current in the accelerometer [25], but it reduces the amount of data handling in the MCU. It is possible to simulate a lower sampling frequency by only using every other or every third sample. 4.1667 Hz is however not available as a sampling rate in the used accelerometer [25]. The results are shown in Table 6-5.

Table 6-5. Effects on performance when reducing the sampling frequency.

Frequency	Comment
12.5 Hz	Is just enough for most classification methods. Higher sampling frequency is not believed to increase performance significantly.
6.25 Hz	SVM classification error increases with 5% in general. RMS representation may be false, since some of the noise might be missed.
4.17 Hz	2-3 Hz character turns into 1.16-2.08 Hz because of aliasing and is indistinguishable from other activities.

6.3.3. Segmentation time

The segmentation time is mainly evaluated on the non-linear classifier methods such as M6.XYZ.b. In this case it was examined how the segmentation time affected M2.YZ.b.m. The result is shown in Figure 6-7. There seem to be no correlation between segmentation time and performance, but the segmentation time does however affect the performance for each test, since sensitivity, precision and accumulated rumination time varies with varying segmentation time. A function (6.1) to score which segmentation time that yielded best overall performance was used. In this case, a segmentation time of 8 seconds gave the best result. To better understand the graph, all values should be as close to 100% as possible, where sensitivity and precision can be in the range 0-100% and RT can be from 0% to any percentage.

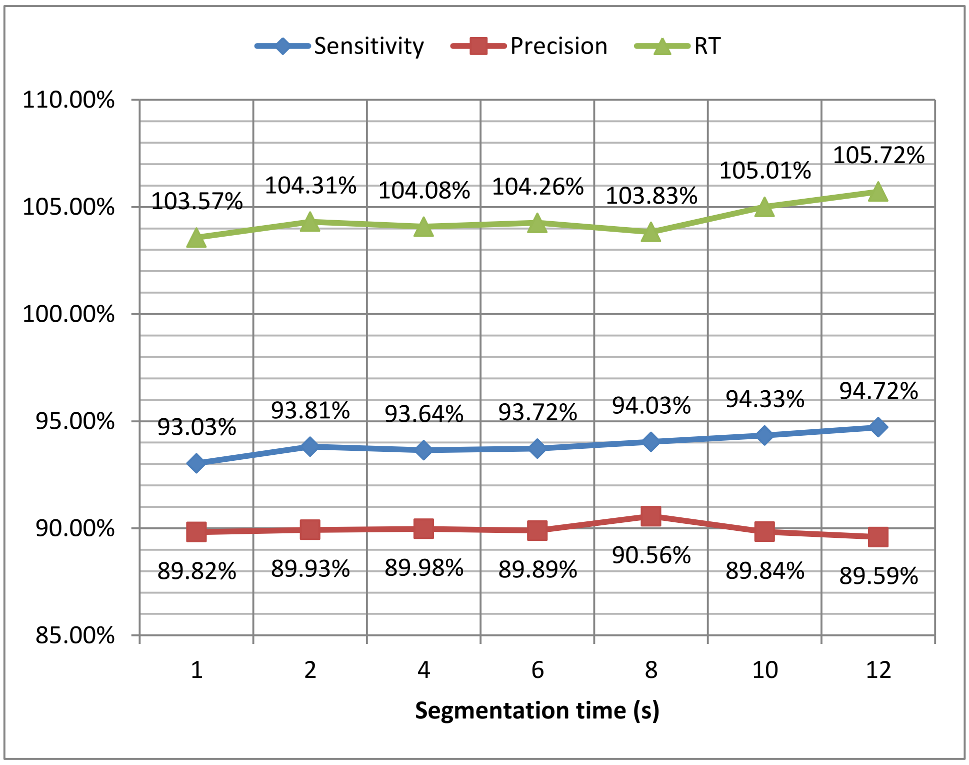


Figure 6-7. Rumination time error as a function of segmentation time in seconds.

$$\text{Score} = \left(1 - \left|1 - \text{RT}\right|\right) \cdot \text{Sensitivity} \cdot \text{Precision} \tag{6.1}$$

7. Discussion/Recommendation

7.1. Hardware

Both memory and flash on the current MCU are insufficient for upgrades. An upgrade within the same family of system-on-chip has 64 kB flash, 4 to 8 kB memory and the same power consumption.

