Institut für Parallele und Verteilte Systeme
Universität Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Masterarbeit Nr. 3089

A Novel Low-power Model for Accelerometry-based Real-time Respiration Monitoring System

Mathankumar Gopalsamy

Course of Study:	INFOTECH
Examiner:	Prof. Dr. Sven Simon
Supervisors:	Dr. Bin Yin, Ir. Haris Duric, Dipl.-Inform. Thomas Falck, *Philips Research, Eindhoven, The Netherlands.* Dipl. Inf. Simeon Wahl *Universität Stuttgart, Germany*
Commenced:	September 23, 2010
Completed:	March 31, 2011
CR-Classification:	I.6, J.3, B.4.1, B.7.1, B.8

Acknowledgement

I wish to thank Prof. Sven Simon and Simeon Wahl for their perpetual support and guidance right from the beginning of this work. I extend my gratitude to Dr. Bin Yin, Haris Duric, and Thomas Falck from Philips Research - Eindhoven, The Netherlands, without whom this work would have not been feasible with this magnitude. They enriched this work through their valuable suggestions and discussions.

I am also grateful to all my colleagues at Philips Research Laboratories in both Eindhoven, The Netherlands and Aachen, Germany for offering assistance and making my stay more lively. Special thanks to Teun van den Heuvel, Rene Derkx, Frank Kneepkens, Albert Geven and Jiang Zhou for their suggestions and support during this work.

Last but not least, my sincere thanks and dedications go to my parents, Mrs. Kamala and Mr. Gopalsamy for their support and dreams that continue to drive me.

Mathankumar Gopalsamy

Abstract

Due to increasing awareness on importance of personal health and recent developments in communication technology, the use of wireless sensor network in healthcare industry increases steadily. Patient comfort and prolonged battery lifetime are the key requirements in battery-powered medical devices irrespective of the methods being applied. Wearable wireless sensor nodes, that are attached to an individual in order to monitor the physiological parameters, are ideal candidates for power conservation.

In this thesis, a configurable power consumption model is developed to estimate average power consumption of the ACcelerometer-based REspiration MOnitoring (ACREMO) system. The model enables the designers to predict the power consumption of each of the hardware components in the system. Various system parameters that impact the average power are identified during the modeling process. The model has estimated that the lifetime of the ACREMO system is 190 hours when driven by a 310 mAh Lithium-ion battery. A new type of respiratory monitoring algorithm is implemented on which current measurements are carried out to validate the model. The model incurs a mean error of -5.2%.

Different power optimization techniques were investigated to prolong the battery lifetime of the ACREMO system and a number of proposals are made. The replacement of the accelerometer and the operational amplifiers with their low-power counterparts would save up to 41% and 7% of the system power respectively. A dynamic power management policy is presented where a set of system states are identified and their usability is defined. The proposed heterogeneous multiprocessor system showed 16% increase in the device lifetime.

Reducing the CPU clock frequency and the sampling frequency resulted in 9% and 15% reduction of average power respectively. Finally, a context-aware power saving strategy is proposed that can save up to 40% of overall power by forcing the system to sleep for a predefined amount of time when a motion artifact is detected on the accelerometer signal.

By carefully selecting and combining the proposed techniques, the lifetime of the ACREMO device can be prolonged from the current estimate of 7.9 days to 14 days or more with few compromises such as the accelerometer noise, hardware cost, system complexity etc.

Contents

List of Figures

List of Tables

1. Introduction

Due to recent developments in sensing materials, computing elements and wireless technologies, the development of sensor systems has been receiving an increased interest and importance from researchers to engineers. A network of wireless sensors (physical, chemical or biological) is employed to sense environmental and physical conditions over the space of interest. Each of the sensor nodes is capable of sensing the parameter of interest, processing locally and transmitting the information over radio to a base station that collects the information and presents it to the user. Market researches foresee that the sale of wireless sensor network (WSN) will grow at 40-50% per year [1] and the number of application areas for WSNs continue to grow over time. Some of applications include [2–4]:

- temperature, humidity and pressure sensing
- intrusion detection
- security and tactical surveillance
- vehicular movement monitoring
- home intelligence
- habitat monitoring
- health monitoring and surgery

With increasing awareness on importance of personal health, the use of wireless sensor network in healthcare industry increases steadily. The healthcare application ranges widely from in-hospital patient monitoring (e.g. measuring vital signs, fall detection etc.) to home care (e.g. health check, elderly monitoring). At present, WSNs in healthcare industry target at chronic disease, aging independence, general health, wellness and fitness [5].

As the sensor nodes are attached to a monitored individual that may be ambulatory, the nodes are powered through a battery. The battery poses major constraints to the device's form factor and lifetime. Though many researches are being carried out to enhance the current battery technologies, in most applications, the battery capacity for the given form factor could not catch up with the ever growing need of computation in systems. Hence, the WSN designers should strive for energy efficient architecture of both the sensor node and the network as a whole.

Figure 1.1 shows a typical example of monitoring an individual with a network of sensors in healthcare. These smart and easy-to-use sensor nodes promote aging independence along with greater overall health, wellness, and fitness awareness. A variety of these products offer continuous monitoring of different physiological parameters and provide information to healthcare professionals like doctors and nurses. Due to recent advancements in communication technologies, one can monitor the healthiness of a person's even when the person is remote from hospital.

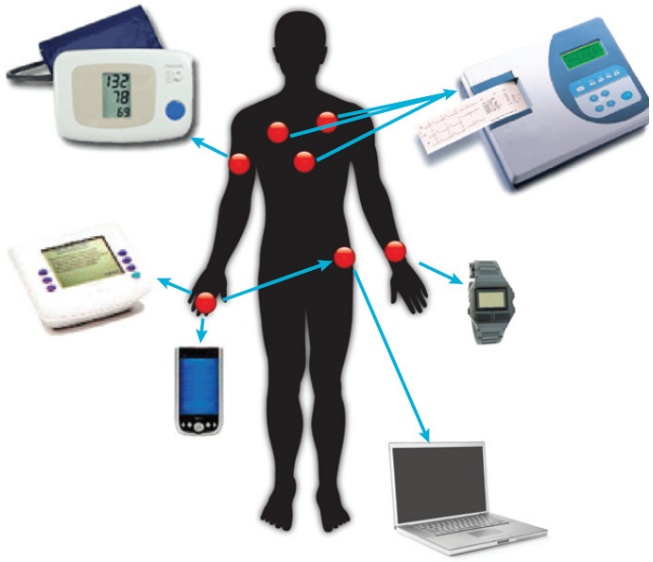


Figure 1.1.: Wireless sensor network (WSN) in healthcare [5]. A number of different sensor modalities are attached to the monitored individual. The processed measurements and/or physiological parameters can be transmitted wirelessly to display units such as bedside monitors, smart phones and computers.

1.1. Motivation

In healthcare applications, the patient comfort and prolonged battery lifetime are the key requirements irrespective of the parameters to be measured or the methods being applied. This is true for any of the sensor modalities that shall be deployed within this context. Using a battery with high capacity will result in extended lifetime but with effects of increased size, weight and cost of the sensor. A bulky wearable sensor will be obtrusive and inconvenient to wear over long time periods, especially for elderly and children.

Before attempting to optimize energy consumption of a sensor node, it is necessary to understand the power consumption of each of its subsystems and different parameters that impact the system battery lifetime. It is therefore necessary to

model the power consumption of the system to identify these parameters which can later be focused to achieve a power-optimized sensor node. With such a configurable power model, it is possible to estimate power consumption of the sensor node much before the actual hardware is realized. The power model helps to identify bottlenecks of the chosen software and hardware architecture much before the actual implementation so that the cost of rework is reduced. Such a model would enable the designers to estimate the battery lifetime of the sensor node under different use cases of the sensor and to set a meaningful target for the battery lifetime.

The system lifetime can only be maximized by incorporating energy awareness into every stage of the system design and operation. Thus, empowering the system with the ability to make dynamic tradeoffs between energy consumption, system performance, and operation fidelity [6]. The conventional low-power schemes [7] at CMOS level are insufficient for these highly energy-constrained systems and hence the more focus in this work is on the energy optimization techniques at architectural and algorithmic levels.

1.2. Related work

Wireless sensor networks is an active research area where many developments are seen over the last decade. Several WSN platforms are being developed and studied by research groups at different universities and industries. Some of the widely discussed platforms are the μAMPS from MIT, the Telos [8] from UC Berkeley, the Aquisgrain [9] from Philips Research, the iMote2 [10] from Intel, the MicroLEAP [11] from UCLA, the MicaZ from Crossbow etc.

Shih et. al. [12] have modeled the average power consumption of the radio used in the μAMPS project. The model incorporates startup time of the radio that can become significant when power cycling the radio between active periods in a low rate wireless network. The effect of the startup transient on the radio's energy per bit is also studied. Such a radio model is used in this work.

Researchers at the University of Tübingen, Germany have developed the AEON, an evaluation tool for calculating power consumption of the sensor nodes and the whole sensor network [13]. The model is based on the direct measurements of current consumptions in the target sensor node. In this work, at the time when the power consumption model was developed, the target hardware was not ready. Hence, the reliance was on the hardware architecture, data sheets of the components and some basic assumptions on the system behavior.

Sinha and colleagues [14] proposed an event prediction and sleep transition policies for a WSN node to exploit the idleness of the CPU with multiple power modes. The authors have developed a sleep-state transition policy that dynamically selects the system' low-power mode depending on the latency and power consumption of the particular mode along with the estimated probability of missing an event during the mode transition. The approach has to be revisited to suit to the data-driven

sensor nodes employing continuous sensing/monitoring.

Corroy et al. [15] analyzed the energy optimization in WSNs through the selection of processing elements. The first case study considered was *accelerometer sensing* where the authors conclude that a low-end microcontroller (MSP430) is more power efficient than a digital signal processor (DSP). The second case study was ECG sensor which possess relatively higher sampling rate and more signal processing in comparison to the first case. For this case, the authors claim that a heterogeneous multiprocessor approach is more power conserving than using a single low-end processor or a single DSP.

1.3. Objectives

The main objectives of this thesis are as follows:

- Modeling and estimation of power consumption of a representative wireless sensor node used in healthcare. The basic requirements of such a power model is that the model should be simple and configurable. The model will yield various parameters that impact the overall battery lifetime of the sensor node. The power consumption of each of the hardware components in the sensor node can be predicted and prioritized the power optimization strategies.

- Implementation of a health monitoring system with the available sensor node hardware. This involves translation of the available biomedical algorithm in MATLAB software to fixed-point arithmetic in *C* language. This system will serve as the basis for confirming the model estimations and perform some of the energy optimization techniques.

- Investigating the power optimization options at architectural and algorithmic levels. The list shall include both generic and system-specific techniques. Based on the studies, several proposals can be made to optimize the system power and prolong the battery lifetime.

1.4. Outline

The remainder of this thesis is organized as follows. Chapter 2 introduces the basic concepts and architecture of the wireless sensor node studied in this work. Chapter 3 focuses on the development of the power consumption model and estimation of battery lifetime. In Chapter 4, different architectural techniques to reduce the power consumption of the system are studied and evaluated. Chapter 5 discusses a set of selected algorithmic techniques for reducing energy consumption in the system. Chapter 6 discusses the observations made during this work and their implications. Finally, chapter 7 concludes this report along with a briefing on the future aspects of this work.

2. Respiration Monitoring System

This chapter introduces the basic concepts and architecture of the respiratory monitoring system on which this entire work of power modeling and optimizations were carried out. The first and second part of this chapter will discuss necessity and nature of the respiratory sensing method used in the system. The last part details about the hardware architecture of this system.

2.1. Needs for a new respiration sensing method

Quality-of-care for in-hospital patients is in the limelight over recent years, especially, in the developed world. Various measures are being introduced in regions like USA, Canada, and Europe to reduce the number of adverse events in hospitals, in particular at general wards. Such events include cardiac arrests, do-not-resuscitate (DNR) deaths and unscheduled admissions to intensive care units (ICU). Studies estimate that, in USA alone, about 320,000 patients die unnecessarily in medicine each year [16, 17].

The regular monitoring of vital body signs (namely respiratory rate, pulse rate, blood pressure, and temperature) is a key activity in assessing the healthiness of an individual. Multiple studies have shown that breathing and heart rate have high clinical values as an early indicator of patient's health deterioration [18–20]. Traditionally, patients in intensive and medium care units are well monitored while the general wards are often poorly monitored due to unavailability of nursing and suitable automatic monitoring systems. The suboptimal monitoring in low acuity units is a major contribution for morbidity and mortality in hospitals [19].

The continuous monitoring of patient's respiratory parameters over a long time period has been achieved with several methods such as spirometry, photoplethysmography (PPG) [21], respiratory inductance plethysmography (RIP) [22] and transthoracic impedance plethysmography (TIP) [23]. Most of these methods are often associated with cables, electrodes, belts or masks which make them obtrusive and unsuitable for ambulating patients. Otherwise, patients have to be visually monitored which is an arduous task due to shortage of nursing personnel and lack of supervision in non-critical medical wards. These facts call for a reliable, automatic and easy-to-use device to measure respiratory parameters for the patients supporting mobility.

Recent work has shown the feasibility to measure thoracic movements due to breathing with tri-axial accelerometers [24–27]. Driven by the development of the sensing method and micro-architecture, such an accelerometer-based approach is

well suited for lightweight, cableless and unobtrusive monitoring of vital signs.

2.2. Accelerometric respiration sensing

An accelerometer is an electromechanical sensor that measures the acceleration relative to a free fall known as proper acceleration and is expressed in multiples of acceleration due to gravity ($g = 9.8ms^{-2}$) [28]. For example, the sensor on earth experiences -1g proper acceleration when stationary and 0g while falling freely. With the advent of micro-electro-mechanical systems (MEMS), the use of accelerometers in commercial applications such as smart phones, digital media players, game consoles etc. has increased manyfold.

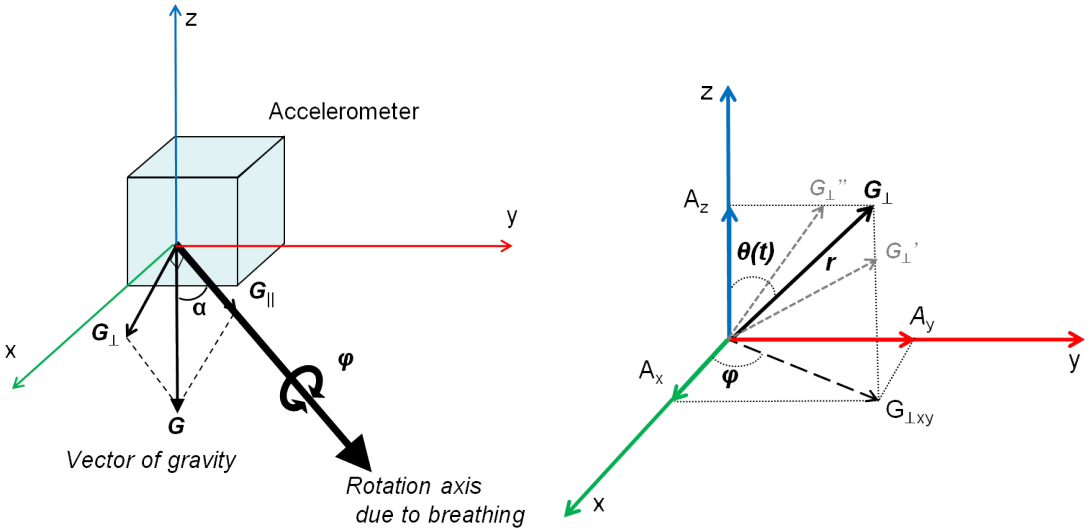


Figure 2.1.: Accelerometer as inclinometer. (a) shows the gravity vector and its components. The accelerometer can detect only the component perpendicular to the rotation axis. (b) shows the projections of $G_\perp$ onto the accelerometer's Cartesian coordinates x, y and z.

A tri-axial accelerometer measures proper acceleration as a vector and outputs its projections on each of the three orthogonal sensor axes. When attached to the patient's thorax, the accelerometer senses the vector of gravity $\vec{G}$ and its projections onto the sensor axes depending on the initial orientation of the sensor. Due to the movement of the rib cage during breathing, the attached sensor rotates around a nominal position. Assuming that the inertial acceleration pertaining to breathing as negligible, the sensor continues to measure the gravity vector but with changes in the axes projections. From the sensor perspective, this can be seen as rotation of the gravity vector along an arbitrary axis R as shown in Figure 2.1.

Since the parallel component of gravity vector $G_\parallel$ aligns itself with the axis of rotation, the accelerometer cannot sense this parallel component. Only the $G_\perp$

component is measured by the sensor. From the representation of the gravity vector in the spherical coordinate system (r, θ, ϕ) (Figure 2.1b), the sensor axes measurements at a nominal position can be formulated as:

$$\begin{aligned} A_x &= G_\perp sin\theta(t) cos\phi \\ A_y &= G_\perp sin\theta(t) sin\phi \\ A_z &= G_\perp cos\theta(t) \end{aligned} \tag{2.1}$$

The breathing effort is measured as θ changes over time when the gravity of vector swings around a nominal position during breathing.

This method of respiration sensing has the potentials of becoming a cableless, unobtrusive, low cost, and continuous monitoring device in the coming years. However, accelerometer's sensitivity to inertial accelerations challenges the reliability due to motion artifacts since the target group (patients in general wards and users at home) is more active than traditionally monitored subjects in ICUs.

2.3. System architecture

This section presents the architecture of the current ACcelerometric REspiration MOnitoring system (referred as ACREMO from here on). Understanding the details of each software and hardware components will play a vital role in modeling the system, identifying its limitations and exploring different possibilities for power optimizations as discussed in later chapters.

2.3.1. Signal processing

Figure 2.2 shows the processing steps involved in the accelerometric respiration monitoring system. As a first step, the accelerometer signals are digitized at a specific sampling rate and buffered for a definite period of time. The buffered signals are low-pass filtered at an upper bound of the breathing frequency (typically 2 Hz) and are down-sampled (known as decimation). Axes fusion techniques are used to reduce the triaxial data to single dominant axis data without losing much of information. A set of features are calculated from pre-processed signal to support classification of the signal into breathing and non-breathing segments. Then, respiratory parameters such as respiratory rate, pattern etc. are extracted from the classified breathing signal and presented to the external world through wireless transmission.

In continuous respiration monitoring, the trend of respiratory parameters over a period of time (for example, few hours to several days) is more important than the individual measurements. This justifies the use of a classification algorithm that discards bad signals rather than trying to recover them.

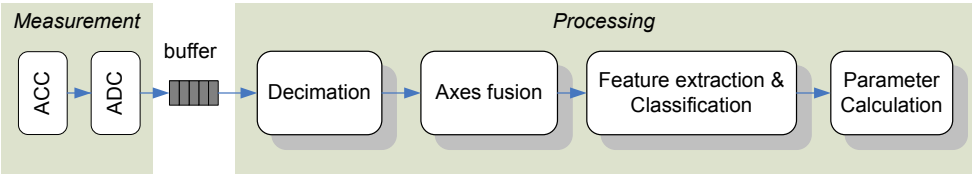


Figure 2.2.: Signal processing chain in the accelerometric respiration monitoring system

2.3.2. Hardware architecture

The ACREMO node is composed of,

Acceleration sensor which measures the proper acceleration of the sensor node;

Analog-digital converter that digitizes the outputs of the accelerometer;

Microcontroller that implements signal processing and parameter calculations;

Radio which consists of a short-range radio frequency circuit and allows wireless communication with the base station;

Battery that drives current to all the other components in the sensor node;

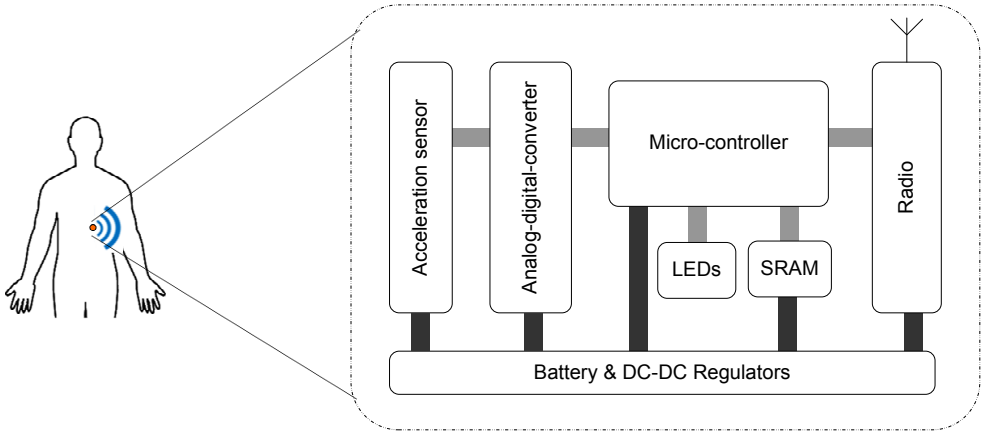


Figure 2.3.: Hardware architecture of ACcelerometric REspiration MOnitoring system (ACREMO)

Acceleration sensor

The ACREMO prototype system uses the LIS344ALH, an analog triaxial accelerometer based on the MEMS technology from ST Microelectronics [29]. The sensor includes a sensing element and an IC interface as an ultra-compact and low-power module. When the sensor is operated at a supply voltage (V_{dd}) of $3V$ with acceleration range of $\pm$2g, the sensitivity of the device is 600 μV/g. Typical current consumptions of LIS344ALH in active and power-down mode are 680 μA and 1 μA respectively.

The bandwidth of the device is selected by adding an external capacitor (C_{load}) to the sensor output. These capacitors in combination with output impedance of the sensor ($110K\Omega$), form RC filters that improve noise reduction and act as anti-aliasing filters. The capacitor values for output channel at a chosen cut-off frequency of f_c can be computed as:

$$C_{load} = \frac{1}{2\pi R_{out} f_c} \tag{2.2}$$

Analog-digital converter

The ACREMO prototype is built on *AD7689*, a 16-bit, 4-channel, successive approximation register (SAR) analog-to-digital converter that can be communicated via serial peripheral interface (SPI). Due to automatic power-down between conversions, the consumption of *AD7689* scales with throughput. For example, AD7689 consumes about 1 μA at sampling rate of 200 samples per second (sps) [30].

