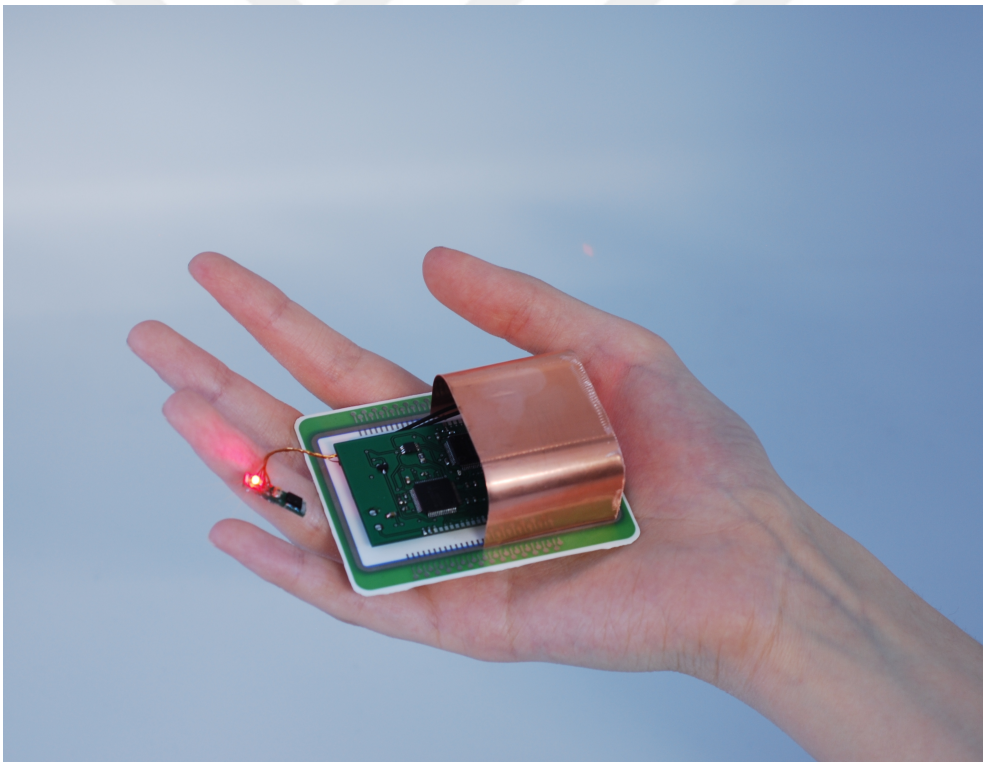
IMTEK

Master's Thesis

Design and Realization of an Implantable Blood Pressure and ECG Sensor Read-Out



Zeynep Duygu Sütgöl

UNI FREIBURG

A master's thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science of Microsystems Engineering

according to the examination regulations at the University of Freiburg for the Master's degree in Microsystems Engineering of July 20, 2001.

Laboratory for Micro-optics
Department of Microsystems Engineering (IMTEK)
University of Freiburg
Freiburg im Breisgau, Germany

Author Zeynep Duygu Sütgöl

Thesis period June 1, 2011 to May 31, 2012

Referees Prof. Dr. Hans Zappe, Laboratory for Micro-optics
Prof. Dr. Peter Woias, Laboratory for Design of Microsystems

Supervisor Dipl.-Ing. Michael Theodor, Laboratory for Micro-optics

Title page The title picture represents the concept of the realized implantable sensor read-out system.

Declaration

according to the Examination Regulations:

I hereby confirm to have written the following thesis on my own, not having used any other sources or resources than those listed. All passages taken over literally or correspondingly from published sources have been marked accordingly. Additionally, this thesis has not been prepared or submitted for another examination, neither partially nor completely.

Freiburg, May 31, 2012

Zeynep Duygu Sütgöl

Zusammenfassung

Die kontinuierliche Überwachung von Vitalparametern über lange Zeiträume ermöglicht Medizinern eine verbesserte Beurteilung des Gesundheitszustandes eines Patienten, eine frühzeitige Diagnose als auch optimierte Therapie Möglichkeiten. Invasive, medizinische Messverfahren sind von großem Interesse in Industrie und Wissenschaft. Sie messen reproduzierbarer und genauer als nichtinvasive Messverfahren und ermöglichen eine ununterbrochene Überwachung des Patienten. In dieser Arbeit wird eine implantierbare Ausleseeinheit vorgestellt, die chronisch Elektrokardiogramme (EKG), Photoplethysmogramme (PPG), den arteriellen Blutdruck und die Köpertemperatur in einem Lebewesen aufnehmen kann.

Um den Blutpuls subkutan zu detektieren ist die präsentierte Ausleseschaltung mit implantierbaren PPG-Sensoren, die am Lehrstuhl für Mikrooptik entwickelt wurden verbunden. Zur Messung der weiteren Parameter sind zwei goldbeschichtete Kupferfolien als EKG-Elektroden und ein intraarterieller Tip-Katheter als Blutdrucksensor angeschlossen. Ein hermetisch versiegeltes, biokompatibles Implantatgehäuse und eine elektronische Schaltung zur Datenerfassung –und speicherung, sowie die zugehörige Software wurden entwickelt.

Die Sensor-Ausleseschaltung wurde erfolgreich auf der Körperoberfläche getestet. Bei in vivo Messungen, die an einem Hausschwein durchgeführt wurden, konnte das entwickelte System ungefilterte PPG-Signale von subkutanem Gewebe, mit sehr geringem Rauschanteil aufnehmen. Des Weiteren wurde die Abhängigkeit des subkutan gemessenen EKG-Signals in Abhängigkeit des Elektrodenabstandes -und deren Fläche evaluiert. Um die Eignung des Systems für den chronischen Einsatz nachzuweisen wurde es 14 Tage in einem Hausschaf getestet.

Das implantierte System verursachte keine ungewöhnliche Gewebereaktion und wurde in seiner Funktion durch den Einfluss des Körpers nicht beeinträchtigt. Mit der Ausleseeinheit konnten photoplethysmographisch pulsatile Signale im wachen Tier aufgezeichnet werden. Die Ergebnisse der Langzeitimplantation zeigen, dass das Ziel dieser Arbeit erreicht und ein implantierbarer Patientenmonitor entwickelt wurde.

Stichwörter: Patientenmonitoring, Implantat, Photoplethysmographie, Electrocardiographie, biokompatibel, Sensor Read-Out.

Abstract

Continuous long-term monitoring of vital signs provides physicians the possibility of an improved evaluation of a patients's health state, an early diagnosis and an optimized therapy. Invasive measurement methods in medical applications have been a research subject of much interest in industry and science. They provide more stable and accurate results than the non-invasive ones, with the ability of incessant tracking. In this thesis, an implantable read-out system is introduced that records chronic electrocardiogram (ECG), photoplethysmogram (PPG), arterial blood pressure (BP) and temperature measurements from a living organism.

In order to acquire the mentioned physiological parameters subcutaneously, the presented read-out system is connected to implantable PPG sensors that were developed by the Laboratory of Micro-optics, two gold coated, copper foil ECG electrodes and a tip catheter BP sensor. A hermetically sealed, biocompatible implant package and an electronic circuitry for data acquisition and recording, along with the implemented software were designed.

The sensor read-out system was successfully tested on the body surface. The conducted acute in vivo measurements on a pig provided raw PPG-signals from subcutaneous tissue which exhibited a low noise profile, acquired by the system. Additionally, the dependencies of the ECG signal on inter-electrode distances and electrode dimensions were evaluated by the performed, subcutaneous ECG measurements. For the examination of its chronic performance, the sensor read-out system was implanted to a domestic sheep for a 14-day long-term measurement.

The implanted system did not cause a distinguished tissue response or exhibit malfunction due to the impact of the body environment. Chronic, pulsatile signals were recorded by the system from the unsedated animal. The results of the long-term implantation indicate that the objective of this thesis has been achieved and an implantable medical monitor has been developed.

Keywords: medical monitoring, implantable, photoplethysmography, electrocardiography, biocompatible, sensor read-out

Contents

Nomenclature

Abbreviations

abbreviation	meaning
BP	Blood Pressure
PPG	photoplethysmogram
ECG	electrocardiogram
AC	Alternating Current
DC	Direct Current
LED	Light Emitting Diode
PTT	Pulse Transit Time
ADC	Analog-to-Digital Converter
SPI	Serial Peripheral Interface
USCI	Universal Serial Communication Interface
I^2C	Inter-Integrated Circuit
UART	Universal Asynchronous Receiver/Transmitter
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
VLO	Very Low Oscillator
RAM	Random Access Memory
TCK	Test Clock
TDI	Test Data Input
TDO	Test Data Output
PGA	Programmable-Gain Amplifier
RLD	Right Leg Drive
SPS	Sample per second
SNR	Signal-to-Noise Ratio
DAC	Digital-to-Analog Converter
DFN	Dual-Flat No-Leads
BJT	Bipolar Junction Transistor
SDA	Serial Data Line
SCLK	Serial Clock
SOMI	Slave Out Master In
SIMO	Slave In Master Out
PCB	Printed Circuit Board
IDE	Integrated Development Environment
ACLK	Auxiliary Clock
SMCLK	Subsystem Master Clock

abbreviation	**meaning**
MCLK	Master Clock
CPU	Central Processing Unit
ISR	Interrupt Service Routine
PUC	Power Up-Clear
GIE	General Interrupt Enable
SR	Status Register
PC	Program Counter
CS	Chip Select
TAIFG	Timer_A Interrupt Flag
TAIV	Timer_A Interrupt Vector
TBIFG	Timer_B Interrupt Flag
MSB	Most Significant Bit
MSB	Least Significant Bit
RDATAC	Read-Data-Continously
SDATAC	Stop-Continous-Read
RDATA	Read-Data
DOUT	Data Out
FIR	Finite Impulse Response
RH	Relative Humidity

1 Introduction

Monitoring systems have been an essential part of clinical medicine for diagnosis and consequential therapy. In recent years, long-term monitoring of vital signs has undergone a substantial change and has been subject to a large amount of research activities. Novel advances in hardware miniaturization, wireless power transmission and embedded systems have lead to the development of implantable biomedical monitoring systems that have broad and important implications for medical applications.

Long-term, continuous monitoring of blood pressure (BP) and other cardiovascular parameters is significantly needed for treatment and control of many cardiovascular diseases. Having a monitoring system that samples a number of parameters can increase the efficiency of diagnosis. By implanting the monitoring systems and fixing the measurement sensors, motion artifacts and powerline interference coupled to the measurement wires can be minimized (Mokwa and Schnakenberg 2001), and the patient comfort can be increased. To date, several implantable electrocardiogram (ECG) monitoring systems that reduce the amount of disturbance of non-invasive measurement devices, have been introduced (Hutten et al. 1998), (Klein et al. 1998), (Song et al. 2004), (Riistama et al. 2007), (Lobban and Stafford 2008). Furthermore, various approaches have been taken for blood pressure monitors, in science and industry. The most common methods are through the use of an external cuff or through a catheter-based system. The fundamental reasons that these methods fail for long term, continuous pressure measurement are comfort, intermittent measurement, occlusion of blood flow, and percutaneous connections (Potkay 2007). Thus, employing implantable devices enables continuous and accurate data monitoring.

Within the framework of VITAMON and OPT4LIFE projects at the Laboratory of Micro-optics, an implantable photoplethysmogram (PPG) sensor was developed. The PPG sensor emits light into the artery and detects the intensity changes caused by the blood flow, in the backscattered light (Fiala et al. 2010). In the presented sensor concept, along with the measurement of ECG signal, BP is obtained through pulse transit time (PTT), which is derived from the time between the R-peak of the ECG and a characteristic point of a PPG. It has been shown in earlier projects, that the PPG sensor is capable of measuring heart rate, blood oxygen saturation and pulse transit time, which displayed high correlation to the blood pressure (Fiala et al. 2010). However, these measurements could only be performance during acute in vivo experiments.

In this Master project, an implantable read-out system is presented for the mentioned PPG sensors and subcutenaous measurement of ECG, intra-arterial BP and body temperature. The battery-powered read-out system in a hermetic and biocom-

patible packaging acquires, and stores vital signs of the implant subject periodically over long term. The read-out system receives the signals from the connected PPG sensors, ECG electrodes, intra-arterial BP sensor and an internal temperature sensor. After a period of data acquisition and storing in the memory card that is included in the electronic circuitry, the system enters to an idle mode with a power consumption of 614 μA, in order to maintain the operation lifetime.

In the context of this Master project, the design of an implant package, the hardware that is constructed of commercially available electronic components and a software program to regulate the operation of the read-out, were performed. The laboratory and in-vivo animal measurements under approved animal experiment license were conducted in order to examine the long-term monitoring capabilities of the read-out system. The acute PPG signals, acquired by the sensor read-out system from the in-vivo measurements, exhibited a very low noise profile and were accurately identified without any filtering. Additionally, the subcutaneous ECG measurements were conducted to investigate the optimum interelectrode distance and electrode dimensions. A 14-day animal implantation of the sensor read-out system was performed and provided long-term, subcutaneous measurement results and confirmed the biocompability and hermeticity of the implant package design. Thus, pulsatile signals were collected from an unsedated individual with an implanted sensor and data acquisition system.

The content of this Master thesis is organized as follows: Chapter 2 describes the basic principles of conventional pulse oximetry, ECG and the design of the adopted PPG sensor. Chapter 3 introduces the components of the read-out system with respect to its design and power consumption. The software design is covered in Chapter 4. The concept and fabrication of the biocompatible, hermetic implant package is shown in Chapter 5. Chapter 6 presents measurement results from the laboratory and in vivo tests and evaluates the long-term animal implantation. Final discussion is done in Chapter 7.

2 Theory

In this chapter, the biomedical signals to be recorded by the implantable read-out system are described with respect to their waveforms and acquisition methods. Moreover, the design of the adopted photoplethysmographic (PPG) sensor is evaluated briefly.

2.1 Biomedical Signals

The major objective of the developed read-out system is to record periodic information from an implant subject. Therefore, it is designed to be connected to the PPG sensors, electrocardiography (ECG) electrodes and a tip catheter sensor. The subcutaneous recording of the ECG and PPG signals enables pulse transit time (PTT) calculation, and blood pressure (BP) estimation. The PPG signals are also used to observe the changes in blood oxygen saturation. Meanwhile, the included tip catheter sensor delivers BP measurement from the artery, for comparison. The basic physics of these concepts are outlined in this section.

2.1.1 Electrocardiography

The Electrocardiography (ECG) is the tracing of the electrical activity generated by the heart through the cardiac cycle. Repeating waves of the ECG signal, displays the depolarization and polarization of the heart chambers. It measures voltage changes from a baseline voltage, rather than absolute voltages. The ECG can be used to determine the regularity and rate of heart beats, as well as the presence of abnormalities in heart rhythm.

The sino-atrial node located at the heart, triggers action potentials that propagate through the heart, causing sequence of excitations and contractions. The ECG waveform displays the patterns of this propagation and the phases of the cardiac cycle. During the cardiac cycle, blood pressure increases and decreases through the cardiovascular system. The blood that is pumped from the right ventricle, flows to the lungs that is involved in gas exchange between blood and alveoli. The newly oxygenated blood, then enters into the left side of the heart from where it is pumped into the blood vessels within and outside of organs (Klabunde 2005). A single cycle of cardiac activity can be divided into two basic phases: diastole and systole.

Diastole represents the period of time when ventricles of the heart are relaxed. The venous blood from the body and the oxygenated blood from lungs refill the right and left heart chambers. The P wave of the ECG occurs at the last phase of

diastole, and normally ranges from 0.12-0.2 seconds (Klabunde 2005). The P wave (See 2.1) represents the atrial depolarization, which leads to a rapid flow of blood into the ventricles.

The systole represents the time during which the left and right ventricles contract and eject blood into the aorta and pulmonary artery, respectively (Klabunde 2011). It begins with the QRS complex of the ECG, which represents polarization of the right and left ventricles. It is a sharp biphasic or triphasic wave of 80 ms duration. Repolarization (relaxation) of the ventricles causes the slow T wave, with the duration of 120 - 160 ms (Rangayyan 2002).

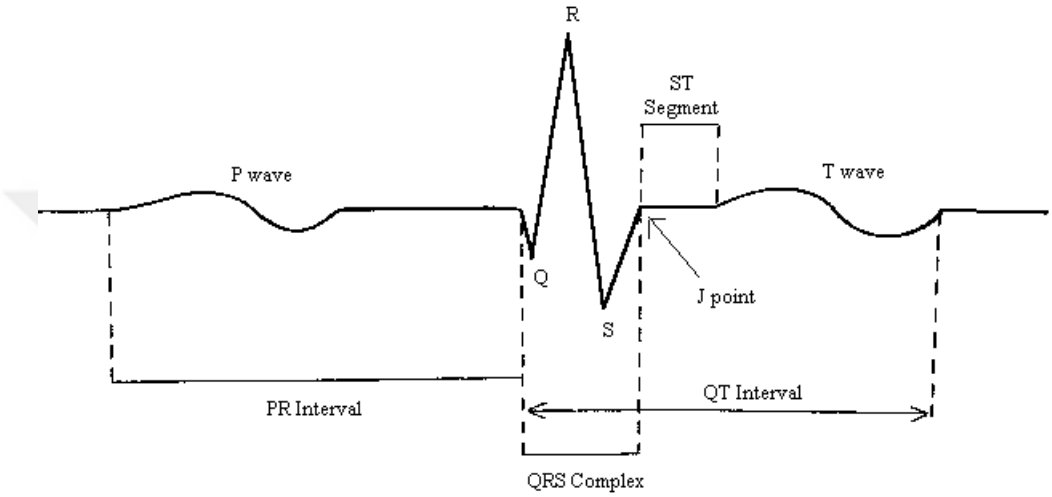


Figure 2.1: A typical ECG signal with its characteristic intervals

2.1.2 Photoplethysmography

Photoplethysmography (PPG) is an optical method to detect blood volume changes in a peripheral vascular bed. In this method, the emitted light, which is made to transverse the skin, is reflected, absorbed and scattered in the tissue and blood (Kyriacou 2006). It has been applied to clinical physiological monitoring of various parameters such as blood oxygen saturation, heart rate, blood pressure, cardiac output and respiration. The PPG waveform comprises a pulsatile ('AC') physiological waveform attributed to cardiac synchronous changes in the blood volume with each heart beat, and is superimposed on a slowly varying ('DC') baseline with various lower frequency components attributed to respiration, sympathetic nervous system activity and thermoregulation (Allen 2007). Figure 2.2 shows a typical PPG waveform, which starts starts with a positive steep due to the blood inflow during systole, followed by a descending runoff phase of arterial pressure, attributed to closure of the aortic valve (Murray and Foster 1996).

The PPG sensors often utilize semiconductor technology with light-emitting diodes

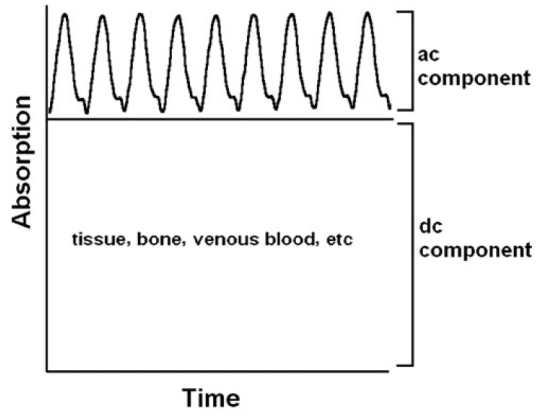


Figure 2.2: PPG waveform as measured through tissue (Murray and Foster 1996)

(LED) and matched photodetector devices working at the red and/or near infrared wavelengths (Allen 2007). Measurements can be performed in either transmission or reflection modes. In transmission mode, the sensor can be attached across a fingertip, foot, or earlobe. In this configuration, the LED and photodetector in the sensor are placed on opposite sides of a peripheral pulsating vascular bed. Alternatively, in reflection mode, the LEDs and photodetectors are both mounted side-by-side on the same planar substrate to enable readings from multiple body locations where transillumination measurements are not feasible (Mendelson et al. 2006).

Pulse oximetry is a method that relies on the detection of PPG signals and is used for monitoring the arterial blood oxygen saturation (SpO_2). It estimates SpO_2 by the ratio of the PPG signals at red and infrared wavelengths, since the light absorption of oxygenated and deoxygenated hemoglobin varies at these two wavelengths.

Together, the use of pulse oximeters with ECG measurements, provides the pulse transit time (PTT) for continuous blood pressure (BP) monitoring. In general the PTT refers to the time it takes a pulse wave to travel between two arterial sites and is measured by the temporal difference between the R wave in an ECG and the beginning of the following PPG wave (Marcinkevics et al. 2009). An acute rise and fall in BP, influences the arterial wall and changes the PTT value. This relation acknowledges PTT as a convenient parameter for continuous blood pressure measurement.

2.2 Implantable PPG Sensor

A reflection mode PPG sensor was designed and manufactured at the Laboratory of Micro-optics for pulse oximetry and PTT measurements. It consists of two pairs of photo diodes with LEDs at the wavelengths of 660 nm for the red, 935 nm for the infrared one. The photodetectors connect to a low noise dual transimpedance amplifier and filtering circuitry to remove the unwanted higher frequency noise, caused by the electrical interference. A transparent, biocompatible silicone rubber

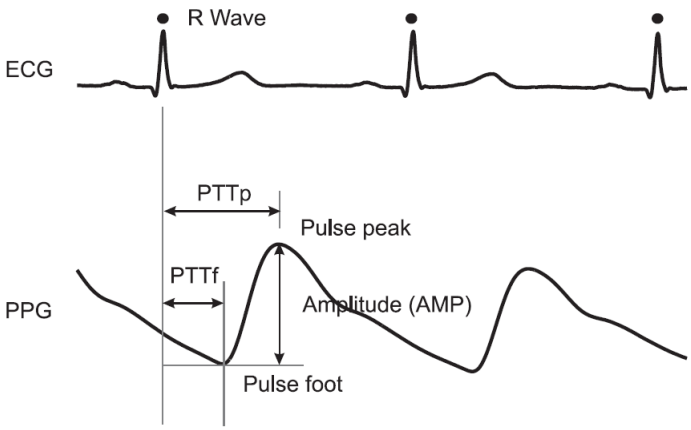


Figure 2.3: Pulse transit time to the foot of the pulse (PTTf), Pulse transit time to the peak of the pulse (PTTp), and foot-to-peak amplitude (AMP) (Allen 2007).

(MED-1000) covers the parylene coated PPG sensor against moisture. The sensor is connected to the sensor read-out sytem by thin, copper wires. The implantable PPG sensor is displayed in Figure 2.4.

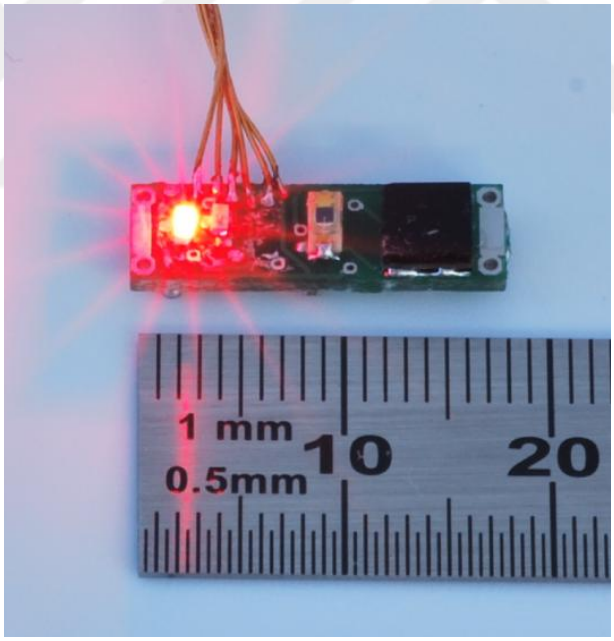


Figure 2.4: The PPG sensor with total dimensions of 18 mm×6 mm.

3 Hardware

Within the framework of this Master's project, an electronic circuitry is designed to perform functions of the sensor read–out system. This chapter presents the hardware design of the system by introducing the included analog and digital components, and describing the printed circuit board design.

3.1 Overview

The implantable sensor read-out system incorporates the following features that are required for long-term in vivo measurement of ECG, PPG and blood pressure signals:

- Acquisition:
  - ECG
  - Pulse Transit Time for blood pressure estimation
  - PPG Signals from up to 4 channels
  - Tip-Catheter blood pressure reference
  - Internal humidity and temperature of the implant package
- Configurable Data Rate: 250 sps to 32 ksps
- Standby current 614 μA
- LED-Drivers with automatic brightness adjustment
- 32 GB Data storage capacity
- Biocompatible hermetic package
- Internal battery life time of 14 days with 20 % duty cycle

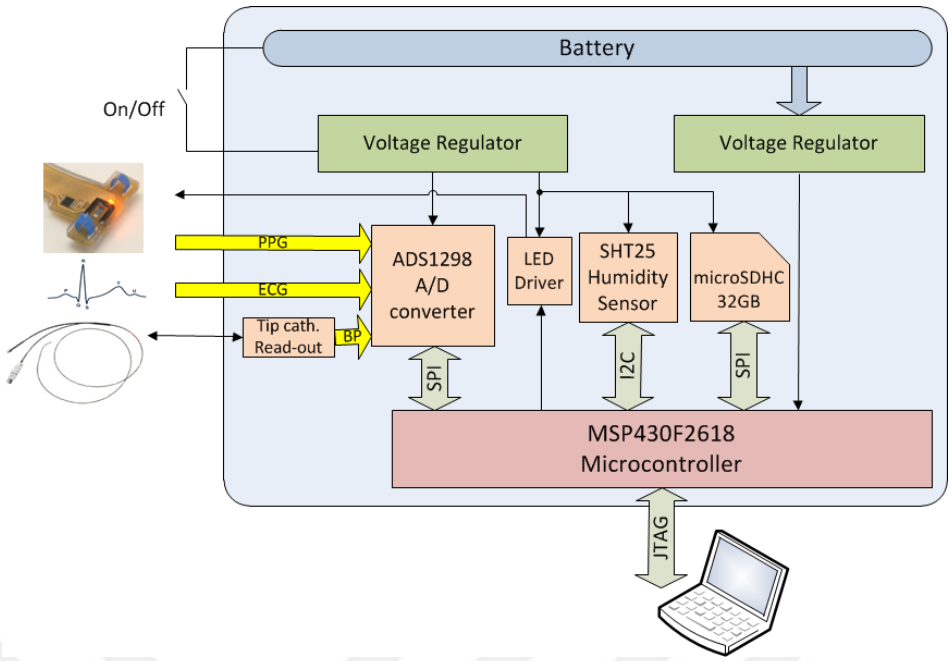


Figure 3.1: Overview of the Sensor Read-Out Hardware

Figure 3.1 shows the overview of the sensor read-out system. Biomedical signals, which are the blood pressure signal from a tip catheter, the PPG signal from the implanted sensors and the ECG signal, are delivered to the system to be sampled and converted into digital by ADS1298 analog-to-digital converters (ADCs). The microcontroller gathers these signals by serial peripheral interface (SPI) communication from ADS1298 and stores them in a 32 GB microSDHC card. The MSP430F2618 is used also for receiving internal temperature and humidity information of the implant package from an implemented sensor. The implanted system drives the LED currents of the connected PPG sensors. Powering the system is achieved by the internal rechargable lithium ion batteries. A JTAG port is provided for on-board debugging of the microcontroller. An on/off switch manages the powering of all the peripheral components.