7.2. Software

RTOS was thought of as a possible software base in either a tag or receiver, but it was advised against by the supervisor and the technical architect of the current software. The main reasons is that current software is without RTOS and it would take too much time to add an RTOS, knowing that the added complexity can produce unexpected issues that are difficult to identify. An RTOS also creates more overhead code, which might be a problem when the flash is small and minimum instructions is of importance. It was therefore never researched if it could be implemented and what it would contribute to. The microcontroller uses a scheduler instead.

7.3. Finite word length effects in IIR filters

Low-pass filtering integers with IIR and very low cut-off frequency can give low-level limit cycles [32]. These can cause oscillations of the filter output, or the output to remain stuck at a non-zero value, even when there is no input. Limit cycles are primarily of concern in fixed-point recursive filters. The problem can be solved by minimizing the steady-state error to an acceptable level. This is done by scaling up the signal before filtering and scaling it down after filtering. For example, changing an unsigned 8-bit integer to an unsigned 16-bit integer and multiplying it by 256 or shifting it up 8 times.

7.4. Observation

The observation was initially intended to include more cows and more hours per day than what it became. It was initially intended to include 4 cows simultaneously for 4 hours per day for 10 workdays, summing up to 160 observation hours. But in reality it turned out to be difficult to observe more than 3 cows simultaneously because the cows were not always at the same place and there was only room for 3 columns in the notepad. The observation also required a lot of concentration since everything is continuously written down and you can’t really start a thought because then you might miss something that happens. This led to a maximum of 3 hours of observation per day. There was also a lot of work at the computer that needed time. After a few days, one of the tags, 1992, stopped transmitting, probably due to an error without fault handling. As a result it was only possible to observe 2 cows for a few days until that tag was replaced, meaning that even more observation time was lost. The decreasing observation time can be overlooked in Table 7-1.

Table 7-1. Reasons why the observation time was decreased and how much observation time remained.

Comment	Observation time
Initial	160
3 cows limitation	120
2.3 hour average	69
Malfunctioning tag	45

The detail of the observation with two-minute intervals was chosen on advice from the supervisor since a cow doesn’t change activity that often. Extra information about sudden events was noted anyway in order to help puzzling sensor readings with observations.

The system controller was set to only save sensor information with the time in minutes in the filename instead of with seconds, since the second value changed irregularly even though it was intended to be a new file every 60 seconds. This was initially a problem in MATLAB but a solution was later programmed, checking for files with every possible name. Having the resolution with seconds might have helped to puzzle sensor data with observations. Although small glitches would still appear due to failed transmissions and/or varying sampling frequency.

7.5. Classifier

The classifier methods used in this project are just proposed options. Different signal features are not limited to any classifier, such as a decision tree. A frequency analysis could be done together with bolus detection to increase the accuracy. The band pass method could be used in any other algorithm. The number of possible algorithms had to be limited due to the limited time and the time it takes to test, adjust and make statistics.

7.6. Rumination characteristics

During the project, it has repeatedly been asked if the frequency that characterizes rumination always is between 2-3 Hz, even for other cows. Only sensor data from Holstein[10] cattle and only from 3 individuals have been gathered. It seems likely that any other breed of the same size would have the same characteristics, but it could also be the opposite. Before making this a product, it would be recommended to investigate differences between different breeds in order to ensure that the device can detect rumination in all breeds.

Another question that also was raised was why the first overtone was strongest. Other reports [33] suggest that the chewing frequency is around 0.95 Hz during rumination and around 1.15 Hz during eating. The same study also suggested that there is 54 seconds between every

[10] Holstein cattle is a cow breed, known today as the world’s highest-production dairy animals.

bolus, something can vary greatly between 40 to 90 seconds as seen during this project. It is probable that also the chewing frequency can vary between cows. It is important to keep in mind that the sensor is not sensing the movements of the jaw (the actual chewing), but the movements of the neck. It might be that the cow shakes the neck for both vertical and horizontal movements of the jaw. It doesn't have to be explained in order to work – the main thing is that there is a signal characteristic that discriminates rumination from other activities.

7.7. Requirements

The main concern in the requirements, which are found in Appendix A – Requirements, is the battery life and performance. It's a bit of a tradeoff. It might be hard to reach 10 years battery life with the chosen 3000 mAh battery currently selected, since the product without the rumination monitoring is expected to last 10 years. Discussions have been made that for an upgraded product, a lower battery life would be acceptable. Reaching above 90 % sensitivity and specificity will probably require heavier algorithms than the lightest one. It remains to be seen how well the lightest algorithm can be tuned and if it needs additional features.