The outputs of the sensor are connected to ADC using operational amplifiers (OPAMP) in voltage follower configuration to avoid loading effects. This ensures that only a little current is drawn from the sensor while ADC is still driven with substantial current [31]. The OPAMP in use is OPA348 from Texas Instruments which consumes about $45\mu A$ per amplifier [32].

Microcontroller

The main computing module of the ACREMO platform is Atmel's SAM3U which is a Cortex-M3 based 32-bit microcontroller. Due to its integrated low-power modes, variety of peripherals, dedicated DMA, and high speed USB, SAM3U is becoming popular as processing unit for sensing and control platforms. It is a high performance controller with maximum clock frequency of 96 MHz and 1.25 DMIPS/MHz.

The Cortex-M3 core is based on the Harvard architecture with a 3-stage pipeline and features a single cycle multiplier and a hardware divider. The core offers efficient SRAM utilization through unaligned access and low latency interrupt handling [33].

The SAM3U provides up to 256 kB Flash and 52 kB of SRAM with maximum operating frequency of 96 MHz. The peripherals include USARTs, TWIs, SPIs, PWMs, 16-bit timers, RTC, a 12-bit and a 10-bit ADC. The device can select

main and auxiliary clocks from internal RC oscillators (32 kHz and 4/8/12 MHz) or external crystal-based oscillators (32 kHz and 24/36/48/96 MHz). In addition to normal (active) mode, the controller has 3 other low power modes: sleep, wait and backup modes. The properties and usage of the clock sources, peripherals and power modes are discussed in Section 4.2. The current consumption of the microcontroller scales with the main clock frequency at a rate of ∼0.5 mA/MHz, for example, with all peripherals deactivated the SAM3U consumes 48 mA at maximum operating frequency of 96 MHz (Crystal oscillator) and 10 mA when clocked at 12 MHz (RC oscillator) [34].

External memory

The system's memory is augmented externally with a static SRAM. *CY62157EV30* is an ultra low-power CMOS memory (8 MB) that consumes about 1.8 mA ($f = 1MHz$) when active and 2 uA when idle [35]. Its memory organization in 16-bit words and high speed of operation (45 ns) makes it suitable for battery operated data processing applications. The minimum value of read/write cycle time (45 ns) limits the maximum operating frequency to 22.22 MHz.

Radio

The radio on board is TI's low-power RF transceiver CC2420 which is an IEEE 802.15.4 compliant radio that operates in the 2.4 GHz ISM band. The CC2420 features several dedicated hardware for packet handling, data buffering, data encryption (AES-128), clear channel assessment etc. Such coprocessor in the radio chip reduces computational load on the CPU. The radio consumes about 17.4 mA (transmit power 0 dBm) in transmit mode and 19.7 mA in receive mode. The transmit and receive energy per bit are 226 nJ and 209 nJ respectively [36]. High data rate (effectively 250 kbps), low power consumption and freely available radio stack make the CC2420 widely used radio in sensor platforms [37].

Battery

Biomedical wearable devices, in general, have constraints on size, weight, and heat dissipation. The battery, which contributes the majority of the device's form factor and weight, must be carefully chosen to meet the power requirements. The respiratory sensing in ACREMO is assumed to be continuous and last over several days or weeks. This makes it impractical to drive the system by a button cell or disposable battery.

Lithium-ion batteries are widely used for power-hungry portable devices. Varta's *LPP402934E* is a Lithium-ion battery pouch that can meet our requirements [38]. This battery has a rated capacity of 310 mAh (at 0.5 C) and nominal voltage of 3.7 V. The cut-off voltage of the battery is $3.0V$ matching the operating range of the hardware components in the ACREMO system.

2.4. Summary

The accelerometric respiratory monitoring has the potentials to become a low-cost, unobtrusive and cableless solution to sensing respiratory activities in individuals. The system is also suitable for out-of-hospital monitoring where nursing is often unavailable. The work in the following chapters focuses on modeling and optimizing power consumption of a prototype system. The system measures the acceleration signal, processes, filters breathing information, compute respiratory parameters, and finally transmit the results over radio to a central station.

3. Power Modeling and Estimations

The objective of power modeling of the ACREMO platform is to estimate average power consumption of the system and thereby arrive at average lifetime of the device between battery recharge cycles. Power modeling is essential when the system is in development stage. A power model facilitates the selection of choices for system features, hardware components, and software design strategies before the actual hardware is implemented. Modeling reduces amount of time and cost spent on hardware or software rework. Since direct hardware measurements are not possible during designing phase of a system, the power model will have to rely on theoretical calculations and data sheets of the components.

This chapter starts with introducing a system level picture of the power consumption in the ACREMO system before going for power modeling at each part of it. In the last section, we will show model estimations of system's average power and battery lifetime under different use cases.

We follow a bottom-up approach that begins with formulating consumption of each hardware component in the architecture (described in Section 2.3.2). In this way, the contributions of each of the components to the average system power can easily be identified. During the modeling process, tuneable parameters that affect a component's power consumption are identified and their effects on device lifetime are studied. With few assumptions on specifications of system's behavior, the model parameters can be configured for various application scenarios and system's average power can be estimated for each of the scenarios.

The hardware components in the sensor node can be grouped under one of the following subsystems based on their functionality [14, 39]:

- Power supply subsystem
- Sensing subsystem
- Computing subsystem
- Communicating subsystem

This approach allows easy reconfiguration of the model for scalable systems. The model can be adapted for inclusion or modification of hardware components and system features.

When a subsystem is temporarily unused, it can either be power cycled to a low-power mode or turned off in order to conserve power. Cycling to a low-power mode is preferred over complete shutdown in applications where a fast response is desired during transition between active and idle modes. The model incorporates two basic

power modes, active and idle, for each subsystem with system's power cycling capability in mind. Following sections detail the power consumption modeling of each of the aforementioned subsystems.

3.1. Power supply subsystem

This subsystem comprises a battery as the only source of power and voltage regulators that regulate the battery voltage (V_{batt}) to provide a stable supply (V_{dd}) to each hardware component. Figure 3.1 shows the distribution of battery current to other subsystems through linear regulators. At any point in time, the current drawn from the battery is the sum of currents driving each subsystem in the platform.

$$I_{sys} = {I_{sens}}' + {I_{comp}}' + {I_{comm}}' \tag{3.1}$$

$$= \frac{I_{sens}}{\eta_{sens}} + \frac{I_{comp}}{\eta_{comp}} + \frac{I_{comm}}{\eta_{comm}} \tag{3.2}$$

where I_{sys} is the system current consumption,
$I_{sens}, I_{comp}, I_{comm}$ are the current consumptions of sensing, computing and communicating subsystems respectively,
$\eta_{sens}, \eta_{comp}, \eta_{comm}$ are the conversion efficiencies of the voltage regulator supplying sensing, computing and communicating subsystems respectively.

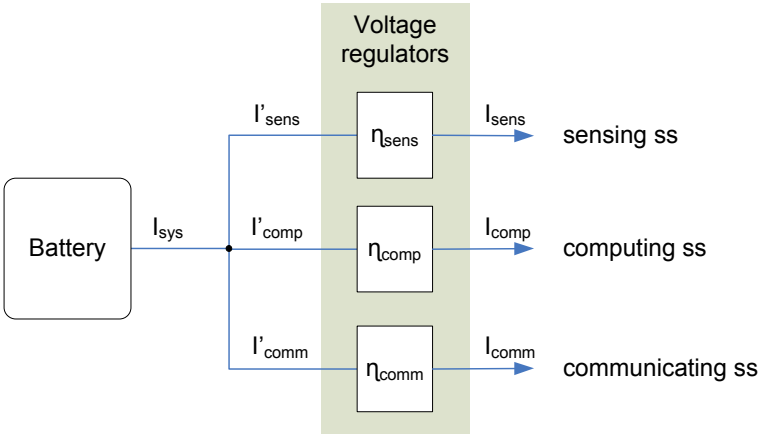


Figure 3.1.: An overview of current consumption in ACREMO platform among the four subsystem

The conversion efficiency of a linear voltage regulators depends on output current of the regulator that drives a subsystem. Low drop-out (LDO) regulators are suitable candidates for our application as they carry low noise along with minimal

drop out voltage ($V_{batt}-V_{dd}$). In such linear regulators, the power dissipation occurs in the form of heat which is equivalent to input current times the dropout voltage of the regulator. Hence, with a non-ideal linear regulator, energy delivered to the subsystems is less than the energy drawn from the battery. Since battery voltage in a typical Li-ion battery changes (slightly) over time, the power dissipation and the conversion efficiencies of the regulators also vary over time.

When the battery voltage and the supply voltage of a component/subsystem differ considerably, it is wise to employ a buck voltage converter.

3.1.1. Battery model

This work uses a linear capacity battery model, where the battery is considered as linear storage of current. Also, it is assumed that the battery capacity after every recharge cycle is same as its initial rated capacity. With the knowledge of average discharge current, the linear model allows us to compute lifetime of the battery directly from its rated capacity (C_{rated}) as:

$$LIFE = \frac{C_{rated}}{I_{sys}} \tag{3.3}$$

Lifetime of a battery depends on several factors including dimensions, density, types of electrolyte material and electrodes. In practice, a battery exhibits non-linear behavior that may defy the calculation in Equation 3.3. Such variation is also application dependent. Two major phenomena causing non-linearity in batteries are *rated capacity* and *relaxation* effects.

Rated capacity effect explains why lifetime of a battery is shortened when discharge current is higher than rated current. When a target draws current from a battery, charged ions are released from the anode by oxidation and are reduced at the cathode after being conducted through electrolyte. The reduction process produces inactive spots in the cathode that are not useful anymore. The distribution of such inactive spots depends on the rate of current discharge. For example, when discharge rate is low, distribution of reduction spots at the cathode is uniform. That allows the entire cathode to be used for the reduction process. On the contrary, when discharge rate is higher than the rated value, outer layers of the cathode will be depleted quickly. This causes later current discharges to fail as inner layers of the cathode (which still have active material) are non-reachable for further reduction. Due to this phenomenon, the battery dies earlier than expected even though some active material is left over [6, 40].

Relaxation effect explains how intermittent discharge cycles allow the battery to regain some of its lost capacity. During a discharge cycle, active ions that are consumed at the cathode are replaced by new ions from the electrolyte through a diffusion process. When the discharge current rises, oxidization at the anode increases to produce more active ions whereas the electrolyte is incapable of diffusing equivalent amounts of active ions to the cathode, causing imbalance in amounts of

active ions between the two electrodes. As a result, the potential across the battery terminals drops. If the discharge cycles are discontinuous, i.e. every discharge period is followed by an idle period, the idle time can be used by the electrolyte to diffuse active ion and equalize their concentration levels between electrodes. This behavior of recovering charge, like a capacitor, in a battery is called *relaxation effect*. Several works have shown that the use of battery power-aware applications that periodically switch between active and idle periods (low power or shutdown) increases overall battery lifetime significantly [40, 41].

Based on these phenomena, it is clear that reducing average discharge current alone is not sufficient to ensure optimal battery lifetime. Other important parameters are maximum discharge current, duration of the current discharge and the idle cycles.

From the ACREMO algorithm in Section 2.3.1, it can be observed that the ACREMO system is a strong candidate for power cycling between sampling activities and buffer-full events. In other words, the system can be designed for pulsed current discharge [41] which will improve battery capacity due to relaxation effect. As long as the drawn current is below the rated capacity, we can expect that our linear model will yield estimations that are more close to practical measurements.

3.2. Sensing subsystem

This subsystem is composed of accelerometer, ADC and amplifiers. The average current consumed by the sensing subsystem is the sum of its active and idle period consumptions.

$$I_{sens} = D_{sens}\, I_{sens,active} + (1 - D_{sens})\, I_{sens,idle} \tag{3.4}$$

$$D_{sens} = \frac{T_{active}}{T_{active} + T_{idle}} \tag{3.5}$$

where D_{sens} is duty cycle of the sensor subsystem, $I_{sens,active}$ and $I_{sens,idle}$ are current consumptions of the subsystem in active and idle modes respectively.

The total current consumed by the sensing subsystem is calculated as the sum of average current consumptions of accelerometer I_{acc}, amplifiers I_{amp}, and analog-to-digital converter I_{ADC}. For example, the current consumption of the sensing subsystem in active mode can be written as:

$$I_{sen,active} = I_{acc,active} + n_{amp}\, I_{amp,active} + I_{ADC,active}(F_s) \tag{3.6}$$

where n_{amp} is the total number of amplifiers used and F_s is the sampling rate expressed in SPS (samples per second). Since AD7689 automatically powers down after every conversion, its active mode current is a function of sampling rate F_s, the number of conversion requests from CPU per second.

3.3. Computing subsystem

3.3.1. Sources of power consumption

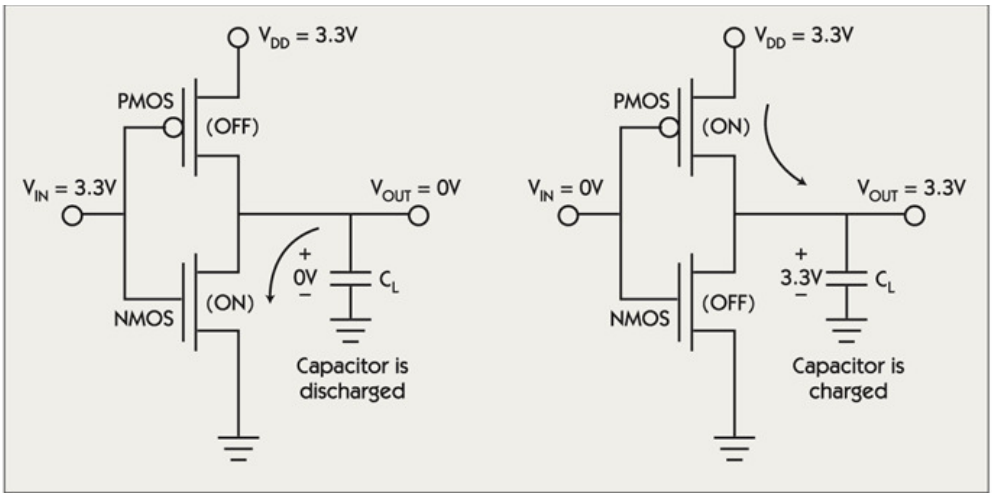


Figure 3.2.: Dynamic current consumption in CMOS

Both CPU and SRAM that form this subsystem are digital chips which are most often fabricated with complementary metal oxide semiconductor (CMOS) technology. A simple CMOS (static) gate is constructed from a PMOS and an NMOS transistors as shown in Figure 3.2.

In a CMOS circuit, the average power dissipation is a combination of four different components, namely dynamic $P_{dynamic}$, short P_{short}, leakage $P_{leakage}$ and static P_{static} power consumptions [7]. The amount of contribution from each of these components vary depending on the CMOS technology and application of the system. Following section briefly discusses each of these components.

$$P_{avg,CMOS} = P_{dynamic} + P_{short} + P_{leakage} + P_{static} \tag{3.7}$$

The dynamic power, $P_{dynamic}$, is the power dissipation during output switching and is equivalent to flow of electric charge between power supply V_{dd} and the output capacitance C_{out} which can be written as:

$$P_{dynamic} = KC_{out}{V_{dd}}^2 f \tag{3.8}$$

where K is the average number of output transitions in half a clock duration and f is the clock frequency.

Dynamic power reduces linearly with frequency of the circuit and quadratically with supply voltage. Most of the modern microcontrollers operate at lowered core voltage ($\simeq 0.7V$) when compared to input-output ports ($3V$). Further reduction in operating voltage impacts the maximum frequency of operation, which directly affects performance of the system. Dynamic power component is also dependent

on switching activities in the entire circuit. The more the gates being switched at output, the more is the dynamic consumption. In heavy-duty systems, dynamic power is the dominant component of circuit's overall power.

The transient behavior of the transistors during switching cause both transistors turned *on* for a short period of time before one switches off completely. As a result, a short-circuit current I_{sc} flows from V_{dd} to ground. Short-circuit dissipation amounts only a small part ($P_{short} < 10\%$) of the circuit's overall power.

Leakage power component plays a dominant role in low-duty systems as this component is present even in the absence of switching activity ($f = 0$). As modern CMOS chips continue to shrink in size and scale operating voltage, leakage becomes an increasingly important phenomenon in CMOS circuits. There are estimates that, in the coming years, leakage power can contribute up to 40% of total power consumption of a chip.

Static power dissipation occurs due to the current flowing from the supply to the ground through the transistors in the absence of output switching. As there is no conducting path from the supply to the ground when output logic is steady ('0' or '1'), static current is zero in fault-free circuits.

The average current consumed by the computing subsystem is the sum of average currents of CPU and SRAM:

$$I_{comp} = I_{CPU} + I_{SRAM} \tag{3.9}$$

3.3.2. CPU

The SAM3U microcontroller has three low power modes (LPM): sleep, wait and backup. Each of these modes are characterized by idle consumption, transition time, wake-up sources, and core status after recovery. Hence, the average CPU current depends mainly on the duty cycle, the chosen LPM, and frequency of operation. Figure 3.3 shows the SAM3U's current consumption in active and sleep modes at 3.3. The wait and backup modes are frequency independent and consume only in the orders of few micro Ampere.

In the ACREMO system, the SAM3U controller is supplied with two different power supplies:

- V_{DD} (typically 3 V) which drives the input-output ports, analog converters, etc. A linear voltage regulator is used to regulate this voltage from the battery whose conversion efficiency is η_{comp}.
- V_{DDCORE} (typically 1.8 V) that drives the Cortex-M3 core, on-chip memories, peripherals, main oscillators etc. A synchronous step-down (buck) regulator is used to convert the battery voltage to this core voltage.

The CPU has two major tasks to perform in the ACREMO system: sampling and processing of the accelerometer signal. The following sections show the estimation of the CPU power consumption during these activities. See A.4 in Appendix A for details.

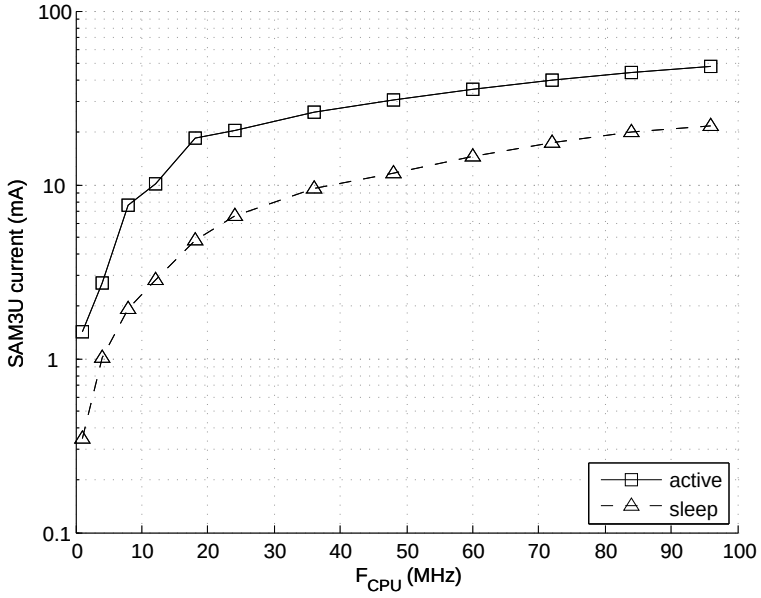


Figure 3.3.: SAM3U current consumption in normal and sleep (one of LPM) modes

Sampling

The sampling task represents a low-utilization phase of the CPU. When the sampling is triggered, the CPU

- wakes up from the low power mode with default/preset clock frequency.
- switches the clock source and set the clock frequency to a value suitable for active mode operation. This involves a clock startup (t_{clkst}) and a switching (t_{clkup}) delay.
- requests ADC conversion(s), reads and stores the sample(s) once the conversion is over. The time delay between the request of conversion and availability of the the sample is t_{adc}.
- After saving the sample(s) into memory, the CPU switches the clock source again and select a frequency suitable for idle mode before entering the low power mode. The clock switching delay in this case is $t_{clkdown}$

The power model predicts the power consumed during sampling by estimating the time delays involved in each of steps mentioned above along with their corresponding current consumptions. It is observed that the clock switching delays dominated the entire time required to perform the sampling task. The switching between RC oscillator and crystal oscillator, a considerable amount of processor cycles

Processing

The calculation of processing part of the CPU power consumption is straightforward. Since the CPU will be operated at active mode during while executing the algorithm, the consumption depends on the clock frequency, peripherals in use, and duty cycle for the algorithm execution.

$$I_{CPU,samp} = D_{proc} \cdot \left(\frac{I_{io}(F_{CPU})}{\eta_{comp}} + \frac{I_{core}(F_{CPU})}{\eta_{cpu}} \right) \tag{3.10}$$

where
D_{proc} is the duty cycle for processing excluding the sampling task,
F_{CPU} is the CPU clock frequency in active mode,
I_{io} is the current consumed by input-output ports at V_{DD} (typically 3.0 V),
I_{core} is the current consumed by the core, on-chip memories and peripherals at V_{DDCORE} (typically 1.8 V),
and η_{cpu} is the conversion efficiency of the buck voltage converter that supplies the core voltage V_{DDCORE}.

3.3.3. SRAM

As part of the system assumptions, the SRAM is expected to be active while the CPU is active. So, the duty cycle of the CPU (D_{proc}) is used to estimate the average current consumed by the SRAM from its active and idle mode consumptions as:

$$I_{SRAM} = D_{proc} \cdot I_{SRAM,active} + (1 - D_{proc}) \cdot I_{SRAM,idle} \tag{3.11}$$

The SRAM's active mode current ($I_{SRAM,active}$) depends on the selection of clock frequency F_{SRAM}.