3.2 Components

This section inspects functions and features of the components that are chosen for the sensor read-out implant.

3.2.1 MSP430F2618 Microcontroller

The 64 pin Quad-Flat-Pack MSP430F2618 microcontroller from the MSP430 family, manufactured by Texas Instruments is chosen as the microcontroller unit. It features a 16-bit RISC architecture, 116-kilobyte main memory, 256-byte flash memory

and 8-kilobyte RAM in von Neumann architecture, with four Universal Serial Communication Interfaces (USCI) that support asynchronous and synchronous communication protocols such as SPI (3 pin or 4 pin), I^2C (Inter-Integrated Circuit), and UART (Universal Asynchronous Receiver/Transmitter). Various sets of peripherals for different applications such as sensor systems and control applications are present in the MSP430. Less than 1 μs wake up time from standby mode and five low power mode operations are strong motivations for choosing this microcontroller in a battery powered implant. By using three internal clock signals, the user can achieve a multi-functional system with low power consumption. In the sensor read-out implant application, one internal and one external clock signals are used in addition to two 16-bit timer/counters. Low power consumption characteristics of the MSP430, are listed below (Texas Instruments Incorporated 2011):

- Active Mode: 365 μA at 1 MHz, 2.2 V
- Standby Mode (VLO): 0.5 μA
- Off Mode (RAM Retention): 0.1 μA

Another microcontroller configuration that meets one requirement of the sensor read-out design is the dual 12-bit D/A converter to allow driving the LEDs of the PPG sensors. The MSP430 has been debugged by ist 4-wire JTAG interface port, using a PC. Furthermore, with the capability of serial on-board programming, the microcontroller did not require external power supply while programming. The other prominent features of the MSP430 are given as:

- Operating Voltage 1.8-3.6 V
- 16-bit RISC Architecture
- Dual 12-Bit Digital-to-Analog (D/A) Converters
- Two 16-Bit Built-In Timers
- Synchronous SPI and I^2C Communication Interfaces
- Serial On-board Programming
- 8-kilobyte RAM

MSP430 JTAG Interface Debugging of the microcontroller has been achieved by connecting the JTAG signals to a computer through the MSP-FET430UIF, a flash emulation tool that includes USB debugging interface. Dedicated embedded emulation logic resides on the device itself and is accessed via JTAG without additional system resources (Texas Instruments Incorporated 2004).The 14-pin connector has been used to connect the USB-FET interface module to the PCB in 4-Wire JTAG configuration as shown in Figure 3.2.

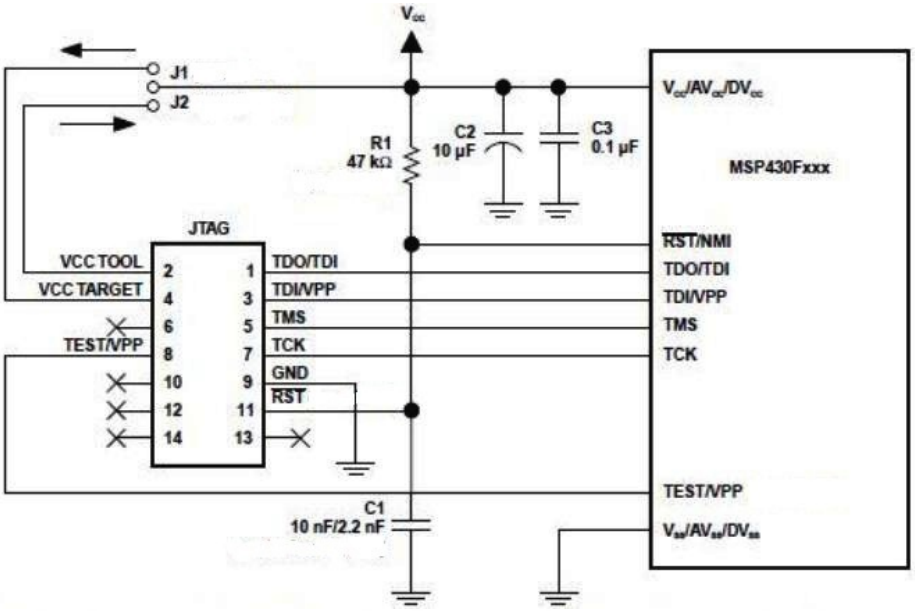


Figure 3.2: MSP430 JTAG 4 Wire Pin-Out (Texas Instruments Incorporated 2010)

The JTAG interface signals for data transmission are seen in Figure 3.2. The JTAG clock input (TCK) is provided from the system clock of the MSP430, while the JTAG data input (TDI) and output (TDO) signals are connected to the JTAG port pins on the microcontroller. Besides of these four standard connections, the 14-pin connector also carries power, ground and reset/nonmaskable interrupt signals of the interface. During the development process, the J2 connection was used for powering the system from the programming adapter. The final system uses the J1 connection and powers the JTAG interface with 3 V analog supply from the PCB.

3.2.2 ADS1298 Analog Front End for EEG/ECG

The ADS1298 is an 8-channel, simultaneous sampling, 24-bit, delta-sigma analog-to-digital converter (ADC), with a built-in programmable gain amplifier (PGA), an internal reference that can be programmed to either 2.4 V or 4 V, and an on-board oscillator. It supports analog supply between 2.7 V and 5.27 V and various ECG functions such as Goldberg and Wilson Center terminal, pace detection and Right Leg Drive (RLD) support. The ADS1298 is used in analog-to-digital conversion of biopotential input signals of the sensor read-out system and for transferring digital data to the microcontroller by using SPI. The PGA gain of each channel can be chosen from one of the seven settings (1, 2, 3, 4, 6, 8, and 12). The ADCs in the device offer data rates from 250 samples per second (sps) to 32 ksps (Texas Instruments Incorporated 2012). Its marketing for ECG use, adjustable gain of the device, different programmable sample rates, and 0.75 mW low power consumption per channel are the main reasons for choosing ADS1298 in this application.

For this project, four channels of the ADS1298 are connected to the implanted

ECG leads, 2 PPG sensors and the tip catheter in order to receive the ECG, PPG and blood pressure signals. The ADS1298 is powered by the unipolar supply, that comes from the output of the voltage regulator on the PCB. The device was configured to run in low power mode using 2.4 V internal reference for the ADCs with a sampling rate of 250 sps.

Further features of ADS1298 can be listed as follows:

- 112 dB Signal-to-noise ratio (SNR)
- -115 dB Common-mode rejection
- Input referred noise: 2.4-3481 μVpp and 0.4-332 μVrms for different sample rates.

The negative input pins of the PPG and the tip catheter signals are connected to the ground node of the PCB, while the positive single-ended input of these channels are connected to the relevant sensor outputs. The ECG signal was measured by the two connected electrodes on the subcutaneous tissue, and delivered to the ADS1298 channel as a fully differential input. Each supply of the ADS1298 is bypassed with 10 μF and 0.1 μF ceramic capacitors, as it is recommended in the datasheet of the ADS1298. Figure 3.3 illustrates the ADS1298 connection to an unipolar supply, while analog supply (AVDD) is referenced to analog ground (AVSS) and digital supplies (DVDD) are referenced to digital ground (DGND).

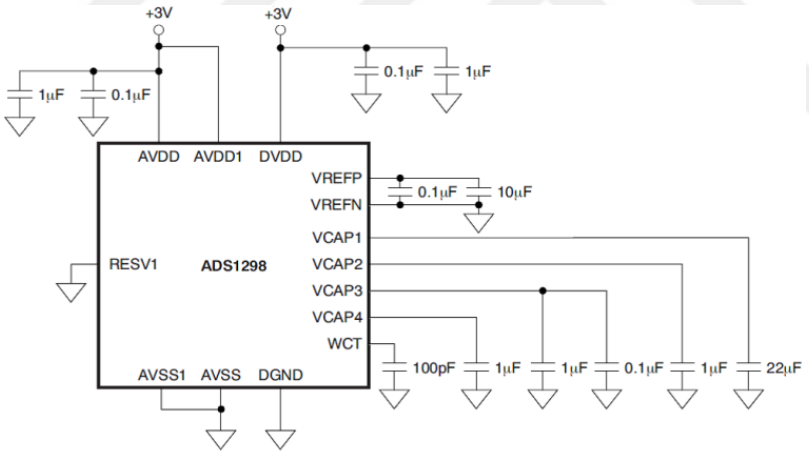


Figure 3.3: Single-Supply Operation of ADS1298 (Texas Instruments Incorporated 2012)

The analog-to-digital conversion in each channel of the ADS1298 is performed by a second-order sigma-delta modulator, followed by a digital decimation filter. As shown in Figure 3.4, a second-order sigma-delta modulator consists of integrators, digital-to-analog converters (DACs) and a quantizer. The modulator quantizes the difference between the *V_in* and the *V_mod* signals. Filtering out the noise at higher frequencies, eliminates the quantization noise and the redundant data due

to the oversampling at the comparator. The sampling rates of the ADS1298 are obtained from the divisions of the internal clock frequency: $f_{clk}/4$ for low resolution and $f_{clk}/8$ for high resolution modes.

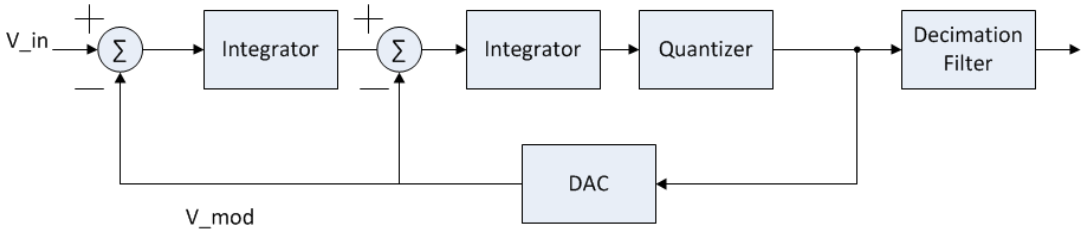


Figure 3.4: Block diagram of a 2nd order $\triangle\Sigma$ modulator

3.2.3 SHT25 Temperature and Humidity Sensor

In order to examine the hermeticity of the implant package and record temperature measurements, the SHT25, a humidity and temperature sensor from Sensirion is added to the system. The SHT25 sensor can operate with supply voltage in the range of 2.1 – 3.6 V and communicates by I^2C protocol, providing calibrated, linearized signals. The sensor is embedded in a reflow solderable Dual Flat No leads (DFN) package of 3 x 3 mm foot print and 1.1 mm height. The SHT25 consumes 8 μA supply current at the most, in idle state.

Figure 3.5 shows the sensor connection with the microcontroller. The serial clock (SCL) and serial data signals provide I^2C interface. The 10 kΩ pull-up resistors (R_p) pull the signal high, during the transmissions.

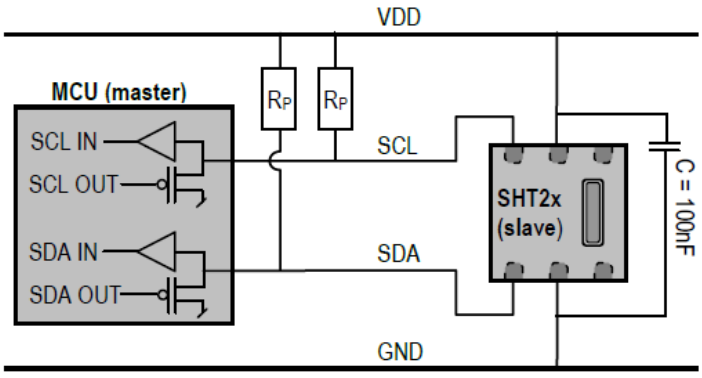


Figure 3.5: The SPI connection between the MSP430 and the SHT25 (Sensirion 2011)

3.2.4 LED Driver Block

The PPG sensors emit light via their included LEDs. These LEDs are driven by DAC12 modules of the MSP430. Two DAC12 modules are present as voltage output

DACs in the microcontroller, which convert the assigned register values into voltages. Port 6.5 and port 6.6 drive the complementary circuit which is a voltage-controlled current source arrangement of a low noise operational amplifier and a bipolar junction transistor (bjt). The two DAC12 modules control the current through the LEDs and thereby their light output level.

Figure 3.6 shows the LED driver circuit that is implemented on the sensor readout system. The 12-bit DAC voltage drives the noninverting input of the opamp. This voltage also appears across the 10 ohm resistor. The negative feedback configuration of the opamp, causes the base node to be one base-emitter junction potential (approximately 0.7 V) higher than the reflected output voltage of the DAC. The combination of the 100 ohm resistor and the coupling capacitor provides DC isolation for the input. By neglecting the common-base current gain value of the transistor which is close to unity, the LED current is derrived by:

$$Iled = \frac{Vdac}{10\Omega} \tag{3.1}$$

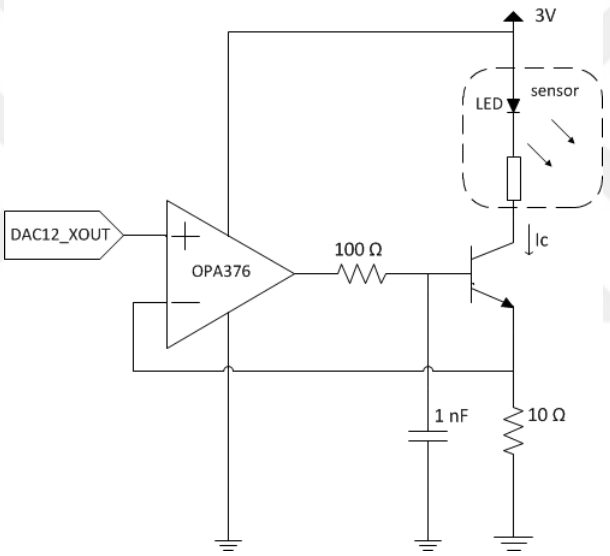


Figure 3.6: LED Drive Circuit

In 12-bit mode, the maximum usable value for DAC12_xDAT is 0FFFh. The DAC12 is configured in this mode with the reference voltage of *(Vref)* of 2.5 V, therefore the output voltage *(Vout)* formula is derived from the DAC12_xDAT data register by Eq. (3.2). In the software, the light intensities of the LEDs are forced to stay in a desired range by changing the value of the DAC12_xDAT registers.

$$Vout = \frac{Vref}{4096} \times DAC12\_XDAT, \tag{3.2}$$

Figure 3.7 displays the LED current over the DAC12 output which is simulated in OrCAD, CapturerCIS Demo, from Cadence Design Systems. As viewed in the

simulation graph, a linear voltage-current characteristics can be achieved with a slope of 0.1 $^{V}/_{A}$. Regulating the forward current maximum at 20 mA is sufficient for the operation of the PPG sensor.

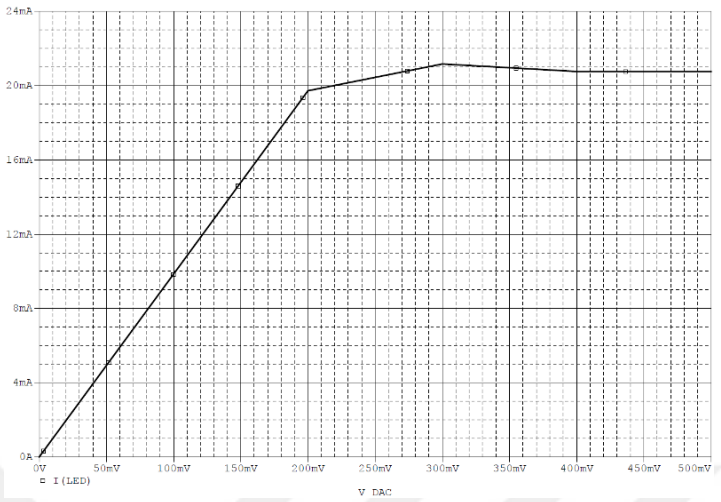


Figure 3.7: Current-Voltage characteristics of the LED drive circuit

3.2.5 Tip Catheter

An implantable Millar Mikro-Tip® catheter measures and transmits blood pressure to the sensor read-out system, providing a reference blood pressure for Pulse Transit Time (PTT) assessments. The piezo-resistive sensor has a current-driven Wheatstone bridge configuration as shown in Figure 3.8 and outputs a differential voltage.

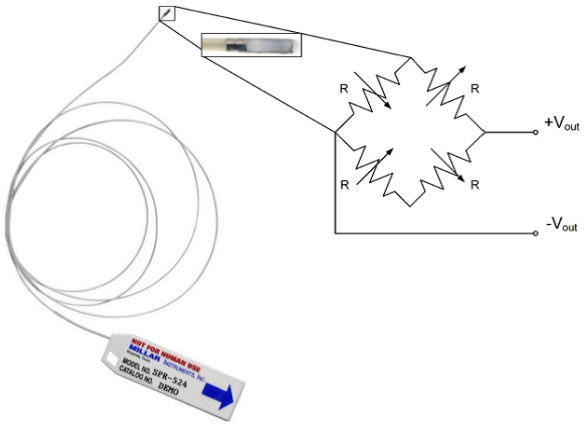


Figure 3.8: The Millar tip-catheter pressure sensor (Karakas 2012)

A read-out circuit had been designed in the Labaratory for Micro-optics, within the framework of the VITAMON project (Karakas 2012) and implanted on the PCB

design of the sensor read-out system as shown in Figure 3.9. A stable current source for the bridge is generated by the voltage regulator LT 6650, and reflected across Rset by the opamp. Then, the differential output voltage at the bridge is fed to the gain-programmable Analog Devices AD8555 differential amplifier. The gain of the amplifier has been set to 800 by digitally programming the chip with the MSP430.

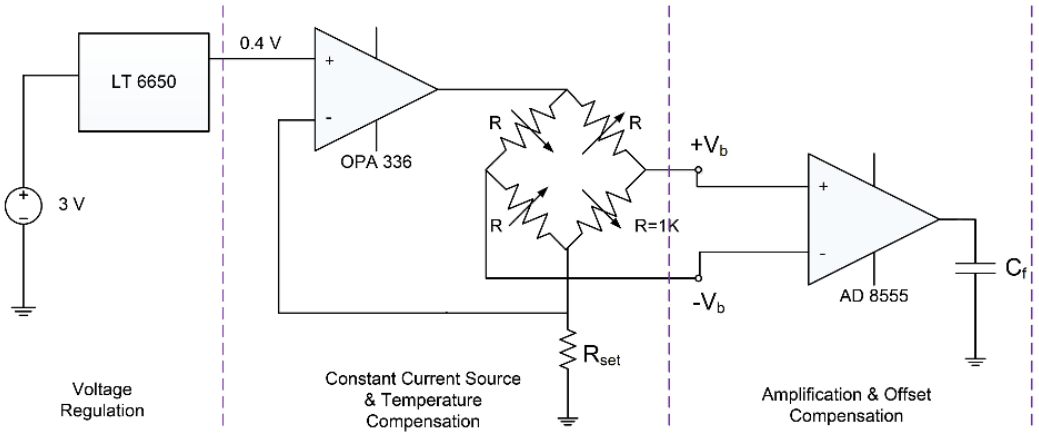


Figure 3.9: The simplified schematic of the read-out circuit (Karakas 2012)

3.2.6 Memory Card

The long term implantation of the sensor read-out system intends to store several biomedical signals for in vivo evaluation. For this purpose, a 32 GB microSDHC card is used to communicate to the MSP430 via SPI. A microSDHC card is the smallest, non volatile memory card with capacities up to 32 GB, developed for use in portable devices. The standard dimensions of a microSDHC card are declared to be 15 mm x 11 mm x 1.0 mm, by the SD Card Association (SDA). The memory card, implemented on the sensor read-out system, belongs to the Speed Class 4, thus minimum write speed is 4 MB/s.

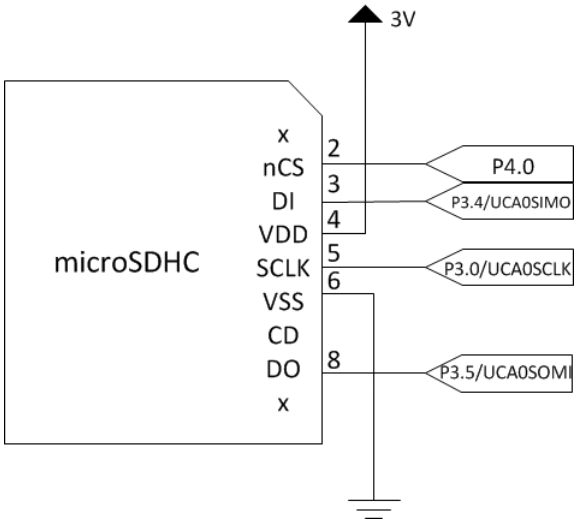


Figure 3.10: Connection between MSP430 and microSDHC

Figure 3.10 depicts the pin configuration of the microSDHC card for the SPI transmission. The microcontroller is configured to be the master device and initiates data transfer, sets the clock frequency, polarity and phase. The four logic signals that are used in the transmission, are: serial clock (SCLK), master output, slave input (SIMO), master input, slave output(SOMI), and an active low chip select line (nCS).

The sensor read-out system operates for ten minutes in every hour. The sufficient memory capacity for two weeks implant duration, can be calculated as following:

$$\textit{Capacity} = \textit{Implant duration in hours} \times \textit{Size of one measurement} \times \textit{Number of channels} \tag{3.3}$$

This calculation lead us to:

$$(14 \times 24)\,\text{hours} \times (250\,\text{sps} \times 600\,\text{sec} \times 2\,{}^{\text{Bytes}}/_{\text{sec}})^{1}/_{\text{hour}} \times 4\,\text{channels} = 403.2\,MB \tag{3.4}$$

Correspondingly, a 32 GB microSDHC card not only fulfills the required memory size but also allows longer recording time, higher ADS1298 sampling rates, and sample resolutions. As an example, the implemented software can be modified to 16 ksps and 3 Bytes/sample resolution which would result in approximately 26 GB.

3.2.7 Battery

One of the most severe restrictions on biomedical implants is power consumption. A reliable and medically safe source of energy is essential for implant operations. Supplying implanted medical devices with electric power for continuous operation is achieved either by equipping them with a battery or by using an external source such as wireless electromagnetic energy transfer. For the sensor read-out application, a rechargeable lithium-ion battery pack from Panasonic was chosen and implanted within the package.

A rechargeable lithium-ion battery consists of four layers: positive and negative electrodes, polymer electrolyte to transport lithium ions, and separator layer. During charging of lithium-ion batteries, lithium in the lithium oxide (the positive electrode), is ionized and reaches to the internal carbon layer (negative electrode).

The lithium-ion batteries have been preferred in medical implants for a wide variety of applications: pacemakers for cardiac arrhythmia, neurostimulators, cochlear implants and further devices (e.g. (Goto et al. 2001)). Moreover, lithium ion rechargeable batteries have the highest energy density (energy/volume) and specific energy (watt-hours per kilogram) compared to other types of rechargeable batteries such as Nickel Cadmium and Nickel Metal Hydride which approximately produce three times less voltage. With very low rate of self-discharge, the rechargeable battery pack that was hermetically sealed inside the device, sustained the read-out operation. Therefore, using an internal battery pack provided steady supply voltage, high energy density and a practical solution with a simple design that met the power requirements of the system.

On the other hand, a battery powered implant caused several design restrictions. Service life and dimensions of the battery pack defined the implant size and operating time in active mode. A wireless power transmission could have reduced the bulkiness and dimensions of the implant package, with longer measurement duration. However, internal power sources are not influenced by any RF interferences and more compliant with biomedical tissues.

For the implantation the two cylindrical, rechargeable lithium-ion batteries, surrounded by a soft battery pack were hard-wired before the device was hermetically sealed. The safety pack within the batteries provided strength and rigidity necessary to resist the abuses to which batteries may be subjected, without resulting in a risk of fire and injury to the biomedical tissue. The internal safety circuit of the battery pack contains thermistors and a controller IC that can prevent overcharging or overdischarging by measuring the voltage for each cell, and FET structures as switches. The charge current is controlled through the resistance value of the thermistors, which is measured during the charging process. The technical features of the battery pack are given as:

- Nominal voltage 3.7 V
- Nominal capacity 5800 mAh
- Standard charge and discharge current 540 mA
- Weight 90 g, dimensions 42 mm x 67 mm

Figure 3.11 represents the placement of the battery pack under the metal lid. In the implanted device, battery terminals are connected to the Pt/Au tracks on the alumina substrate.

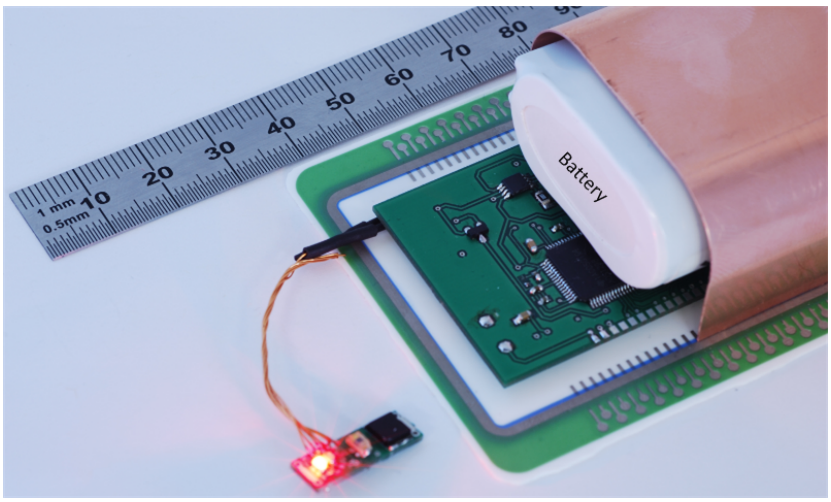


Figure 3.11: The battery pack inside the package

3.3 Power Management

Internal batteries power all the connected sensors and the PCB components. Short-circuiting two pins of the implant package connects the positive terminal of the battery pack to the input of the two voltage regulators on the PCB. The two REG102 linear regulators from Texas Instruments supply 3 V to the system. The MSP430 receives its supply voltage from the regulator that is enabled by the short-circuit of the on/off switch, therefore the microcontroller is initialized as soon as the battery pack is connected. However, all the other components of the PCB are supplied with the second regulator, enabled by the PWR_EN signal from the microcontroller, as shown in Figure 3.12.