The requirement that data must be transmitted at least every 15 minutes is ensured if the transmission is time-triggered instead of location or event triggered. By using ZigBee as wireless communication, it is also ensured that communication always can be established. The requirement that the system must be able to detect rumination activity and calculate rumination time is ensured by sensing the movement of the neck with an accelerometer and applying preprocessing, feature extraction and classification. The system is still scalable not having a fixed set of connections, but by using a bus network with identifiers and wireless communication with identifiers. The wireless communication does not interfere with other equipment or regulations by using the standard 433 MHz band as long as the farmer doesn't have multiple equal systems at the same farm. The tags will withstand the tough environment by being filled with plastic, making it mechanical shock resistant and short circuit proof. The system doesn't cause discomfort to the cow by having the tag on the collar, instead of on a noseband or on the leg. The requirement about portability is ensured by using a high capacity battery and wireless communication.

7.8. How to proceed

The next step is to investigate how much time it takes for an MCU of the same family to compute the simplest algorithm and make more accurate estimations of battery life. After this is done, more detailed hardware and software requirements can be made. The external validity should also be tested. This can be done by testing on different cow breeds and in different environments. Also, a new approach is needed towards detecting heat in tied-up cows. There might be other signatures such as social activities that are easier to detect.

8. Conclusion

A test system has been set up, data has been gathered, algorithms have been developed and tested and system aspects have been analyzed.

From the test results it is clear that the signals required for detecting rumination are available using an accelerometer on the collar. This sensing method is sufficient to reach acceptable performance on rumination monitoring, where acceptable performance is defined in Appendix A – Requirements. The highest performing algorithms are suspected to draw too much energy or memory to be feasible, based on the profiler results in section 6.3.1 on page 48. The true battery life has to be estimated by programming the tag with the actual algorithm and measure the computation time.

The mounting turned out to affect the rumination monitoring performance but not radio transmissions. The best mounting of the tag was on the upper right or left side of the collar, which is also what other manufacturers have concluded with similar systems. Proper mounting has to be ensured, perhaps with a redesign of the tag housing.

It is realistic that the battery life can reach 10 years with modifications on hardware and optimizing software. If a battery life of 5 years could be accepted for an upgraded product, it is recommended to include frequency analysis in the software, increasing the accuracy of rumination time.

An accelerometer on the collar is not feasible for detecting lie-downs or stand-ups since there is little, random or no significant movement that can’t occur during other activities. A much more effective on-cow motion sensor is a pedometer sensing the angle of the legs.

9. Appendices

9.1. Appendix A – Requirements

Table 9-1. Non-functional system requirements.

No#	Description	Priority	Domain
3	The system must be scalable, using any number of tags.	1	Scalability
4	Wireless communication must not interfere with other equipment or regulations.	1	Regulatory
5	The tags must withstand tough environments.	1	Environmental
6	The system must not cause damage or discomfort for the cows.	1	Environmental
7	The tags must be completely portable. No wires are permitted.	1	Portability
9	The battery life of the tags must exceed 10 years.	2	Performance
10	The sensitivity and specificity of the rumination detection must be above 90 %.	2	Performance
11	The performance of different algorithms must be documented in the form of tables.	2	Documentation
12	The tags shall be designed so that no maintainance is needed throughout its lifetime.	2	Maintainability
13	The receivers shall be easy to install and connect for a service technician.	2	Maintainability
14	The information from the tags must reach the receivers even if a transmission fails.	2	Fault tolerance

Table 9-2. Functional system requirements.

No#	Description	Priority
1	Cow rumination and activity information must be transmitted from the tag to the base unit least once every 15 minutes.	1
2	The system must be able to detect rumination activity and calculate rumination time.	1
8	The system must be able to detect motion patterns that could mean heat in tied-up cows.	3

9.2. Appendix B – Tag battery life

Some of the tags ran stopped working before the battery had run out. Only two tags in the Hamra farm test installation worked until the battery was fully discharged. Below is a list of how long each tag worked (transmitted messages) and how good the transmission quality was.