3.4. Communication subsystem

The communication subsystem is formed by light emitting diodes (LEDs) and radio transceiver which are used to exchange information between the ACREMO system and external world. So, the subsystem consumption can be written as:

$$I_{comm} = I_{LED} + I_{radio} \tag{3.12}$$

LEDs are used in many consumer applications to visually signal the user during different system activities such as acknowledging received user commands, signaling the system activeness, showing healthy or poor battery status etc. The advantage of LEDs is that they can be operated at low frequencies and low duty

cycle with negligible startup time which makes them suitable for battery-powered applications. The average LED current can be written as:

$$I_{LED} = f_{LED}\ D_{LED}\ I_{LED,on}(LI) \tag{3.13}$$

where f_{LED} is the turn-on frequency, D_{LED} is the duty cycle, $I_{LED,on}$ is the forward current and LI is luminous intensity.

The forward current required to drive a LED varies with desired amount of luminous intensity which is often expressed in milli-candela (mcd). For example, a compact surface mount LED consumes about 20 mA at full intensity of 60 *mcd* while consuming only 2 mA at 6 *mcd* [42].

There are several factors affecting a radio's consumption such as data rate, transmission distance, modulation scheme, and duty cycle. In most of the systems, leaving a radio in idle mode is equally expensive as radio in receive mode and can be more expensive than a CPU at full operation. Hence, it is highly preferred to operate a radio in a scheme where the radio can periodically wake up, transmit or check for reception and shutdown soon after.

Shih et. al. [12] have formulated the average power consumption of a radio as:

$$\begin{aligned} I_{radio} =& N_{tx}[I_{tx}(T_{on-tx} + T_{start}) + \frac{P_{out}}{V_{dd,radio}} T_{on-tx}] \\ &+ N_{rx}[I_{rx}(T_{on-rx} + T_{start})] \end{aligned} \tag{3.14}$$

where,
$N_{tx/rx}$ is the average number of transmission/reception cycles per second,
$I_{tx/rx}$ is the current consumed by radio during transmission/reception,
$T_{on-tx/rx}$ is the duration the radio is on for transmission/reception,
T_{start} is the startup time of the radio,
P_{out} is output transmit power,
and $V_{dd,radio}$ is the supply voltage of transceiver.

The turn-on time of the radio for transmission or reception can be estimated as,

$$T_{on-tx/rx} = \frac{B_{tx/rx} * 8}{R} \tag{3.15}$$

where R is the data rate of the radio, and $B_{tx/rx}$ is the total number of bytes to be transmitted or received per cycle of operation.

3.5. Lifetime estimation

From the component level currents (shown in the sections above), we can now deduce the overall average consumption of the device under different application scenarios. Before introducing the application scenarios considered for this estimation, it is worthwhile to brief the known system behaviors and assumptions made during this estimation process.

3.5.1. System behavior and assumptions

The respiratory monitoring system under discussion is expected to continuously monitor the user's respiration activity and record the respiration rates. This requires continuous sampling of accelerometer outputs as long as the device is operating. The sensor platform's startup time (Equation 3.16) is commanded by the accelerometer's output RC filter [29] which is calculated as:

$$t_{acc,on} = 5 * R_{out}C_{load} + 0.3\ ms \tag{3.16}$$

For example, a cut-off frequency of 44 Hz can be achieved with a load capacitance of 33 nF. Then, the startup time is 18.425 ms. However, the inter-sample duration when sampled at Nyquist rate (2^*f_c = 88 Hz) is only 11.4 ms. As the startup time exceeds the inter-sampling duration, power cycling the accelerometer between samples becomes impractical. It can be concluded that, in normal circumstances, the accelerometer is always *on* (D_{sens}=100%).

As previously mentioned, the model accounts for power cycling of the microcontroller where the duty cycle of the subsystem and the idle mode selection are among model parameters. The application can choose either of three available power modes of SAM3U, sleep, wait and backup. Both *sleep* and *wait* modes are suitable for normal operation as they keep memories powered whereas *backup* mode requires a core reset on wake-up.

In sleep mode, only the core clock is stopped while peripheral clocks can be enabled. Hence, idle power in sleep mode is dependent on master clock frequency where major power consumption is due to PLL and peripherals. This mode is preferred when external peripherals require continuous clocking from the CPU.

In wait mode, clocks to core, peripherals, and on-chip memories are stopped while all power supplies are *on*. The purpose of the wait mode is to achieve very low power consumption (8μ A) while maintaining the whole device in a powered state for a startup time of less than 10μ s.

The CPU's computation time in the ACREMO application can be divided into two parts, sampling and processing (with the assumption that the load on CPU for radio transmission is negligible). The ACREMO algorithm operates on a fixed sampling rate of 125 Hz. From the SAM3U's clock switching times and the worst case time for external ADC data acquisition, t_{sample} is estimated to $\simeq 600\mu$ s.

The ACREMO algorithm (discussed in Section 2.3.1) occupies only 0.25% of processor's time on a SAM3U evaluation board running at full speed of 96 MHz without a RTOS. A realistic and little conservative value for duty cycle of the computing subsystem D_{proc} was derived as 5%.

As radio transmission per bit is more expensive than local computations, wireless sensor nodes often process the measured signal locally, store intermediate results and periodically transmit them to a base station. The ACREMO system also wakes up the radio every few minutes to transmit stored results since the last transmission. Also, the sensor node wakes up the radio at a predetermined periodicity to listen for any message from the base-station. Once communication is over, the radio is turned off immediately.

The communication between the ACREMO sensor platform and base-station uses the IEEE 802.15.4 standard [43], for low-rate wireless personal area networks, which defines the medium access control (MAC) and physical (PHY) layers. The ZigBee specification which defines the network and application layers is laid over the IEEE 802.15.4. In our work, the radio chip CC2420 is operated at $V_{dd} = 3$V with data rate R=250 kbps. CC2420's startup time is the sum of oscillator startup (1 ms) and PLL lock up time (192 μs).

Espina et.al. [44] have used the same radio chip with ZigBee for a biomedical application and showed that packet error rate while transmitting data within 5 meters was only 1% in most of the cases. This work assumes that the packet error rate, network startup time and channel access time are negligible.

3.5.2. Power budget

In this work, an optimistic target of 7 days of operation (168 hours) is set as lifetime budget for the ACREMO system. This means discharging an average current of $\simeq 1.845$ mA from a Li-ion battery with capacity of 310 mAh under linear battery model. In other words, the current and power budgets for the ACREMO prototype system are 6.827 mW and 1.845 mA. Since capacity loss due to self-discharge of Li-ion batteries ranges typical from 5-10%, we expect that the effect of self-discharge on this power budget to be negligible.

3.5.3. Use cases

Two different use cases of ACREMO system, namely *UC1* and *UC2*, were considered to illustrate the capability of power estimation with the power model.

Use case 1 is defined with generic system specification and power cycling in mind. For example, it uses all the four current amplifiers in the hardware, one for each ADC input from accelerometer and one for providing ADC reference voltage. The advantage of choosing *sleep* as CPU idle mode is that when CPU is idling, internal and external peripherals can still be serviced with (reduced) clock. LED is used as visual message for active operation and is configured for blinking for every 3 seconds with *on* period of 300 milliseconds.

Use case 2 is a little more energy-aware configuration. The ADC which is configured as bipolar in *Use case 1* is operated in unipolar mode in this configuration. The bipolar configuration needs an additional current amplifier to drive the ADC reference voltage. However, using the CPU to convert unipolar ADC values to bipolar ($\simeq 125 * 3$ subtractions) is less expensive than the average power consumed ' this additional amplifier ($45\mu\ A \times 3.0\ V$). Employing *wait* mode for CPU idling will significantly reduce the system's idle power. In a hospital setup, the sensor node is attached to the patient's thorax and is visually hidden most of the time under clothes or blanket. This motivated us to consider removal of LED in this use case. Also, the periodicity and exchange data size of the radio are lowered to represent compute-at-node approach of biomedical sensor networks.

Table 3.1.: Use cases for the ACREMO system and choice of values for model parameters

Subsystem	Parameter	Use Case 1 (UC1)	Use Case 2 (UC2)	Units
power	C_{rated}	310	310	mAh
	$\eta_{sens}, \eta_{comp}, \eta_{comm}$	81.08	81.08	%
sens	D_{sens}	100	100	%
	n_{amp}	4	3	
	F_s	125	125	Hz
comp	D_{proc}	1	1	%
	$F_{CPU,active}$	96	96	MHz
	$F_{CPU,idle}$	4	4	MHz
	Idle mode	sleep	wait	
	F_{SRAM}	1	-	MHz
comm	f_{LED}	1/3	-	Hz
	D_{LED}	30	-	%
	LI	6	-	mcd
	N_{tx}	1/60	1/900	
	N_{rx}	1/60	1/60	
	Btx	20	40	bytes/cycle
	Brx	20	10	bytes/cycle
	P_{out}	0	0	dBm

Table 3.1 shows the specific values used for important model parameters. The power and sensor subsystem parameters are more or less the same for both cases. The regulator efficiencies are computed as ratio of output and input voltages ($3.0/3.7 = 0.81$). The CPU is operated at its full speed when active and at a reduced frequency while in idle mode.

3.5.4. Results

Table 3.2 lists the estimated average current consumption of hardware components and the system as a whole. The table also shows the contribution of these component as a percentage of the available current budget. In both configurations, the major contributors for system's average current are CPU, accelerometer and regulators. The ACREMO system consumption in UC2 is expected to be meeting the current target which guarantees the battery lifetime of 7 days.

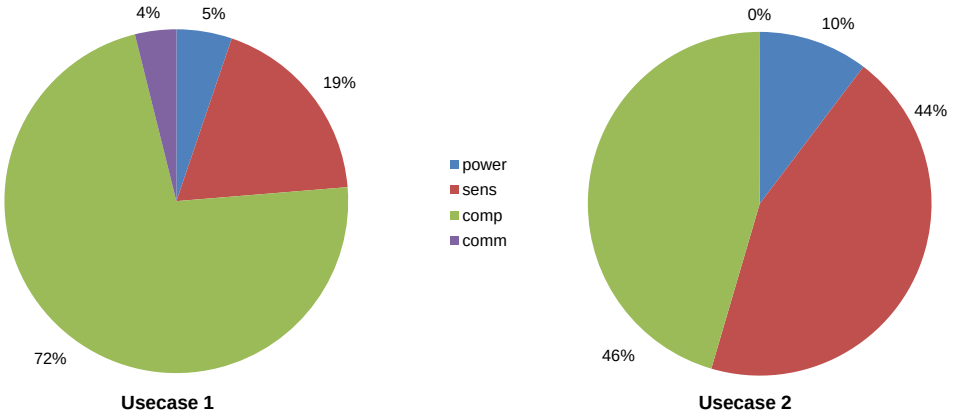
Figure 3.4a shows the average power consumption of the hardware subsystems as percentage of the system current. The component-wise contributions to system power are plotted in Figure 3.4b. Figure 3.4c shows the estimated battery lifetime of the ACREMO system under both cases. The sensor node can serve up to 67 (2 days 19 hours) and 168 hours (7 days) under *use case 1* and *use case 2* configurations respectively. It is important to remember that both the configurations assumed a conservative value for duty cycle for CPU processing

Table 3.2.: Model estimations of component-wise currents in units of μA

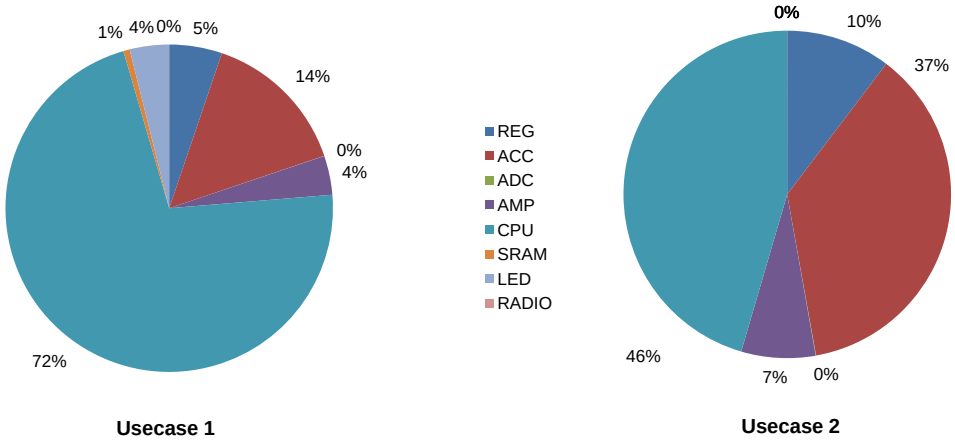
component	Usecase 1		Usecase 2	
	I_{avg}	% budget	I_{avg}	% budget
Regulators	243.1	13.18	190.5	10.32
Accelerometer	680.0	36.86	680.0	36.86
ADC	0.7	0.04	0.7	0.04
Amplifiers	180.0	9.76	135.0	7.32
CPU	3344.4	181.27	838.2	45.43
SRAM	28.8	1.56	0.0	0.00
LED	180.0	9.76	0.0	0.00
Radio	1.1	0.06	0.5	0.03
System	4658.1	252.47	1844.9	99.99

(D_{proc} = 1%). Since the existing algorithm run on the SAM3U platform with processing duty of D_{proc} = 0.23%, the system can achieve a lifetime of about **190 hours** (7 days 22 hours).

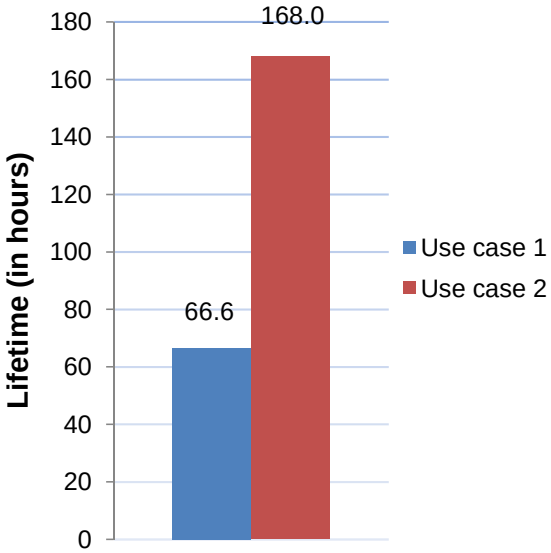
The variation in the average system current against some of the model parameters are plotted in Figure 3.5 for both use cases. The sensor subsystem, which is assumed to be continuously operational in these configurations, can considerably reduce the system current when power cycled (Figure 3.5a). Although the processing duty cycle of the CPU is set to a low value (1%), both the configurations show that the CPU is the most expensive component in the system. When the processing duty D_{proc} increases, the average system current increases significantly as shown in Figure 3.5b. For example, the system with D_{proc} = 30% consumes about 10 mA which bring down the battery lifetime to approximately 31 hours. Hence, it is important to maintain the processing duty as low as possible. Figures 3.5c and 3.5d shows the system current consumption against the model parameters from the communication subsystem. The figures show that the radio transmission can be very expensive while increasing the number of bytes exchanged or/and the number of communication cycles per second. The choice of transmitting the raw accelerometer samples over radio ($B_{tx} = 750\ bytes, N_{tx} = 1$) would be the most expensive operating point for the communication subsystem.



(a)



(b)



(c)

Figure 3.4.: Model estimations of ACREMO system

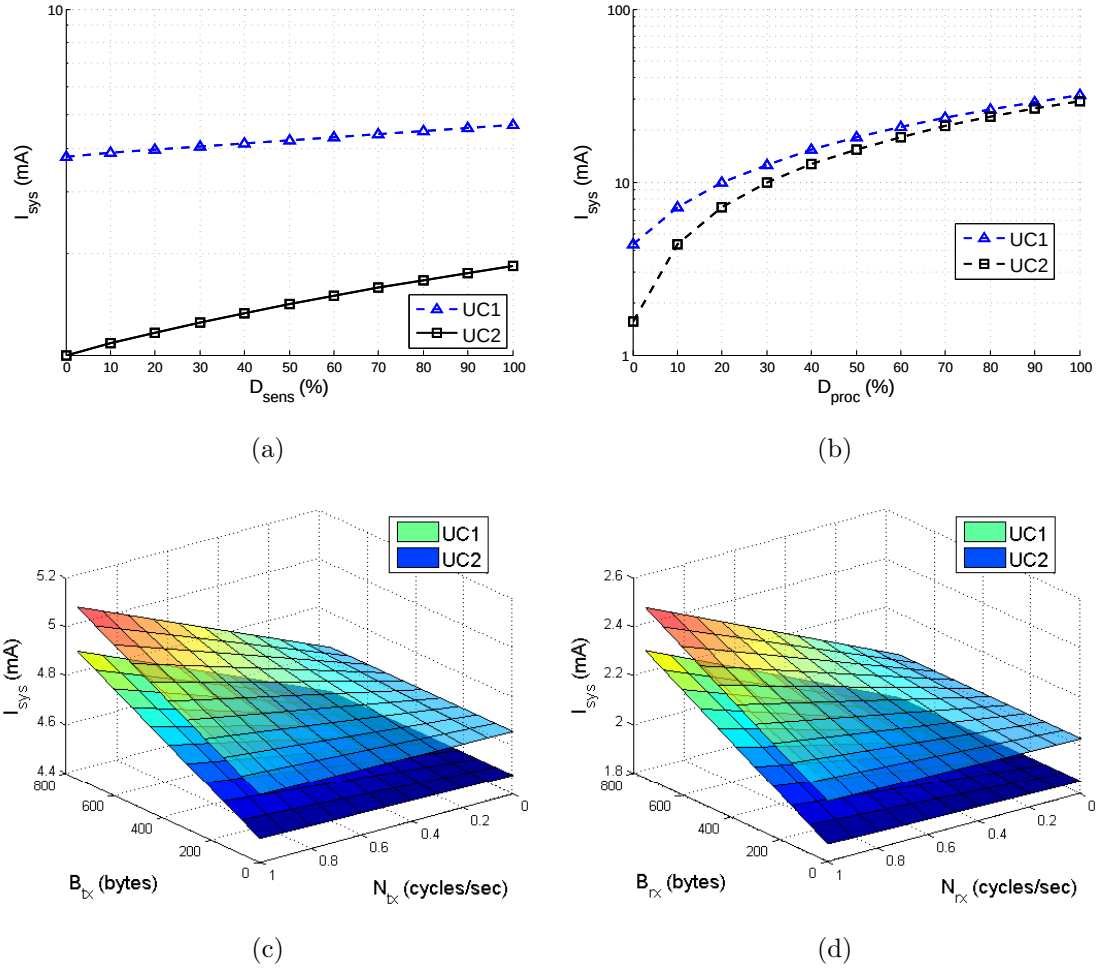


Figure 3.5.: System current vs. model parameters (a) Duty cycle of the sensing subsystem (b) Duty cycle of the CPU processing (excluding sampling task) (c) Number of transmit cycles per second and number of bytes per transmission (d) Number of receive cycles per second and number of bytes per transmission

3.6. Summary

The hardware components in the ACREMO system are divided into four subsystems, namely power, sensing, computing, and communication. The average consumption of each of these subsystems is modeled in order to estimate the battery lifetime of the device between successive recharge cycles. The model is based on a linear battery model for simplicity and its promise of an acceptable accuracy in low duty systems. A number of configurable model parameters are identified that considerably affect the system consumption. The accelerometer and the CPU turned out to be the major contributors of system's power consumption. The

radio can be very expensive when raw data or processed data are transmitted to the central station instead of sending only the physiological parameters. The power model is illustrated by assuming two different and practical use cases of the ACREMO system. Depending on the system requirements and complexity of the algorithms, the prototype ACREMO device can be operated continuously for 67 to 168 hours with a 310 mAh Lithium-ion rechargeable battery.

4. Power Optimization at Architectural Level

This chapter introduces some of the architectural choices for optimizing the power consumed by the ACREMO system. By studying these choices, we evaluate their applicability to the system and gain in battery lifetime.

4.1. Component selection

The selection of a hardware component plays a vital role in system's static and dynamic power consumptions. Some of the important criteria to look for while choosing a component are performance, active or idle consumption, startup time, power cycling capability under the intended application etc. This section discusses few open choices for ACREMO hardware components and their tradeoffs.

4.1.1. Accelerometer

Factors that affect power consumption of a MEMS based inertial sensor are type of transduction (sensing) technique, detection circuit, and interface circuit (for digital output). Commonly used transduction techniques in accelerometers are piezo-resistive, capacitive and tunneling. The piezo-resistive sensors demand higher static current and have poor overall sensitivity compared to other two techniques. A capacitive acceleration sensor has high sensitivity, low-cost, low power consumption and is widely used in low-g applications. Tunneling accelerometers are used in high-sensitivity applications and require high supply voltages.

Capacitive accelerometers are the suitable candidates for our application considering their low power operation at reasonable sensitivity. The detection and interface circuits contribute significantly to energy consumption of the sensor as a function of their detection or conversion time. Table 4.1 shows few of the triaxial accelerometers available in the industry. The list includes both analog and digital accelerometers.

The change in ACREMO's lifetime between battery charge cycles when replacing the currently used accelerometer $LIS344ALH$ with a new accelerometer can be estimated as:

$$\Delta LIFE \approx \frac{I_{sys}}{I_{sys} - \eta_{sens} D_{sens} (I_{acc,active} + I^{new}_{acc,active})} \tag{4.1}$$

Table 4.1.: Different options for accelerometers and their properties

Properties	Currently used LIS344ALH	ACC1 KXSC7-2050	ACC2 LIS3DH	Units
Range	2	2	2	± g
Output	analog	analog	digital	
Non-linearity	0.5	5	-	%FS
Noise density	50	100	220	$\mu g/\sqrt{Hz}$
$I_{acc,active}$	680	230	11	μA
$\Delta LIFE$	-	↑ 24.6	↑ 41.6	%

where $I^{new}_{acc,active}$ is the active consumption of new sensor component.