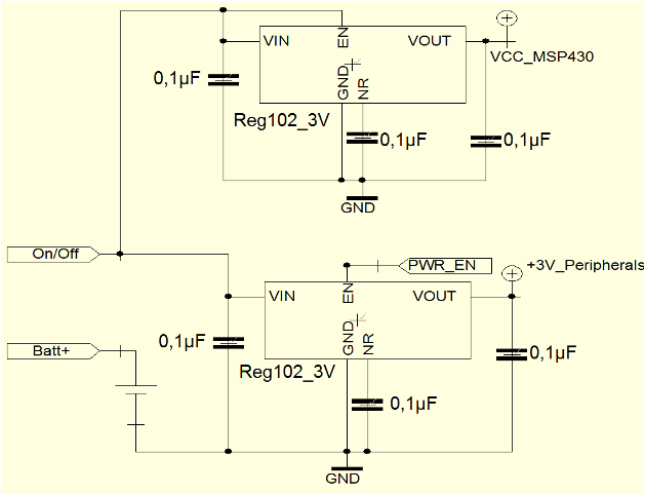


Figure 3.12: Voltage Regulator Configuration

The power consumption of the circuit needed to be optimized to fulfill 14 days of implant duration. Consequently, the current consumption of the read-out system was measured in the LabView program from National Instruments. The voltage over a 1.4 Ω resistor, which was connected between a 3.7 V supply and the read-out system, was recorded. Figure 3.13 displays the supply current over time. The graph shows the low power mode and active mode current consumptions, each in 60 seconds, without the connected sensors.

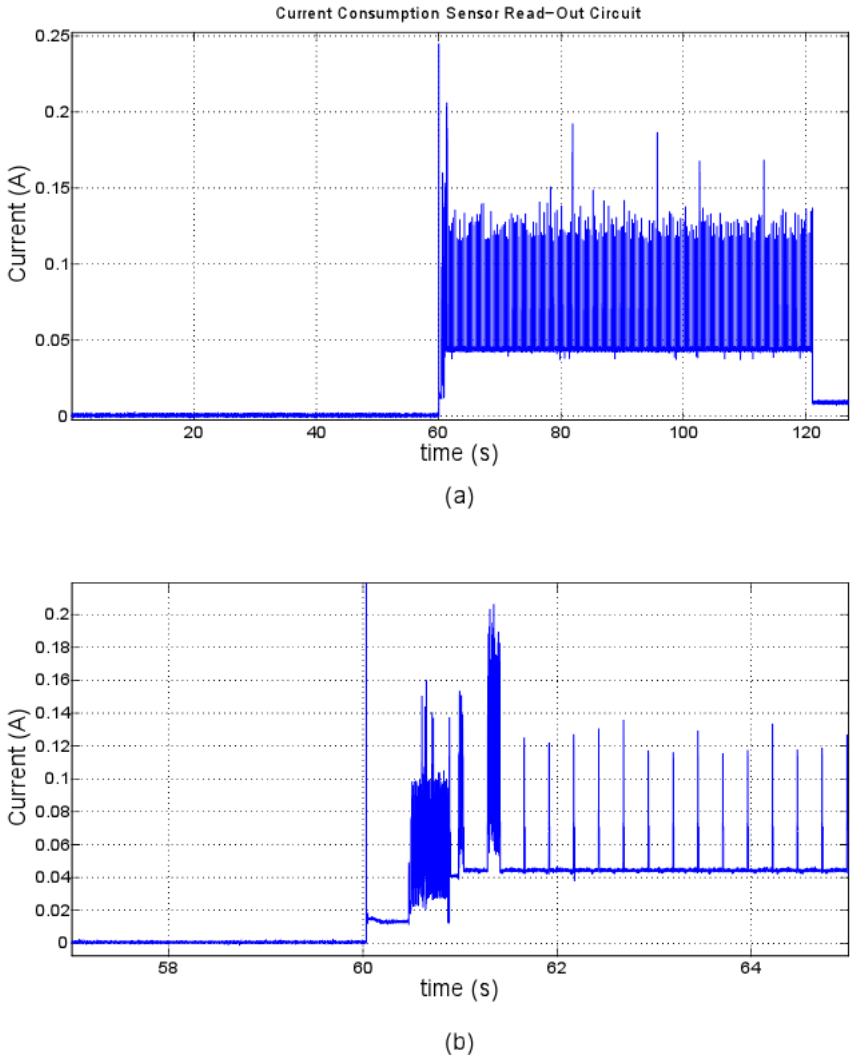


Figure 3.13: a) Current consumption of the sensor read-out sytem with 50 % duty cyle. b) Closer view of the active mode operation.

The mean values of the consumed current in the low power mode and the active mode are 614 μA and 45.4 mA, respectively. The periodic peaks in the Figure 3.13.b are due to the power consumption of the memory card-writing operation. A write operation takes approximately 8 milliseconds with the mean value of the current during a write operation, as 75 mA. When 20 mA contribution from the two con-

nected PPG sensors to the active mode supply current is taken into account, the battery pack with 5800 mAh nominal capacity, allows the system to operate at most 20 % of the implant duration. Accordingly, one period of the system is optimized as follows: The sensor read-out system operates for 10 minutes, and becomes idle for 50 minutes when the microcontroller enters to a low power mode.

3.4 PCB Design

All the electronic components, performing the sensor read-out functions, are mounted on a printed circuit board (PCB) inside the package. The dimensions of the implemented battery were the main parameter in defining the dimensions of the PCB and the implant package. The double-sided, 36 mm x 65 mm PCB layout was generated manually, using TARGET 3001! CAD program, and is composed of 2 layers. The constraints by the pin widths of the mounted components and overall dimensions of the PCB, lead to the design specifications below:

- minimum copper track width: 0.2 mm
- minimum distance between signal tracks: 0.125 mm,
- FR-4 (Woven glass and epoxy) laminated surface
- 99 drill holes for via connections
- smallest drill hole diameter: 0.2 mm
- smallest drill hole spacing: 0.125 mm

The board was manufactured by a PCB supplier company called Beta LAYOUT, according to the Gerber data extracted from the layout files (see Appendix A.3 for the PCB layout and Appendix A.2 for the schematic of the system). Figure 3.14 shows the manufactured PCB.

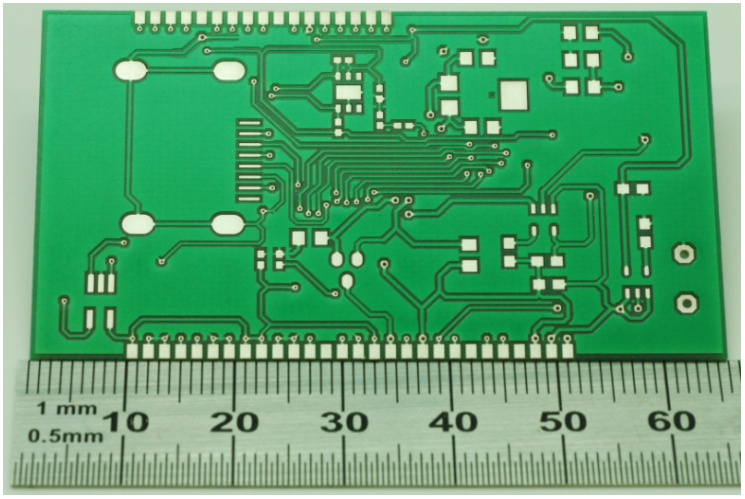


Figure 3.14: Top side of the PCB that contains microSDHC card, 32.768 kHz crystal, SHT25 temperature and humidity sensor and voltage regulators.

All the components that are involved in the PCB layout design, were mounted manually by soldering at the workstation of the Labaratory for Micro-optics. The Sn-Pb alloy ($Sn_{60}Pb_{40}$) was used at 320°C. Figure 3.15 shows the partially assembled PCB.

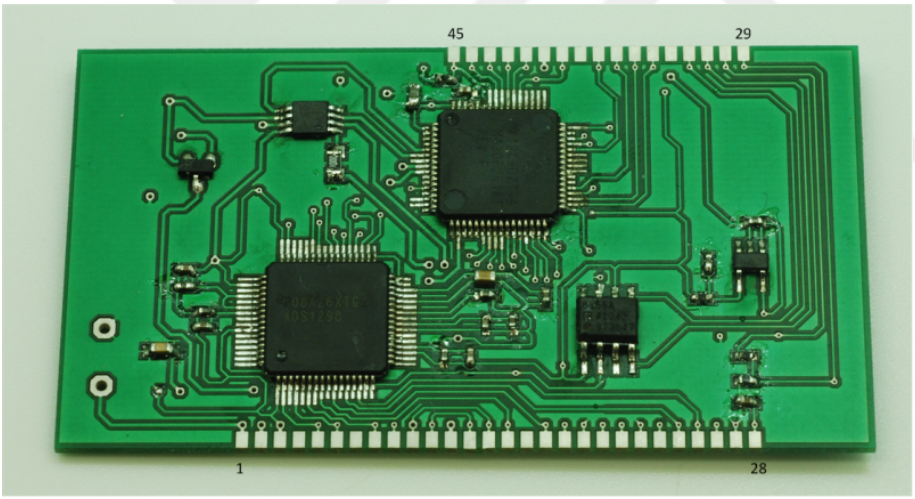


Figure 3.15: Bottom side of the PCB that contains MSP430F2618, ADS1298, AD8555, LT6650 and the passive components.

Forty-five input and output pins are located in both sides of the PCB. The purposes of these pins are supplying voltage to the PPG and the tip catheter sensors, the JTAG interface for debugging the microcontroller, power switch of the system, acquisition of bio-medical signals and a SPI port interfacing the microcontroller. The PCB design supports interconnection of up to five PPG sensors and regulation of two connected LEDs. The PCB pins are connected to the Pt/Au tracks printed

on the alumina substrate of the implant package, in order to transport the connections to outside of the package. Figure 3.11 shows the numbering of the pins with first and last number displayed on each edge. The pin configuration is included in Appendix A.4

4 Software

This chapter, focuses on the implementation of the developed software for the sensor read-out system. As a part of this master project, a software was designed and implemented in the read-out system, in order to organize the operations and communications of the used components. The performed algorithms are explained in details. Firstly, the system operation is introduced in brief. The content of the software design is then described through the source code.

4.1 General Description

The software design of the read-out system is constructed in the Code Composer Studio (CCStudio) Integrated Development Environment (IDE) v4 from Texas Instruments, by using the C programming language. The CCStudio comprises an editor, compiler, linker, debugger, real-time operating system and many other features that are necessary for developing software in embedded systems. Programming the MSP430 is done by using the conventional 4-wire JTAG interface, as mentioned in section 3.2.1 of the previous chapter. Displaying and modifying the registers of the microcontroller is possible by the JTAG interface of the IDE. The microcontroller program runs in real-time under the control of the debugger of the CCStudio. The flash emulation tool connects the PCB with the computer. The debugged program can be stopped, started, and single-stepped within the IDE.

The aim of the sensor read-out system is the acquisition and storage of the biomedical signals on a periodic basis and to sustain these operations during the implant duration. The implemented source code is designed to provide this functionality by interfacing several peripherals to the microcontroller, using timer counters and different communication protocols (see Appendix A.1 for the complete source code). Figure 4.1 shows the flow diagram of the main() function that starts up the read-out system and organizes its operation. The main() function is called immediately after the system is powered.

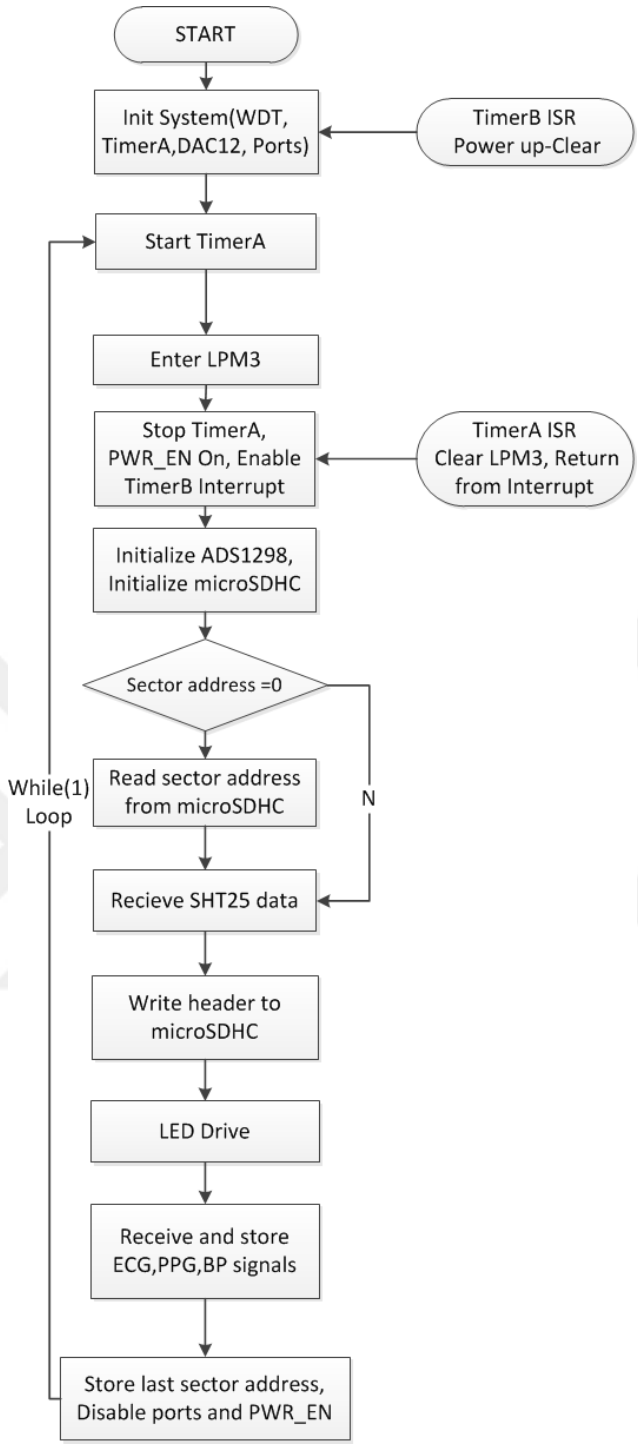


Figure 4.1: Basic flowchart of the source code

At first, the timer modules of the MSP430 are configured. The watchdog timer of the microcontroller restarts the system if a software problem occurs. Since this fail-safe operation is obtained by another timer setup, the watchdog function is stopped through the watchdog timer register. The next configuration defines the operation

of the basic clock module. The MSP430 offers three clock sources:

- Low-frequency/high-frequency oscillator (LFXT1CLK) that can be obtained by 32768 Hz watch crystals or 450-kHz to 8-MHz standard crystals
- High-frequency oscillator (XT2CLK) that can be obtained by standard crystals, resonators, or external clock sources in the 450 kHz to 8 MHz range
- Internal digitally controlled oscillator (DCOCLK) that is a built-in RC-type oscillator

The basic clock module provides the following software selectable, internal clock signals, that can be divided by 1, 2, 4, or 8.

- Auxiliary clock (ACLK) can be selected by the peripherals. It is distributed from LFXT1CLK.
- Subsystem master clock (SMCLK) can be supplied by any of the clock sources, for the peripherals.
- Master clock (MCLK) is used by the CPU and system. It can be derived from any of the clock sources. Typically both SMCLK and MCLK run at the same frequency.

The software design uses DCO at 16 MHz, which sources the MCLK and the CPU. The Timer_A module regulates the low power mode operation. Frequency of the Timer_A is taken from the ACLK that is generated by an 32.768 kHz external crystal on the PCB. The configuration of the clock and timer parameters, is followed by enabling the Timer_A interrupts and assigning the DAC control register value. As it is described in section 3.3 of the previous chapter, all the peripheral components of the PCB are supplied by a voltage regulator that is enabled by a microcontroller signal. The MSP430 outputs this enable signal at pin port 1.1. Thus, the port direction and function are defined as output.

The eternal while(1) loop runs the sensor read-out operation. This loop arranges the periodical behavior of the microcontroller while entering and waking up from the low power mode for saving power. The first step within the eternal loop is to start the Timer_A and to enter the low power mode LPM3. In LPM3, the central processing unit (CPU) and all the clock sources except of ACLK are disabled. The previously enabled Timer_A interrupts call the interrupt service routine (ISR). Through the ISR, the LPM3 is sustained.

Upon the return from the Timer_A ISR with waking up from LPM3, the microcontroller enables all the peripheral components by its PWR_EN output signal. For securing the system operation against software failures, the Timer_B module registers are assigned to generate an interrupt that would trigger a power up-clear (PUC) sequence. After the fail-safe operation is configured, the ADS1298 and the microSDHC card are initialized. The flow of the active mode operation continues

with reading the current sector address of the microSDHC card to be written. The relative humidity and temperature information is collected from the SHT25 sensor by the I^2C protocol. A header sector, which is constituted of the SHT25 data and DAC output values, is written in the microSDHC card, to indicate the beginning of the data acquisition. Afterwards, the biomedical signals are gathered from the channels of the ADS1298 and sampled values are written to the microSDHC card, inside a for-loop. The writing speed of the microSDHC card is obtained by collecting a timestamp in every cycle of the for-loop. The data acquisition and storage is based on the SPI communication of the microcontroller with the ADS1298 and the microSDHC card. At the end of the for loop, the last written sector address is stored in a specific location of the card, in order to be read in the next active mode operation. The duty cycle in the active mode ends by returning to the eternal loop, after halting the Timer_B and disabling all the used microcontroller ports along with the peripheral components.

4.2 Interrupt Handling

The software design initiates two interrupts: one for waking the MSP430 from the low power mode and the other for a fail-safe operation. After a peripheral generates an interrupt, when the related bits and general interrupt enable (GIE) bits are set, the microcontroller completes its current instruction, stores the content of the status register and address of the next location to resume its operation later and calls an interrupt service routine (ISR). The MSP430, fetches the next instruction after an interrupt, from a vector table at the program memory (addresses 0xFFC0–0xFFFF), in which every ISR vector has a predefined address. Before the program counter is loaded with the content of the interrupt vector, the status register (SR) is cleared. This terminates the existing low-power mode. Upon return from an interrupt, the SR and PC pops back from the stack, restoring the previous low-power mode.

In this section, periodic main loop processing with the low-power mode and the fail-safe operation will be explained in the context of the interrupt service routines.

4.2.1 Timer_A Interrupt Service Routine

In the source code, the microcontroller enters to the low power mode at the beginning of the eternal loop. The sensor read-out system sustains at an idle state during this mode. The duration of the low power mode depends on the Timer_A control register (TACTL) settings and the ISR operation. A 32 kHz watch crystal, as the ACLK, is associated with the timer. Setting the ID_3 bit of TACTL register divides the timer frequency by 8, and the MC2 bit selects the continous mode operation. Therefore, the timer/counter register (TAR) counts from 0 to 0FFFFh, at which point it overflows and restarts counting from 0. With $\frac{32}{8}$ kHz clock frequency, 65,536 counts give 16 seconds as the timer period. After the _bis_SR_register(LPM3_bits + GIE) function which puts the microcontroller into LPM3 and sets the GIE bit

of the SR, the Timer_A interrupt flag (TAIFG) will be raised upon an overflow at the counter. At this point, the current SR is stacked and is cleared for the interrupt service routine, including the low-power bits (Davies 2008). Therefore, the low power mode is suspended and will be re-entered at the end of the ISR.

Figure 4.2 shows the Timer_A ISR operation. The line that starts with the #pragma expression, at the main.c file, associates the main() function with the Timer_A interrupt vector (TAIV). The ISR needs to read TAIV to acknowledge the interrupt, even if only one source has been enabled and there is no other need to read the vector (Davies 2008). The continuous mode operation of the Timer_A enables interrupts on TAIFG when the TAR counter returns to 0, in this case at every 16 seconds, as mentioned above. For 50 minutes low power mode, an if-statement within the TAIV switch, checks whether TAIFG is raised 185 times. When the low power mode duration ends, the ___bic_SR_register_on_exit(LPM3_bits) function clears the low power mode bits in the SR, and the microcontroller returns to the active mode from the interrupt.

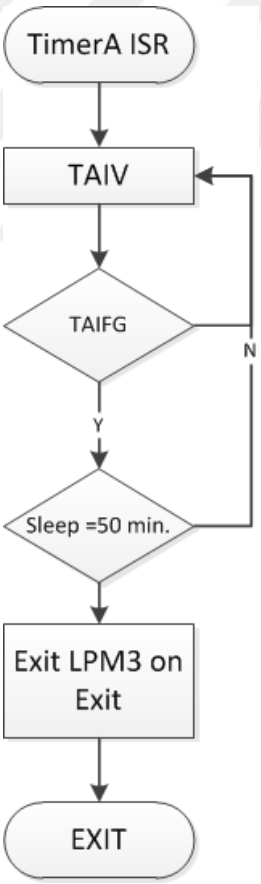


Figure 4.2: Flowchart of the Timer_A ISR

4.2.2 Timer_B Interrupt Service Routine

The microcontroller must operate accurately during the in-vitro measurement. For the reason that the implanted system can not be maintained manually, a fail-safe algorithm must reset the system when a failure occurs due to a hardware or timing problem. As seen in Figure 4.1, the microcontroller starts data acquisition and storage after several function blocks. Any system failures that can occur before receiving the biomedical signals, are compensated by the Timer_B ISR. The microcontroller generates a software reset which is called power up-clear (PUC) when the Timer_B interrupt vector (TBIFG) is set due to overflow. The Timer_B period is set to 16 seconds with identical clock configurations to the Timer_A module. Since the normal operation of the MSP430 until it starts data acquisition is faster than the Timer_B interrupt, an overflow occurs only in the presence of a failure.

Figure 4.3 shows the Timer_B ISR. The PUC is generated by writing a security key violation in the watchdog timer control register. Once this step is run by the microcontroller, the SR is reset, and the PC is loaded with the reset vector which provides the address of the first instruction of the main function.

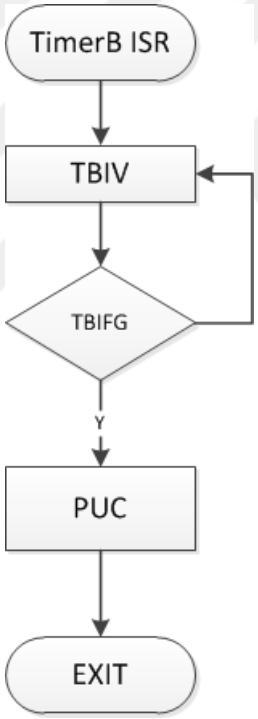


Figure 4.3: Flowchart of the Timer_B ISR

The Timer_B ISR secures the system by halting and restarting the timer in every cycle of the data acquisition and storage. Meanwhile, reading the Timer_B counter collects timestamps to observe even very short variations in reading and writing time of the biomedical signals. This prevents errors in the calculation of the Pulse Transit Time (PTT).

4.3 I2C Communication with the SHT25 Sensor

The sensor read-out system is hermetically sealed to avoid any gas or fluid leakage during the in-vivo measurements. Considering that the system is battery powered, a hermetical seal is crucial also for the biological safety of the implant. The SHT25 digital humidity and temperature sensor is implemented inside the implanted package to investigate the hermeticity of the environment. The USCI_B has a state machine in hardware that generates the start and stop conditions for data transfer, when required.

The I²C bus uses two, bidirectional lines: serial data (SDA) and serial clock (SCL) that are connected to the USCI_B1 port of the microcontroller. The data transfer between the master and slave device consists of a sequence of bytes, with the msb first and is clocked by the master. After successful reception, the recipient writes an acknowledgment bit of 0 to SDA. Each transmission sequence begins while SCL is high and SDA is in a high to low transition. Similarly, data transfers end by a rising edge on SDA while SCL is high. These operations are called as start and stop conditions, respectively. The 7-bit I²C device address ('1000'000' for the SHT25) selects the slave that should respond. A data transfer sequence starts with one byte that consists of the slave address and a final bit ($R/\overline{W}$) which defines whether a read or write operation will be executed. Figure 4.4 presents the I²C transfer sequence.

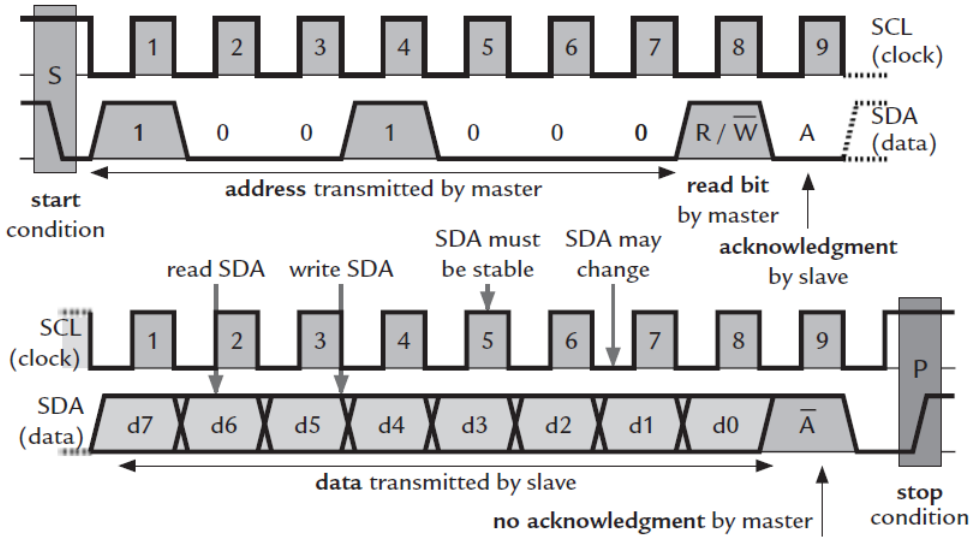


Figure 4.4: I²C transfer sequence (Davies 2008)

The resolution of the measured data is set to 12-bit for relative humidity and 14-bit for temperature, while data is transferred in two byte packages. For implementing the I²C interface in the software, the code libraries from an application note (Kretzschmar and Hernitscheck 2007) by Texas Instruments, are modified according to the the requirements of the system. The following steps are followed for the temperature and humidity measurements, separately:

1. The USCI_B module is initialized for master-transmit operation by TI_USCI _I2C_DMA_transmitinit() function. The address of the slave is attended and the baud rate is set to 0.8 MHz.

2. After the slave address is acknowledged, the MSP430 sends the 1-byte measurement command. The SHT25 pulls down the SCL line while measuring, forcing the microcontroller to wait. The TI_USCI_I2C_notready() function checks the USCI_B registers and data transfer is halted until the bus is free. Transmission is started by setting the UCTXSTT bit within the TI_USCI_I2C_transmit() function. The USCI vector for transmission is used for the transmit flag UCB1TXIFG, that is generated by the internal state machine of USCI_B. When the transmission is completed, the associated ISR issues a stop bit.