Tag	**Life**	**Problems**[11]
1992	9	■■■■
1993	38	■
1994	0	-
1995	4	■■■■
1996	16	■■
1997	39	■■

Tag 1994 activated 2014-02-21	Dead on arrival
Tag 1995 activated 2014-02-21	Died 2014-02-25
Tag 1992 activated 2014-02-21	Died 2014-03-02
Tag 1996 activated 2014-02-21	Died 2014-03-08
Tag 1993 activated 2014-02-21	Died 2014-03-31
Tag 1997 activated 2014-03-06	Died 2014-04-14

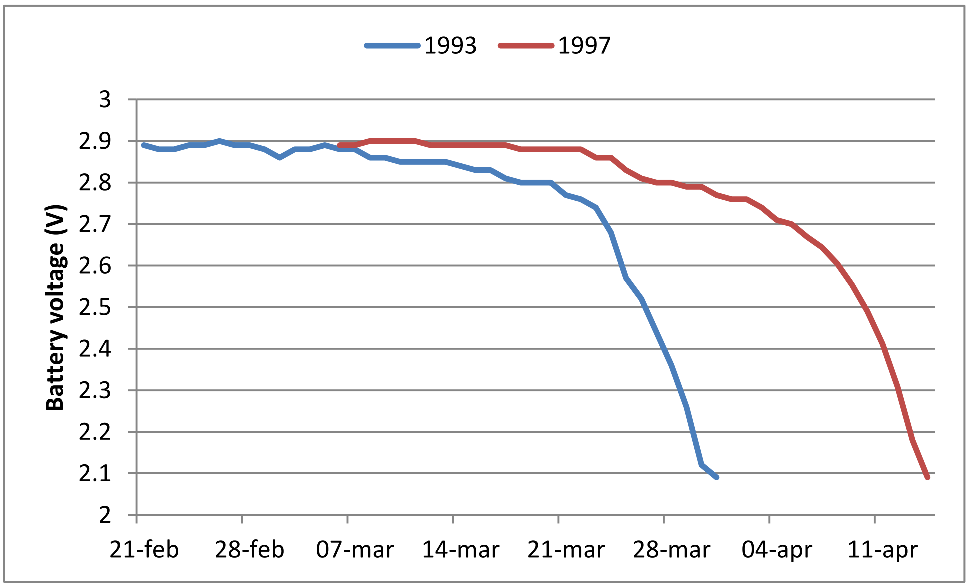


Figure 9-1. Activation and death of tags 1993 and 1997, the only tags to not die from software bug or other.

[11] Failed transmissions, lost transmissions, low amount of data.

9.3. Appendix C – SVM classification methods

Method	Description	SVM	Aftertreatment	Cost
Full method	LP filtering with butterworth. AC = raw-DC. RMS = windowrms(AC).	Xrms, Yrms, Zrms, Vrms, f25p(max(x,y,z,v))	Median 180s, 15% over 20s. No eating overlap.	26
M3.YZP	LP filtering with butterworth. AC = raw-DC. RMS = windowrms(AC).	Yrms, Zrms, f25p(max(x,y,z,v))	Median 180s, 15% over 20s. No eating overlap.	20
M2.ZP	LP filtering with butterworth. AC = raw-DC. RMS = windowrms(AC).	Zrms, f25p(max(x,y,z,v))	Median 180s, 15% over 20s. No eating overlap.	17
M2.VP	LP filtering with butterworth. AC = raw-DC. RMS = windowrms(AC).	Vrms, f25p(max(x,y,z,v))	Median 180s, 15% over 20s. No eating overlap.	17
M2.ZZ	LP filtering with butterworth. AC = raw-DC. RMS = windowrms(AC).	Zrms, f25z	Median 180s, 15% over 20s. No eating overlap.	7
M2.YY	LP filtering with butterworth. AC = raw-DC. RMS = windowrms(AC).	Yrms, f25y	Median 180s, 15% over 20s. No eating overlap.	7
M2.YY.b	LP filtering with butterworth. AC = raw-DC. var = butter(|AC|).	Yvar, f25y	Median 180s, 15% over 20s. No eating overlap.	6
M2.ZZ.b	LP filtering with butterworth. AC = raw-DC. var = butter(|AC|).	Zvar, f25z	Median 180s, 15% over 20s. No eating overlap.	6
M2.ZZ.b.R	LP filtering with butterworth. AC = raw-DC. var = butter(|AC|).	Zvar, f25z	Median 180s, 15% over 20s.	6
M2.XZ.b	LP filtering with butterworth. AC = raw-DC. var = butter(|AC|).	R: Zvar, f25z. E: Xf, f25z	Median 180s, 15% over 20s. No eating overlap.	7
M7.XYZ.b	LP filtering with butterworth. AC = raw-DC. var = butter(|AC|).	Xf, Yf, Zf, Xvar, Yvar, f25y, f25z	Median 180s, 15% over 20s. No eating overlap.	15
M6.XYZ.b	LP filtering with butterworth. AC = raw-DC. var = butter(|AC|).	Xf, Xvar, Yvar, Zvar, f25y, f25z	Median 180s, 15% over 20s. No eating overlap.	15
M7.XYZV.b.NF	LP filtering with butterworth. AC = raw-DC. var = butter(|AC|).	Xf, Yf, Zf, Xvar, Yvar, Zvar, Vvar	Median 180s, 15% over 20s. No eating overlap.	11
M4.YZ.b	LP filtering with butterworth. AC = raw-DC. var = butter(|AC|).	Yvar, Zvar, f25y, f25z	Median 180s, 15% over 20s. No eating overlap.	12
M2.YZ.b.m	LP filtering with butterworth. AC = raw-DC. var = butter(|AC|).	max(Yvar, Zvar), max(f25y, f25z)	Median 180s, 15% over 20s. No eating overlap.	11