It is evident that both choices ACC1 and ACC2 result in significant improvement in device lifetime. However, the concerning factor is that there is possible loss of signal quality as both of the new accelerometers carry high noise densities. Moreover, digital accelerometers at $\pm 2g$ range often come only with less ADC resolution (12-bits or less). LIS344ALH was chosen for the current system as it carries low spectral noise density. However, the accelerometer selection should be revisited considering the fact that the consumption of the accelerometer dictates the system consumption significantly.

The RMS value of the accelerometer noise is given by:

$$N_{RMS} = N \sqrt{f_c \times 1.57} \tag{4.2}$$

where N is the noise density and f_c is the cut-off frequency of the sensor output (Equation 2.2).

The factor *1.57* in Equation 4.2 corresponds to the roll-off characteristics of a single pole filter. Based on Equation 4.2, a simulation is performed to understand expected noise performance of the accelerometers under the choice list. Also two different filters, 1-pole and 3-pole low pass filters, are assumed. Figure 4.1 shows the estimated RMS noise for three different noise densities (see Table 4.1) as a function of cut-off frequency. Obviously 3-pole filters result in better performance. With a 44 Hz anti-aliasing filter, the RMS noise with current configuration of ACREMO is $415.6\mu g$ which is marked by a square symbol in Figure 4.1. It can be observed from Figure 4.1 that a noisy accelerometer may result in approximately the same noise performance as the current configuration by operating at a lower cut-off frequency. Hence, it is a valuable suggestion to replace the power hungry LIS344ALH with an ultra-low power accelerometer (such as ACC2) available in the industry.

If noise dominates the lower most bits of digitized acceleration signals, we can afford to sample the sensor with an ADC smaller than 16-bits. SAM3U has a 12-bit on-chip ADC peripheral that can be used in conjunction with peripheral direct memory access controller (PDC). The PDC can be programmed to sample the accelerometer periodically and buffer the sample onto a memory block. Upon collecting predefined amount of samples, PDC can wake up the core and start

processing. Thus, on-chip ADC sampling can significantly reduce CPU overhead during sampling activity. However, with lower number sampling bits, the amount of precision loss in accelerometric respiratory signal and its impact on respiration rate extraction are to be studied.

Though the currently used ADC consumes negligible power, it imposes sampling duty on CPU. Due to periodic wake up and sampling, CPU spends most of its sampling time in power mode transitions. Modern digital accelerometers come with integrated FIFO buffers and interrupt generating capability. By programming such a sensor to interrupt the CPU once a certain number of samples are collected, we can reduce the CPU's load due to sampling.

It is therefore advocated here that future studies on ACREMO should investigate replacing the current accelerometer with an ultra-low power and/or digital accelerometer. In such a modified system, along with inspecting the savings in the sensor power consumption, the possible compromise in the data quality will have to be studied.

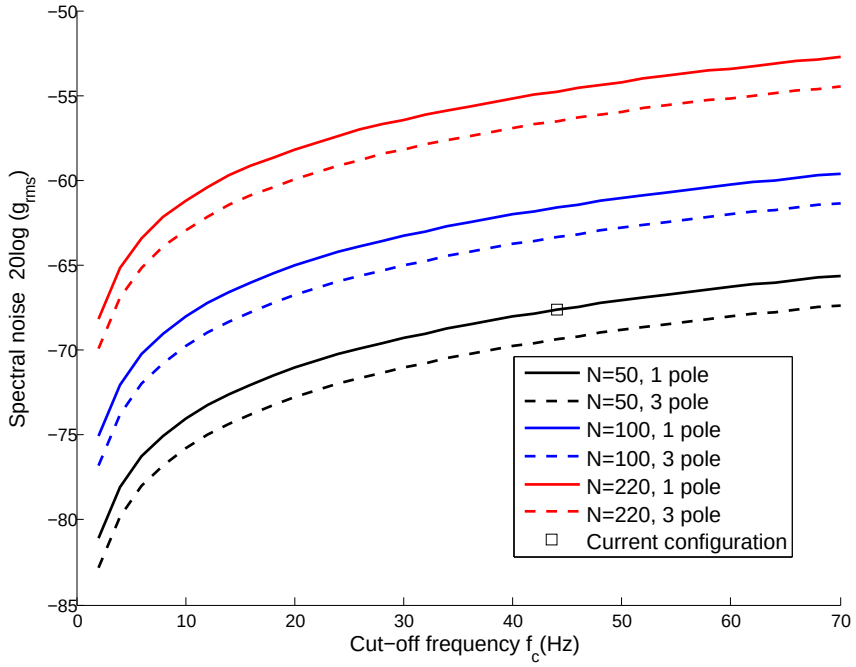


Figure 4.1.: Accelerometer spectral noise (RMS) versus cut-off frequencies for different accelerometers

4.1.2. OPAMP

OPAMPs are used for impedance matching between accelerometer output and ADC inputs. As mentioned in Section 3.5.1, the sensor subsystem is always *on* as

Table 4.2.: ACREMO system states identified for dynamic power management

id	state	CPU	SRAM	sensor	radio
$s0$	processing	active	active	on	off
$s1$	transmit	active	active	on	tx
$s2$	receive	active	active	on	rx
$s3$	observe	sleep	idle	on	off
$s4$	deep observe	wait	idle	on	off
$s5$	no-sense	wait	idle	off	off
$s6$	soft-off	backup	off	off	off
$s7$	mechanical-off	off	off	off	off

long as sampling is desired. This makes the current amplifiers to operate continuously. Hence selection of OPAMPs is an important factor in influencing the system power and hence the battery lifetime. For example, using Maxim's MAX954 amplifier ($I_{amp,active} = 7\mu A$) instead of OPA348 would result in an increased lifetime of 7% to that of current budget.

4.2. Dynamic power management

The concept of exploiting *idleness* of a system (as whole or its components) to reduce average power consumption is known as *dynamic power management* (DPM). For unfertilized systems, DPM is an effective technique which dynamically selects power states for each of the components and hence the system's state. DPM allows to shutdown the system as whole or its parts upon reaching idleness. The system can be woken up using asynchronous external inputs or pre-defined hardware time-outs.

Advanced configuration and power management interface (ACPI) is an operating system centric specification that defines handling of multiple components in a system each with different power modes [45]. ACPI is a joint initiative from HP, Intel, Microsoft, Phoenix and Toshiba which targets at managing power consumption of mobile, desktop, and server platforms. This work identifies and defines usable system modes that are ACPI compatible under the given application context (Table 4.2). Only states s0 and s4 were implemented in this work and are used as active and idle system modes respectively.

The system states ($s0$, $s1$, $s2$) represent global ACPI state G0 (working). The low power states ($s3$, $s4$, $s5$) are similar to global sleep states with global state G1 of ACPI. Similarly, $s6$ and $s7$ are counterparts of ACPI global states G2 and G3 respectively. The low power modes ($s3$, $s4$) can be used to exploit short idle period such as inter-sample duration given the requirements of peripheral clocking.

A short description of the identified system states for ACREMO are as follows:

- *processing* is the normal operational mode of the system where sensing and

computing subsystems are fully functional with communication subsystem being turned off.

- *transmit/receive* is the communicating state where radio module is active along with the CPU. Due to the nature of radio communication, this is the most power consuming system state.

- *observe* is a low power state with SAM3U using its *sleep* mode where power consumption is relatively low and is frequency dependent. In *sleep mode* core clock is turned off while the master clock and power supplies are kept ON allowing the peripherals to be clocked. The processor can be programmed to wake up on interrupts or events.

- *deep observe* state utilizes *wait* mode of SAM3U where both core and master clocks are stopped. As a result, the consumption is very low and the processor can be woken only by events or through pre-configured wake-up inputs. However, internal power supplies are left *on* so that the CPU start up is quick.

- *no-sense* is another low power system state where the sensor is turned-off. This mode is useful, in particular, when the application had decided to stop sampling the accelerometric data.

- *soft-off* is the least power consuming system state where all components except CPU are turned off. The CPU (SAM3U) is put in *backup* mode, lowest power mode, where the cortex core is powered off. Leaving this mode causes a core reset while IOs remain unaffected.

- *mechanical-off* states defines the condition where the system is completely turned off either through user input action or upon reaching low battery.

4.3. Dynamic frequency scaling

The ACREMO system is not driven by peak performance as the computations and result updates are required to be carried out cyclically with relatively long time period (in the order of seconds). Moreover, frequent act of sampling can be implemented in a fashion where CPU wakes up from a low power mode, sample the sensor, store results and transition back to low power mode again. Such a behavior involves mode transitions where most of CPU's active time is being lost (about 95%). Hence, it becomes important to study dynamic frequency scaling (DFS) in the ACREMO system.

Figure 4.2 shows variation in system level current consumption of ACREMO system against CPU clock frequency. The selected system configuration is *Use case 2* (see Section 3.5.3). It is important to note that the chosen idle mode of CPU is *wait* mode running at 4 MHz clock from fast RC oscillator.

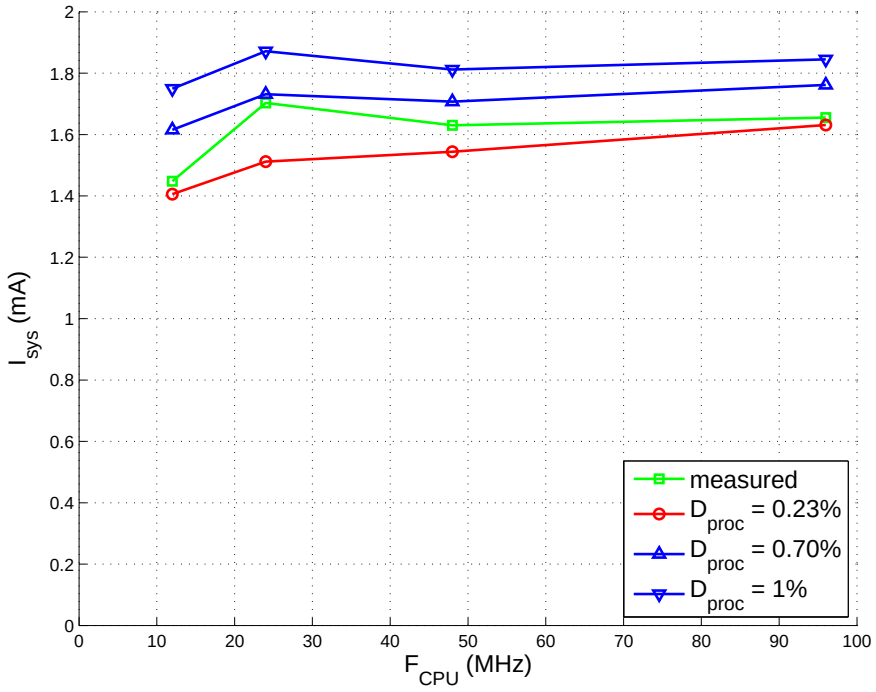
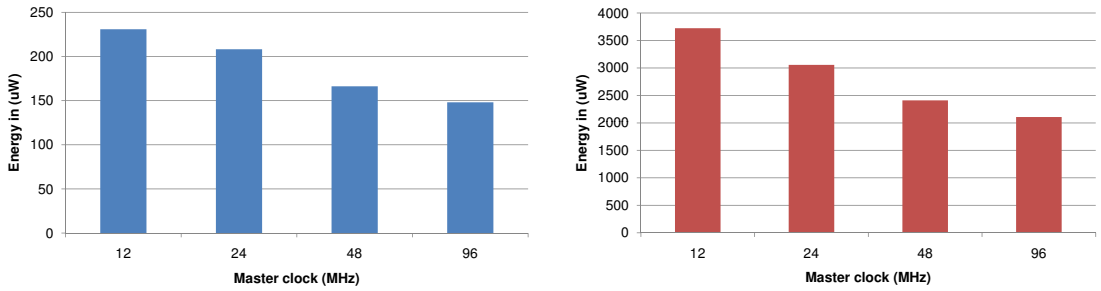


Figure 4.2.: System level current consumption of the ACREMO system against clock frequency frequency F_{CPU} (active mode) under *Use case 2* configuration. The clock frequency in wait mode is set to 4 MHz from fast RC oscillator. D_{proc} is the duty cycle of the processor to accomplish the signal processing and parameter extractions. The D_{proc} excludes the sampling load on the CPU.

It is clear that there is a non-linear dependance on clock frequency for system's current consumption which is corroborated with the correlation of estimated and measured current curves. The most power conserving choice is to use 12 MHz clock from the fast RC oscillator as the CPU main frequency which saves about 8.95% and 3.5% of system power with $D_{proc} = 0.23\%$ and $D_{proc} = 1\%$ respectively. This selection allows us to turn-off 12 MHz crystal oscillator and PLL which consume relatively higher currents. The system consumption at 24 MHz clock is higher than that of 48 MHz clock. This is due to the fact that clock switching delay during transition between active and wait modes is dependent on a fixed number of master clocks. When the master clock is low, this switching delay is higher and causes more energy during transition. We have only modeled PLL frequencies that are multiples of crystal frequency (12 MHz). If intermediate values are to be used, the pre-scaler feature will be incur an additional delay (max. 64 clock cycles) during clock switching.

The difference in ACREMO's current consumption between model estimation and actual measurement is due to multiple reasons such as:

- The measurement was carried out on system without radio functionalities;
- The power model assumed a linear battery model. Hence, there was an approximation in consumption of voltage regulators whose conversion efficiencies depend on instantaneous battery voltage;
- The power model was designed with few conservative global parameters;



(a) foreground task (b) background task

Figure 4.3.: Energy consumed to perform tasks vs. dynamic frequency scaling (active mode) under *Use case 2* configuration. The clock frequency in wait mode was set to 4 MHz from fast RC oscillator.

The algorithm is executed in two different cyclic tasks, namely foreground and background processes, with periodicity of 1 sec and 32 sec, respectively. The energy consumed by the ACREMO algorithm was measured directly from the hardware (using measurement setup shown in Figure 6.1) and is plotted in Figure 4.3. The energy consumption lowers with increasing master clock frequency. Though time taken to execute each of these tasks halves with doubling the frequency, the processor active current increases with frequency by a factor less than two. Since the CPU is underutilized, energy consumption of sampling activity dominates that of algorithm computation.

Based on the observations above, the wise options for SAM3U's active mode frequency are 12 MHz (Fast RC oscillator) and 48 MHz (Crystal oscillator driven PLL). Although choosing RC oscillator clock leads to low power consumption, its accuracy is less than the crystal oscillator clock. Moreover, the microcontroller must be operating on a PLL clock for Universal Serial Bus (USB) applications.

Alternatively, an even more dynamic frequency scaling manager can be devised which keeps running on internal RC oscillator (4/8/12 MHz) during sampling activities. The manager can switch CPU clock to a PLL clock when the foreground or background task is ready in order to compute the algorithm quicker and cheaper. Such a system design would be unconventional and difficult to maintain given the fact that changing clock frequency more frequently may also require frequent reprogramming or restating of peripherals (for example ADC). However, such a

manager can be an effective and tailored solution for increasing the ACREMO system's lifetime considerably.

4.4. Heterogeneous multi-processor system

The CPU utilization in the ACREMO system can be classified into a low utilization phase (LUP) and a high utilization phase (HUP). It was also previously mentioned that the low utilization part involves frequent power mode transitions that dominate the power consumed during HUP in SAM3U microcontroller. In addition to the objective of lowering idle power, there are additional CPU requirements for each of these phases. For example, LUP requires a power mode that has fast startup and shutdown. Such a mode should be consuming very low power during transitions as the number of expected transitions is high (i.e two times the sampling frequency). On the contrary, HUP implicates a high performance power mode whose transition time or transition energy is relatively unimportant. A high performance is desired to meet time requirements, if any, in result calculation and to reduce computational energy. Such a wide span of requirements could be difficult to achieve efficiently with single microcontroller. Based on these motivations, this work proposes a heterogenous multiprocessor (HMP) system as a solution to the problem of meeting both the power and the timing requirements.

When a system is characterized by different processing requirements, the hardware platform can be redesigned with different processors varying from 8-bit to 32-bit microcontrollers and/or digital signal processors (DSPs). A HMP system exploits the properties of different processors to accomplish each of the system's processing requirements efficiently to conserve system power and improves the battery lifetime. HMP systems are becoming popular in embedded sensing applications and have been studied in several works such as the PASTA project [46], the LEAP [47], the mPlatform [48] and the hybrid sensor module [49].

In this section, we propose and investigate a heterogeneous multi-processor system that uses a low-end microcontroller (TI's MSP430) for the low utilization phase of the algorithm and a high-end microcontroller (Atmel's SAM3U) for the high utilization periods. Figure 4.4 shows the CPU current consumption versus time in both current system and the proposed multiprocessor system.

4.4.1. Existing system

The current ACREMO system uses a SAM3U microcontroller to perform sampling, foreground, and background tasks. The processor runs at full speed of 96 MHz when in *active* mode and is put on *wait* mode whenever it is idle. Before going to *wait* mode, the processor clock is switched to 4 MHz from fast RC oscillator source which is also the frequency, by default, when the processor wakes up from *wait* mode. A timer is used to trigger an interrupt periodically that wakes up the core and prompts to go for sampling the accelerometer data. When enough

data samples are buffered (1 sec), a foreground task is executed that performs preprocessing and some basic feature calculations. A background process executes every time a frame capture is complete (32 sec). The execution times of foreground (t_{fg}) and background tasks (t_{bg}) were obtained from direct measurements.

It is important to note that both the model and measured hardware include buck regulator which drives the CPU core (1.8 V) from battery voltage (3.7 V). The CPU core at its full speed draws about 48 mA (at 1.8 V) from buck regulator while current drawn from battery is only about 26 mA.

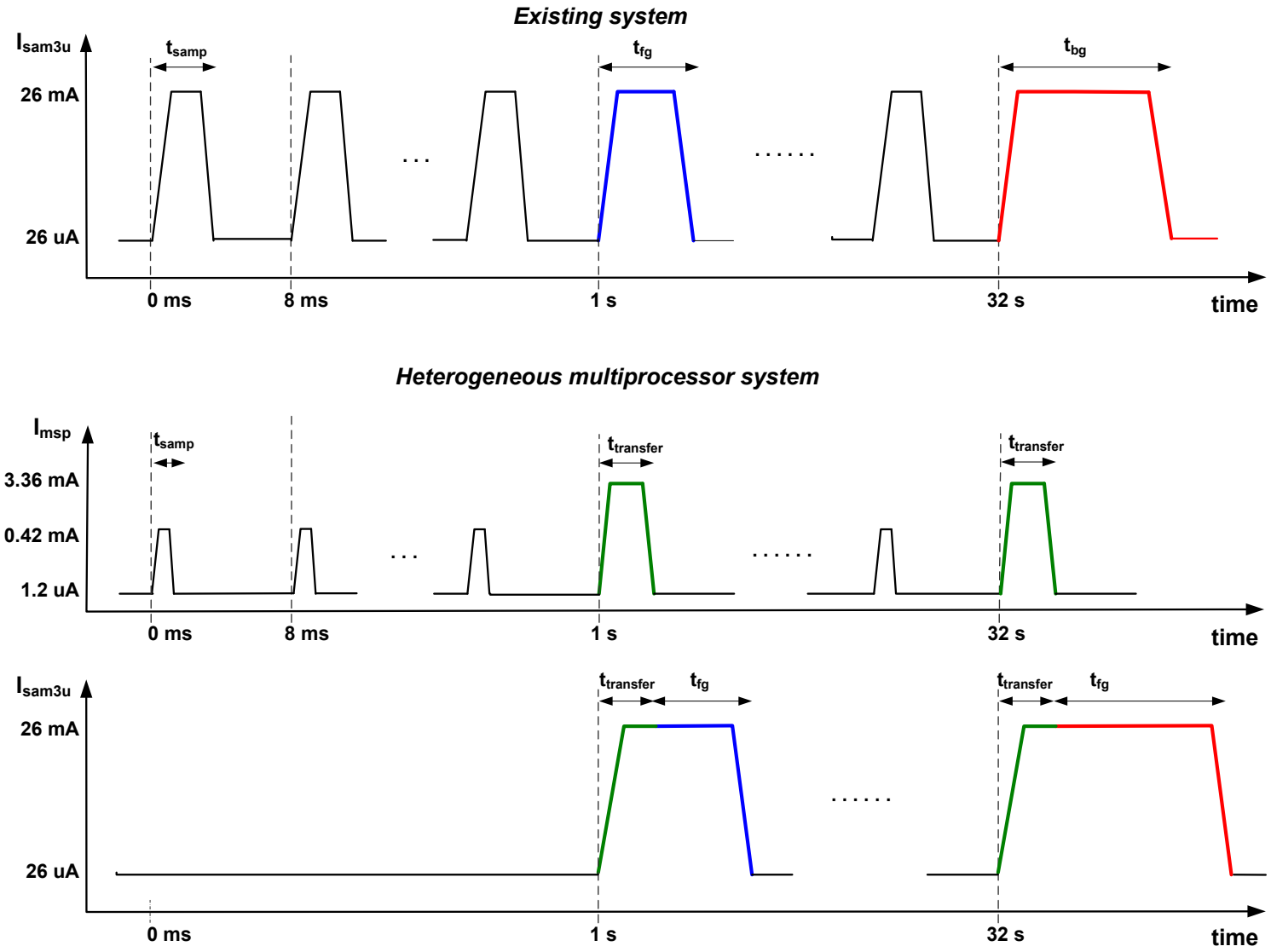


Figure 4.4.: Typical processing cycles and current consumptions in existing system and in proposed multiprocessor system

4.4.2. Proposed system

The proposed system uses a low-end microcontroller (MSP430) to sample and buffer data. By exploiting the nature of low-utilization system the high-end controller shall be put onto low power mode when there is no processing required. At the same time, the low-end controller is effectively active in sampling the accelerometer data. Once a predefined amount of samples are collected, the low-end controller can wake up the high-end controller though an interrupt and transfer all the buffered samples to initiate respiratory processing. Figure 4.4 show the modeled utilization and current consumption in the existing system and in proposed multiprocessor system.