3. The TI_USCI_I2C_receiveinit() function initializes the USCI_B module for master-receive operation. The same slave address and baud rate from the first step, are set for the receive operation.

4. The 2-byte measurement data and 1-byte checksum are received from the SHT25 by USCI_B module. When a received byte is copied into the receive buffer (UCB1RXBUF), the receive flag (UCB0RXIFG) is raised. The associated ISR with the receive vector (USCIAB0RX) sends a stop condition if the not-acknowledgment flag (UCNACKIFG) is raised. The microcontroller generates a start condition by the UCTXSTT bit, and waits until this bit is cleared. Then a stop condition is generated until receiving of one byte is completed.

4.4 LED Driver

The two digital-to-analog converters of the DAC12 module drive the LEDs of the connected PPG sensors, as it is described in the section 3.2.4 of the previous chapter. The linear proportion of the LED current to the DAC output voltage is obtained by keeping the DAC output below 0.2 V. The 12-bit DAC output is calculated according to the DAC12_xDAT data register (see Eq. (3.2)). Figure 4.5 shows the flow chart for setting the output voltage. At the beginning of each duty-cycle the intensities of the LEDs of the PPG-Sensor are regulated, so that an ideal PPG-Signal level is accomplished. The set current is stored in the header byte on the memory card. The procedure for that is as follows: Attending 327 as DAC12x_DAT register value results to 0.2 V as the maximum voltage within the linear range (See Fig.3.7). Afterwards, the ADS1298 reads the input signal from the channels that are connected to the PPG sensors. The DAC outputs are re-configured until, the amplitudes of the PPG signals are below 2 V, for an efficient measurement.

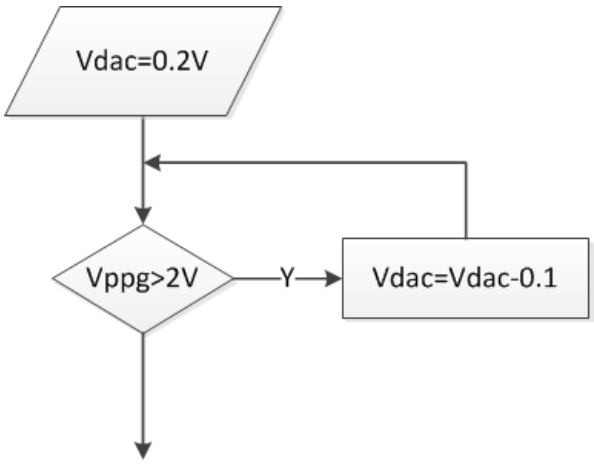


Figure 4.5: Setting the DAC output voltage

4.5 Data Acquisition and Storage

The sensor read-out system is designed to receive and store in-vivo signals. Figure 4.6 shows this sequence within the main function. The ADS1298 samples the biomedical signals that are received from its channels and stored in 512-byte memory locations at the microSDHC card. The initialization and the SPI interface of the ADS1298 and the microSDHC card are evaluated in the following subsections. The associated files ads1298.c, mmc.c and hal_MMC_hardware_board.c with corresponding header files (see Appendix A.1) contain the functions and register settings that are necessary to perform initialization of the devices, SPI interface, data sampling and storage.

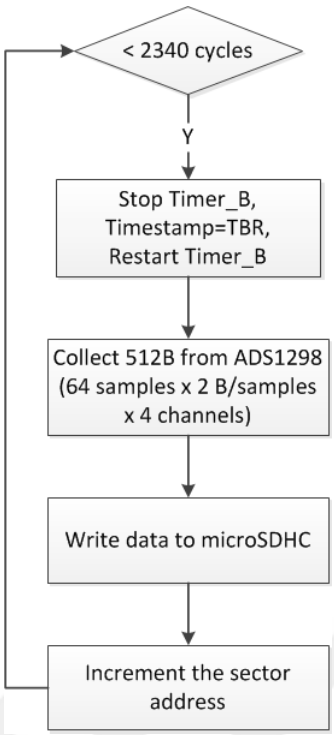


Figure 4.6: Data acquisition and storage sequence

4.5.1 Analog-to-Digital Conversion

The ADS1298 is initialized after the microcontroller wakes up from LPM3. Power-up sequence of the device is as follows:

1. The CLKSEL pin of the ADS1298 is set high by the microcontroller, since the ADS1298 uses its 2.048 MHz internal clock. Additionally, not-reset and not-power-down pins are pulled high.

2. The 3rd port of the MSP430 is assigned and the not-chip-select pin is set low for SPI communication with the ADS1298. During the initialization, the start pin is kept low, halting data conversions.

3. The ADS1298 retrieves the data in two modes. The data can be read from channel continuously without sending opcodes by the Read-data-continuously (RDATAC) command. This default mode is stopped at the initialization by the Stop-continuous-read (SDATAC) command, for configuring the ADS1298 registers. The ADS1298 decodes several SPI commands to configure its operation.

4. The Register-read (RREG) command is issued by the ADS1298, to read the factory programmed ID control register at the address 00h.

The above steps are repeated until the ADS1298 outputs its identification bits at the ID control register. Then the ADS1298 registers are written to indicate the device operation. The device is configured to run in low power mode, that enables setting the data rate to 250 sps. The internal reference buffer is powered and the value is set to 2.4 V. The built-in PGA gain is set to 1 for the channels, that are connected to the sensors, and to 12 for the ECG channel. The input channels are configured as the normal electrode inputs. After the register configurations, the start pin is set to 1 to enable data retrieval and conversions.

As it is seen in Figure 4.6, a for-loop completes 2340 cycles of reading the sampled input signals from the ADS1298 and writing them into the memory card. The channel data is read by issuing the Read-data (RDATA) command inside the ads_sendrdata8() function. The ADS1298 outputs a data-ready ($\overline{\text{DRDY}}$) signal that goes low when new data is available at the DOUT pin. After the low transition of the $\overline{\text{DRDY}}$ signal, the opcode of the RDATA command is moved to the data transmit buffer of the USCI_B0 port of the MSP430. The ADS1298 latches the data in its DIN pin on the falling edge of SCLK that is set to 8 MHz for the SPI communication. The sampled data is outputted to the SOMI signal line, as twos complement and MSB first from each channel. The size of the output data is 216-byte, consisting of 3-byte sampled signal from 8 channels and 24 status bits. Figure 4.7 shows the SPI bus signals of the ADS1298. The ads_sendrdata8() function acquires the first 2-bytes of the samples, skips the LSBs and status bits. This increases the number of the stored sample in one cycle.

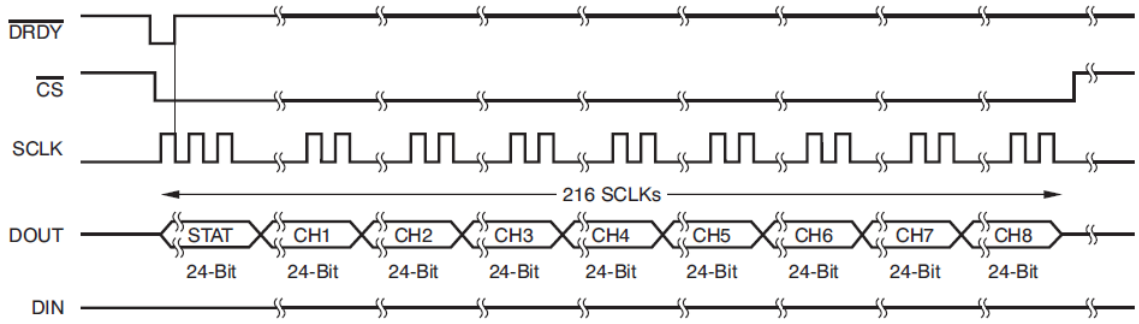


Figure 4.7: SPI bus data output (Davies 2008)

4.5.2 The Memory Card Operation

The microSDHC card is initialized before any writing attempt is taken in the main function. For SPI communication with the MSP430, the memory card uses a command set that is declared by the SD Card Association. The assertion of the CS signal puts the card in SPI mode. The connected card not only responds to every SPI command but also responds with data tokens to the sent data blocks during write operations. Figure 4.8 displays the SPI commands that are sent to the microSDHC card during the initialization by the mmcinit() function. The halSPIsetup()

function selects the USCI_A0 port of the MSP430 for transporting the SPI signals. Asserting CS to low, starts the SPI bus transaction. The SEND_OP_COND command rejects the cards that can not operate in the supply voltage range provided by the host device. The SEND_IF_COND and APP_CMD commands must be issued previously for initialization of a microSDHC card.

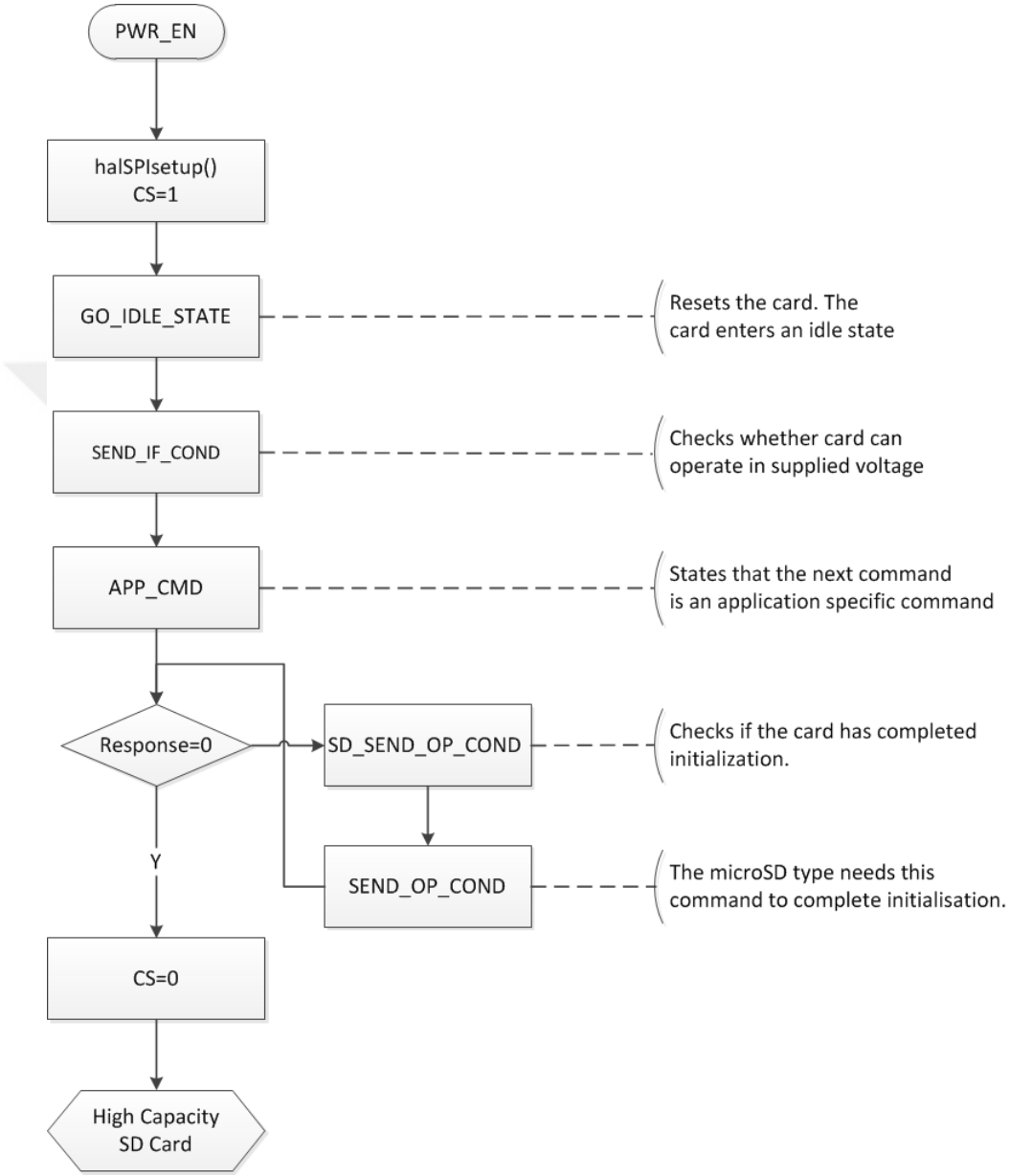


Figure 4.8: SPI mode initialization sequence

The microSDHC card receives and transmits the data as 512-byte blocks. This fixed 512-byte block length is called a sector. In the SPI mode, the mmcWriteBlock() function sends a single-block-write command, and waits for a response token from

the card. The data transmission starts upon the receipt of the start block token by the card. After transmission of every sector, the card sends a data-response token that indicates if any error has occurred during the operation. While the card is busy with writing the sector, it holds the SOMI line low. Similarly, the mmcReadBlock() function reads one sector sized data, after the memory card responds to the single-block-read command without any errors.

In the while(1) loop of the main() function, 64 sampled values are collected from the ADS1298 channels, the MSP430 prepares a buffer with 512 bytes of data and writes it into the sector of the card. This operation is repeated 2340 times in the main function. The number of the cycles is chosen to obtain 10 minutes of data acquisition and storage. The speed of the SPI communication between the microcontroller and the device defines the period of a cycle. Here, it might be suitable to emphasize that, the full potential of the SPI is restricted in the MSP430, as a consequence of only one shift register in the USART peripheral interface, The data transmission never occurs to be full duplex, which results in a slow data rate.

Figure 4.9 represents the organization of the acquired biological data within the microSDHC card. In one period of the active mode operation, 2340 sectors are written in the shown order.

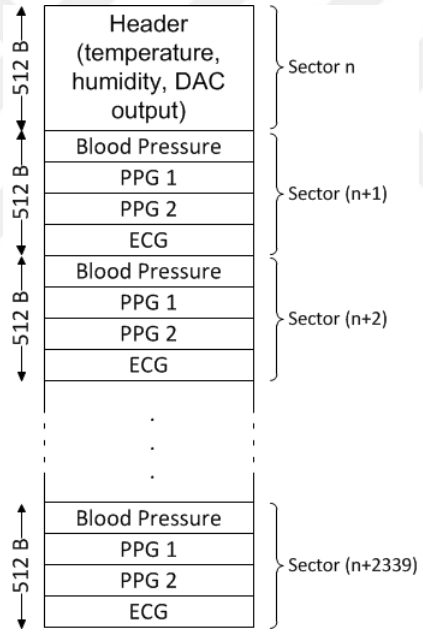


Figure 4.9: Organization of the acquired biological data within the microSDHC card

5 Design and Fabrication of the Hermetic Implant Package

This chapter describes the design and fabrication steps of the hermetic implant package of the sensor read-out system. The presented implant package is based on a concept that is adopted from the the Laboratory for Biomedical Microtechnology of the University of Freiburg, which also contributed to the fabrication process. The hermetic and biocompatible package was realized and implanted as a part of this master project and the OPT4LIFE project.

5.1 Concept

The sensor read-out system is designed to be implanted for acquisition of biomedical signals in an animal. The system contains several electronic components and conductor cables that require hermetic encapsulation against humidity and physiological fluids. The packaging materials must be biocompatible in means of nonallergenic, nonthrombogenic, nontoxic behavior. Therefore, a miniature hermetic package concept (Schuettler et al. 2010) is developed at the Laboratory for Biomedical Microtechnology in University of Freiburg, as a variation of a previously presented method (Donaldson 1976), is adopted to the read-out system.

As the implantable monitoring systems and biomedical devices have been researched and developed through the years, several materials and packaging technologies have accordingly been presented to provide biocompatibility and prevent the diffusion of substances into the devices and vice versa. The most common approach utilizes a hermetic enclosure that is coated with a biocompatible material. The hermetic enclosure protects the sensitive portion of the system from the body and the biocompatible coating is used to improve the interface of the implant with body (Potkay 2007). A hermetic package can be defined as one that prevents the diffusion of moisture and water vapor through its body, and can be formed of various materials such as metal, ceramic, or glasses. In this project, ceramic, which is an inert, insoluble material, was used in combination with a metal lid. In contrast to metal packages, ceramics have better transmission properties for alternating electromagnetic fields. This enables the use of RF-data-telemetry for the next version of the read-out circuit.

Several patterned laminates are co-fired together to form a ceramic package. A thin laminate sheet is prepared by screen-printing the layers that compose the ceramic substrate. As one of these layers, electrical interconnections through the

package must also conform to hermetic sealing. In ceramic packages, metal pins are embedded and brazed within the ceramic laminates (Maluf and Williams 2004). Accordingly, platinum-gold (Pt-Au) traces can be used in this design to establish the electrical connectivity, due to its good adhesion on the substrate. The hermetic sealing can be achieved by several techniques including metal or glass welding and glass soldering. However, these techniques require high sealing temperatures that would damage the electronic components of the read-out system. Therefore, the used design concept prefers a metal solder seal to enclose the implant package with a metal lid.

Another fundamental element of the implant packages is the encapsultant. It must cover the feedthroughs, the hermetic seal and the enclosure against corrosion during long term implantation. Additionally, any sharp edges of the implant packages are smoothed by coating the encapsultant. One of the widely used encapsultant material is silicone rubber due to its good water absorption and adhesion characteristics and toxicity records.

Eventually, the mentioned requirements of an implant package lead to employ the package design based on an alumina (Al_2O_3) substrate that carries the PCB, the screen printed Pt-Au tracks as feedthroughs and a metal lid. The PCB is fixed to the substrate and its 45 input/output pins are attached to the feedthroughs to establish electrical connectivity. The metal lid is soldered to a screen-printed metal layer on the substrate. The insulation of the Pt-Au and metal is obtained by a printed dielectric. As the finishing layer, glass is printed on the dielectric parts at the outside of the lid. The glass layer ensures a good adhesion between substrate and rubber and also protects the dielectric against moisture traveling through the rubber once the package is implanted (Schuettler et al. 2010). Figure 5.1 represents a cross-section of the applied packaging concept with an attached PPG sensor. In the real application, the sensor and the PCB pads are wired to the Pt/Au track and the battery pack resides above the PCB.

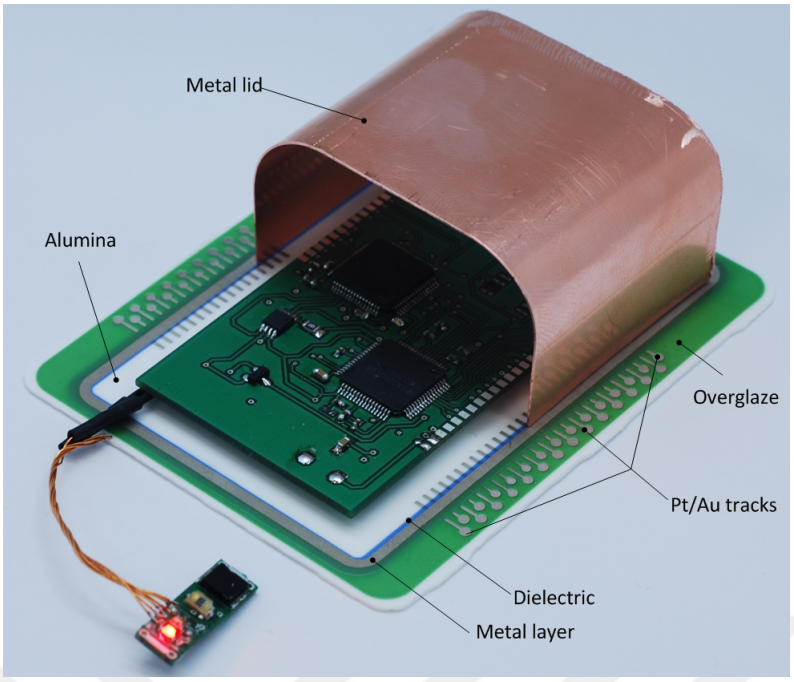


Figure 5.1: The electronic components are encapsulated by the metal lid, on the screen-printed substrate

5.2 Design and Fabrication

An alumina substrate is designed in L-Edit program from Tanner EDA Software Tools, and sent to a screen manufacturer in Gerber format. The 61×75 mm plate of 96 % alumina serves as the substrate which consists of four layers: the Pt-Au feedthroughs, the metal layer which provides soldering of the lid, the dielectric and the glass layer. It would be suitable to mention that the alumina is one of the most preferred materials in surgical implants because of its corrosion resistance, biocompability and high strength. For patterning feedthroughs with the diameter of 0.75 mm and width of 0.5 mm, the Pt-Au paste (type 5837-G by ESL Europe) was screen-printed by the Laboratory for Biomedical Microtechnology. Respectively, the dielectric paste (type 4913-G, ESL), the metal paste (Pt/Au, paste 5837-G, ESL) as the soldering base and the glass paste (G-486-1, ESL) were printed and fired. The copper layer has the track width of 2 mm, fitting to the dielectric layer in 3 mm width.

Figure 5.2 shows the entire fabrication procedure of the implant package concept from the Laboratory for Biomedical Microtechnology. After obtaining the electrical feedthroughs that reach inside the encapsulated area, the PCB was glued to the substrate in the manner that its contact pads are aligned with the Pt/Au tracks. The contact pads of the PCB were connected to these tracks by soldering thin copper wires. Following the attachment of the PCB and fixation of the battery pack above it, the medical grade condensation cure silicone rubber MED-1000 was coated on

both of them, to protect the electronics in case of a leakage through the hermetic seal. A copper lid that covers an 50×75 mm area, was designed and manufactured at the Laboratory for Micro-optics. Two holes with the diameter of 1 mm were drilled through the lid, to allow silicone rubber filling into the package under the lid. This filling was useful to ensure that there is no moist air encapsulated in the package. The assembled substrates were placed on a 120 °C hotplate and the lid was gently pushed to the screen printed metal ring, using a solder iron at 450 °C and the low melting point solder ($Sn_{99.3}Cu_{0.7}$) (Schuettler et al. 2010). The lid punctures were also sealed with a solder iron after drying the applied silicone rubber. The package was cleaned and rinsed in isopropyl-alcohol and deionised water before encapsulation.

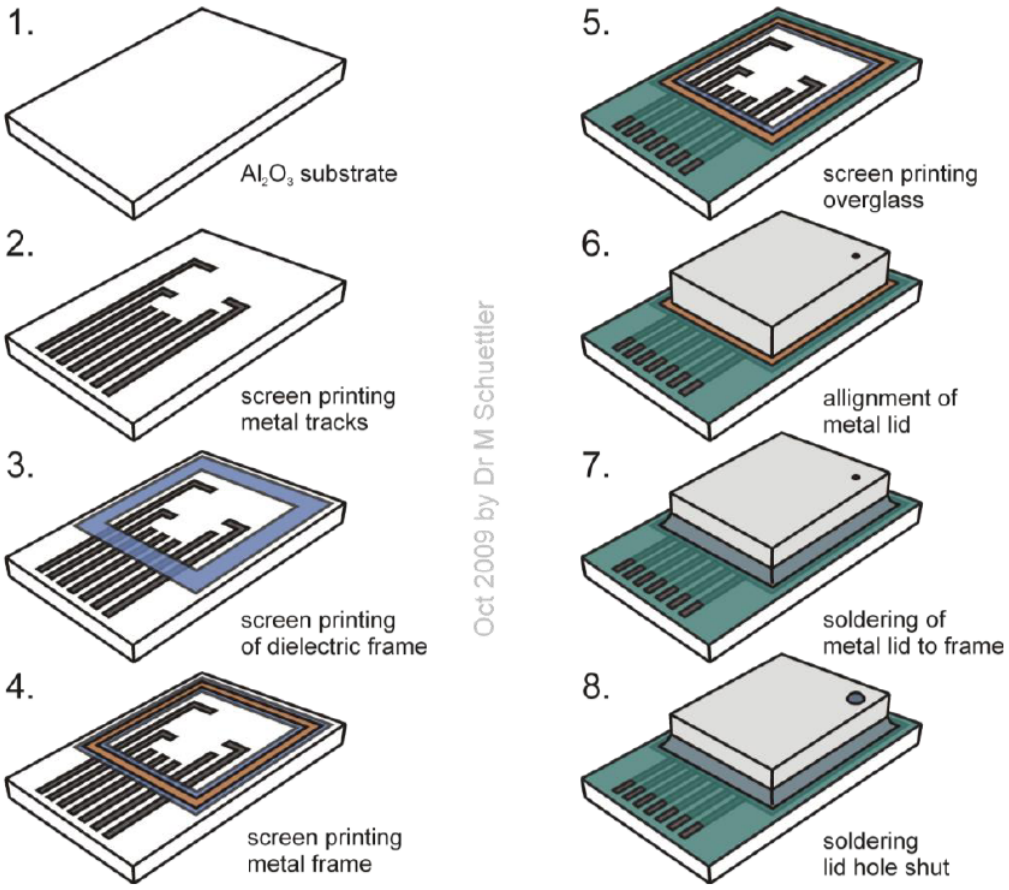


Figure 5.2: Fabrication steps of the implant package (Schuettler 2009)

The Pt/Au pins were connected to the implant sensors after lidding and sealing. Considering the implantation surgery, placing the PPG sensors requires flexibility and robustness against any detachment of the cables. Therefore a waterproof connector system was used for PPG sensors. The four-conductor unshielded cable with waterproof and biocompatible coating carried the PPG signals to the connector

which was soldered to the tracks of the implant package. The two gold-coated, copper foil electrodes were also soldered to the tracks for ECG measurements. The tip catheter was fixed next to the package lid. Assembling the package was finalized by coating it with the connected sensors entirely in MED-1000A to obtain biocompatibility. The package was then submersed into a water container for 12 hours to examine its hermeticity. The sensor read-out system within the package successfully operated after the water bath. Figure 5.3 shows the assembled hermetic package that is ready for implantation. The only remark about this figure is that the shown JTAG connection was used for debugging the microcontroller, and is detached before the implantation.