10. Bibliography

[1] T. Choudhury, S. Consolvo, B. Harrison, J. Hightower, A. LaMarca, L. LeGrand, A. Rahimi, A. Rea, G. Borriello, B. Hemingway, P. Klasnja, K. Koscher, J. Landay, J. Lester and D. Wyatt, "The Mobile Sensing Platform: An Embedded Activity Recognition System," IEEE Pervasive Computing, 2008.

[2] S. Hoy and S. Reith, "Analysis of physical activity, rumination and body weight of dairy cattle during oestrus using sensor-aided systems," Department of Animal Breeding and Genetics, Justus-Liebig-University Giessen, Giessen, Germany, 2011.

[3] F. Nydegger, L. Gygax and W. Egli, "Automatic Measurement of Rumination and Feeding Activity using a Pressure Sensor," Agroscope ART Tänikon, Ettenhausen, Switzerland.

[4] "CowManager SensOor," Agis Automatisering BV, [Online]. Available: https://www.cowmanager.com. [Accessed 2014].

[5] M. Gustafsson, C. Lindahl, B. Berglund and H. Gustafsson, "Stå- och liggtider för brunstdetektion i uppbundna system - en pilotstudie," JTI - Institutet för jordbruks- och miljöteknik, 2007.

[6] A. Spink, B. Cresswell, A. Kölzsch, F. v. Langevelde, M. Neefjes, L. Noldus, H. v. Oeveren, H. Prins, T. v. d. Wal, N. d. Weerd and W. F. d. Boer, "Animal behaviour analysis with GPS and 3D accelerometers," Precision Livestock Farming, Wageningen, The Netherlands, 2013.

[7] F. Oudshoorn, C. Cornoou, A. Hellwing, H. Hansen, L. Munksgaard, P. Lund and T. Kristensen, "Estimation of grass intake on pasture for dairy cows using tightly and loosely mounted di- and tri-axial accelerometers combined with bite count," Elsevier, 2013.

[8] D. T.-P. Fong and Y.-Y. Chan, "The Use of Wearable Inertial Motion Sensors in Human Lower Limb Biomechanics Studies: A Systematic Review," Sensors, Hong Kong, China, 2010.

[9] A. Bulling, U. Blanke and B. Schiele, "A Tutorial on Human Activity Recognition Using Body-worn Inertial Sensors," ACM Computing Surveys, 2013.

[10] A. Mannini and A. M. Sabatini, "Machine Learning Methods for Classifying Human Physical Activity from On-Body Accelerometers," Sensors, 2010.

[11] S. Marsland, Machine Learning: An algorithmic perspective, Boca Raton, FL: Chapman & Hall/CRC, 2009.

[12] MathWorks, "Supervised Learning Workflow and Algorithms," [Online]. Available: http://www.mathworks.se/help/stats/supervised-learning-machine-learning-workflow-

and-algorithms.html. [Accessed 23 05 2014].

[13] C.-C. Chang and C.-J. Lin, "LIBSVM - A Library for Support Vector Machines," [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm. [Accessed 23 05 2014].

[14] J. A. Ward, P. Lukowicz, G. Tröster and T. Starner, "Activity Recognition of Assembly Tasks Using Body-Worn Microphones and Accelerometers," IEEE, 2006.

[15] K. Murphy, "Hidden Markov Model (HMM) Toolbox for Matlab," [Online]. Available: http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html. [Accessed 24 05 2014].