MSP430 is an ultra low-power 16-bit RISC microcontroller platform which is widely used in medical, portable and wireless sensor applications. The controller is designed with active mode(s) and low power modes suitable for power cycling and can wake up from a low power mode in less than 6 μs.

MSP430FG47x is a configuration designed specifically for portable medical applications and comprises 16-bit timers, 16-bit sigma-delta ADC, two OPAMPs, two universal serial communication interface, and a liquid crystal display driver. The microcontroller has one active mode (AM) and five low power modes (LPM0-LPM5).

In the proposed system, MSP430 can be operated with clock frequency of 1 MHz during sampling task and 8 MHz during data transfer with SAM3U. The additional time overhead in the new system is the time required to transfer (t_{trans}) the collected samples from MSP430 to SAM3U which is estimated as:

$$b_{trans} = F_s \cdot N_{ax} \cdot N_{ADC} \tag{4.3}$$

$$t_{trans} = b_{trans}/D \tag{4.4}$$

where F_s is the sampling frequency, N_{ax} is the number of accelerometer axes, N_{ADC} is the number of ADC bits, b_{trans} is the number of bits to be transferred per second, and D is data rate of the inter-processor communication. With triaxial accelerometer sampling digitized with 16 bits at 125 Hz, the size of transfer data per second is 6,000 bits. This could mean a transfer time of 750 μs at a data rate of 8 Mbps.

The average currents consumed by different tasks in both the existing and the proposed system are listed in Table 4.3. It can be observed that the additional processor improves consumption of sampling task significantly. The average currents of foreground and background consumptions increase only by small fractions suggesting the overhead of sample transfer is negligible.

The proposed system will result in achieving the respiratory monitoring functionality with 57.8% less computational average power than the existing system. This means an improvement of about 16% (27 hours) in battery lifetime budget of 168 hours under the system configuration *use case 2*. In an optimized system, this improvement can achieve up to 33 hours more battery lifetime. Hence, it is suggested that the current system be redesigned to include a dedicated low-

Table 4.3.: CPU consumption estimations in multiprocessor system

Tasks	Existing system		Proposed system	
	t_{cyc} (ms)	$I_{avg}(\mu A)$	t_{cyc} (ms)	$I_{avg}(\mu A)$
sampling	0.658	302.4	0.037	19.4
foreground	1.7	47.7	2.450	68.7
background	17.3	15.2	18.050	15.8
idle	-	57.6	-	60.1
transfer (MSP)	-	-	-	1.1
TOTAL		422.9		165.1

end controller to sampling and other frequent activities while other expensive and heavy processing be performed at a high-end controller. Since most of the computations performed by SAM3U are related to signal processing, future studies can consider a digital signal processor (DSP) as a replacement of SAM3U so the execution is much faster and efficient. The multiprocessor system under proposal has disadvantages such as increase in cost and size of the device, need for synchronization, code maintenance, etc.

4.5. Local vs. remote processing

The ACREMO sensor node operates under frugal power budget with on-board battery as the only power source whereas the main station that receives data/results from the sensor node has nearly unlimited power source and computational resources. Hence, it could be worthwhile to investigate partitioning of the signal processing task between sensor node and base station.

The CC2420 transceiver's transmission energy is approximately 226 nJ per bit. The ARM's Cortex-M3 processor core executes 1.25 DMIPS/MHz while consuming 0.19mW/MHz. Consider an example of transmitting the raw accelerometer data collected over a second. The energy required to transmit 750 bytes of data over radio is about 1.4 mJ. With the same amount of energy, Cortex-M3 can execute nearly 9 million instructions. This indicates that the nodal processing would be advantageous over radio transmission of raw samples.

Four different transmission schemes are identified, namely, TX1-TX4 along with the scheme in *use case 2* (referred as TX0). Table 4.4 shows the parameter details and current estimations under the identified transmission schemes.

In TX1, raw samples collected from the accelerometer are transmitted over the radio channel every second. Though the CPU need not have to perform any algorithmic computation, the CPU has to run a radio stack that supports the wireless communication. The CPU current in this scheme is higher than the *use case 2* following the reason that the CPU is active during communication along with the radio. The battery lifetime is reduced by 56% which suggests that the radio

Table 4.4.: Estimation of average system current for various transmission schemes

Scheme		B_{tx} (bytes)	N_{tx} (cycles/sec)	I_{radio} (μA)	I_{cpu} (μA)	I_{sys} (μA)	$LIFE$ (hours)	$\Delta LIFE$ %
TX0 - use case 2		40	1/900	0.5	624.4	1631	190	0.00
TX1 - raw		750	1	446.8	1226.9	2680	116	-64.30
TX2 - decimated		750/8	1	74.4	649.4	1730	179	-6.06
TX3 - features	wc	40	1	43.9	660.0	1710	181	-4.84
	ac	40	1/5	9.2	631.5	1647	188	-0.97
	bc	40	1/10	4.8	628.0	1639	189	-0.48
TX4 - results	wc	4	1	23.5	628.0	1658	187	-1.63
	ac	4	1/5	5.1	625.1	1636	189	-0.32
	bc	4	1/10	2.8	624.8	1634	190	-0.16

transmission is more expensive than computation at sensor node in the ACREMO system.

As briefed in Section 2.3.1, the ACREMO algorithm samples the accelerometer at 125 Hz and then downsamples it to a suitable rate before starting the main algorithm. Such sampling may well be oversampling for the ACREMO processing but it is necessary for the pulse signal processing which is performed on the same sampled (accelerometric) data. In this work, the sampling frequency F_s is set to 125 Hz and a downsampling factor is set to 8.

In scheme TX2, decimated samples instead of raw samples are transmitted over radio. This reduces the radio transmission time by factor of 8 (decimation factor) which shows an improvement in radio consumption compared to TX0. However, the consumption of CPU is almost equal to TX1 where complete processing is carried out. As a result, this scheme gains no advantages over power saving.

Transmission scheme TX3 processes the decimated signal, computes a set of features and transmits them over radio. Periodicity of transmission (N_{tx}) can be selected arbitrarily or based on typical breathing frequencies. We considered three different cyclic rates, namely bc - best case, ac - average case and wc - worst case, based on breathing rates of 6, 12 and 60 breaths per minute (bpm) respectively. The translation of decimated signal into feature space can be considered as a type of data compression technique. Such a compression with our respiratory algorithm can achieve a compression factor from 2.3 (features per 1s) to 70 (features per 30sec). This scheme also results in reduced lifetime ($\Delta LIFE$) in comparison to existing scheme TX0.

TX4 scheme is similar to TX3 except that only algorithm results are being sent over radio. When the transmission periodicity is high, the estimated consumption in this scheme is close to that of TX0. The consumption slowly increases as N_{tx} is increased since frequent radio transmissions are expensive.

Figure 4.5 shows the estimated current consumptions under each of the transmission schemes. It is evident that the local computation at a sensor node is more

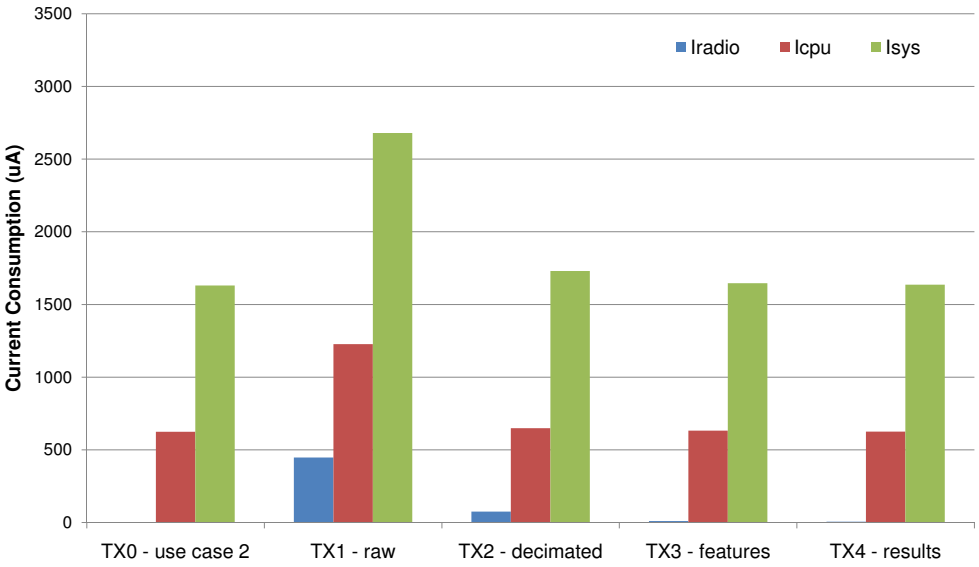


Figure 4.5.: Average current estimations for different radio transmission schemes. The power consumption of the current system is dominated by sampling instead of computation. The data transmission scheme is expected to have gain when the computational load gets more significant.

power conservative than the transmission of raw sample data. However, an optimal operating point can be identified in the system so that a part of the computation is done at the node while the remaining is carried out at the base station. It is also important to note that the system consumption depends on the main processor which is a high-end controller (SAM3U) in our system. It might be an interesting study for future work to investigate the consumption under similar schemes in a modified system where a low-end controller (for example, MSP430) performs for signal processing and/or transmission.

4.6. Summary

The architectural level of power optimization, in most of the cases, is the foremost approach to significantly reduce a system's average power consumption. The selection of the hardware components such as accelerometer and OPAMP play a vital role in improving the ACREMO device's lifetime. For example, choosing a low power digital accelerometer increase device lifetime by about 48%. By dynamically selecting the system and component power states significant amount of power saving can be achieved in systems where idle duration dominates the active duration. A set of ACPI compatible system states for ACREMO are identified and are partly implemented in this work.

Due to large startup times of SAM3U microcontroller during transition from a low power mode into a PLL clocked active mode, the energy spent by the CPU

for sampling increases with the system's clock frequency. However, the energy consumption for executing the ACREMO algorithm decreases with increasing clock frequency. Hence, the optimal choice of clock frequency for lowering the CPU consumption depends on finding the balance between sampling and computational power. Another open choice would be to operate the SAM3U at two different frequencies, one for sampling task and another for processing tasks.

The ACREMO system architecture can be redefined as a multiprocessor architecture where the computational tasks can be partitioned among a low-end processor and high-end processor depending on the task's complexity and timing requirements. A simple heterogeneous multiprocessor system with MSP430 and SAM3U microcontrollers is proposed. It was also shown that such an architecture modification can increase the system's battery lifetime by more than 16% in comparison to the current architecture with single processor (SAM3U).

It is also shown that processing at the sensor node is more efficient for the current architecture of the ACREMO system than transmitting accelerometric data (raw or compressed) over the radio and processing at a central station.

5. Power Optimization at Algorithmic Level

This chapter introduces some of the algorithmic choices for optimizing the power consumption of the ACREMO system. By studying these choices, we evaluate their applicability to the system and gain in battery lifetime. The first section will explain the existing respiratory monitoring algorithm and how it is optimized in this work. The second section will study the gain in power saving with reduction of sampling frequency of the accelerometer. In the last section, a context-aware power saving strategy is proposed that can optionally power down the entire system during specific system conditions.

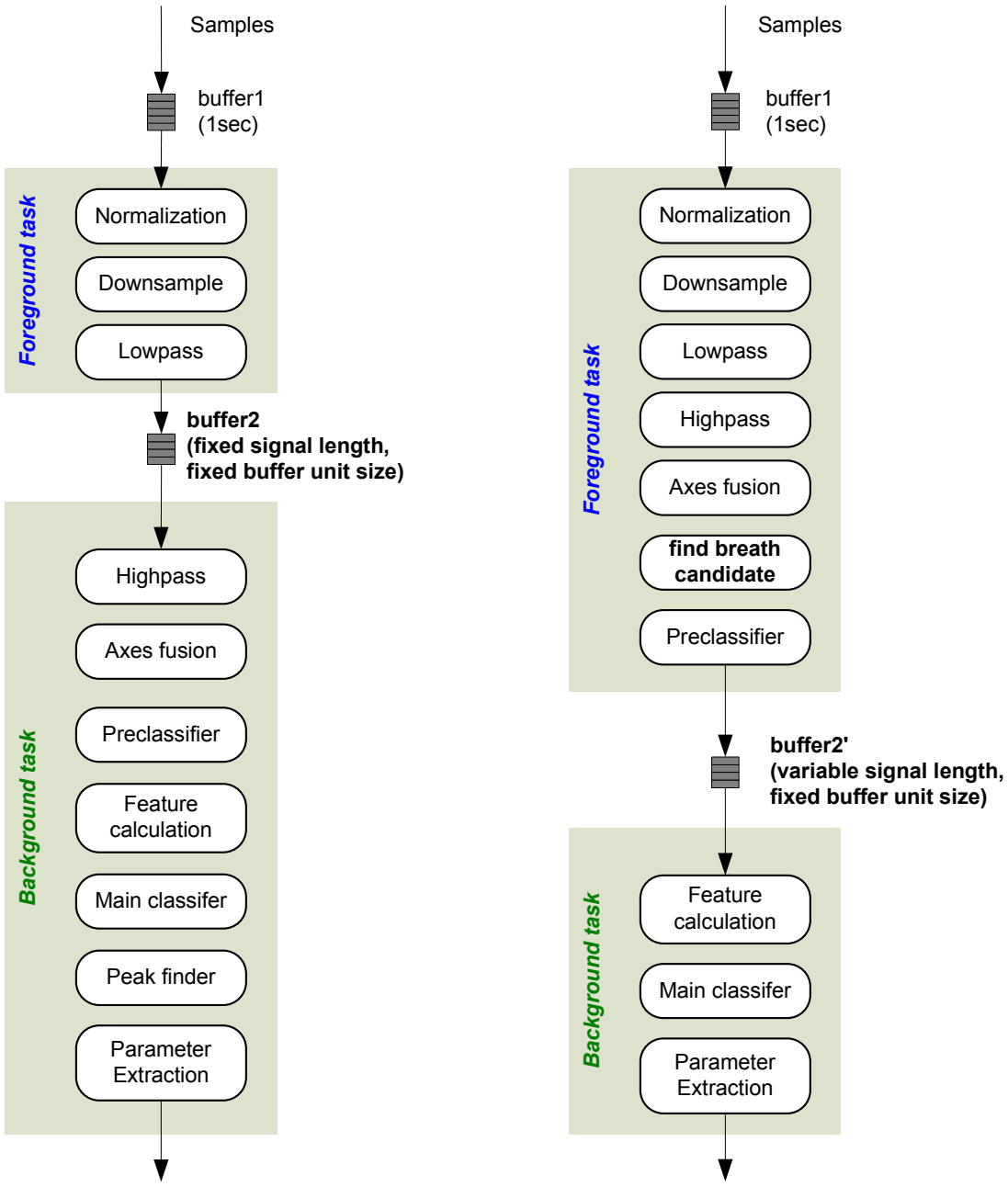
5.1. Respiratory processing algorithm

5.1.1. Fixed window classification

The reference algorithm that generates respiratory parameters from accelerometric measurements is called *fixed window classification algorithm (FWC)*. The signal processing chain is divided into two main tasks, namely foreground and background tasks as shown in Figure 5.1a. The accelerometric samples are buffered in *buffer1* that can hold one second long triaxial signal. When *buffer1* is full, the foreground task is triggered where signal preprocessing is performed and few of the critical features are computed. The foreground process buffers the preprocessed signals onto *buffer2* that can hold 32 sec long (down-sampled) triaxial signal. When buffer2 is full, the background process is invoked where the feature calculations, signal classification, and respiratory parameter for the corresponding 32 sec signal frame were computed. Since the size of a signal segment undergoing feature calculation and classification is fixed (32 sec), the algorithm is called fixed window classifier.

During the preprocessing step, the samples are normalized to appropriate signed number system and accelerometer calibrations are performed. Then, the signal is downsampled by a factor of 8 that makes later processing simpler. A low pass filter with cut-off frequency of 2 Hz is applied on the downsampled signal before buffering onto *buffer2*.

The background task retrieves the signal from *buffer2* and applies a high pass filter with cut-off of 0.1 Hz. It is necessary to remove the DC component before the axes fusion is performed. Then, a feature vector comprising of temporal, spectral and spatial domain features is computed from the 32 sec signal segment



(a) FWC

(b) SBC

Figure 5.1.: ACREMO respiratory monitoring algorithm dataflow (a) Fixed window classification (b) Single breath classification

(one window). Based on the computed feature vector, a signal classifier labels the window into one of the two classes, namely *breathing* and *non-breathing*. For this purpose, a decision tree classifier was generated through training with ground truth signals and manually annotated labels. When a window is labeled as *breathing*, a peak finding algorithm is run on the signal that detects peaks in the signal which are used for respiratory rate calculations.

Though the algorithm looked solid when implemented in MATLAB (The Mathworks Inc.) software, there are some disadvantages of this algorithm when translated to real-time software to be run on a microcontroller. Some of them are listed below:

- use of large buffers and buffer pools. Due to this, FWC demands more on-chip memory which challenges the implementation of the algorithm on low-end microcontrollers which are often low power and low cost devices;
- use of spectral features for classification. Since Fast Fourier Transform (FFT) is involved, computation of spectrum of the time domain signals and calculations of spectral features are expensive in real time, especially with a general purpose microcontroller;
- due to (relatively) longer fixed frame based classification, the algorithm becomes unsuitable for realtime observations. For example, once the algorithm is started, respiratory parameters are available only after 32sec;
- operating on large time frames, we tend to lose the trend tracking or breathing patterns in the signal.

In order to address these issues, FWC algorithm has to be modified to adapt to smaller frame size and to renounce spectral features without losing classifier's decision quality. In addition, we will investigate the power savings achievable by such a modified algorithm.

5.1.2. Single breath classification

The modified respiratory monitoring algorithm under study is called *single breath classification (SBC)*. As the name suggests, the basic unit of classification is a single breath. The duration of a valid breath typically varies from 1 sec (1 Hz or 60 bpm) to 10 sec (0.1 Hz or 6 bpm). Figure 5.1b shows the processing chain of SBC and its basic differences from FWC are highlighted in bold text.

Some of the processing steps from the background task of FWC are moved into SBC's foreground task causing the task slightly heavier. A valley finding algorithm was developed in this work to locate start or end of a potential breath. These modifications allow us to identify the potential breath, called *breath-candidate*, in the foreground task itself. Then basic features and properties of a breath-candidate are computed and buffered onto *buffer2'* which will be transferred to the background task. Hence, the size of the new buffer is very small compared to

FWC as buffer2' contains only properties and features but not the signal anymore. This can save a significant amount of memory during static allocation of buffers.

For example, at 125 Hz sampling frequency, a five second long tri-axial breathing signal segment means 3750 bytes (5*125*2*3) of raw data, 469 bytes of data downsampled by factor of 8, and 156 bytes of fused data. The current structure of the FWC algorithm stores both dowsampled raw and fused data to be transferred for the background task. The storage of data is necessary due to the fact the spectrum calculations in the background task require a definite amount of data collection (for example 512 samples). However, the current SBC algorithm needs only 40 bytes (size of a feature vector). Thus, SBC achieves data compression with factor ranging from 2 to 23 (depending on candidate length) and significantly reduces the amount of data being transferred from the foreground to the background task.

Implementation

The basic single breath classification algorithm was available in MATLAB® where the main classifier (decision tree) was generated using WEKA [50]. For this work, the algorithm was translated to fixed point C language using Microsoft Visual Studio 6.0 under Windows platform. Finally, the algorithm was merged with the commercial RTOS μC/OS-II (Micriμm Inc.) using IAR Embedded Workbench for ARM v6.10.

Valley finding

Detecting a breath-candidate basically starts with finding local valleys in the time-domain signal segments. A simple C function was developed which locates local minima below the mean of signal amplitude. Hence, a running mean of signal amplitude is to be computed to supplement valley detection. Exponential moving average (EMA) filter or forgetting filter is widely being used in signal processing to compute instantaneous mean values as the samples keep coming.

$$EMA = \alpha \times \frac{\sum_i^n s_i}{n} + (1 - \alpha) \times EMA \tag{5.1}$$

where $\alpha = \frac{2}{N+1}$ is the forgetting factor, n is the number of samples in the signal segment and N is the number of segments in the window of interest. The algorithm is currently configured to compute EMA for every 1 sec ($n = F_s/8$) segment with filter width of 32 segments ($N = 32$). The advantage of EMA is that it enables precalculations at smaller signal segments (1 sec) which distributes computational load of the CPU over the length of breath-candidate.

A simple valley detector finds local minima in every 1 sec signal segment below EMA of the segment. This is to avoid false valleys that might have been caused by motion artifacts. In order not to miss valleys at the edge of signal segments, the last two samples (at least) from the end of a segment are carried forward to the next segment.

On finding a valley inside a 1 sec signal segment, a potential breath-candidate is detected which will be packed into a candidate data structure and buffered on to buffer2'. The data structure contains the precalculated features and basic information on the breath-candidate. The background task picks up the candidate from the buffer, computes other features, and classifies the candidate as either breath or non-breath.

Feature vector

The SBC algorithm aims at computational load reduction without compromising on quality of decisions made by the algorithm. It was measured that the computation of spectrum and spectral features in the FWC algorithm costed about 91.1% of feature calculation energy which is 5.4% of total energy consumed by the FWC algorithm. It is therefore decided to drop spectral features in the SBC algorithm and thereby avoid spectrum computation. The set of features used in both FWC and SBC algorithms are listed in Appendix B.