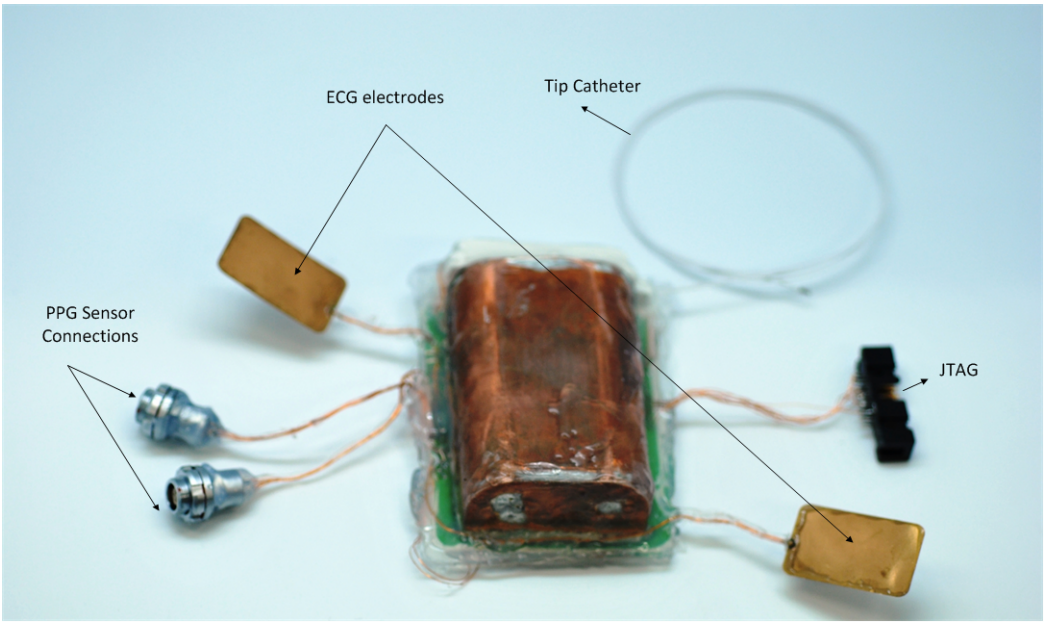


Figure 5.3: The assembled implant package, encapsulated in silicon rubber

Despite the fact that the used copper as the sealing and lidding material is not biocompatible, the silicon rubber encapsulation prevents emission of any toxic materials into the blood or body tissue. For human use, titanium can be used instead, since it is a biocompatible and inert metal. As an example, solder itself covered by silicone rubber has been successfully implanted as part of the Finetech Bladder Controller at least in a few thousand patients since the 1970s without any known adverse effects (Schuettler et al. 2010).

6 Measurements of Vital Parameters

In this chapter, the measurements that were performed with the sensor read-out system are explained and presented. During the development of the implant device, the vital parameters that are planned to be received from subcutaneous tissue, were initially measured in the laboratory environment. The long-term measurements are gathered from a 14-day implantation of the sensor read-out system in a sheep. Additionally, a subcutaneous tissue measurement on a pig provided PPG signals and comparison of ECG signals by copper electrodes in different sizes.

6.1 Laboratory Measurements

The laboratory measurements were performed using the PCB, which is described in chapter 3.4. The PPG sensors or two ECG electrodes, depending on the measurements, were connected to the channel inputs of the ADS1298 analog-to-digital converter on the PCB. Figure 6.1 shows a PPG setup. The MSP430 microcontroller on the PCB was programmed via the USB debugging interface that was connected to a computer. A DC power supply, corresponding to the battery pack of the implant package, sourced 3.7 V to the PCB. Once the LED current was driven by the microcontroller, the finger PPG signals were written to the microSDHC card on the PCB. The PPG sensors that were used in the laboratory tests, were identical to the implanted ones.

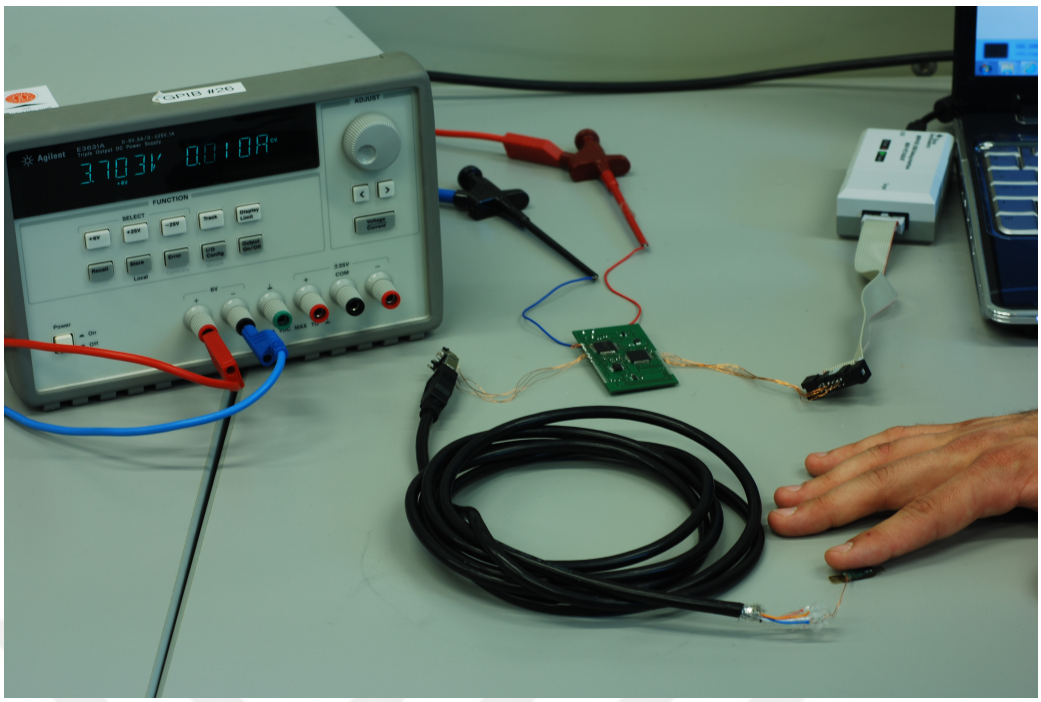


Figure 6.1: Setup for a finger PPG measurement.

The human PPG signal , shown in Figure 6.2, was acquired by the presented laboratory setup, from the index finger of a subject. The received PPG data was extracted from the microSDHC card and plotted in Matlab. For this measurement, the ADS1298 was configured to operate at 250 sps data sampling rate with no channel gain. The pulsatile (AC) component of the PPG signal can be observed in Figure 6.2, with its its fundamental frequency around 1 Hz. The slow variation in the PPG baseline was caused by the respiratory-related fluctuations and motion artifacts.

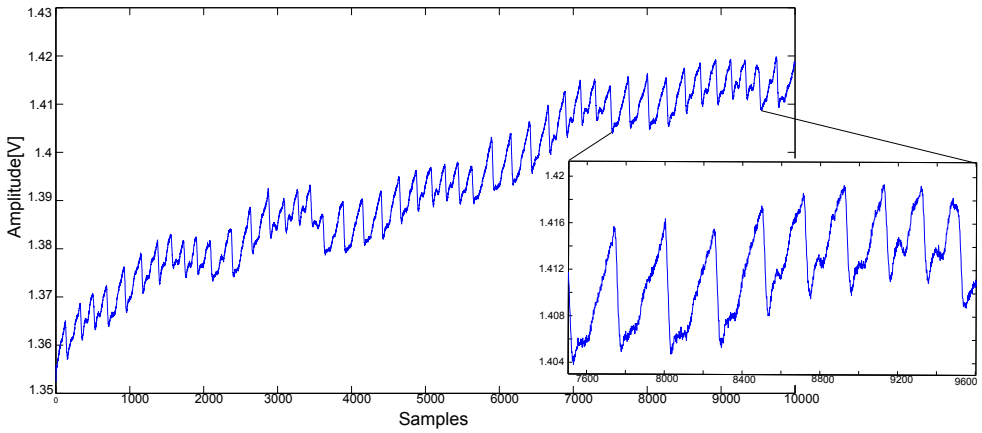


Figure 6.2: The PPG waveform.

The raw signal was processed by a finite impulse response (FIR) low-pass filter

with 20 Hz cut-off frequency, to remove the motion artifacts and baseline wandering. A Hanning window was used in the filter. The Figure 6.3 shows a portion of the filtered and inverted PPG signal. In the displayed graph, the primary peak, known as the systolic rising edge, and the dicrotic notch that is caused by the closure of aortic valve, identify the results as a PPG waveform.

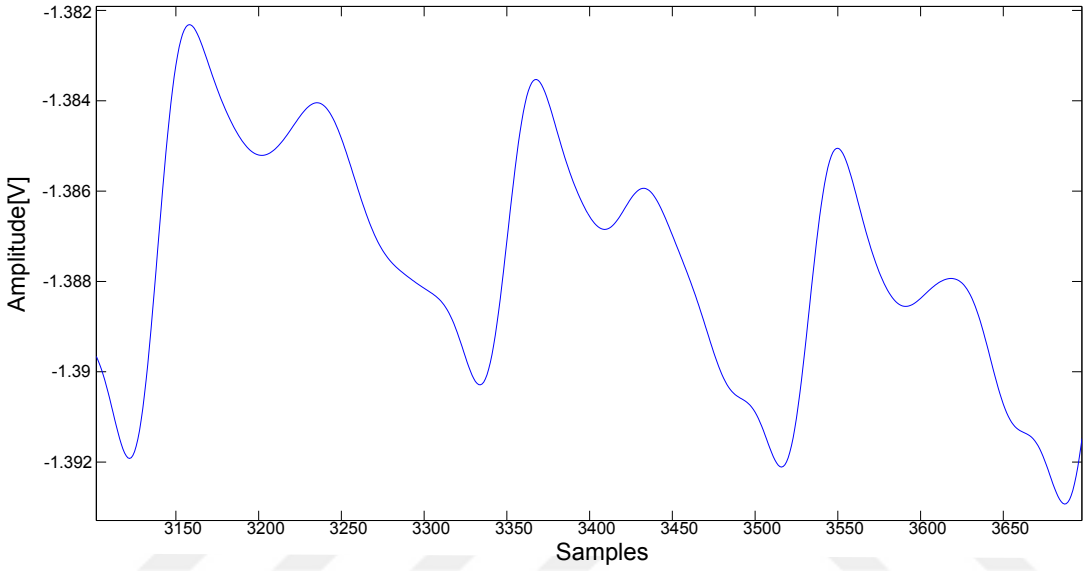


Figure 6.3: The processed PPG waveform with clear dicrotic notch.

The ECG measurements were held, using the mentioned setup with two dry electrodes, placed on each arm of a human subject. The ADS1298 sampled the received signal at the rate of 500 sps, with the channel gain of 12. The signal is filtered with a 50 Hz low-pass filter to eliminate power line interferences. The obtained ECG signal, with distinct QRS complex waves, is shown in Figure 6.4. The amplitude and duration of the QRS complex meet the characteristics of a typical cardiac cycle. The displayed R-R interval is calculated to be 1.112 seconds, which complies with the resting heart rate of humans. The distortions in the shown ECG waveform show the difficulty in stabilizing the ECG pattern, due to motion artifacts.

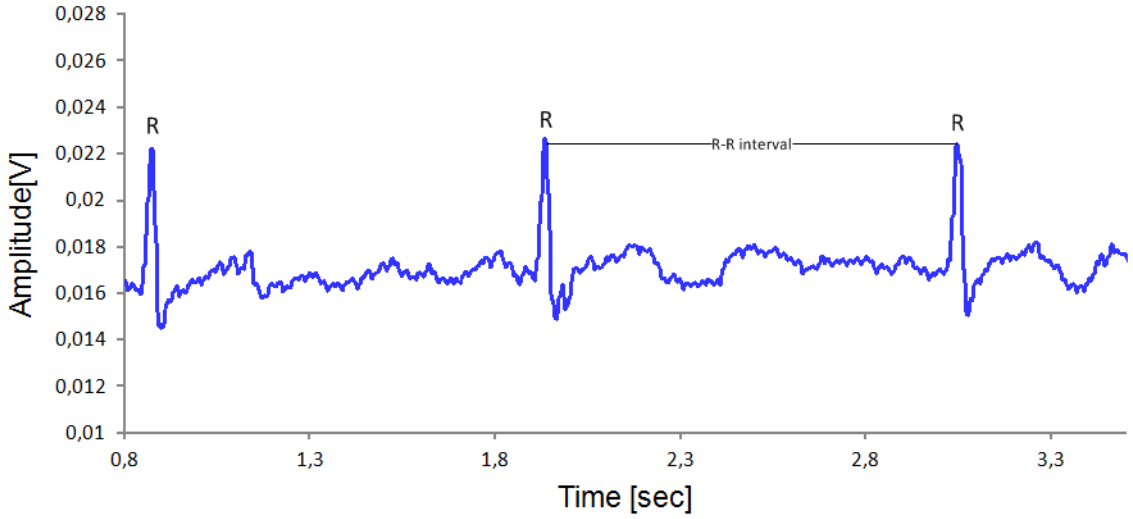


Figure 6.4: ECG measured from the body surface

As it is presented in the above measurement results, the sensor read-out system is able to acquire and store the PPG and ECG signals, in means of software and hardware design. The accuracy of the ECG signal is sufficient as long as the R peaks are obtained, since the PTT can be calculated in presence of these peaks and a PPG signal from the same time interval.

6.2 Long-term In-Vivo Investigation

To examine the functioning and capabilities of the sensor read-out system in an organism, a 14-day in-vivo measurement with a domestic sheep was done. This long-term measurement operation was conducted under the approval of the ethical committee of the University Medical Center (the registration code: 35-9185.81/G-10/67). The implant surgery was performed in the University Medical Center Freiburg, by a veterinary surgeon. For sterilization, the implant package encapsulated in silicon-rubber, and the cables to be implanted were entirely immersed in an alcohol solution (70 % ethanol, 30 % DI-water) until their implantation. The animal was intravenously put to sleep with an anaesthetic for the operation. During the surgery, the animal under anesthesia, was connected to a respirator.

The sensor read-out system was implanted into the neck of the animal with approximately 7 cm of incision. The two ECG electrodes that were connected to the sensor read-out system were also subcutaneously placed. The two PPG sensors were attached on different muscle tissues, with smaller incisions. The cables that are connecting the PPG sensor and the read-out system were placed under the skin, as well. Unfortunately, the tip catheter could not be applied on the artery of the animal due to the deficiency of the surgery duration. As a result, measurement of the blood pressure, that would be used as reference for PTT calibration, was not

performed. Additionally, it was recognized that using shorter cables to connect the PPG sensors to the implant package, would be sufficient and less invasive.

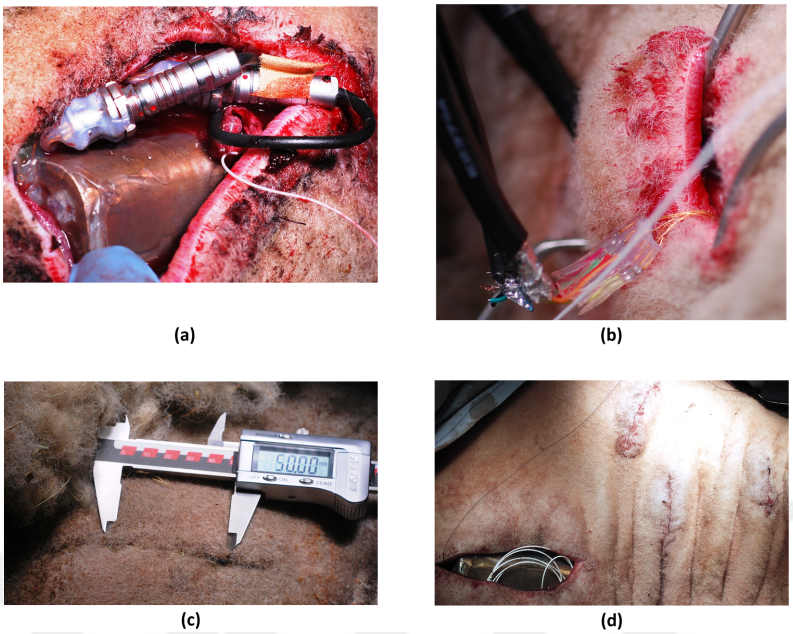


Figure 6.5: Implantation of the sensor read-out system (a) The package was implanted on the neck of the animal (b) The PPG sensors were implanted to the muscle tissues (c) Implant incision after 2 weeks (d) Positioning of the implant package and two PPG sensors on the tissue. The fourth incision carried the cable connections.

After the surgery, the subject animal was brought to an animal shelter at open air. The 14-day implant measurements were made while the subject was resting comfortably and breathing at a normal tidal rate.

6.2.1 Evaluation of the Implant Package

In order to evaluate the measurements, the implanted system was removed from the subject animal after 14-days. It was observed that the implant did not exhibit a significant tissue reaction. As seen in Figure 6.5 (c), the wounds healed well within two weeks. The animal with the implanted device behaved normal, did not need pain-medication and did not show any signs of obstruction.The only distinguished tissue response was the swelling that was formed around the implant package.

After the removal, the implant package, as it is seen in Figure 6.6, was examined. The silicone rubber encapsulation was found to have remained completely stuck to the system, including the alumina, the solder seal, the copper lid and cables. The copper lid and the remaining ECG electrode became slightly darker but appear unaffected. Some corrosion of the solder seal, at the joints of the lid and the substrate, was seen, however the silicone rubber encapsultant prevented early malfunction and

the escape of the metal ions into the body. This corrosion is suggested to be activated by the used MED-1000 silicone, that produces acetic acid during curing. It can be prevented by choosing another type of silicon rubber (Dow Corning 3140) as the encapsultant which adheres more strongly to metal surfaces.

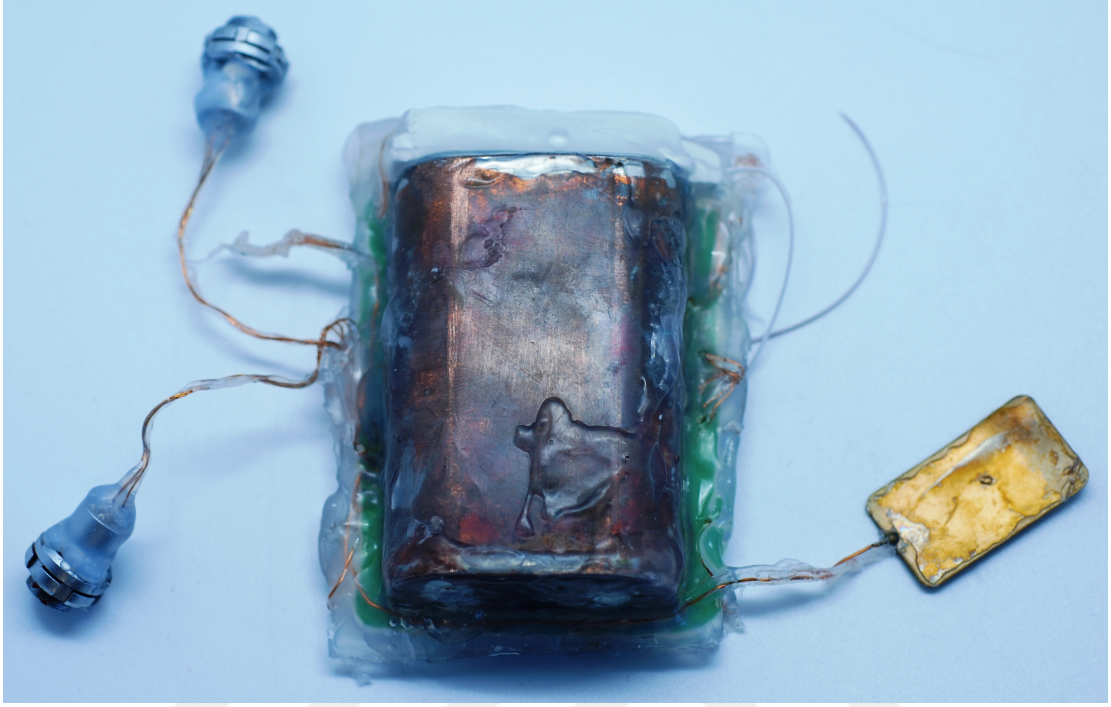


Figure 6.6: Evaluation of the implant package after its removal

One of the ECG electrodes was found to be broken at a point outside the encapsultant. Inside the encapsultant, several fractures of the copper cables, connected to both PPG sensors, were observed. Those cables were still slightly in contact, causing unstable signal acquisition and noise generation. The copper cables did not exhibit any failures in welding to the Pt/Au pins, but certainly were affected by mechanical stress at the time of surgery or implant duration. They need to be replaced by a more flexible cable and connector system. There are several implantable cable and connector systems that are presented and have been commercially available. A form of cable (Donaldson 1983) that withstands bending, twisting, stretching and crushing and a relevant connector system (Donaldson 1985) can be employed in future designs. The mentioned Cooper cable is a practical compromise in which the conductors that are insulated with a polyimide resin, are arranged as simple helixes inside a silicon rubber cable body. The cable terminates in the Craggs connector, which is an implantable plug-and-socket assembly, realized by using a special type of adhesive sealant on the mating surfaces of the receptacles and pins, that would be scraped off when the pins and receptacles are pushed towards each other.

The electronic components inside the package were not exposed to any moisture effects. The PCB was tested by debugging the microcontroller with the JTAG

interface, before the lid was removed, and operated successfully. In conclusion, the long-term investigation of the system confirms that the implanted package was biocompatible with sufficient hermeticity.

PPG As it was mentioned earlier, the tip catheter could not be implanted and accordingly, a reference blood pressure signal was not acquired. Additionally, the detachment of one ECG electrode prohibited the measurement of an ECG-signal. The fractures at the connections of the PPG-sensors distorted most of the measurements. Hence, only a small portion of the acquired data was useful. Figure 6.7, shows an acquired pulsatile signal, that was sampled at 250 sps and stored in the memory card, 3 hours after the sheep woke up from narcosis. The responsible veterinary surgeon interpreted the low frequency signal changes seen in Figure 6.7.a as slow head movements. The pulsatile signals from figure 6.7.b were identified as cardiac pulse signals. This indicates that photoplethysmographic signals could be be measured by an implanted sensor in an awake animal for the very first time, as no such measurement has ever been published. The shown signal was processed by using a median filter to remove the far outlier values. In sequence, a FIR filter and cut-off frequency of 8 Hz was applied to remove the high frequency noise.

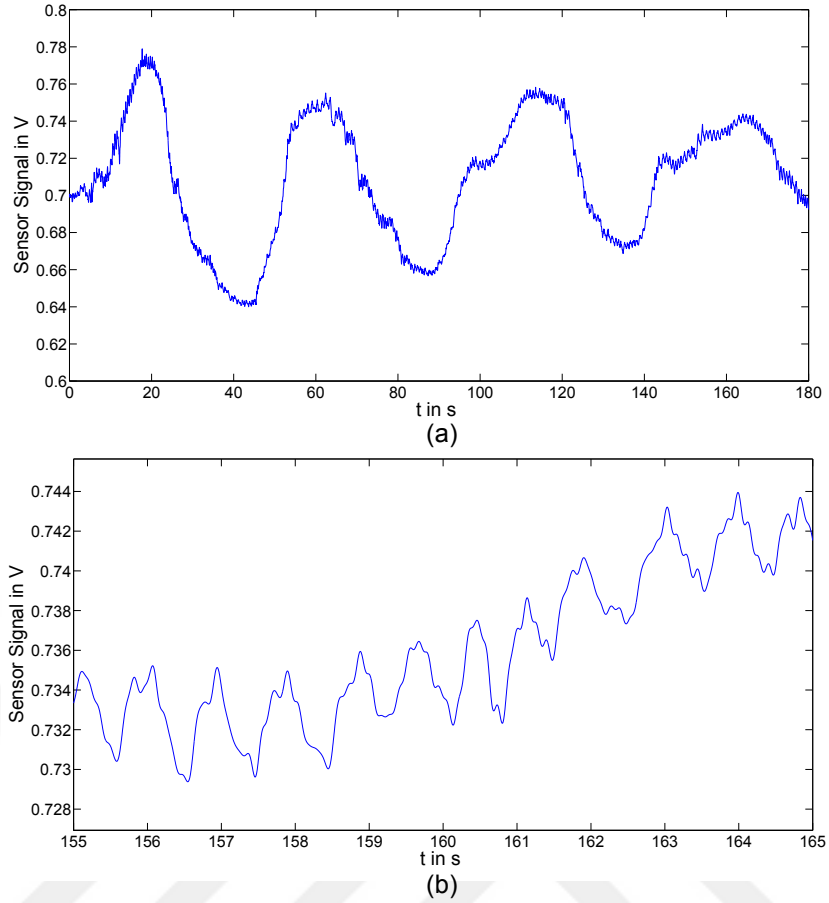


Figure 6.7: (a) Pulsatile signal from the in-vivo measurements. (b) Closer view of the signal.

The overall measured PPG sensor signals from the 14-day implant duration, were observed to be very noisy. Although the origin of the high noise is not clear, it is suggested to occur from intermittent contact of the sensor cables and interference of the physiological sources inside the body, such as muscles. Ultimately, during the implant duration, the sensor-read out system achieved to acquire, sample and store the PPG signals. The in-vivo measurements indicated that, the sensor read-out system is capable to obtain biomedical signals. With improvement of the sensor connections, a stable read-out system with high signal accuracy can be maintained.

The relative humidity and temperature inside the implanted package was measured by the internal SHT25 sensor. Figure 6.8 shows the temperature and humidity measurements that were held at the first 105 hours of the implant duration. The sensor read-out system was powered by its internal battery pack at 8 hours before the implantation into the animal body, as the steep rise of the measurement values indicates.

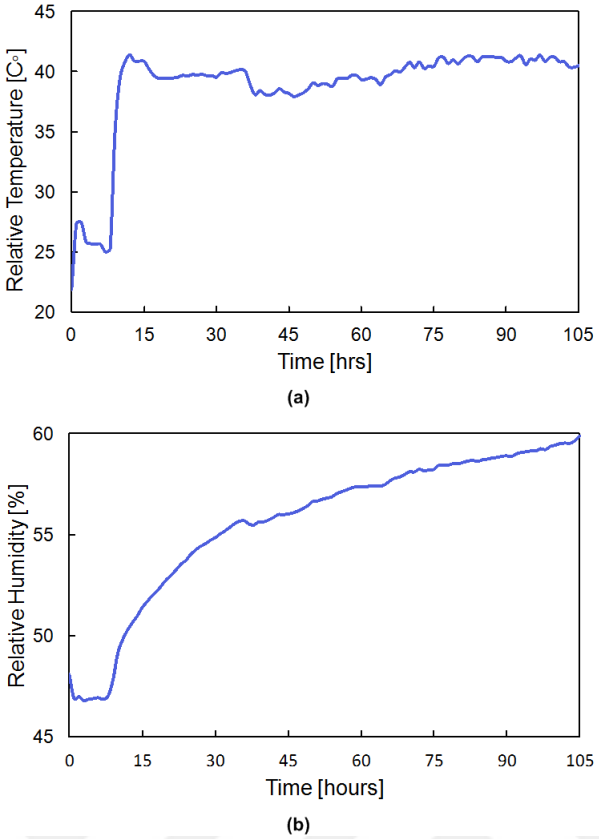


Figure 6.8: (a) Relative temperature measurement. (b) Relative humidity measurement.