[16] Wikipedia, "Microelectromechanical systems," [Online]. Available: http://en.wikipedia.org/wiki/Microelectromechanical_systems. [Accessed 02 06 2014].

[17] M. S. Y. Siva Prasad, "Design simulation and fabrication of micromachined acceleration sensor," Jawaharlal Nehru Technological University, 2011.

[18] A. Burg, A. Meruani, B. Sandheinrich and M. Wickman, "MEMS Gyroscopes and their applications," Northwestern University, Evanston, IL, 2004.

[19] A. J. da Silva and R. S. de Oliveira, "Methods and guidelines for embedded system processor selection," in *VI Induscon - Conferência Internacional de Aplicações Industriais,* Recife, Brazil, 2006.

[20] F. Salewski and S. Kowalewski, "Hardware Platform Design Decisions in Embedded Systems - A Systematic Teaching Approach," in *WESE'06 - Embedded Systems Education,* Seoul, South Korea, 2006.

[21] Texas Instruments, "Choosing the Right Architecture for Real-Time Signal Processing Designs," 2002. [Online]. Available: http://www.ti.com/lit/wp/spra879/spra879.pdf. [Accessed 11 06 2014].

[22] P. Bishop, "A tradeoff between microcontroller, DSP, FPGA and ASIC technologies," EE Times, 25 2 2009. [Online]. Available: http://www.eetimes.com/document.asp?doc_id=1275272. [Accessed 11 6 2014].

[23] Silicon Labs, "Implementing Energy Harvesting in Embedded System Designs," [Online]. Available: http://www.silabs.com/Support%20Documents/TechnicalDocs/implementing-energy-harvesting-in-embedded-system-designs.pdf. [Accessed 02 06 2014].

[24] Wikipedia, "Thin film rechargeable lithium battery," [Online]. Available: http://en.wikipedia.org/wiki/Thin_film_rechargeable_lithium_battery. [Accessed 02 06 2014].

[25] Freescale Semiconductor, "MMA8451Q," 10 2013. [Online]. Available: http://cache.freescale.com/files/sensors/doc/data_sheet/MMA8451Q.pdf. [Accessed 13 05 2014].

[26] Silicon Laboratories, "Battery Life Estimator," [Online]. Available: http://www.silabs.com/support/pages/batterylifeestimator.aspx. [Accessed 26 05 2014].

[27] K. Pothuganti and A. Chitneni, "A Comparative Study of Wireless Protocols," *Advance in Electronic and Electric Engineering,* vol. 4, no. 6, pp. 655-662, 2014.

[28] Future Electronics, "fut-electronics.com," [Online]. Available: http://www.fut-electronics.com/wp-content/plugins/fe_downloads/Uploads/Comparison%20of%20Wireless%20Technologies.pdf. [Accessed 25 Maj 2014].

[29] Wikipedia, "Radial basis function kernel," [Online]. Available: http://en.wikipedia.org/wiki/Radial_basis_function_kernel. [Accessed 28 05 2014].

[30] MathWorks, "svmclassify - Classify using support vector machine (SVM)," [Online]. Available: http://www.mathworks.se/help/stats/svmclassify.html. [Accessed 14 05 2014].

[31] D. Geebelen, J. Suykens and J. Vandewalle, "Reducing the Number of Support Vectors of SVM Classifiers using Smoothed Separable Case Approximation," [Online]. Available: ftp://ftp.esat.kuleuven.be/stadius/dgeebele/SCA.pdf. [Accessed 28 05 2014].

[32] University of Newcastle, "Design of IIR Filters," [Online]. Available: http://www.staff.ncl.ac.uk/oliver.hinton/eee305/Chapter5.pdf. [Accessed 20 05 2014].

[33] Luginbuhl, Pond, Russ and Burns, "A Simple Electronic Device and Computer Interface System for Monitoring Chewing Behavior of Stall-Fed Ruminant Animals," *Journal of Dairy Science,* vol. 70, no. 6, pp. 1307-1312, 1987.

[34] Silicon Labs, "Si1012-A short," 2010. [Online]. Available: http://www.silabs.com/Support%20Documents/TechnicalDocs/Si1012-A-short.pdf. [Accessed 13 05 2014].

[35] FDK Corporation, "CR8-LHC technical information," [Online]. Available: http://www.fdk.co.jp/battery/lithium_e/tech_info/CR8LHC%20Technical%20Information.pdf. [Accessed 13 05 2014].