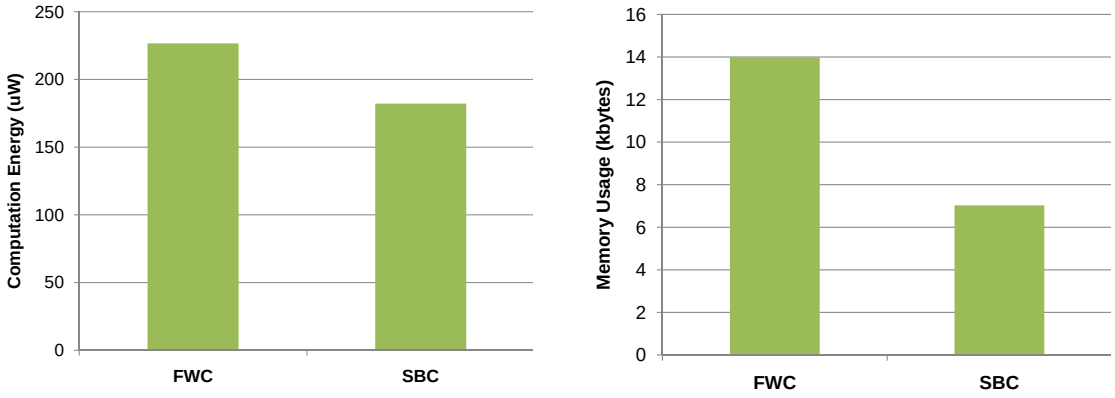
The *peak finder* and *parameter extraction* activities of the background task in the FWC algorithm (see Figure 5.1a) contribute about 9.3% of the total energy consumption of the algorithm. SBC algorithm incorporates a simple peak finder code within valley detection function. Hence, the peaks are detected along with detection of breath-candidates at the foreground task itself. Moreover, respiratory parameters such as respiratory rate (RR) can directly be computed from the SBC output labels and candidate parameters. For example, consider a signal capture of 20 sec where SBC algorithm has assigned four consecutive breath labels (and no non-breath labels anywhere in the capture). Then, RR can be estimated as $4/20 \times 60 = 12\ bpm$. Therefore, SBC not only saves energy in feature computation but also on parameter extraction.

Figure 5.2a shows the energy consumed by the two algorithms running on the ACREMO prototype hardware with the SAM3U microcontroller running at 96 MHz. The SBC algorithm can save up to **25%** of the SAM3U energy consumption in the ACREMO system.

As an added value, SBC brings down the memory consumption of the algorithm by 50% (14 kb to 7 kb) in comparison to that of FWC by reducing both static and dynamic allocations (Figure 5.2b). The major actions through which the memory savings came from are:

- replacing buffer2 in FWC that holds 32 sec triaxial data by buffer2' in SBC that holds feature vector;
- removing buffers used for spectral calculations

Further savings in memory usage can be achieved if the classification function is also moved into the foreground task. This will lead to the removal of the background task completely to save significant amount of memory (about 1.5 Kbytes).



(a) (b)

Figure 5.2.: A comparison of FWC and SBC algorithms. The measurement is carried out on the ACREMO prototype hardware where SAM3u microcontroller was running at maximum frequency of 96 MHz

Such reduction in memory footprint of the ACREMO software algorithms suggests that the algorithm can be comfortably implemented on a smaller on-chip memory provided by low-profile microcontrollers. It was previously shown in this work that the algorithm imposes a very low duty on the CPU which is SAM3U in the current system. Based on these observations, it is suggested to conduct a study of the modified ACREMO hardware architecture with a low-profile microcontroller as the main CPU and investigate savings in power consumption of the new system.

5.2. Power vs. sampling frequency

Currently used ACREMO software architecture is designed with sampling rate of 125 SPS (F_s). The respiratory processing chain downsamples the accelerometric samples by a factor of 8. Hence, sampling frequency of signal on which the axes fusion, classification, peak and valley findings are carried is only 15.625 Hz ($F_s/8$). The high sampling frequency, 125 Hz, was chosen in order to account for other physiological signals, for example, pulse signal, present in the sampled signal along with respiratory signal. Also, the process of oversampling, filtering and then downsampling again can help us to effectively reduce aliasing without increasing complexity of the filter. However, from power consumption point of view, the ACREMO system spends more power in sampling the accelerometer data rather than processing it. This is because SAM3U microcontroller is expensive when operated in a fashion of waking up from idle mode, sample data and transition back to idle mode. Most of the sampling power spent in transitions between idle and active modes is attributed to wait for PLL lock, clock switching, processor mode

change etc.

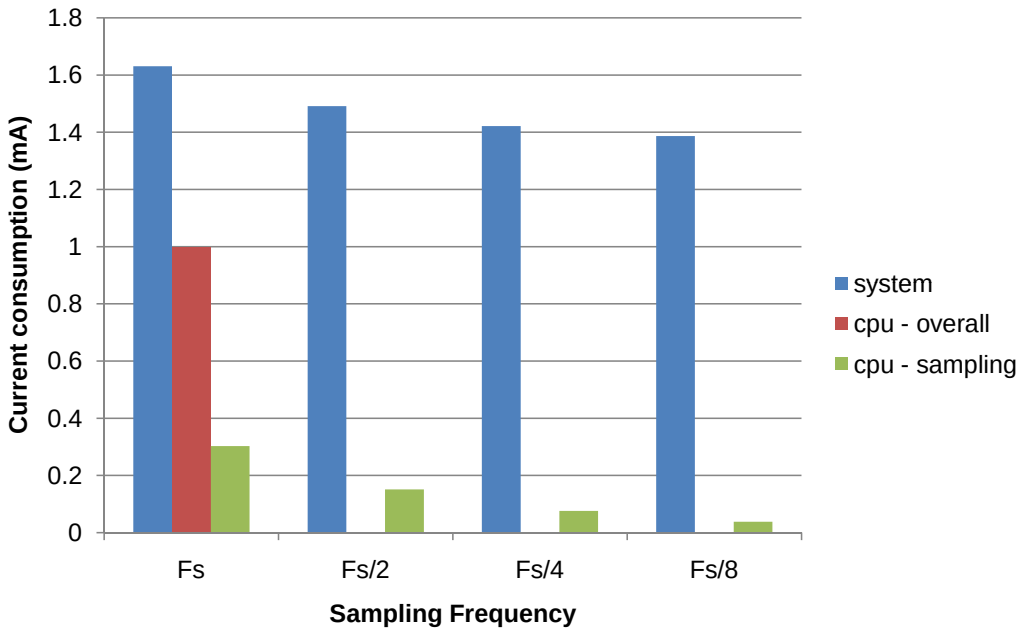


Figure 5.3.: Sampling frequency vs Current consumptions

The sampling frequency (F_s) was identified as one of the model parameter in power model described in Chapter 3. Figure 5.3 shows the system power consumption as F_s is varied. The CPU consumption owing to sampling reduces linearly when sampling frequency is decreased. However, its impact on average CPU consumption (including processing and idle consumptions) and the system level current consumption is less. For example, an improvement 15% in battery lifetime can be achieved by choosing sampling frequency of 15.625 Hz ($F_s/8$) under *use case 2* system configuration.

5.3. User activity-driven power saving

One of the disadvantages in accelerometric physiological parameter monitoring is that the sensor is susceptible to body movements and vibrations that can easily override the motion induced by the physiological activity of interest. There was wide presence of artifacts in the measurements especially in active individuals. Only 56.8% of the training data set (manually annotated) represented breathing information while the remaining measurements were all artifacts due to various activities such as body movements, walking, speaking, posture change etc. It is important to note that both FWC and SBC algorithms do not attempt at retrieving information from artifact induced signals. The algorithms detect and discard the non-breathing signal segments through classification. The physiological parameter calculations are performed only on the breathing segments of the signal.

Most of the strong motion artifacts can easily be identified from other artifacts or breathing signal by comparing their amplitude variance. When such motion artifacts are detected early, the sensor subsystem can be put onto a low power mode or even turned off for a definite period of time until the expected end of artifact section. Such a sleep duration can be pre-determined from training data set or learnt online to adapt to different users and time of day during measurement. In this section, we propose and study a simple sleep strategy for the ACREMO system in order to save power during high user-activity.

In this thesis, a simple sleep strategy is developed as illustrated in Figure 5.4. Using the knowledge that such a strong motion artifact takes some time to damp which is in the order of seconds, the scheme acknowledges an artifact only when a particular number ($nart\_th$) of consecutive one second segments showed their variances above the threshold value var_th. Once these conditions are met, the system stops the sampling activity, starts the wake-up timer for a predefined sleep duration and then transitions into the low power mode. Thus, the strategy acts as a pre-classifier based on the two features that detects motion artifacts.

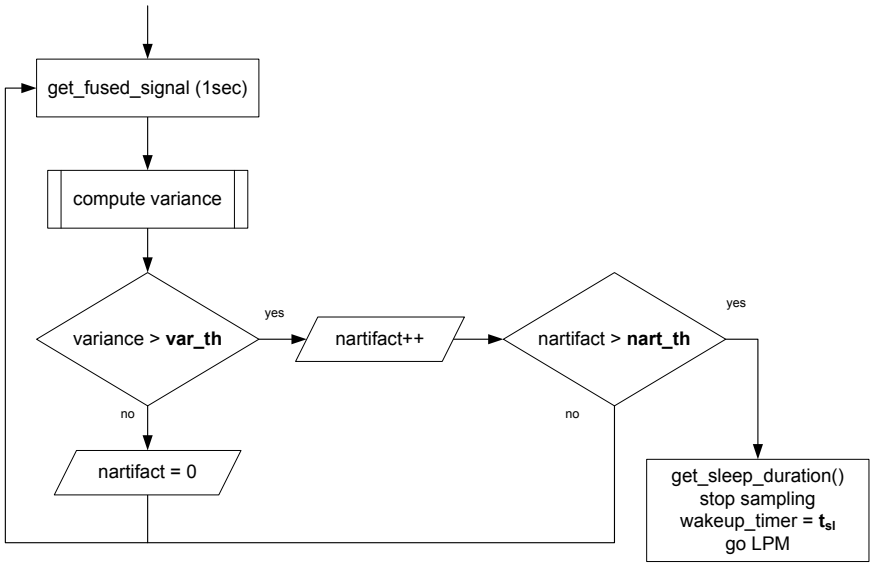
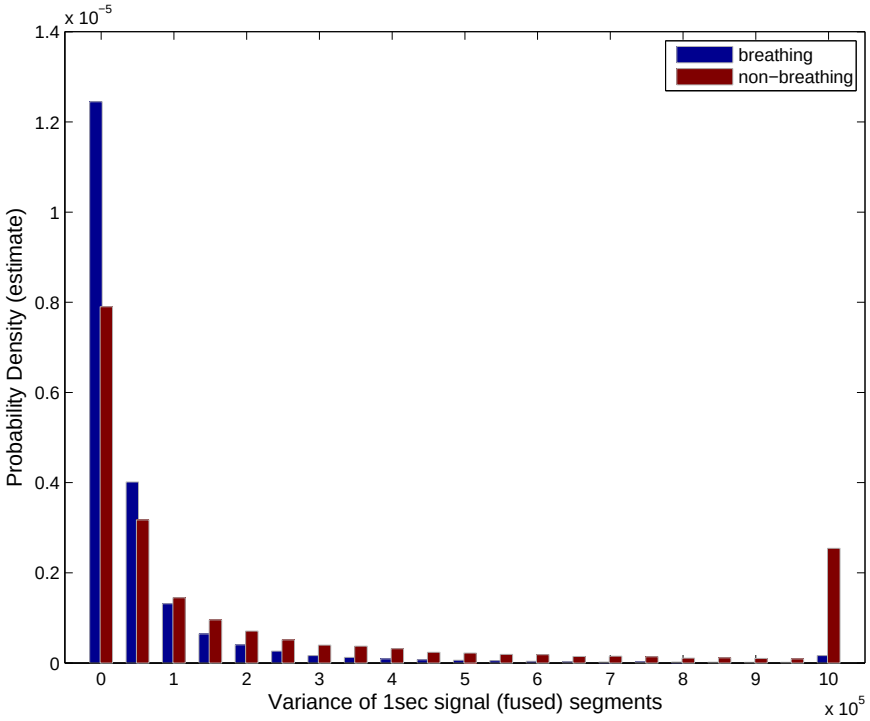


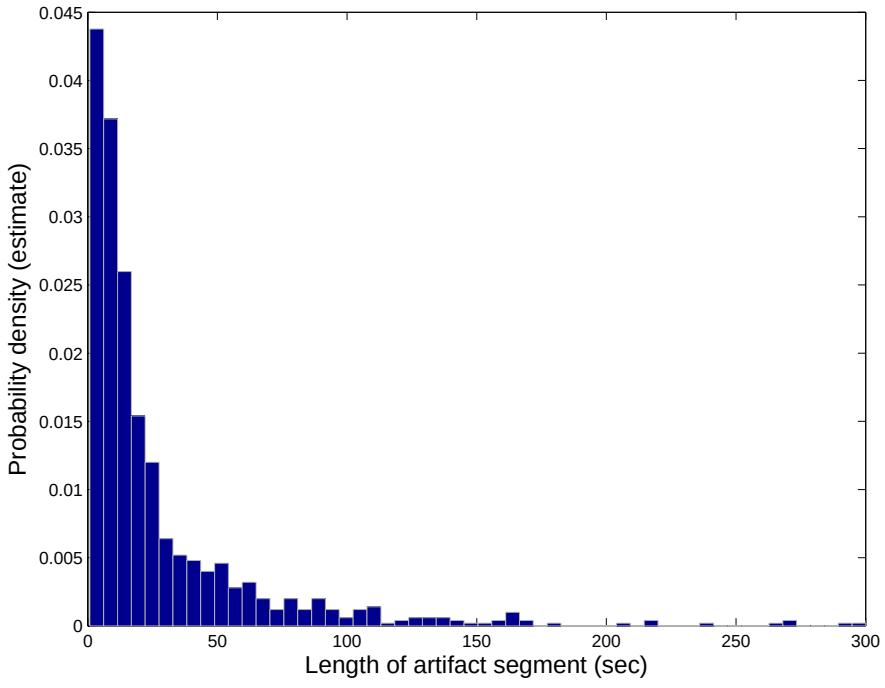
Figure 5.4.: User activity-driven power saving strategy

An estimation of probability density function (pdf) of amplitude variance for one second signal segments (fused) available in the training data set is shown in Figure 5.5a. This estimation is used to choose the threshold value for signal amplitude variance, var_th. Figure 5.5b shows the density estimate of the artifact durations. It is observed that the mean, median and mode of the artifact segments found in the training data set are 31.8 s, 14.3 s and 3.2 s respectively. Since the mode and median are less than 15 sec, the threshold for artifact length ($nart\_th$) should be few seconds to achieve maximum power saving.

Simulations were carried out by varying the minimum artifact length threshold



(a) Amplitude variance



(b) Artifact duration

Figure 5.5.: Artifact detection - density estimations

nart_th and duration of sleep t_{sl} forced each time the strong artifact section is detected. From the training data set, an estimate of average power saving that can be achieved was found and is shown in Figure 5.6a. More than achieving power saving by sleep, the amount of data loss that might occur due to the predefined sleep duration is of more interest to justify this scheme. Figure 5.6b shows how much of the accelerometric breathing data in the entire data set was lost due the sleep enforced. Figure 5.6c shows how much amount of sleep duration is breathing information. One of the objectives of the simulation is to find an operating point $\mathbf{O}(nart\_th, t_{sl})$ that would result in an acceptable data loss while yielding maximal power saving. Some of the candidates for such an operating point are learnt from running the simulation on the entire training data set and are as following:

- **O1(nart_th = 5s, t_{sl} = 6s)** resulted in saving 8% of system power on the training data set. However, the scheme lost of 1.7% of the entire breathing data available in the training data set.
- **O2(nart_th = 4s, t_{sl} = 20s)** saved 20.9% of system power with loss of 5.3% of the entire breathing data.

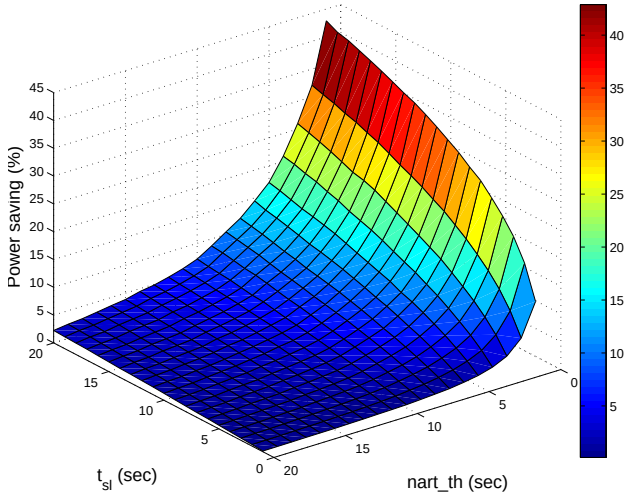
Figure 5.6 shows the power saving achieved and loss of breathing information during sleep, computed on the training data set, against the variable operating point $\mathbf{O}(\text{nart\_th}, t_{sl})$. The amount of power saving drops along with the data loss as the artifact length threshold, *nart_th*, increases. This could be due to the fact that the strong motion artifacts persist longer. Also, when *nart_th* is sufficiently long, the sleep scheme can be more reliable.

On other other hand, increasing the sleep duration, t_{sl}, results in saving more power. However, when the sleep duration is higher than the artifact duration threshold, more breathing information are being lost.

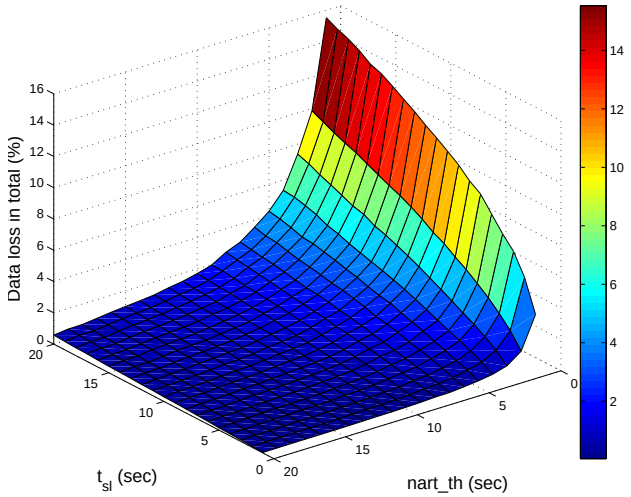
The simple scheme discussed above gives a good starting point in understanding the system's capability to power cycle itself on finding high user activity. However, the scheme can be improved further to achieve similar or more power saving with reduced loss of breathing data. Instead of sleeping for fixed duration each time, the system can adopt a strategy to increase sleep duration progressively. Also, it is worthwhile to investigate the possibilities to predict the required sleep duration without losing much data. Dynamic learning based on the past events and statistical prediction methods are some of the candidates for future improvement of the proposed strategy.

5.4. Summary

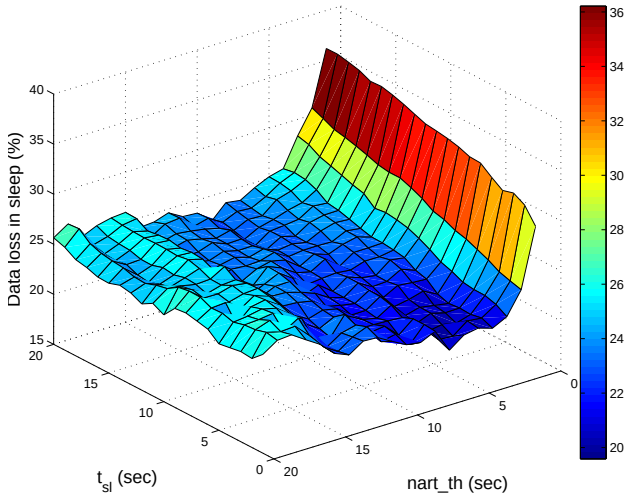
The implementation of the single breath classifier algorithm was detailed in this chapter. SBC is seen as a potential replacement for the existing ACREMO algorithm (FWC). The new algorithm can reduce the energy consumed for respiration processing by 25%. Such an improvement is made through structural change of



(a)



(b)



(c)

Figure 5.6.: User activity-driven power saving against loss of breathing data

the algorithm implementation and by avoiding expensive features during feature extraction before the main signal classifier.

The current ACREMO system performs oversampling with respect to respiration processing. A high sampling rate was chosen to cater to the pulse signal processing and to simplify the filter design. The reduction of sampling frequency, for example by factor of 8, can improve the device lifetime by 15%. The choice of reduction factor in the actual system is straight forward if the system contains only respiratory processing. Otherwise, the factor will have to be chosen depending on the pulse processing chain.

An application specific solution, namely user *activity-driven power saving*, is proposed to dynamically conserve power with little or no data loss. The strategy identifies the periods of motion artifact early enough to force the device to sleep until the artifact ends. The threshold parameters used in this strategy can be chosen based on the system constraints of power and acceptable data loss.

6. Discussions

This chapter summarizes the observations made during the power modeling and optimization processes and discusses their implications. The first section describes the measurement setup used in this work for target current measurements. In the second section, the power model developed during this work and its outcomes are discussed. Finally, the optimization techniques which are studied as a part of this thesis and their results are discoursed.

6.1. Measurement setup

The ACREMO hardware was available after the power model was developed. Hence, in order to validate the model predictions, current measurements were carried out on the ACREMO prototype system. The ACREMO hardware was modified to support the current measurements (Figure 6.1). A high resolution shunt resistor ($R_{shunt} = 5\ \Omega$) was placed in series with the battery to measure the overall current drawn from the battery as a voltage drop (V_{shunt}). At room temperature, an error of 1% in the measured current may occur owing to tolerance of the shunt resistor. The battery and shunt voltages were sampled using two different channels of the National Instrument's digital acquisition device (NI USB-6353 X) that comes with an ADC resolution of 16 bits. The sampling frequency was set to 15 kHz for most of the measurements while some of the measurements were done at 25 kHz (for example, CPU mode transitions). A LabVIEW application was used to control the acquisition device and log the current measurements into a PC through USB port. Various MATLAB scripts were developed during this work to extract and analyze the log data.