The plotted relative humidity values demonstrate the relative humidity (RH) above liquid water according to World Meteorological Organization (WMO), and were calculated from the output signal (S_{RH}) of the SHT25, by the following formula (Sensirion 2011):

$$RH = -6 + 125 \times \frac{S_{RH}}{2^{16}}. \tag{6.1}$$

The temperature (T) values in °C, were calculated by inserting temperature signal output S_T into the following formula (Sensirion 2011):

$$T = -46.85 + 175.72 \times \frac{S_T}{2^{16}}. \tag{6.2}$$

The body temperature of the subject animal was externally tracked and was recorded as stable since the morning of the second day after the implantation. The mean value of the internal temperature of the implanted package from Figure 6.8, is calculated as 39.93 °C. Former studies on different types of the domestic sheep animal shows that they maintain a body temperature of 38.5 to 39.5 °C (Miller and Monge 1946), which complies with the measurement results taken from the read-out system. As it was mentioned in Chapter 5.2, the implant package was filled with

silicone rubber inside the copper lid. Although, the used silicone rubber was put into a vacuum chamber formerly, the measurement results show an increasing profile of the relative humidity above 50 %. Accordingly, it is suggested that some air still got trapped inside the silicone rubber during pouring it through the holes of the copper lid before sealing.

As it is discussed above, the system was able to record the body temperature of the implant subject and the relative moisture level inside the implant package. The results demonstrated that, in order to reduce the relative humidity inside the implant package, moisture which is trapped inside the package during hermetic sealing, needs to be removed. This could be done by a method that involves pre-drying, heating and soldering the package inside a chamber which monitors and controls its internal temperature and humidity (Schuettler et al. 2010).

6.3 Subcutenous Tissue Masurements

The dependency of the ECG measurements on electrode dimensions and interelectrode distance was studied by an acute in-vivo examination on a pig. The aim of this study was to investigate the optimum positioning and dimensions of implant electrodes for bipolar ECG measurements. The same medical procedure that was mentioned in Chapter 6.2 was applied on the subject animal. The study was conducted with copper foil electrodes of three different sizes: surface areas 1 cm^2, 4 cm^2 and 9 cm^2. The ECG signals were recorded using LabView program, from of the basic bipolar biopotential measurements, subcutaneously on the neck of the animal. For each electrode pair, the measurements were carried out in three different placement configurations: the two electrodes were located in parallel, at the incisions that were 1 cm, 3 cm and 8 cm apart from each other. The recorded signals were processed in Matlab by filtering with a FIR low-pass filter with 40 Hz of cut-off frequency.

Figure 6.9 shows the ECG signals that were obtained from the 4 cm^2electrodes in 3 different subcutaneous placements. The amplitude of the R-peaks in the shown graph are 7.2 mV, 12 mV and 24 mV for 1 cm, 3 cm and 8 cm distances respectively. As it is demonstrated by the measurement results, the smaller interlectrode distance reduces the signal amplitude. Nonetheless, the QRS interval is explicit in all of the three distances, enabling sufficient accuracy for a PTT calculation.

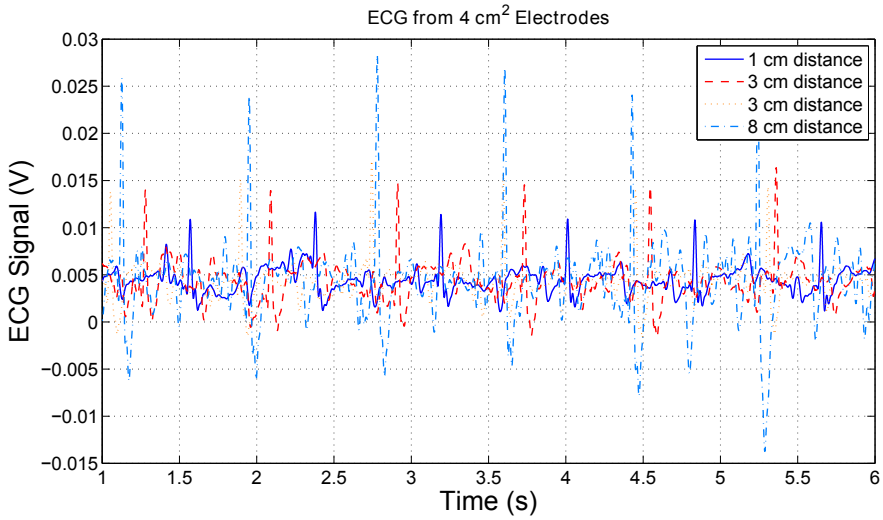


Figure 6.9: ECG measurement with different interelectrode distances.

In Fig 6.10, the electrode dimensions with 3 cm interelectrode distance are compared. The smallest electrode exhibits 0.4 mV R-peak with very rough signal details. The 4 cm^2 and 9 cm^2 electrodes provide ECG measurements with distinct P and T waves, in addition to R peaks with amplitudes of approximately 12 mV and 23 mV, respectively.

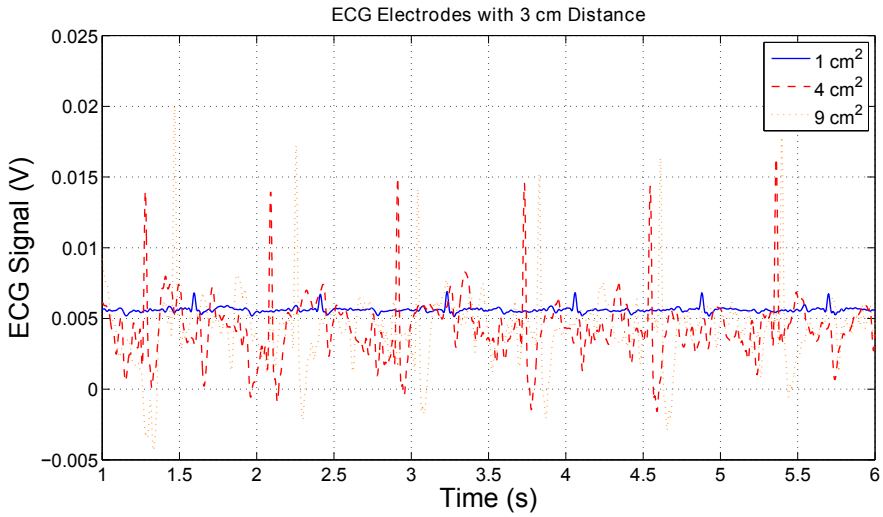


Figure 6.10: Comparison of the electrode dimensions.

Power of the ECG signal from a 4 cm^2 electrode with 3 cm distance is given in Figure 6.11. The ECG overtones are present up to 50 Hz. An interference peak is present at 50 Hz.

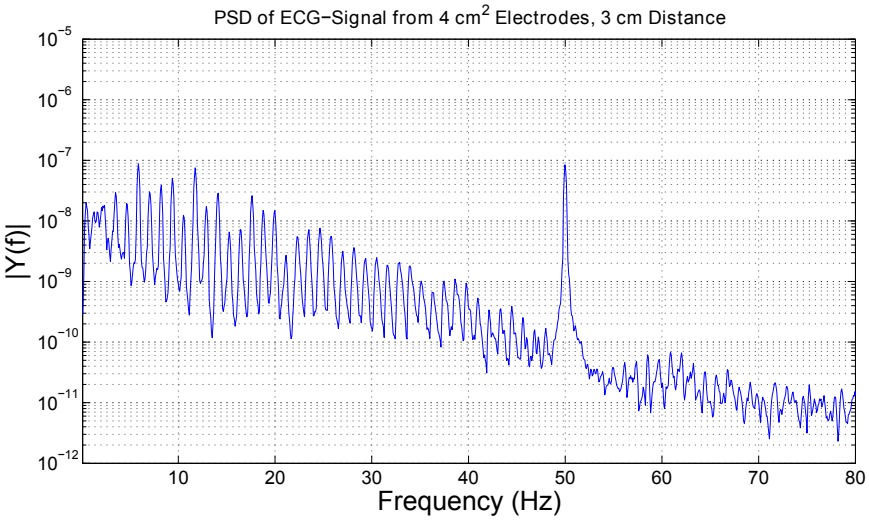


Figure 6.11: Power Spectral Density of the ECG signal.

Table 6.1 gives the R-peak amplitudes for the conducted measurements. According to these results the distance of 3 cm with 4 cm^2 and 9 cm^2 electrodes provide accurate and sufficient extraction of the important features from the ECG signal, and hence, for applications requiring the heart beat and PTT calculation, the electrode pairs around these range would be even more than adequate.

Surface Area \ *Interelectrode distance*	3 cm	8 cm
1 cm^2	0.47 mV	1.09 mV
4 cm^2	12.01 mV	23.94 mV
9 cm^2	15.72 mV	23.54 mV

Table 6.1: Comparison of the ECG signal strength for different electrode setups.

The PPG signal measurements for the subcutaneous tissue on the subject animal was also conducted by the PPG sensors that were used in the long-term implantation and the PCB of the sensor read-out system. The measured signal was sampled at 250 sps and written into the microSDHC card. The raw PPG signal is displayed in Figure 6.12. As seen, the subcutaneous tissue provides smooth PPG signals even without any signal processing. The respiratory cycle of approximately 5 seconds which was applied to the subject animal during the operation, is visible in the plotted signal. Each PPG cycles exhibit an almost identical primary peak amplitude of approximately 6 mV. Once more, the capability of acquiring in-vivo PPG signals within the sensor read-out system is confirmed by the performed subcutaneous tissue measurements.

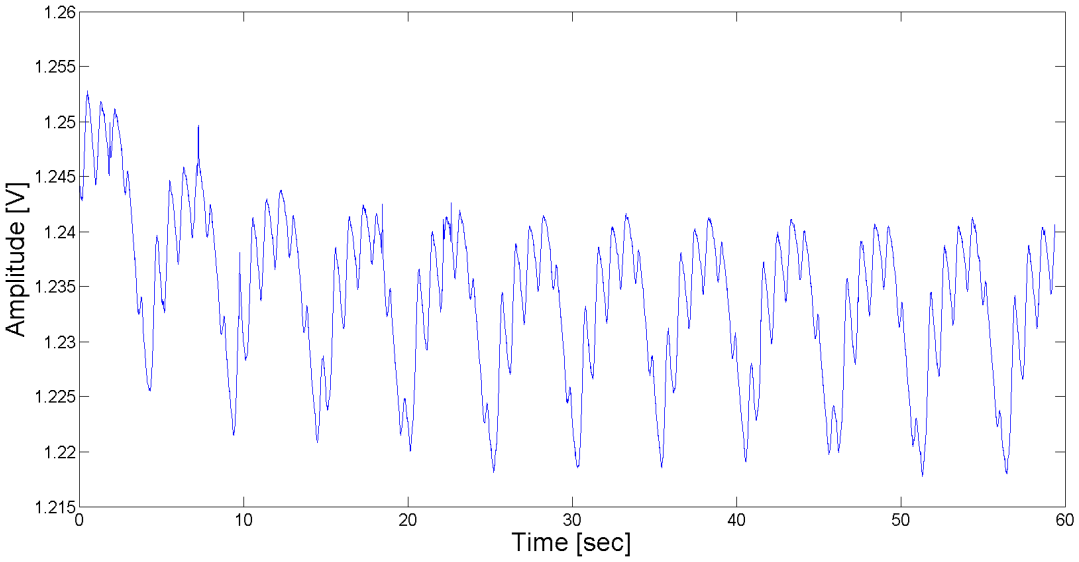


Figure 6.12: The subcutaneously measured PPG signal.

7 Conclusion & Outlook

7.1 Conclusion

Monitoring and recording physiologic events in a patient's body facilitates research, diagnostic and therapeutic approaches that have significant importance in modern medicine. The objective of this Master thesis was to achieve an implantable read-out system that would gather and record vital signs from the attached sensors and electrodes in a chronic, subcutaneous embodiment. In this context, an implantable system, having a hermetically sealed housing, that is capable of recording chronic, subcutaneous temperature, ECG, PPG and arterial BP signals, was developed and is confirmed to be accomplished by laboratory tests, in-vivo measurements and a long-term implantation.

Development The implantable read-out system consists of a hermetically sealed, biocompatible package, an internal, rechargeable battery pack and a PCB with surface mounted electronic circuitry for data acquisition and recording. To achieve the recording of the subcutaneous parameters, a hardware design and a software program were developed. The system was preferred to be battery powered to maintain the power consumption of the electronic circuitry. The implemented microcontroller is programmed to enter a low power operating mode, after a period of recording of the vital signs. Once the sensor read-out system is externally turned on, it drives the LED currents on the attached PPG and BP sensors, measures the temperature and humidity by an implemented sensor inside the implant package, acquires ECG, PPG and arterial BP signals and stores them in a memory card. During the 10 minutes of this duty cycle, the system consumes of 44 mA in addition to the LED currents. This is followed by an idle state for 50 minutes with power consumption of 614 μA. Hence, the periodical operation saves the battery power. The implemented multi-channel analog-to-digital converter allows a range of sampling rates from 250 sps to 32 ksps, and different data resolutions, as well.

An implantable system must have a minimal impact on the host body and vice-versa. Accordingly, a fabrication technology based on screen-printing on an alumina substrate was employed for the hermetically sealed implant package. The PCB and the battery pack were fixed on the screen-printed alumina substrate which carries 45 electrical feedthroughs and a metal lid, covering the components. The Pt/Au feedthroughs, connected to the PCB under the lid, provide the attachments of the two ECG electrodes, the tip catheter and up to 5 PPG sensors to the read-out system. A JTAG interface for debugging the implemented microcontroller, an SPI

port for external data transmission and an on/off switch for external management of the system are also introduced by the feedthroughs. The sealing of the package was done by soldering the metal lid to the metal layer on the alumina substrate. The implant package was joined to thin copper wires for the sensor and electrode connections. The entire package was encapsulated in a transparent, biocompatible silicone rubber (MED-1000).

Measurements The introduced implantable sensor read-out system was successfully tested in the laboratory measurements. The in vitro tests showed that the system is able to acquire and store the PPG and ECG signals with sufficient accuracy for BP monitoring. To evaluate the sensor read-out system in an organism, it was implanted in a domestic sheep, with subcutaneous PPG sensors and gold-coated, copper foil ECG electrodes for 2 weeks. The only distinguished tissue response after the implant duration, was the swelling that was formed around the implant package. After the explantation, the read-out system was found to operate normally without any malfunctioning or moisture effects. The signals that were subcutaneously recorded from the animal were quite noisy due to the fractures which occurred in the sensor and electrode connections outside of the implant package. However, the implanted sensor read-out system achieved to record subcutaneous pulsatile signals in the animal's body, without any sedation. The sensor inside the implant package delivered the body temperature and humidity measurements.

In Addition, acute in-vivo measurements of PPG and ECG signals were performed on a pig under anesthesia. The examination of the optimum positioning and dimensions of implant electrodes for bipolar ECG measurements, was conducted with copper foil electrodes with surface areas of 1 cm^2, 4 cm^2 and 9 cm^2. The measurements were carried out with interelectrode distances of 1 cm, 3 cm and 8 cm for each electrode pair. Accordingly, the measurements demonstrated that 3 cm distance with 4 cm^2 and 9 cm^2 electrodes provide accurate and sufficient extraction of the important features from the ECG signal, although the QRS complex was recognizable in all the three distances except for the electrode with 1 cm^2 surface area. During the operation, the acute, subcutaneous measurement of the PPG signals were conducted by the PPG sensors and the read-out system. The recorded PPG signals are observed to be smooth and stable without any signal processing.

7.2 Outlook

The sensor read-out system is currently under progress to utilize wireless power transmission and communication. The system will be miniaturized and its operating time will be extended by inductive powering with antenna-based transmission. Furthermore, the exchange of the internal memory card with wireless data transmission will introduce real-time, continuous monitoring without the need of explantation of the device. The used alumina provides transmission properties for RF-signals and is therefore well suited for this concept.

Considering the fractures of the cables that were connecting the sensors and electrodes to the system, a more robust configuration must be presented. For this reason, the commercially available, silicone rubber coated conductor cable and its termination, developed especially for implantation and known as the Cooper cable (Donaldson 1983) and the Craggs connector (Donaldson 1985), can be employed as a plug-and-socket type of connector. Moreover, the corrosion which was observed on the solder seal of the implant package is supposed to be prevented by choosing a different type of silicon rubber (Dow Corning 3140) as the encapsultant which adheres more strongly to metal surfaces than the used Med-1000 silicone.

A Appendix

A.1 Source Code

A.1.1 main.c

```
extern char ads_stdconfig[0x1A];
#include "prototype.h"

unsigned char sd_status =1 ;
unsigned int timeout=0, time=0;
unsigned char header[512], channel_0[128], channel_4[128], channel_5[128], channel_7[128];
unsigned char buffer_write[512], last_sector[512];
int timestamp, i=0, q=0, g, n, sample_id, sd_id, led1 ,led2 ;
char *regs_pointer;
char ads_stdconfig[0x1A];
unsigned long int sector_id=0x00000000;
long int  led_in1, led_in2;
unsigned int recT[4]={0x00,0x00, 0x00, 0x00};
unsigned int recRH[4]={0x00,0x00, 0x00, 0x00};
float   temperatureC;
float   humidityRH;
unsigned int cmdT[1]={0xE3}; //temperature measurement command
unsigned int cmdRH[1]={0xE5}; //humidity measurement command

//-------------------------------------------------------------------------------
void main(void)
{

 char reg;
 char byte0[8];
 char byte1[8];
 //Configuration of internal peripherals
 WDTCTL = WDTPW + WDTHOLD; // Stop Watchdog Timer
 BCSCTL1 = CALBC1_16MHZ; // Set internal oscillator to 16MHz
 DCOCTL = CALDCO_16MHZ;  // Set internal oscillator to 16MHz
 BCSCTL2 = 0;            // MCLK = DCO = SMCLK
 TACTL = TASSEL_1 + MC_2 + TAIE + ID_3 ; // Set Timer_A to 4096Hz in continous mode
 ADC12CTL0 = REF2_5V + REFON; // Internal reference for DAC is 2.5V
 DAC12_0CTL =  DAC12CALON | DAC12IR | DAC12AMP_5 | DAC12ENC; // VRef+, Medium speed/current
 DAC12_1CTL =  DAC12CALON | DAC12IR | DAC12AMP_5 | DAC12ENC; // VRef+, Medium speed/current
 P1DIR |= 0x41; //P1.0 in output direction
 P1OUT &= ~0x41; //P1.0 is off

  while (1) //eternal loop
  {

__bis_SR_register(LPM3_bits + GIE); // Enter LPM3, wait for wakeup

    P1OUT |= 0x40; // PWR_EN
    // ADS1298 uses its internal clock
    P5DIR |= BIT4 ;
P5OUT |= BIT4 ;

TBCTL =CNTL_0 + TBSSEL_1 + MC_2 + TBIE + ID_3; // Set Timer_B to 4096Hz in continous mode
    //ADS1298 initialisation
     reg=0;
    while (reg != 0x92) {
       P1OUT &= ~0x40;
  P1OUT |= 0x40;
        ads_spi_init();
        P2OUT |= BIT7;
ads_set_ncs(0);
ads_set_startpin(0);
  ads_sendsdatac();
   reg=ads_sendreadreg(0); }
 ads_getstdconfig(ads_stdconfig);
 regs_pointer= &ads_stdconfig[0];
 ads_sendwriteallreg(regs_pointer);
 ads_set_startpin(1);
 //microSDHC initialisation
 while (sd_status != 0)
  {
    sd_status = mmcInit();
```

```
    timeout++;
    if (timeout == 150)
    { break; }   }
  sd_status = 1;
  timeout=0;
    // Read last sector address after a restart
if (sector_id == 0) {
    memset(&last_sector,0,512);
mmcReadBlock(0x3BC5FFF,512,last_sector);
sector_id = (last_sector[3] << 8)+ last_sector[2];
sector_id =( sector_id  << 8)+last_sector[1];
sector_id =( sector_id  << 8)+last_sector[0]; }
     //Receive humidity and temperature data from SHT25
 TI_USCI_I2C_transmitinit(0x40,0x12);     // init transmitting with USCI
 while ( TI_USCI_I2C_notready() );        // wait for bus to be free
 if ( TI_USCI_I2C_slave_present(0x40) )   // slave address may differ from
  {
    TI_USCI_I2C_transmitinit(0x40,0x12);  // init transmitting with USCI
    while ( TI_USCI_I2C_notready() );     // wait for bus to be free
    TI_USCI_I2C_transmit(1,cmdT);
    while ( TI_USCI_I2C_notready() );     // wait for bus to be free

    TI_USCI_I2C_receiveinit(0x40,0x12);   // init receiving with
    while ( TI_USCI_I2C_notready() );     // wait for bus to be free
    TI_USCI_I2C_receive(4,recT);          // start receiving
    while ( TI_USCI_I2C_notready() );

    TI_USCI_I2C_transmitinit(0x40,0x12);   // init transmitting with USCI
    while ( TI_USCI_I2C_notready() );      // wait for bus to be free
    TI_USCI_I2C_transmit(1,cmdRH);
    while ( TI_USCI_I2C_notready() );      // wait for bus to be free

    TI_USCI_I2C_receiveinit(0x40,0x12);    // init receiving with
    while ( TI_USCI_I2C_notready() );      // wait for bus to be free
    TI_USCI_I2C_receive(4,recRH);          // start receiving
    while ( TI_USCI_I2C_notready() );
  }
   //Write the header sector to the card
   memset(&header,0,512);
   for (g=0; g < 16 ;g ++)
{
header[g]= 170;
header[g+20]= 170;
}
header[12]= recRH[0];
    header[13]= recRH[1];
    header[14]= recT[0];
    header[15]=recT[1];
   // LED_Drive
   DAC12_0DAT = 327; // DAC0 output// 0.2V ~20mA
   DAC12_1DAT = 327;
   led1=327;
   led_in1 = 0x6AAAA9; // 2V
   while (led_in1 >= 0x6AAAA9 && led1 > 0 ) {
ads_sendrdata8(byte0,byte1);
    led_in1 = (byte0[4] << 8) + byte1[4];
    led_in1 =(led_in1 << 8)  ;
led1 = led1 - 16; //--0.01V
DAC12_0DAT = led1;
}
   led2=327;
   led_in2 = 0x6AAAA9;;
   while (led_in2  >= 0x6AAAA9 && led2 > 0 ) {
ads_sendrdata8(byte0,byte1);
    led_in2 = (byte0[5] << 8) + byte1[5];
    led_in2 =(led_in2 << 8)  ;
    led2 = led2 - 16; //--0.01V
DAC12_1DAT = led2;
}
//Write DAC values to the header
     header[16]= led1 & 0x00FF;
     header[17]= led1 / 0x0100;
     header[18]= led2 & 0x00FF;
     header[19]= led2 / 0x0100;
     mmcWriteBlock(sector_id, 512,header);
 sector_id=sector_id+1;

 TBCTL = MC_0 | TBCLR; // stop timer B

for (q=0; q < 2340  ;q ++) // 10 min. daata acquisition and storage
      {
TBCTL =  MC_0; // Halt Timer_B
    timestamp= TBR; //Read the counter
    TBR = 0;
TBCTL =CNTL_0 + TBSSEL_1 + MC_2 + TBIE + ID_3; //Restart Timer_B

       // Collect 128-bytes from 4 channels
for (sd_id=0; sd_id <32 ;sd_id++)
  {
     for (sample_id=0; sample_id < 2; sample_id++)
```

```
     {
        ads_sendrdata8(byte0,byte1); // Read all channels

        channel_0[4*sd_id+2*sample_id]   = byte0[0]; // channel 1=tip cath.
      channel_0[4*sd_id+2*sample_id+1] = byte1[0];

        channel_4[4*sd_id+2*sample_id]   = byte0[4]; // channel 5 = PPG1
      channel_4[4*sd_id+2*sample_id+1] = byte1[4];

        channel_5[4*sd_id+2*sample_id]   = byte0[5]; // channel 6 = PPG2
      channel_5[4*sd_id+2*sample_id+1] = byte1[5];

        channel_7[4*sd_id+2*sample_id]   = byte0[7];// channel 8 = ECG
      channel_7[4*sd_id+2*sample_id+1] = byte1[7];

     }
}

 for (n=0; n <128 ;n ++) // prepare 512-bytes
   {
    buffer_write[n]    = channel_0[n] ;
    buffer_write[n+128]= channel_4[n] ;
    buffer_write[n+256]= channel_5[n] ;
    buffer_write[n+384]= channel_7[n] ;
    }
// Write the timestamp to the card
buffer_write[1]= timestamp & 0x00FF; //lsb;
buffer_write[0]= timestamp / 0x0100; //msb;
     mmcWriteBlock(sector_id, 512,buffer_write); // Write the sector to the card
    sector_id = sector_id + 0x01;
  }
   DAC12_0DAT = 0; //Turn off the LED
   DAC12_1DAT = 0; //Turn off the LED
   //store sector number
   last_sector[0] = sector_id & 0x00FF; //lsb
   last_sector[1] = sector_id / 0x0100; // 16-23 bits
   last_sector[2] = sector_id / 0x10000; //8-15 bits
   last_sector[3] = sector_id >> 24 ; //msb
   mmcWriteBlock(0x3BC5FFF, 512,last_sector);

TBCTL = MC_0 | TBCLR; // stop timer B
//Disable the ports
P1OUT &= ~0x41; //Disable PWR_EN
P2SEL = 0;
P2OUT = 0;
P3SEL = 0;
P3OUT = 0;
P4SEL = 0;
P4OUT = 0;
P5SEL = 0;
P5OUT = 0;
P6SEL = 0;
P6OUT = 0;

 }
}

#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)

{
switch (TAIV) { // Define the interrupt spurce

case 0: // No interrupt pending
break;  // No action

case 2: // Capture/compare
break;  // No action

case TAIV_TAIFG: // Timer overflow
time ++;
if (time==184) { // sleep for ~50 min.
time=0;
__bic_SR_register_on_exit(LPM3_bits); } //wake up, disable interrupt
break;
  }
}

#pragma vector=TIMERB1_VECTOR
__interrupt void Timer_B (void)

{
switch (TBIV) { // // Define the interrupt spurce

case 0: // No interrupt pending
break;  // No action

case 2: // Capture/compare
break;  // No action
```

```
    case TBIV_TBIFG: // Timer overflow
     WDTCTL = 42; // Reset and restart the system
break;
  }
}
```