6.2. Power consumption model

The power consumption model detailed in chapter 3 has shown that the key hardware components contributing to the system's average power are the accelerometer (LIS344ALH) and the microcontroller (SAM3U) which consume about 37% and 46% of the total power respectively. LIS344ALH is one of the low noise accelerometers available, but, has a relatively high supply current (680 μA). The accelerometer can not be cycled to the low-power mode between the samples as long as the Nyquist criterion is to be satisfied. Hence, an easy way to reduce the consumption of the sensing subsystem is to choose a different accelerometer that consumes less current than the LIS344ALH.

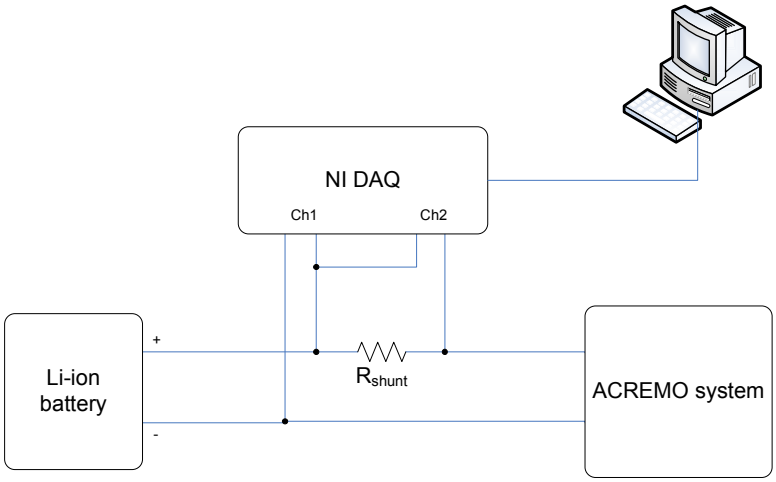


Figure 6.1.: Measurement setup for target current measurements

Similarly, the SAM3U controller in the ACREMO system can be replaced with a low-end microcontroller, for example TI's MSP430. Such a CPU replacement can significantly improve the system power as the SAM3U is an overfitting for the existing ACREMO algorithm (D_{proc}=0.23%). However, the performance of such a low-end processor while executing DSP routines and the processor's capability to cater to the future software extensions are unknown at this point of the work. Hence, future studies can consider the replacement of the CPU as a candidate for power reduction in the ACREMO system.

The model has assumed two different use cases and predict the battery lifetime in each of those cases. The model estimated the lifetime of the existing ACREMO system as 70 hours for *use case 1* and 190 hours for *use case 2*. One of the important differences between these use cases is the CPU's low-power mode. The *use case 2* employs the *wait* mode of the SAM3U for idling whereas the *sleep* mode is used in the *use case 1*. It is seen that that the use of *wait* mode is very efficient. However, it is important to remember that in this mode, the SAM3U stops the clocks of the core, peripherals and memory.

Other model parameters that impact the power consumption of the computing subsystem are sampling frequency F_s, clock frequency of the CPU in active mode F_{CPU} and duty cycle of the CPU for processing D_{proc}. Figure 6.2 shows the estimated device lifetime between successive battery charge cycles against the CPU's processing duty. D_{proc} excludes the duty imposed by the sampling task. The solid line in the figure shows the lifetime estimation curve for the Lithium-ion battery with rated capacity of 310 mAh while the dotted line represents the estimation for a 150 mAh battery. Increasing the existing algorithm's complexity by large proportions will significantly reduce the device lifetime.

The energy consumed by the SAM3U to execute the ACREMO algorithm scales

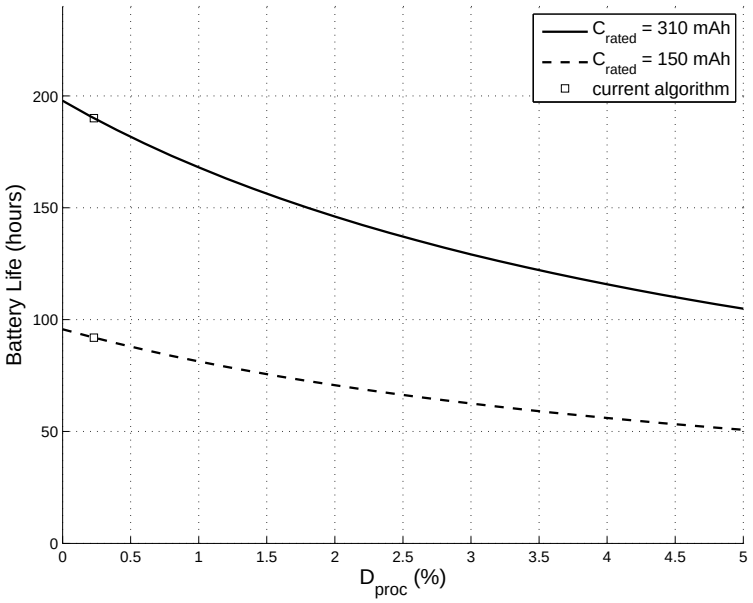


Figure 6.2.: CPU processing duty vs. battery life

down with increasing the clock frequency. However, the energy consumed by the CPU for sampling increases F_{CPU} due to the power mode transitions and fixed delays involved in sampling the data. Hence, an intelligent strategy would be to operate the system with two different clock frequencies in active mode itself, a low frequency during sampling and a high frequency during processing. At least during the sampling task, the PLL-based clock source can be replaced with the internal RC oscillator which will bring the consumption further down.

The energy consumed by the SAM3U to sample the accelerometer scales down with decreasing the sampling frequency F_s. The current choice of $F_s = 125\ Hz$ is an oversampling for the existing respiratory processing algorithm. Depending on the needs of the other processing chains in the system (e.g. pulse), the sampling frequency can be redefined to conserve energy.

The model predictions are compared with the measurements carried out on the ACREMO hardware. Table 6.1 lists the estimations and measurements of the average system current at different clock frequencies. The comparison shows that the model underestimates the system current with a mean error of -5.2%. The model performance is poor during the CPU operation at 24 MHz which is because the estimation of the CPU's sampling power is inaccurate at this frequency. Hence, some of the prospects for extension of this work in the future are improvement of the modeling of sampling power and refinement of the power model on the whole based on the target measurements.

Table 6.1.: Model error in estimating the average system current

F_{CPU}	I_{sys} (μA)		error
(MHz)	estimated	measured	(%)
12	1406	1448	-2.9
24	1512	1703	-11.2
48	1544	1630	-5.3
96	1631	1655	-1.4

6.3. Power optimization

Several power optimization techniques were studied and some of the relevant techniques are investigated in Chapters 4 and 5. The power conservation techniques such as the choosing lowest power mode for idling, scaling the CPU clock frequency, and reducing the sampling frequency are discussed already in the section above.

In Section 4.4, a heterogeneous multiprocessor sensor system is proposed as a solution to conserve power during both low and high utilization phases of the computing subsystem. A low-end microcontroller (MSP430FG47x) is assigned the sampling task while the algorithmic functions were executed by a high-end processor (SAM3U). The proposed system would save up to 16% of the overall average power consumed by the current ACREMO system.

The problem of software partitioning in a multiprocessor system is addressed in [51]. The partitioning is formulated as problem of optimizing the energy cost of the system. The system constraints, power consumptions and transition energies of each of the processors are supplied to the formulated Integer Linear Program (ILP). Such an approach is useful for the ACREMO system when the number of tasks and functionalities are increased in future versions of the system. In such a case, use of a dedicated operating system for multiprocessor system can be considered to improve the timely processing of the events and to reduce overheads of the operating system. Han et al. [52] have presented the design and implementation of CoMOS, a component messaging based operating system for heterogeneous multiprocessor sensor systems. Such an operating system would reduce complexity of the software that supports synchronization and communication between the heterogenous processors in the proposed ACREMO system.

The tradeoffs between processing at the sensor node against the processing at the base station are studied in Section 4.5. Under the the existing ACREMO hardware architecture, the local processing (TX0) is much more efficient than transmitting the raw data (TX1) over the radio to the base station. The scheme (TX2), which transmits the downsampled data and avoids further local computations, consumes only 6% more power than the local processing. It is important to remember that the assumption in the scheme TX2 is that the SAM3U is running the radio stack and would be active during the radio communication. Hence, it is expected that the system lifetime target of 7 days or more can easily be achieved if the sampling

Table 6.2.: Optimization techniques and power savings

Technique	Details	Δ LIFE (%)
Component selection	accelerometer1	24.6
	accelerometer2	41.6
	OPAMP	7
Processor frequency scaling	F_{CPU}=12 MHz	9
Heterogeneous multiprocessor	SAM3U + MSP430	16
Algorithm redesign	SBC	1
Reduction of sampling frequency	F_s=125/8	15
User activity-driven	O2(nart_th=4s, t_{sl}=20s)	20.9

and the radio stack is executed on a low-end processor (e.g. MSP430).

When a high-end processor (e.g. SAM3U) is used to execute the ACREMO algorithm, the attempts to optimize the algorithm for power will have little impact on reducing the system's average power. Section 5.1.2 detailed the SBC algorithm, a modification on the current processing chain and classification algorithm. Though, the SBC algorithm brings down the memory footprint considerably, it reduces the system's current consumption by only 1%.

Section 5.3 proposed a simple power saving strategy by power cycling the system during motion artifacts. The proposed scheme enables the system to dynamically choose the operating point depending on the requirements of preferred power saving and allowable data loss. Though the scheme is simple and primitive, it serves as a starting point for a context-aware operation of the ACREMO system with the goal of conserving power. Currently, the scheme imposes system sleep for a fixed amount of time when an artifact of substantial length is detected. In future work, predictive shutdown techniques can be employed to determine a safe value for the sleep duration so that the data loss is minimal. Benini et al. have studied predictive shutdown (static, adaptive and stochastic) techniques to achieve dynamic power management in real-world systems [53].

Table 6.2 shows a summary of power savings estimated during this work on the ACREMO system using different techniques. The component selection, the system partitioning with a heterogeneous multiprocessor, and the context-aware power saving technique outperform other optimization efforts made during this work.

7. Conclusions and Future work

7.1. Conclusions

A configurable power consumption model was developed for the accelerometric respiration monitoring platform to estimate the battery lifetime. The model followed a simple, modular, and scalable architecture. The hardware components in the system were divided into four different subsystems, namely power, sensing, computing, and communication. The average consumption of each of the subsystems and hence the entire system was modeled. The power model enabled identification of the system parameters that impact the power consumption of the entire device. Some of those parameters are sampling frequency of the accelerometer, clock frequency of the CPU, duty cycle of the sensing subsystem, processing (algorithmic) duty cycle of the CPU, idle mode of the CPU, radio communication periodicity, communication data size etc. The model has estimated that the ACREMO platform with existing respiratory algorithm draws an average current of about 1.63 mA. Under a linear battery model, the device lifetime would be 190 and 92 hours when powered by a Lithium-ion battery with rated capacity of 310 mAh (*BATT1*) and 150 mAh (*BATT2*) respectively. Increasing the complexity of the existing algorithm by a factor of 4 would reduce the lifetime of *BATT1* to 168 hours and *BATT2* to 67 hours.

The power consumption modeling is primarily based on the values from data sheets of the components and some basic assumptions on the system behavior. Such an approach enabled the model to estimate the power and the lifetime even before the hardware was available. The later current measurements on the target showed that the model incurs a mean error of -5.2%. This error can be attributed to various reasons such as non-linearity in voltage regulators, use of conservative values for parameters while modeling the CPU's sampling power etc.

The power model developed in this work will serve as a tool for identifying the power consumption of each of the components, subsystems and hence the entire wireless sensor node much before the hardware realization. This will enable the designers to choose better hardware components and reduce the costs involved in rework. Also, the possible future modifications of the sensor node such as additional hardware components, new system features, changes in the algorithm complexity etc. can easily be incorporated into this model and their effects on the lifetime of the wireless sensor node can be predicted much before implementing the corresponding modification.

Different power optimization techniques were studied and investigated in this work with the aim of prolonging the battery lifetime of the ACREMO system

between consecutive charge cycles. The replacement of the accelerometer and the OPAMP with their low-power counterparts would save up to 41% and 7% of the system power respectively. A dynamic power management policy is proposed where a set of system states are identified and their usability is defined. While the algorithmic redesign and remote processing techniques showed only little improvement in power saving, a proposed heterogeneous multiprocessor system showed 16% increase in the device lifetime.

Reducing the CPU clock frequency and the sampling frequency resulted in 9% and 15% reduction in average power respectively. However, these reductions will have to be decided depending on the system applications and processing tasks other than the currently focussed respiratory monitoring. Finally, a context-aware power saving strategy is proposed which forces the system to sleep for a predefined amount of time when a motion artifact is detected on the accelerometer signal. Such a strategy can dynamically choose different tradeoff points depending on priorities over the importance of power saving and severity in data loss.

The results of the optimization studies reveal that the lifetime of the ACREMO device can be prolonged from the current estimate of 7.9 days to around 14 days with some compromises such as higher accelerometer noise, cost of an additional microcontroller, slight increase in system complexity etc. We believe that a battery lifetime of two weeks or more, for wireless sensor nodes employed in healthcare application, is a promising feature.

7.2. Future work

The power consumption model developed in this work can be perfected based on the target measurements. The model can also be extended to predict the lifetime of wireless sensor networks by incorporating the wireless network simulations. The simple sleep strategy proposed in this work for user activity-driver power saving can be improved by including statistical methods that predict the forced sleep durations more accurately.

Bibliography

[1] Multimedia Research Group Inc., "U.S. Industry & Market Survey Comprehensive Analysis of Wireless Sensor Systems Market," 2006.

[2] M. Haenggi, "Opportunities and challenges in wireless sensor networks," *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, pp. 1.1–1.14, jul. 2005.

[3] J. Al-Karaki and A. Kamal, "A taxonomy of routing techniques in wireless sensor networks," *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, pp. 6.1–6.24, jul. 2005.

[4] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[5] ZigBee Health, "ZigBee Wireless Sensor Applications for Health, Wellness and Fitness," mar. 2009.

[6] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava, "Energy-aware wireless microsensor networks," *Signal Processing Magazine, IEEE*, vol. 19, no. 2, pp. 40–50, 2002.

[7] L. Benini and G. De Micheli, *Dynamic Power Management: design techniques and CAD tools*. Kluwer Academic Publishers, 1998.

[8] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, (Piscataway, NJ, USA), pp. 48+, IEEE Press, 2005.

[9] T. Falck, J. Espina, J.-P. Ebert, and D. Dietterle, "Basuma - the sixth sense for chronically ill patients," in *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, pp. 4 pp. –60, apr. 2006.

[10] L. Nachman, J. Huang, J. Shahabdeen, R. Adler, and R. Kling, "Imote2: Serious computation at the edge," in *Wireless Communications and Mobile Computing Conference, 2008. IWCMC'08. International*, pp. 1118–1123, IEEE, 2008.

[11] L. Au, W. Wu, M. Batalin, D. Mclntire, and W. Kaiser, "Microleap: Energy-aware wireless sensor platform for biomedical sensing applications,"

in *Biomedical Circuits and Systems Conference, 2007. BIOCAS 2007. IEEE*, pp. 158 –162, nov. 2007.

[12] E. Shih, S. H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," in *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 272–287, ACM, 2001.

[13] O. Landsiedel, K. Wehrle, and S. Gotz, "Accurate prediction of power consumption in sensor networks," in *Proceedings of The Second IEEE Workshop on Embedded Networked Sensors (EmNetS-II)*, Citeseer, 2005.

[14] A. Sinha and A. Chandrakasan, "Dynamic power management in wireless sensor networks," *Design Test of Computers, IEEE*, vol. 18, pp. 62 –74, mar. 2001.

[15] S. Corroy, J. Beiten, J. Ansari, H. Baldus, and P. Mahonen, "Energy efficient selection of computing elements in wireless sensor networks," in *Sensor Technologies and Applications, 2008. SENSORCOMM '08. Second International Conference on*, pp. 312 –318, aug. 2008.

[16] L. Kohn, J. Corrigan, M. Donaldson, *et al.*, "To err is human: building a safer health system. A report of the Committee on Quality of Health Care in America, Institute of Medicine," 2000.

[17] J. Weissman, E. Schneider, S. Weingart, A. Epstein, J. David-Kasdan, S. Feibelmann, C. Annas, N. Ridley, L. Kirle, and C. Gatsonis, "Comparing patient-reported hospital adverse events with medical record review: do patients know something that hospitals do not?," *Ann Intern Med*, vol. 149, no. 2, p. 100, 2008.

[18] S. Gandevia and D. McKenzie, "Respiratory rate: the neglected vital sign.," *Med J Aust*, vol. 189, no. 9, 2008.

[19] D. Goldhill, S. White, and A. Sumner, "Physiological values and procedures in the 24 h before ICU admission from the ward," *Anaesthesia*, vol. 54, no. 6, pp. 529–534, 1999.

[20] J. McFadden, R. Price, H. Eastwood, and R. Briggs, "Raised respiratory rate in elderly patients: a valuable physical sign.," *British Medical Journal*, vol. 284, no. 6316, p. 626, 1982.

[21] K. Shelley and S. Shelley, "Pulse oximeter waveform: photoelectric plethysmography," *Clinical Monitoring: Practical Applications for Anesthesia and Critical Care. Philadelphia: WB Saunders Company*, pp. 420–428, 2001.

[22] K. Konno and J. Mead, "Measurement of the separate volume changes of rib cage and abdomen during breathing," *Journal of Applied Physiology*, vol. 22, no. 3, p. 407, 1967.

[23] R. Allison, E. Holmes, and J. Nyboer, "Volumetric dynamics of respiration as measured by electrical impedance plethysmography," *Journal of Applied Physiology*, vol. 19, no. 1, p. 166, 1964.

[24] T. Reinvuo, M. Hannula, H. Sorvoja, E. Alasaarela, and R. Myllyla, "Measurement of respiratory rate with high-resolution accelerometer and emfit pressure sensor," in *Sensors Applications Symposium, 2006. Proceedings of the 2006 IEEE*, pp. 192 – 195, 2006.

[25] D. Morillo, J. Ojeda, L. Foix, D. Rendon, and A. Leon, "Monitoring and analysis of cardio respiratory and snoring signals by using an accelerometer," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pp. 3942 –3945, aug. 2007.

[26] P. Hung, S. Bonnet, R. Guillemaud, E. Castelli, and P. Yen, "Estimation of respiratory waveform using an accelerometer," in *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium*, pp. 1493 –1496, may. 2008.

[27] A. Jin, B. Yin, G. Morren, H. Duric, and R. Aarts, "Performance evaluation of a tri-axial accelerometry-based respiration monitoring for ambient assisted living," in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pp. 5677 –5680, sep. 2009.

[28] N. Yazdi, F. Ayazi, and K. Najafi, "Micromachined inertial sensors," *Proceedings of the IEEE*, vol. 86, no. 8, pp. 1640–1659, 2002.

[29] STMicroelectronics, "LIS344ALH data sheet," 2008.

[30] Analog Devices, "AD7689 data sheet," 2009.

[31] D. Terrell, *Op Amps: design, applications, and troubleshooting*. Newnes, 1996.

[32] Texas Instruments, "OPA2348 data sheet," 2009.

[33] J. Yiu, *The Definitive Guide to the ARM Cortex-M3*. Newnes, 2009.

[34] Atmel Corporation, "SAM3U data sheet," 2010.

[35] Cyprus Semiconductors, "CY62157EV30 data sheet," 2009.

[36] M. Kuorilehto, M. Kohvakka, J. Suhonen, P. Hämäläinen, M. Hännikäinen, and T. Hämäläinen, *Ultra Low Energy Wireless Sensor Networks in Practice*. Wiley Online Library, 2007.

[37] Texas Instruments, "CC2420, 2.4 GHz IEEE 802.15.4/ZigBee-ready RF Transceiver - Data sheet," 2006.

[38] Varta Microbattery, "LPP402934E data sheet," 2009.

[39] J. Hill and D. Culler, "A wireless embedded sensor architecture for system-level optimization," tech. rep., 2001.

[40] T. Panigrahi, D. Panigrahi, C. Chiasserini, S. Dey, R. Rao, A. Raghunathan, and K. Lahiri, "Battery life estimation of mobile embedded systems," in *VLSI Design, 2001. Fourteenth International Conference on*, pp. 57–63, IEEE, 2002.

[41] C. Chiasserini and R. Rao, "Pulsed battery discharge in communication devices," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 88–95, ACM, 1999.

[42] Avago Technologies, "ASMT RF45, Ultra-thin surface mount chipLED - Data sheet," 2008.

[43] IEEE 802.15.4-2006, "IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)," 2006.

[44] J. Espina, T. Falck, and O. Mülhens, "Network topologies, communication protocols, and standards," *Body sensor networks*, pp. 145–182, 2006.

[45] Intel, Microsoft, and Toshiba, "Advanced configuration and power management interface (acpi)," 2010.

[46] B. Schott, M. Bajura, J. Czarnaski, J. Flidr, T. Tho, and L. Wang, "A modular power-aware microsensor with¿ 1000X dynamic power range," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pp. 469–474, IEEE, 2005.

[47] D. McIntire, K. Ho, B. Yip, A. Singh, W. Wu, and W. Kaiser, "The low power energy aware processing (LEAP) embedded networked sensor system," in *Proceedings of the 5th international conference on Information processing in sensor networks*, pp. 449–457, ACM, 2006.

[48] D. Lymberopoulos, N. Priyantha, and F. Zhao, "mPlatform: a reconfigurable architecture and efficient data sharing mechanism for modular sensor nodes," in *Proceedings of the 6th international conference on Information processing in sensor networks*, pp. 128–137, ACM, 2007.

[49] S. Hong, N. Kim, C. Pyo, and W. Kim, "Hybrid Sensor Module and Data Processing Using Low-Power Wakeup in WSN," in *2010 Fourth International Conference on Sensor Technologies and Applications*, pp. 191–195, IEEE, 2010.

[50] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.