A.1.2 ads1298.c

```
#include "prototype.h"
#define t_pause 20000
char ads_sendstart(void)
{
char response;
ads_txbuf = ads_start;
  while(ads_stat&UCBUSY);
  ads_txbuf = 0xFF; // dummy byte
  while(ads_stat&UCBUSY);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  response = ads_rxbuf;
return response;
}
 char ads_sendreset(void)
{
char response;
ads_txbuf = ads_reset;
  while(ads_stat&UCBUSY);
  ads_txbuf = 0xFF; // dummy byte
  while(ads_stat&UCBUSY);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  response = ads_rxbuf;
return response;
}
 char ads_sendwakeup(void)
{
char response;
ads_txbuf = ads_wakeup;
  while(ads_stat&UCBUSY);
  ads_txbuf = 0xFF; // dummy byte
  while(ads_stat&UCBUSY);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  response = ads_rxbuf;
return response;
}
 char ads_sendstandbye(void)
{
char response;
ads_txbuf = ads_standby;
  while(ads_stat&UCBUSY);
  ads_txbuf = 0xFF; // dummy byte
  while(ads_stat&UCBUSY);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  response = ads_rxbuf;
return response;
}
 char ads_sendsdatac(void)
{
char response;
```

```
ads_txbuf = ads_sdatac;
  while(ads_stat&UCBUSY);
  ads_txbuf = 0xFF; // dummy byte
  while(ads_stat&UCBUSY);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  response = ads_rxbuf;
return response;
}

 char ads_sendreadreg(char reg)
{
char response;
ads_txbuf = ads_rreg|reg;
  while(ads_stat&UCBUSY);
  ads_txbuf = 0; // number of regs - 1
  while(ads_stat&UCBUSY);
  ads_txbuf = 0xFF; // dummy byte
  while(ads_stat&UCBUSY);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  response = ads_rxbuf;
return response;
}

 void ads_sendreadallreg(char *buffer)
{
int reg_id;
ads_txbuf = ads_rreg;
  while(ads_stat&UCBUSY);
  ads_txbuf = 0x19; // read all 0x19 regs
  while(ads_stat&UCBUSY);
  for (reg_id=0; reg_id <0x19; reg_id++)
  {
  ads_txbuf = 0xFF; // dummy byte
  while(ads_stat&UCBUSY);
  buffer[reg_id] = ads_rxbuf;
  }
  ads_txbuf = 0xFF; // dummy byte
  while(ads_stat&UCBUSY);
  wait(t_pause);

}

 char ads_sendwritereg(char reg, char value)
{
char response;
ads_txbuf = ads_wreg|reg;
  while(ads_stat&UCBUSY);
  ads_txbuf = 0x00; // number of regs -1
  while(ads_stat&UCBUSY);
  ads_txbuf = value; // send value
  while(ads_stat&UCBUSY);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  response = ads_rxbuf;
return response;
}

 char ads_sendwriteallreg(char *bufferpointer)
{
char response;
int reg_id;
ads_txbuf = ads_wreg|ads_config1;
  while(ads_stat&UCBUSY);
  ads_txbuf = 0x10; // number of regs -1
  for (reg_id=1;reg_id<ads_loff_statp;reg_id++)
  {
  while(ads_stat&UCBUSY);
  ads_txbuf = *(bufferpointer + reg_id); // send value
  }
while(ads_stat&UCBUSY);
ads_txbuf = ads_wreg|ads_gpio;
```

```
  while(ads_stat&UCBUSY);
  ads_txbuf = 0x05; // number of regs -1
  for (reg_id=ads_gpio; reg_id < ads_wct2+1; reg_id++)
  {
  while(ads_stat&UCBUSY);
  ads_txbuf = *(bufferpointer + reg_id); // send value
  }

  while(ads_stat&UCBUSY);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  wait(t_pause);
  response = ads_rxbuf;
return response;
}

 void ads_set_ncs(char set)
{
if (set ==0)
ads_ctl_OUT &= ~ads_ncs ; // set nCS to 0
else
ads_ctl_OUT |= ads_ncs; // set nCS to 1
}

char ads_set_gain(int channel, int value)
{
char ch_reg = ads_sendreadreg(channel+4);
ch_reg &= 0x8F; // set gain-bits 7:5 to 0
if (value < 13)
{
switch ( value)
{
case 1:
ch_reg |= 0x10; // gain = 1
break;
case 2:
ch_reg |= 0x20; // gain = 2
break;
case 3:
ch_reg |= 0x20; // gain = 3
break;
case 4|5:
ch_reg |= 0x40; // gain = 4
break;
case 6|7:
ch_reg |= 0x00; // gain = 6
break;
case 8|9|10:
ch_reg |= 0x50; // gain = 8
break;
case 11|12:
ch_reg |= 0x60; // gain = 12
break;
default:
ch_reg |= 0x00; // gain = 6
break;
}
}
else
{ ch_reg |= 0xE0;} // gain = 6
return ads_sendwritereg(channel+4, ch_reg);

}

void ads_spi_init()
 {
  UCB0CTL1 |= UCSWRST ; // Disable SPI during config
  // init registers: 8bit, msb first,master mode, sync mode, clk starts with msb and data valid on falling clk-edge, ucckph=0 and ucckpl =0
  UCB0CTL0 |=  UCMST  + UCSYNC +UCMSB; //3-pin-mode! //+ UCMODE1;
  UCB0CTL1 |= UCSSEL1;
  UCB0BR1 = 0;
  UCB0BR0 = 2; // Set prescaler to 2
  // Ads1298 spi pins
  ads_spi_SEL |= ads_simo + ads_somi + ads_sclk;
ads_spi_DIR |= ads_simo + ads_sclk;
ads_spi_DIR &= ads_somi;
ads_ctl_DIR |= ads_start +ads_rst +ads_ncs + ads_n_pwrdwn;
  UCB0CTL1 &= ~UCSWRST ; // Enable SPI, to be ready for transfer
  wait(t_pause);
ads_ctl_OUT |= ads_rst; // Set RST-pin high

}
void ads_getstdconfig(char *ads_stdconfig) // Configure ADS1298 registers
{
 ads_stdconfig[ads_config1] = 0x06 ; // 250sps data rate
```

```
 ads_stdconfig[ads_config2] = 0x00 ;
 ads_stdconfig[ads_config3] = 0xC0 ; // Reference voltage =2.4V, RLD disabled
 ads_stdconfig[ads_loff] = 0x40 ;
 ads_stdconfig[ads_ch1set] = 0x10 ; // Channel Gain=1 for tip cath.
 ads_stdconfig[ads_ch2set] = 0x00 ;
 ads_stdconfig[ads_ch3set] = 0x00 ;
 ads_stdconfig[ads_ch4set] = 0x00 ;
 ads_stdconfig[ads_ch5set] = 0x10 ; // Channel Gain=1 for PPG1
 ads_stdconfig[ads_ch6set] = 0x10 ; // Channel Gain=1 for PPG2
 ads_stdconfig[ads_ch7set] = 0x00 ;
 ads_stdconfig[ads_ch8set] = 0x60 ; // Channel Gain=12 for ECG
 ads_stdconfig[ads_rld_sensp] = 0x00 ;
 ads_stdconfig[ads_rld_sensn] = 0x00 ;
 ads_stdconfig[ads_loff_sensp] = 0x00 ;
 ads_stdconfig[ads_loff_sensn] = 0x00 ;
 ads_stdconfig[ads_loff_flip] = 0x00 ;
 ads_stdconfig[ads_loff_statp] = 0x00 ;
 ads_stdconfig[ads_loff_statn] = 0x00 ;
 ads_stdconfig[ads_gpio] = 0x0F ; // all GPIOs are inputs
 ads_stdconfig[ads_pace] = 0x00 ; // disabled
 ads_stdconfig[ads_resp] = 0x00 ;
 ads_stdconfig[ads_config4] = 0x00 ; // lead-off disabled
 ads_stdconfig[ads_wct1] = 0x00 ; // wilson central terminal disabled
 ads_stdconfig[ads_wct2] = 0x00 ; // wilson central terminal disabled
}

char ads_sendrdata8(char *byte0,char *byte1) // Read samples from channels
{
char status;

int ch_id;
while(ads_ctl_IN & ads_ndrdy);
ads_txbuf = ads_rdata;
while(ads_stat & UCBUSY);
// Status bits
ads_txbuf = 0xFF; // Dummy Byte
while(ads_stat & UCBUSY);
status = ads_rxbuf;
ads_txbuf = 0xFF; // Dummy Byte
while(ads_stat & UCBUSY);
ads_txbuf = 0xFF; // Dummy Byte
while(ads_stat & UCBUSY);

for (ch_id=0; ch_id<8;ch_id++)
{
ads_txbuf = 0xFF; // Dummy Byte
while(ads_stat & UCBUSY);
byte0[ch_id] = ads_rxbuf; //MSB of the sample
ads_txbuf = 0xFF; // Dummy Byte
while(ads_stat & UCBUSY);
byte1[ch_id] = ads_rxbuf; // middle byte of the sample
ads_txbuf = 0xFF; // Dummy Byte
while(ads_stat & UCBUSY);
}
return status;
}

 void ads_set_startpin(char set)
 {
  if (set ==0)
ads_ctl_OUT &= ~ads_start_pin ; // set START to 0
else
ads_ctl_OUT |= ads_start_pin; // set START to 1
wait(t_pause);
 }

void wait(int cycles)
{
int i;
int w=0;
for (i=0; i< cycles;i++)
{w=w+1;}
}
```

A.1.3 ads1298.h

```
// use USCI_B0 port for ADS1298
 #define ads_txbuf UCB0TXBUF
 #define ads_rxbuf UCB0RXBUF
 #define ads_stat UCB0STAT
// Define Parameters for ADS1298 on Port2:
 #define ads_ctl_OUT P2OUT
 #define ads_ctl_IN  P2IN
 #define ads_ctl_DIR P2DIR
 #define ads_ctl_SEL P2SEL
 #define ads_ndrdy BIT2
 #define ads_start_pin BIT3
```

```
#define ads_rst BIT4
#define ads_gpio1 BIT6
#define ads_n_pwrdwn BIT7
#define ads_ncs BIT5
// Define SPI-parameters for ADS on port3:
#define ads_spi_OUT P3OUT
#define ads_spi_DIR P3DIR
#define ads_spi_SEL P3SEL
// SPI signal lines
#define ads_simo BIT1
#define ads_somi BIT2
#define ads_sclk BIT3

// Define commands:
#define ads_wakeup 0x02
#define ads_standby 0x04
#define ads_reset 0x06
#define ads_start 0x08
#define ads_stop 0x0A
#define ads_rdatac 0x10
#define ads_sdatac 0x11
#define ads_rdata 0x12
#define ads_rreg 0x20
#define ads_wreg 0x40

// Define Registers
#define ads_id 0x00
#define ads_config1 0x01
#define ads_config2 0x02
#define ads_config3 0x03
#define ads_loff 0x04
#define ads_ch1set 0x05
#define ads_ch2set 0x06
#define ads_ch3set 0x07
#define ads_ch4set 0x08
#define ads_ch5set 0x09
#define ads_ch6set 0x0A
#define ads_ch7set 0x0B
#define ads_ch8set 0x0C
#define ads_rld_sensp 0x0D
#define ads_rld_sensn 0x0E
#define ads_loff_sensp 0x0F
#define ads_loff_sensn 0x10
#define ads_loff_flip 0x11
#define ads_loff_statp 0x12
#define ads_loff_statn 0x13
#define ads_gpio 0x14
#define ads_pace 0x15
#define ads_resp 0x16
#define ads_config4 0x17
#define ads_wct1 0x18
#define ads_wct2 0x19

// Define functions:
char ads_sendstart(void);
char ads_sendreset(void);
char ads_sendwakeup(void);
char ads_sendstandbye(void);
char ads_sendsdatac(void);
char ads_sendrdatac(int samples, char *buffer);
char ads_sendrdata16(signed int *ch_samples);
char ads_sendrdata8(char *byte0,char *byte1);
char ads_sendreadreg(char reg);
char ads_sendwritereg(char reg, char value);
char ads_sendwriteallreg(char *bufferpointer);
void ads_sendreadallreg(char *buffer);
void ads_getstdconfig(char *ads_stdconfig);
void ads_set_ncs(char set);
void wait(int cycles);
void ads_spi_init(void);
char ads_set_gain(int channel, int value);
void ads_set_startpin(char set);
```

A.1.4 hal_MMC_hardware_board.h

```
// microSDHC connections
#include "msp430f2618.h"
// SPI Port
#define SPI_PxSEL         P3SEL
#define SPI_PxDIR         P3DIR
#define SPI_PxIN          P3IN
#define SPI_PxOUT         P3OUT
#define SPI_SIMO          BIT4
#define SPI_SOMI          BIT5
#define SPI_UCLK          BIT0
```

```
// SPI port definitions              // Adjust the values for the chosen
#define MMC_PxSEL         SPI_PxSEL      // interfaces, according to the pin
#define MMC_PxDIR         SPI_PxDIR      // assignments indicated in the
#define MMC_PxIN          SPI_PxIN       // chosen MSP430 device datasheet.
#define MMC_PxOUT         SPI_PxOUT
#define MMC_SIMO          SPI_SIMO
#define MMC_SOMI          SPI_SOMI
#define MMC_UCLK          SPI_UCLK
// Chip Select
#define MMC_CS_PxOUT      P4OUT
#define MMC_CS_PxDIR      P4DIR
#define MMC_CS            BIT0
#define CS_LOW()    MMC_CS_PxOUT &= ~MMC_CS              // Card Select
#define CS_HIGH()   while(!halSPITXREADY); MMC_CS_PxOUT |= MMC_CS //!halSPITXDONE); MMC_CS_PxOUT |= MMC_CS  // Card Deselect

#define DUMMY_CHAR 0xFF
```

A.1.5 hal_SPI.c

```
// SPI Interface functions

#ifndef _SPILIB_C
#define _SPILIB_C

#include "prototype.h"

#ifndef DUMMY_CHAR
#define DUMMY_CHAR 0xFF
#endif

void halSPISetup(void) // SPI config. for the microSHC card
{
  UCA0CTL0 = UCMST+UCCKPL+UCMSB+UCSYNC;     // 3-pin, 8-bit SPI master
  UCA0CTL1 = UCSSEL_2 + UCSWRST;            // SMCLK
  UCA0BR0 |= 0x02;                          // UCLK/2
  UCA0BR1 = 0;
  UCA0MCTL = 0;
  UCA0CTL1 &= ~UCSWRST;                     // **Initialize USCI state machine**
}

//Send one byte via SPI
unsigned char spiSendByte(const unsigned char data)
{
  while (halSPITXREADY ==0);    // wait while not ready for TX
  halSPI_SEND(data);            // write
 // while (halSPIRXREADY ==0);    // wait for RX buffer (full)
  while(halSPITXDONE);
  return (halSPIRXBUF);
}

//Read a frame of bytes via SPI
unsigned char spiReadFrame(unsigned char* pBuffer, unsigned int size)
{
#ifndef withDMA
  unsigned long i = 0;
  // clock the actual data transfer and receive the bytes; spi_read automatically finds the Data Block
  for (i = 0; i < size; i++){
    while (halSPITXREADY ==0);   // wait while not ready for TX
    halSPI_SEND(DUMMY_CHAR);     // dummy write
    //while (halSPIRXREADY ==0);   // wait for RX buffer (full)
       while(halSPITXDONE);
    pBuffer[i] = halSPIRXBUF;
  }
#else
        U1IFG &= ~(URXIFG1 + URXIFG1);      /* clear flags */
        /* Get the block */
        /* DMA trigger is UART1 receive for both DMA0 and DMA1 */
        DMACTL0 &= ~(DMA0TSEL_15 | DMA1TSEL_15);
        DMACTL0 |= (DMA0TSEL_9 | DMA1TSEL_9);
        /* Source DMA address: receive register.  */
        DMA0SA = U1RXBUF_;
        /* Destination DMA address: the user data buffer. */
        DMA0DA = (unsigned short)pBuffer;
        /* The size of the block to be transferred */
        DMA0SZ = size;
        /* Configure the DMA transfer*/
        DMA0CTL =
          DMAIE   |                         /* Enable interrupt */
          DMADT_0 |                         /* Single transfer mode */
          DMASBDB |                         /* Byte mode */
          DMAEN |                           /* Enable DMA */
          DMADSTINCR1 | DMADSTINCR0;        /* Increment the destination address */

        /* We depend on the DMA priorities here.  Both triggers occur at
           the same time, since the source is identical.  DMA0 is handled
           first, and retrieves the byte.  DMA1 is triggered next, and
```

```
          sends the next byte. */
        /* Source DMA address: constant 0xFF (don't increment)*/
        DMA1SA = U1TXBUF_;
        /* Destination DMA address: the transmit buffer. */
        DMA1DA = U1TXBUF_;
        /* Increment the destination address */
        /* The size of the block to be transferred */
        DMA1SZ = count-1;
        /* Configure the DMA transfer*/
        DMA1CTL =
          DMADT_0 |                         /* Single transfer mode */
          DMASBDB |                         /* Byte mode */
          DMAEN;                            /* Enable DMA */

        /* Kick off the transfer by sending the first byte */
        halMMC_SEND(0xFF);
        _EINT(); LPM0;  // wait till done
#endif
  return(0);
}

//Send a frame of bytes via SPI
unsigned char spiSendFrame(unsigned char* pBuffer, unsigned int size)
{
#ifndef withDMA
  unsigned long i = 0;
  // clock the actual data transfer and receive the bytes; spi_read automatically finds the Data Block
  for (i = 0; i < size; i++){
    while (halSPITXREADY ==0);   // wait while not ready for TX
    halSPI_SEND(pBuffer[i]);     // write
   // while (halSPIRXREADY ==0);   // wait for RX buffer (full)
   while(halSPITXDONE);
    pBuffer[i] = halSPIRXBUF;
  }
#else
      /* Get the block */
      /* DMA trigger is UART send */
      DMACTL0 &= ~(DMA0TSEL_15);
      DMACTL0 |= (DMA0TSEL_9);
      /* Source DMA address: the data buffer.  */
      DMA0SA = (unsigned short)pBuffer;
      /* Destination DMA address: the UART send register. */
      DMA0DA = U1TXBUF_;
      /* The size of the block to be transferred */
      DMA0SZ = count;
      /* Configure the DMA transfer*/
      DMA0CTL =
        DMAREQ  |                         /* start transfer */
        DMADT_0 |                         /* Single transfer mode */
        DMASBDB |                         /* Byte mode */
        DMAEN |                           /* Enable DMA */
        DMASRCINCR1 | DMASRCINCR0;        /* Increment the source address */
#endif
  return(0);
}


#ifdef withDMA
#ifdef __IAR_SYSTEMS_ICC__
#if __VER__ < 200
interrupt[DACDMA_VECTOR] void DMA_isr(void)
#else
#pragma vector = DACDMA_VECTOR
__interrupt void DMA_isr(void)
#endif
#endif

#ifdef __CROSSWORKS__
void DMA_isr(void)   __interrupt[DACDMA_VECTOR]
#endif

#ifdef __TI_COMPILER_VERSION__
__interrupt void DMA_isr(void);
DMA_ISR(DMA_isr)
__interrupt void DMA_isr(void)
#endif
{
  DMA0CTL &= ~(DMAIFG);
  LPM3_EXIT;
}
#endif

#endif /* _SPILIB_C */
```

A.1.6 hal_SPI.h

```
// Filename:  hal_SPI.h:
// Declarations for Communication with the microSD card in SPI mode.

#ifndef _SPILIB_H
#define _SPILIB_H

#include "prototype.h"

// Use USCI_A0 port for the microSD card
 #define halSPIRXBUF  UCA0RXBUF
 #define halSPI_SEND(x) UCA0TXBUF=x
 #define halSPITXREADY  (IFG1&UCA0TXIFG)         /* Wait for TX to be ready */
 #define halSPITXDONE  (UCA0STAT&UCBUSY)       /* Wait for TX to finish */
 #define halSPIRXREADY (IFG1&UCA0RXIFG)          /* Wait for TX to be ready */
 #define halSPIRXFG_CLR IFG1 &= ~UCA0RXIFG
 #define halSPI_PxIN  SPI_USART0_PxIN
 #define halSPI_SOMI  SPI_USART0_SOMI

// Function Prototypes
void halSPISetup (void);
unsigned char spiSendByte(const unsigned char data);
unsigned char spiReadFrame(unsigned char* pBuffer, unsigned int size);
unsigned char spiSendFrame(unsigned char* pBuffer, unsigned int size);

#endif /* _SPILIB_H */
```

A.1.7 mmc.c

```
// The microSDHC card functions

#ifndef _MMCLIB_C
#define _MMCLIB_C
#include "prototype.h"

// Function Prototypes
char mmcGetResponse(void);
char mmcGetXXResponse(const char resp);
char mmcCheckBusy(void);
char mmcGoIdle();

char mmc_buffer[512] = { 0 }; // Buffer for mmc i/o
char mmcInit(void) // Initialize MMC card
{
  int i;
  // SPI signal lines
  MMC_PxOUT |= MMC_SIMO + MMC_UCLK;
  MMC_PxDIR |= MMC_SIMO + MMC_UCLK;
  MMC_PxDIR &= ~MMC_SOMI;
  // Chip Select
  MMC_CS_PxOUT |= MMC_CS;
  MMC_CS_PxDIR |= MMC_CS;
  // Init. SPI Module
  halSPISetup();
  MMC_PxSEL |= MMC_SIMO + MMC_SOMI + MMC_UCLK;

  //initialization sequence on PowerUp
  CS_HIGH(); // Raise CS
  for(i=0;i<=9;i++)
    spiSendByte(DUMMY_CHAR);

  return (mmcGoIdle());
}

char mmcGoIdle() // set MMC in Idle mode
{
  char response=0x01;
  CS_LOW();
  mmcSendCmd(MMC_GO_IDLE_STATE,0,0x95); // Put MMC in SPI mode
  if(mmcGetResponse()!=0x01) // Wait for READY RESPONSE
    return MMC_INIT_ERROR;
  mmcSendCmd(0x48,0x1AA,0x87); // SEND_IF_COND command
  sd_delay(8);
  mmcSendCmd(0x77,0,0xFF); // APP_CMD command
  sd_delay(8);
  while(response!=0x00) // card valid?
  {
    CS_HIGH();
    spiSendByte(DUMMY_CHAR);
    mmcSendCmd(0x69,0x40000000,0xFF); // acmd41
    response=mmcGetResponse();
    sd_delay(8);
    mmcSendCmd(0x41,0x40000000,0xFF); // cmd41
    response=mmcGetResponse();
```

```
  }
  CS_HIGH();
  spiSendByte(DUMMY_CHAR);
  return (MMC_SUCCESS);
}

char mmcGetResponse(void)
{
  //Response comes 1-8bytes after command
  //the first bit will be a 0
  //followed by an error code
  //data will be 0xff until response
  int i=0;

  char response;

  while(i<=64)
  {
    response=spiSendByte(DUMMY_CHAR);
    if(response==0x00)break;
    if(response==0x01)break;
    i++;
  }
  return response;
}

char mmcGetXXResponse(const char resp)
{
  //Response comes 1-8bytes after command
  //the first bit will be a 0
  //followed by an error code
  //data will be 0xff until response
  int i=0;

  char response;

  while(i<=1000)
  {
    response=spiSendByte(DUMMY_CHAR);
    if(response==resp)break;
    i++;
  }
  return response;
}

// Check if MMC card is still busy
char mmcCheckBusy(void)
{
  int i=0;
  char response;
  do
  {
    response=spiSendByte(DUMMY_CHAR);
    i++;
  }while(response==0);
  return response;
}

char mmcReadBlock(const unsigned long address, const unsigned long count, unsigned char *pBuffer)
{
  char rvalue = MMC_RESPONSE_ERROR;

  // Set the block length to read
  if (mmcSetBlockLength (count) == MMC_SUCCESS) // block length could be set
  {
    CS_LOW ();
    // send read command MMC_READ_SINGLE_BLOCK=CMD17
    mmcSendCmd (MMC_READ_SINGLE_BLOCK,address, 0xFF);
    // Send 8 Clock pulses of delay, check if the MMC acknowledged the read block command
    wait(8);
    if (mmcGetResponse() == 0x00)
    {
      // now look for the data token to signify the start of the data
      if (mmcGetXXResponse(MMC_START_DATA_BLOCK_TOKEN) == MMC_START_DATA_BLOCK_TOKEN)
      {
        // clock the actual data transfer and receive the bytes; spi_read automatically finds the Data Block
        spiReadFrame(pBuffer, count);
        // get CRC bytes (not really needed by us, but required by the card)
        spiSendByte(DUMMY_CHAR);
        rvalue = MMC_SUCCESS;
      }
      else
      {
        // the data token was never received
        rvalue = MMC_DATA_TOKEN_ERROR;
      }
    }
    else
    {
      // the card never acknowledge the read command
```

```
      rvalue = MMC_RESPONSE_ERROR;
    }
  }
  else
  {
    rvalue = MMC_BLOCK_SET_ERROR;
  }
  CS_HIGH ();
  spiSendByte(DUMMY_CHAR);
  return rvalue;
}// mmc_read_block


char mmcWriteBlock (  unsigned long address, const unsigned long count, unsigned char *pBuffer)
{
  int response = 0;
  char rvalue = MMC_RESPONSE_ERROR;

  if (mmcSetBlockLength (count) == MMC_SUCCESS)   // block length could be set
  {
    CS_LOW ();// CS = LOW (on)
    mmcSendCmd (MMC_WRITE_BLOCK,address, 0xFF); // send write command
    if (mmcGetXXResponse(MMC_R1_RESPONSE) == MMC_R1_RESPONSE) // 0x00 is no errors
    {
      spiSendByte(DUMMY_CHAR);
      spiSendByte(0xfe); // Send the data token to signify the start of the data
  spiSendFrame(pBuffer, count);
  sd_delay(3);

      while(response==0) {
       response=spiSendByte(DUMMY_CHAR);

      }
      rvalue = MMC_SUCCESS;

    }

  }

  CS_HIGH ();
  spiSendByte(DUMMY_CHAR);// Send 8 Clock pulses of delay.
  return rvalue;

}
//

// send command to MMC
void mmcSendCmd (const char cmd, unsigned long data, const char crc)
{
  unsigned char frame[6];
  char temp;
  int i;
  frame[0]=(cmd|0x40);
  for(i=3;i>=0;i--){
    temp=(char)(data>>(8*i));
    frame[4-i]=(temp);
  }
  frame[5]=(crc);
  spiSendFrame(frame,6);
}


//--------------- set blocklength 2^n -------------------------------------------------------
char mmcSetBlockLength (const unsigned long blocklength)
{
  // CS = LOW (on)
  CS_LOW ();
  // Set the block length to read
  mmcSendCmd(MMC_SET_BLOCKLEN, blocklength, 0xFF);

  // get response from MMC - make sure that its 0x00 (R1 ok response format)
  if(mmcGetResponse()!=0x00)
  { mmcInit();
    mmcSendCmd(MMC_SET_BLOCKLEN, blocklength, 0xFF);
    mmcGetResponse();
  }

  CS_HIGH ();

  // Send 8 Clock pulses of delay.
  spiSendByte(DUMMY_CHAR);

  return MMC_SUCCESS;
} // Set block_length


// Reading the contents of the CSD and CID registers in SPI mode is a simple
// read-block transaction.
char mmcReadRegister (const char cmd_register, const unsigned char length, unsigned char *pBuffer)
{
```

```
  char uc = 0;
  char rvalue = MMC_TIMEOUT_ERROR;

  if (mmcSetBlockLength (length) == MMC_SUCCESS)
  {
    CS_LOW ();
    // CRC not used: 0xff as last byte
    mmcSendCmd(cmd_register, 0x000000, 0xff);

    // wait for response
    // in the R1 format (0x00 is no errors)
    if (mmcGetResponse() == 0x00)
    {
      if (mmcGetXXResponse(0xfe)== 0xfe)
        for (uc = 0; uc < length; uc++)
          pBuffer[uc] = spiSendByte(DUMMY_CHAR);  //mmc_buffer[uc] = spiSendByte(0xff);
      // get CRC bytes (not really needed by us, but required by MMC)
      spiSendByte(DUMMY_CHAR);
      spiSendByte(DUMMY_CHAR);
      rvalue = MMC_SUCCESS;
    }
    else
      rvalue = MMC_RESPONSE_ERROR;
    // CS = HIGH (off)
    CS_HIGH ();

    // Send 8 Clock pulses of delay.
    spiSendByte(DUMMY_CHAR);
  }
  CS_HIGH ();
  return rvalue;
} // mmc_read_register

void sd_delay(char number)
{
char i;
/* Null for now */
for (i=0; i<number; i++)
{
/* Clock out an idle byte (0xFF) */
spiSendByte(DUMMY_CHAR);
}
}
//-------------------------------------------------------------------
#endif /* _MMCLIB_C */
```