[51] M. Goraczko, J. Liu, D. Lymberopoulos, S. Matic, B. Priyantha, and F. Zhao, "Energy-optimal software partitioning in heterogeneous multiprocessor embedded systems," in *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, pp. 191 –196, jun. 2008.

[52] C. Han, M. Goraczko, J. Helander, J. Liu, B. Priyantha, and F. Zhao, "Comos: An operating system for heterogeneous multi-processor sensor devices," tech. rep., MSR-TR-2006-117, 2006.

[53] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 8, no. 3, pp. 299–316, 2000.

Appendices

A. Hardware Details

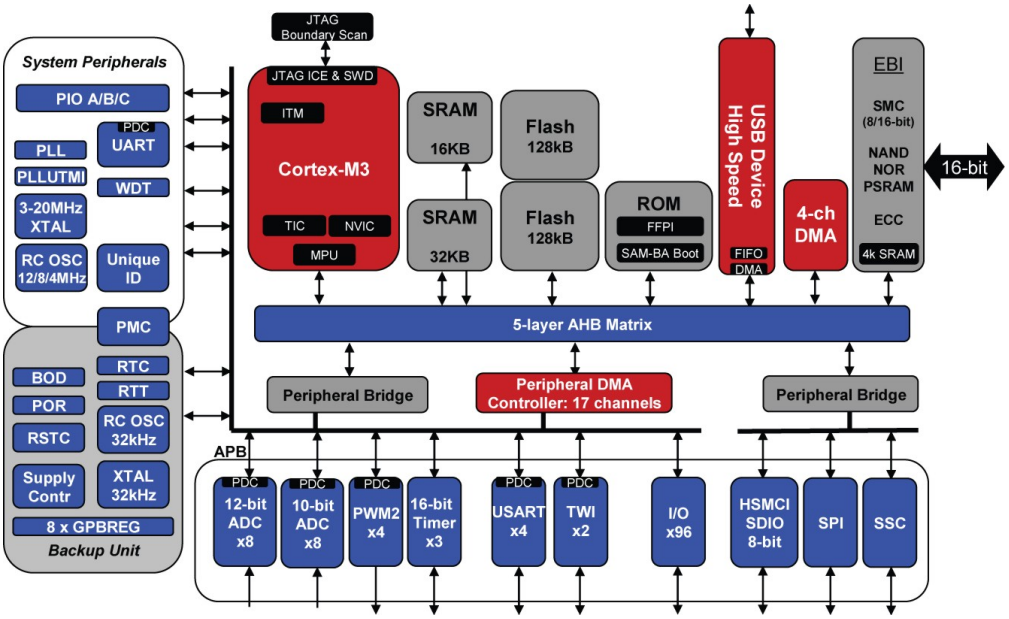


Figure A.1.: Block diagram of Atmel AT91SAM3U

Table A.1.: SAM3U current consumption vs. CPU clock frequency

MCK	Crystal Osc								RC Osc			
(MHz)	96	84	72	60	48	36	24	18	12	8	4	1
I_{active} (μA)	48.3	44.3	39.7	35.4	30.5	26.1	20.6	18.5	10.1	7.5	2.7	1.42
I_{sleep} (μA)	21.9	20.0	17.3	14.5	11.7	9.6	6.6	4.8	2.8	1.9	1.013	0.343

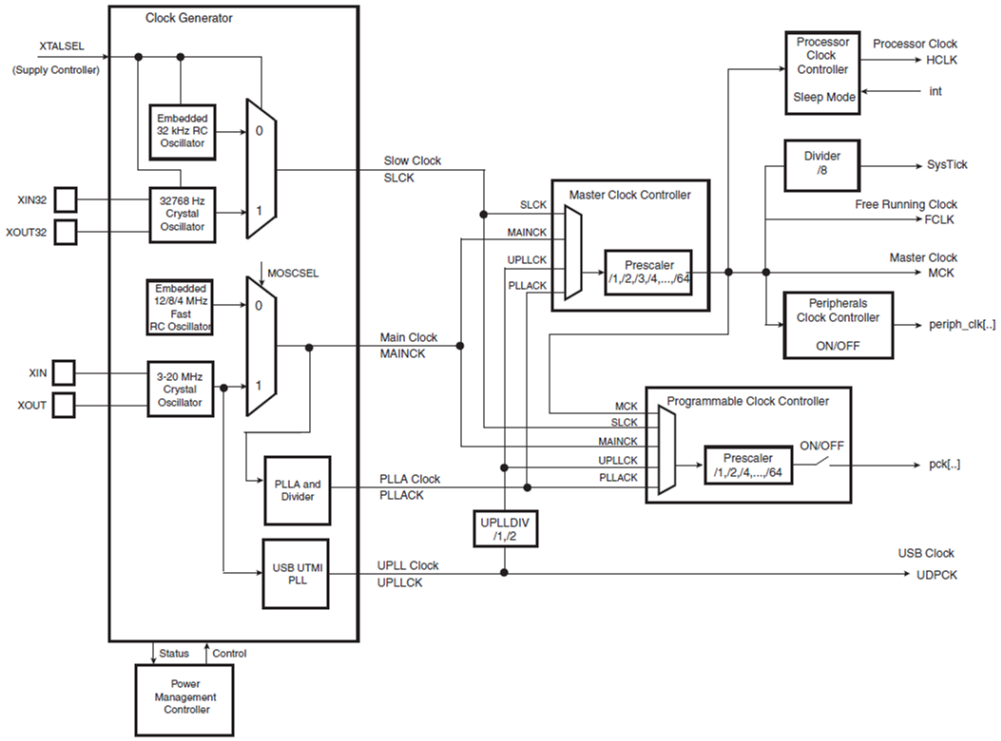


Figure A.2.: Block diagram showing different clock sources and their distributions in AT91SAM3U

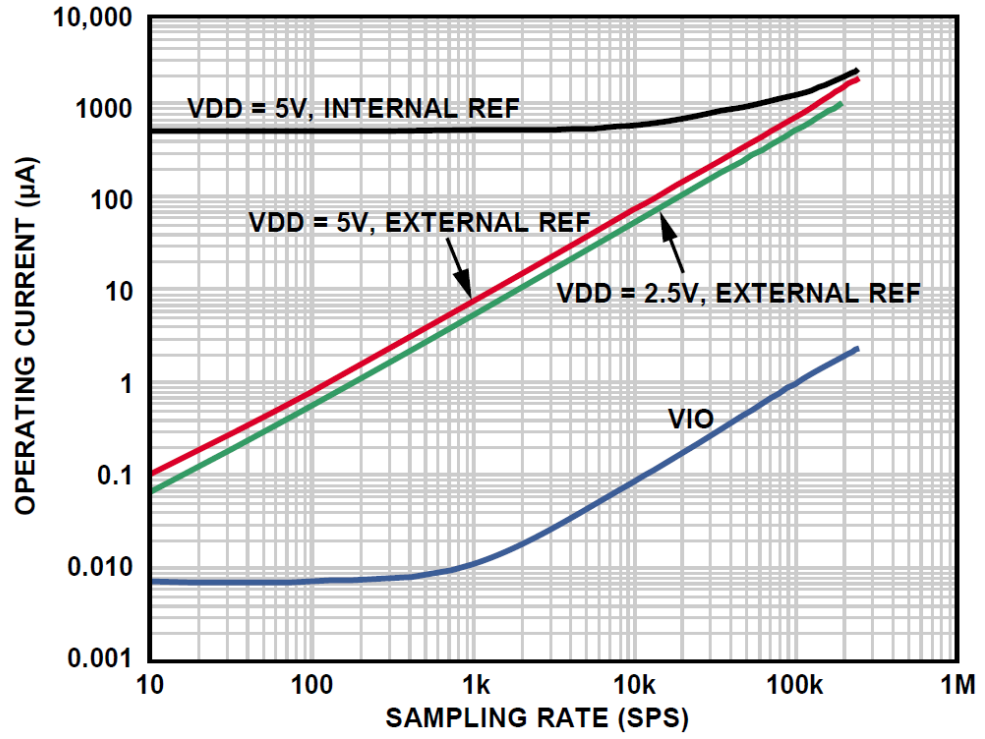


Figure A.3.: ADC current consumption vs sampling rate

From / To	Main Clock	SLCK	PLL Clock
Main Clock	–	4 x SLCK + 2.5 x Main Clock	3 x PLL Clock + 4 x SLCK + 1 x Main Clock
SLCK	0.5 x Main Clock + 4.5 x SLCK	–	3 x PLL Clock + 5 x SLCK
PLL Clock	0.5 x Main Clock + 4 x SLCK + PLLCOUNT x SLCK + 2.5 x PLLx Clock	2.5 x PLL Clock + 5 x SLCK + PLLCOUNT x SLCK	2.5 x PLL Clock + 4 x SLCK + PLLCOUNT x SLCK

(a) between main and PLL

From / To	PLLA Clock	UPLL Clock
PLLA Clock	2.5 x PLLA Clock + 4 x SLCK + PLLACOUNT x SLCK	3 x PLLA Clock + 4 x SLCK + 1.5 x PLLA Clock
UPLL Clock	3 x UPLL Clock + 4 x SLCK + 1.5 x UPLL Clock	2.5 x UPLL Clock + 4 x SLCK + UPLLCOUNT x SLCK

(b) between two PLLs

Figure A.4.: SAM3U clock switching timings (worst case)

B. Software Details

B.1. Filter design

During this work, a low pass filter with cut-frequency 2 Hz was designed and used in the single breath classifier implementation. Following is the MATLAB script that generates the filter coefficients.

```
% LPF FILTER DESIGN
N     = 24;       % Order
Fpass = 2;        % Passband Frequency
Fstop = 2.5;      % Stopband Frequency
Astop = 30;       % Stopband Attenuation (dB)
Fs    = 15.625;   % Sampling Frequency

h  = fdesign.lowpass('n,fp,fst,ast', N, Fpass, Fstop, Astop, Fs);
Hd = design(h, 'equiripple');
```

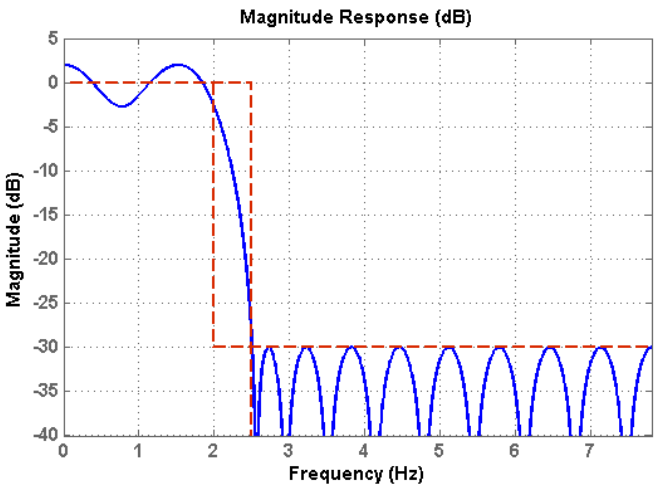


Figure B.1.: LPF - Magnitude Response

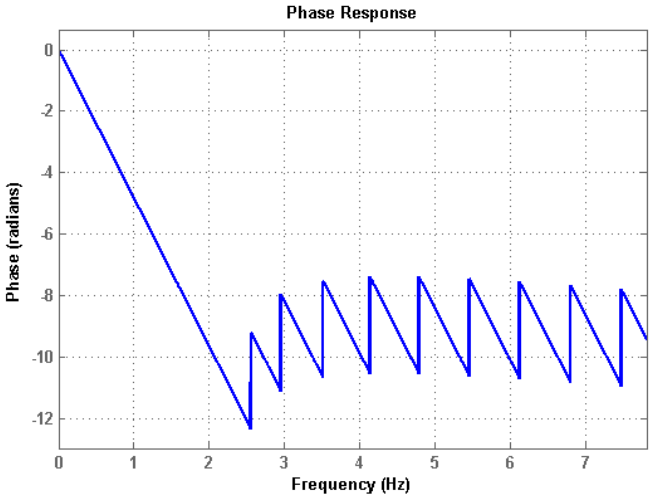


Figure B.2.: LPF - Phase Response

B.2. Single breath classifier

The decision tree classifier was designed using WEKA [50] (J48 tree). The confidence level C was set to 0.05 while the minimum number of object M was 100. The classifier training was carried on a manually annotated training data set that contains 11765 breaths and 9657 non-breath candidates.

Table B.1.: Properties of the trained single breath classifier

SBC properties	
Sensitivity	87%
PPV	88.4%
Accuracy	86.9%
#leaves	23
tree size	45

Table B.2.: Features used for breathing signal classification

FWC (per 32sec frame)	SBC (per breath-candidate)
standard deviation	standard deviation
mean	mean
standard deviation (vector)	width
orientation vector	slope
dominant frequency	relative peak amplitude
spectral entropy	peak position
	number of minima inside candidate

Table B.3.: SBC vs. FWC algorithm - memory usage (in bytes)

Allocation	SW component	FWC	SBC
Static	SegBufPoolFifo	244	124
	FeatBufPoolFifo	-	124
	SegBufFifo1	244	124
	SegBufFifo2	244	124
	SegBuf * N	8400	4200
	FeatBuf * N	-	1560
	WndBuf	1034	-
	subsample, lowpass	732	732
	preclass/precalc	130	88
	Background	572	0
Dynamic	Foreground	124	124
	Background	2588	0
TOTAL		14,312	7,200

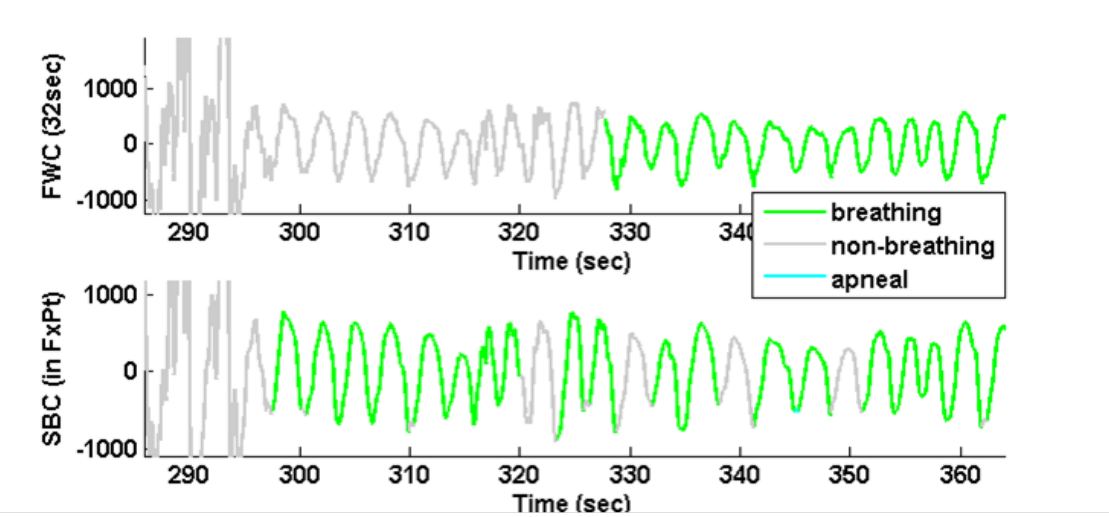


Figure B.3.: SBC vs. FWC algorithm - signal classification, example 1

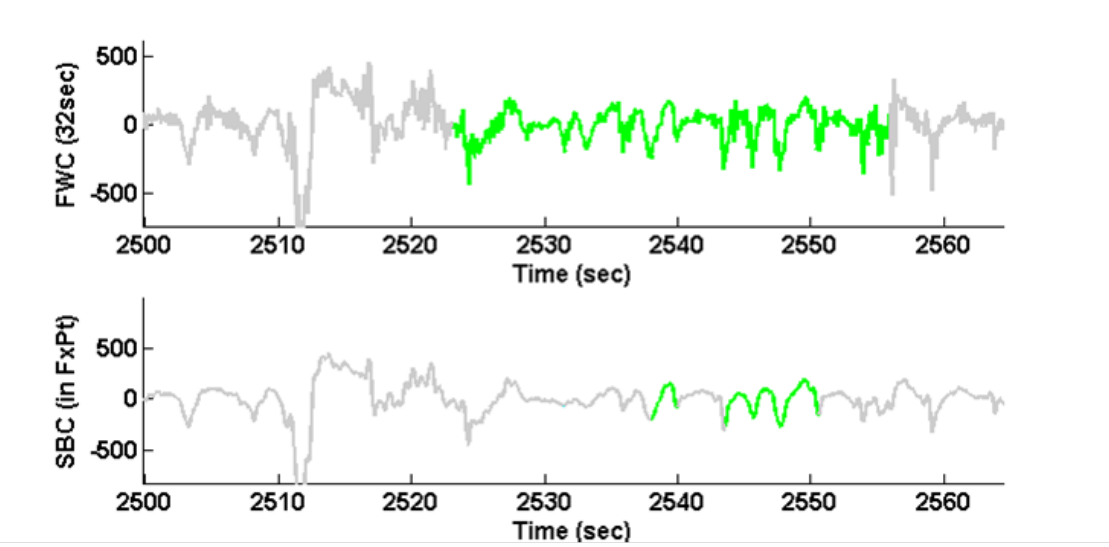


Figure B.4.: SBC vs. FWC algorithm - signal classification, example 2

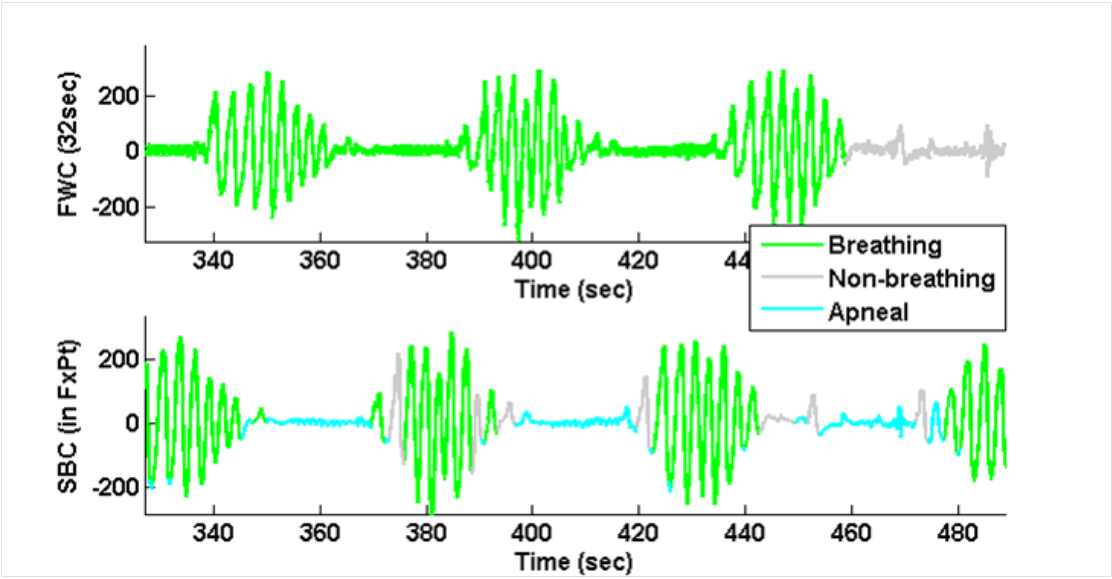


Figure B.5.: SBC vs. FWC algorithm - signal classification, example 3

B.3. Artifact pre-classifier

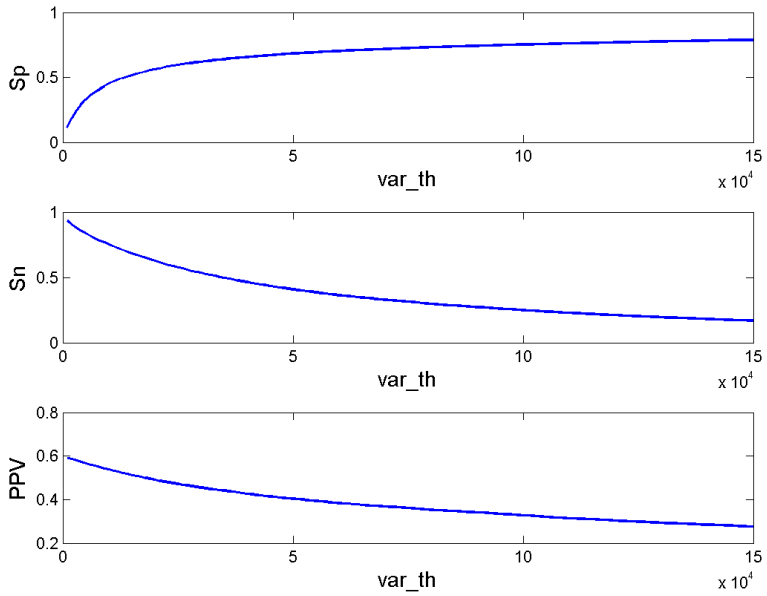


Figure B.6.: Pre-classifier performance curves against the amplitude variance threshold var_th.

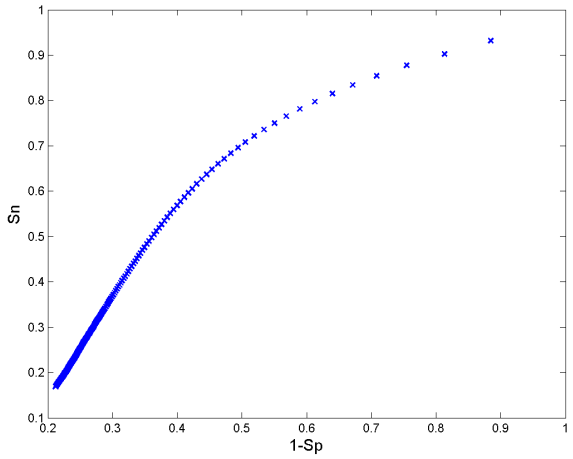


Figure B.7.: Receiver operating characteristic (ROC) curve of the artifact pre-classifier

Declaration

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

(Mathankumar Gopalsamy)