A.1.8 mmc.h

```
// Declarations for Communication with the MMC
#ifndef _MMCLIB_H
#define _MMCLIB_H

#include "msp430f2618.h"
// macro defines
#define HIGH(a) ((a>>8)&0xFF)               // high byte from word
#define LOW(a)  (a&0xFF)                    // low byte from word

#define DUMMY 0xff


// Tokens (necessary  because at NPO/IDLE (and CS active) only 0xff is on the data/command line)
#define MMC_START_DATA_BLOCK_TOKEN          0xfe   // Data token start byte, Start Single Block Read
#define MMC_START_DATA_MULTIPLE_BLOCK_READ  0xfe   // Data token start byte, Start Multiple Block Read
#define MMC_START_DATA_BLOCK_WRITE          0xfe   // Data token start byte, Start Single Block Write
#define MMC_START_DATA_MULTIPLE_BLOCK_WRITE 0xfc   // Data token start byte, Start Multiple Block Write
#define MMC_STOP_DATA_MULTIPLE_BLOCK_WRITE  0xfd   // Data toke stop byte, Stop Multiple Block Write


// an affirmative R1 response (no errors)
#define MMC_R1_RESPONSE       0x00


// this variable will be used to track the current block length
// this allows the block length to be set only when needed
// unsigned long _BlockLength = 0;

// error/success codes
#define MMC_SUCCESS           0x00
#define MMC_BLOCK_SET_ERROR   0x01
#define MMC_RESPONSE_ERROR    0x02
#define MMC_DATA_TOKEN_ERROR  0x03
#define MMC_INIT_ERROR        0x04
#define MMC_CRC_ERROR         0x10
#define MMC_WRITE_ERROR       0x11
#define MMC_OTHER_ERROR       0x12
```

```
#define MMC_TIMEOUT_ERROR     0xFF


// commands: first bit 0 (start bit), second 1 (transmission bit); CMD-number + Offsett 0x40
#define MMC_GO_IDLE_STATE          0x40     //CMD0
#define MMC_SEND_OP_COND           0x41     //CMD1
#define MMC_SEND_IF_COND           0x48     //CMD8
#define MMC_READ_CSD               0x49     //CMD9
#define MMC_SEND_CID               0x4a     //CMD10
#define MMC_STOP_TRANSMISSION      0x4c     //CMD12
#define MMC_SEND_STATUS            0x4d     //CMD13
#define MMC_SET_BLOCKLEN           0x50     //CMD16 Set block length for next read/write
#define MMC_READ_SINGLE_BLOCK      0x51     //CMD17 Read block from memory
#define MMC_READ_MULTIPLE_BLOCK    0x52     //CMD18
#define MMC_CMD_WRITEBLOCK         0x54     //CMD20 Write block to memory
#define MMC_WRITE_BLOCK            0x58     //CMD24
#define MMC_WRITE_MULTIPLE_BLOCK   0x59     //CMD25
#define MMC_WRITE_CSD              0x5b     //CMD27 PROGRAM_CSD
#define MMC_SET_WRITE_PROT         0x5c     //CMD28
#define MMC_CLR_WRITE_PROT         0x5d     //CMD29
#define MMC_SEND_WRITE_PROT        0x5e     //CMD30
#define MMC_TAG_SECTOR_START       0x60     //CMD32
#define MMC_TAG_SECTOR_END         0x61     //CMD33
#define MMC_UNTAG_SECTOR           0x62     //CMD34
#define MMC_TAG_EREASE_GROUP_START 0x63     //CMD35
#define MMC_TAG_EREASE_GROUP_END   0x64     //CMD36
#define MMC_UNTAG_EREASE_GROUP     0x65     //CMD37
#define MMC_EREASE                 0x66     //CMD38
#define MMC_READ_OCR               0x67     //CMD39
#define MMC_CRC_ON_OFF             0x68     //CMD40


// mmc init
char mmcInit(void);

// check if MMC card is present
char mmcPing(void);

// send command to MMC
void mmcSendCmd (const char cmd, unsigned long data, const char crc);

// set MMC in Idle mode
char mmcGoIdle();

// set MMC block length of count=2^n Byte
char mmcSetBlockLength (const unsigned long);

// read a size Byte big block beginning at the address.
char mmcReadBlock(const unsigned long address, const unsigned long count, unsigned char *pBuffer);
#define mmcReadSector(sector, pBuffer) mmcReadBlock(sector*512ul, 512, pBuffer)

// write a 512 Byte big block beginning at the (aligned) address
char mmcWriteBlock ( unsigned long address, const unsigned long count, unsigned char *pBuffer);
#define mmcWriteSector(sector, pBuffer) mmcWriteBlock(sector*512ul, 512, pBuffer)

// Read Register arg1 with Length arg2 (into the buffer)
char mmcReadRegister(const char, const unsigned char, unsigned char *pBuffer);

// Read the Card Size from the CSD Register
unsigned long mmcReadCardSize(void);
void sd_delay(char);

#endif /* _MMCLIB_H */
```

A.1.9 TI_USCI_I2C_master.c

```
//   MSP430 USCI I2C Transmitter and Receiver
#include "prototype.h"

signed char byteCtr;
int k;
unsigned int *TI_receive_field;
unsigned int *TI_transmit_field;

// This function initializes the USCI module for master-receive operation.
void TI_USCI_I2C_receiveinit(unsigned char slave_address,
                             unsigned char prescale){
  P5SEL |= SDA_PIN + SCL_PIN;                // Assign I2C pins to USCI_B0
  UCB1CTL1 = UCSWRST;                       // Enable SW reset
  UCB1CTL0 = UCMST + UCMODE_3 + UCSYNC;      // I2C Master, synchronous mode
  UCB1CTL1 = UCSSEL_2 + UCSWRST;             // Use SMCLK, keep SW reset
  UCB1BR0 = prescale;                        // set prescaler
  UCB1BR1 = 0;
  UCB1I2CSA = slave_address;                 // set slave address
  UCB1CTL1 &= ~UCSWRST;                      // Clear SW reset, resume operation
  UCB1I2CIE = UCNACKIE;
  UC1IE = UCB1RXIE;                           // Enable RX interrupt
```

```
}
// This function initializes the USCI module for master-transmit operation.
void TI_USCI_I2C_transmitinit(unsigned char slave_address,
                              unsigned char prescale){
  P5SEL |= SDA_PIN + SCL_PIN;                 // Assign I2C pins to USCI_B0
  UCB1CTL1 = UCSWRST;                        // Enable SW reset
  UCB1CTL0 = UCMST + UCMODE_3 + UCSYNC;       // I2C Master, synchronous mode
  UCB1CTL1 = UCSSEL_2 + UCSWRST;              // Use SMCLK, keep SW reset
  UCB1BR0 = prescale;                         // set prescaler
  UCB1BR1 = 0;
  UCB1I2CSA = slave_address;                  // Set slave address
  UCB1CTL1 &= ~UCSWRST;                       // Clear SW reset, resume operation
  UCB1I2CIE = UCNACKIE;
  UC1IE = UCB1TXIE;                            // Enable TX ready interrupt
}
// This function is used to start an I2C commuincation in master-receiver mode.
void TI_USCI_I2C_receive(unsigned char byteCount, unsigned int *field){
  TI_receive_field = field;
  if ( byteCount == 1 ){
    byteCtr = 0 ;
    __disable_interrupt();
    UCB1CTL1 |= UCTXSTT;                      // I2C start condition
    while (UCB1CTL1 & UCTXSTT);               // Start condition sent?
    UCB1CTL1 |= UCTXSTP;                      // I2C stop condition
    __enable_interrupt();
  } else if ( byteCount > 1 ) {
   // byteCtr = byteCount - 2 ;
    byteCtr = byteCount - 1 ;
    UCB1CTL1 |= UCTXSTT;                      // I2C start condition
  } else
    while (1);                                // illegal parameter
}
// This function is used to start an I2C commuincation in master-transmit mode.
void TI_USCI_I2C_transmit(unsigned char byteCount, unsigned int *field){
  TI_transmit_field = field;
  byteCtr = byteCount;
  UCB1CTL1 |= UCTR + UCTXSTT;                 // I2C TX, start condition
}
// This function is used to look for a slave address on the I2C bus.
unsigned char TI_USCI_I2C_slave_present(unsigned char slave_address){
  unsigned char uc1ie_bak, slaveadr_bak, ucb1i2cie, returnValue;
   ucb1i2cie = UCB1I2CIE;                      // restore old UCB0I2CIE
  //ie2_bak = IE2;                              // store IE2 register
  uc1ie_bak     =  UC1IE;
  slaveadr_bak = UCB1I2CSA;                   // store old slave address
  UCB1I2CIE &= ~ UCNACKIE;                    // no NACK interrupt
  UCB1I2CSA = slave_address;                  // set slave address
  UC1IE  &= ~(UCB1TXIE + UCB1RXIE);             // no RX or TX interrupts
  __disable_interrupt();
  UCB1CTL1 |= UCTR + UCTXSTT + UCTXSTP;       // I2C TX, start condition
  while (UCB1CTL1 & UCTXSTP);                 // wait for STOP condition

  returnValue = !(UCB1STAT & UCNACKIFG);
  __enable_interrupt();
  UC1IE = uc1ie_bak;                            // restore IE2
  UCB1I2CSA = slaveadr_bak;                   // restore old slave address
  UCB1I2CIE =  ucb1i2cie;                      // restore old UCB0CTL1
  return returnValue;                         // return whether or not
                                              // a NACK occured
}
// This function is used to check if there is commuincation in progress.
unsigned char TI_USCI_I2C_notready(){
  return (UCB1STAT & UCBBUSY);
}


#pragma vector = USCIAB1RX_VECTOR
__interrupt void USCIAB1RX_ISR(void)
{
  if (UCB1STAT & UCNACKIFG){            // send STOP if slave sends NACK
    UCB1CTL1 |= UCTXSTP;
    UCB1STAT &= ~UCNACKIFG;
  }

}


#pragma vector = USCIAB1TX_VECTOR
__interrupt void USCIAB1TX_ISR(void)
{
  //if (IFG2 & UCB1RXIFG){
   if (UC1IFG & UCB1RXIFG) {
    if ( byteCtr == 0 ){
      UCB1CTL1 |= UCTXSTP;                    // I2C stop condition
      *TI_receive_field = UCB1RXBUF;
      TI_receive_field++;
    }
    else {
      *TI_receive_field = UCB1RXBUF;
      TI_receive_field++;
```

```
      byteCtr--;
    }
  }
  else {
    if (byteCtr == 0){
      UCB1CTL1 |= UCTXSTP;                    // I2C stop condition
      //IFG2 &= ~UCB1TXIFG;                     // Clear USCI_B0 TX int flag
      UC1IFG &= ~UCB1TXIFG;
    }
    else {
      UCB1TXBUF = *TI_transmit_field;
      TI_transmit_field++;
      byteCtr--;
    }
  }
}
```

A.1.10 TI_USCI_I2C_master.h

```
#ifndef USCI_LIB
#define USCI_LIB

#define SDA_PIN 0x02                                // msp430x261x UCB01DA pin1
#define SCL_PIN 0x04                                // msp430x261x UCB1SCL pin2

void TI_USCI_I2C_receiveinit(unsigned char slave_address, unsigned char prescale);
void TI_USCI_I2C_transmitinit(unsigned char slave_address, unsigned char prescale);


void TI_USCI_I2C_receive(unsigned char byteCount, unsigned int *field);
void TI_USCI_I2C_transmit(unsigned char byteCount, unsigned int *field);


unsigned char TI_USCI_I2C_slave_present(unsigned char slave_address);
unsigned char TI_USCI_I2C_notready();


#endif
```

A.2 Schematic

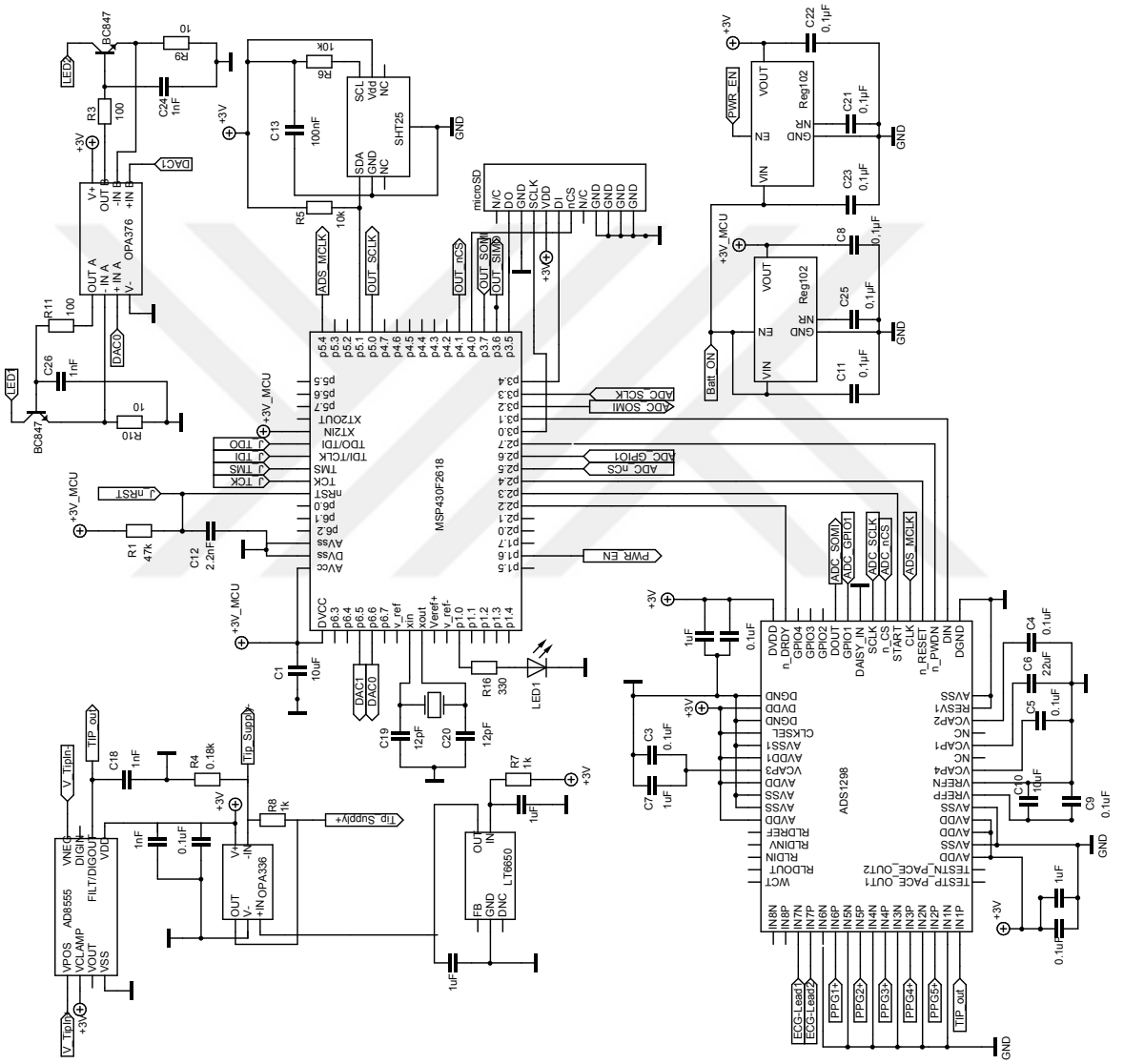


Figure A.1: The schematic of the sensor read-out system

A.3 PCB layout

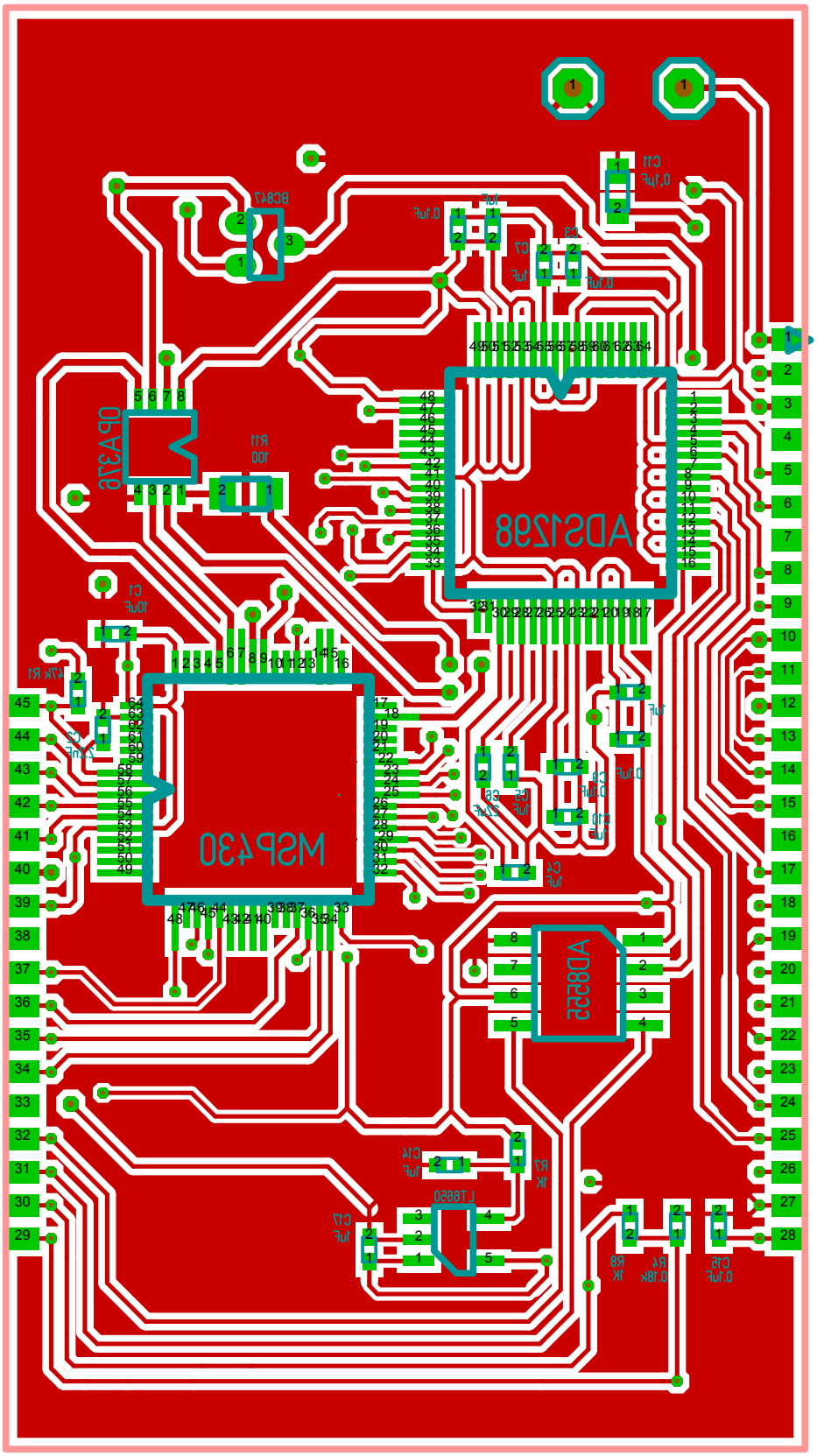


Figure A.2: Top view of the PCB Layout for the sensor read-out system

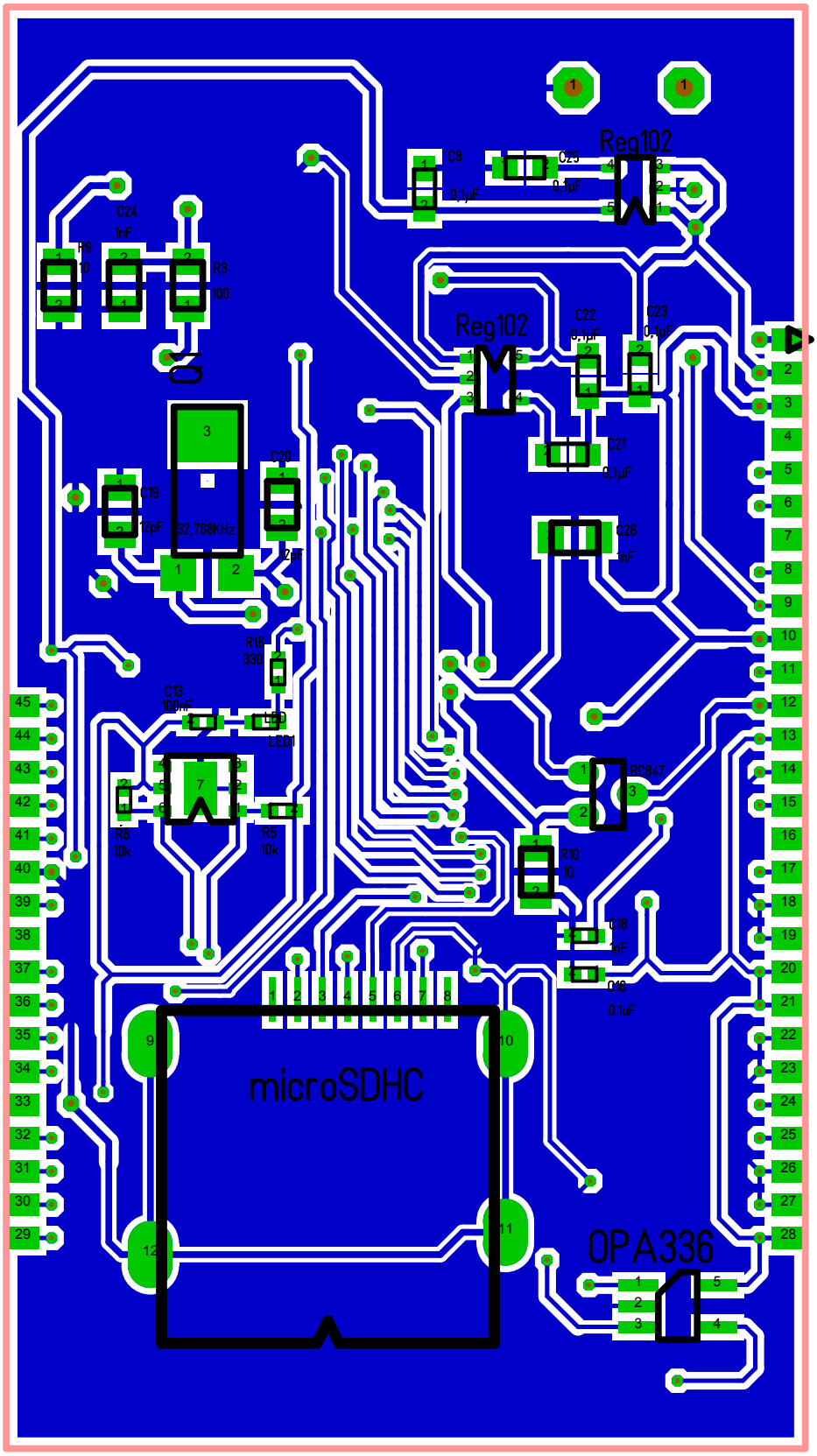


Figure A.3: Bottom view of the PCB Layout for the sensor read-out system

A.4 Pin configuration

Pin No.	I/O	Description
1	I/O	Positive terminal of the battery
2	I/O	On/Off Switch
3	I/O	Ground signal of PCB
4, 7, 16, 33, 38	-	NC
5	input	Positive ECG Lead
6	input	Negative ECG Lead
8, 15, 17, 24, 25	input	Sensor Output
9, 12	input	LED Drive
10, 14, 18, 19, 22, 23, 26, 27	output	Analog ground of the sensors
11, 13, 20, 21, 28	output	Analog supply of the sensors
29	output	Positive Supply of the Tip Catheter
30	output	Negative Supply of the Tip Catheter
31	input	Tip Catheter Output Positive Terminal
32	input	Tip Catheter Output Negative Terminal
34	outpu	SPI slave data in/master out
35	input	SPI slave data out/master in
36	output	SPI chip select
37	output	SPI clock
39	output	Analog supply of JTAG
40	output	Analog ground of JTAG
41	input	JTAG data output
42	output	JTAG data input/TCLK input
43	output	Signal to control the JTAG state machine
44	output	JTAG clock input
45	output	JTAG reset; active low

References

Allen, J. (2007). Photoplethysmography and its application in clinical physiological measurement. *Physiological Measurement 28*, R1–R39.

Davies, J. H. (2008). *MSP430 Microcontroller Basics*. Elsevier Ltd.

Donaldson, P. (1976). The encapsulation of microelectronic devices for long-term surgical implantation. *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING BME-23*, 281–285.

Donaldson, P. E. K. (1983, May). The cooper cable: an implantable multiconductor cable for neurological prostheses. Technical report, MRC Neurological Prosthesis Unit.

Donaldson, P. E. K. (1985, March). The craggs connector: a termination for cooper cable. Technical report, MRC Neurological Prosthesis Unit.

Fiala, J., P. Bingger, K. Foerster, C. Heilmann, F. Beyersdorf, H. Zappe, and A. Seifert (2010). Implantable sensor for blood pressure determination via pulse transit time. In *IEEE Sensors Conference*.

Goto, K., T. Nakagawa, O. Nakamura, and S. Kawata (2001, July). An implantable power supply with an optically rechargeable lithium battery. Technical Report 7, IEEE Transactions on Biomedical Engineering.

Hutten, H., P. Kastner, G. Schreier, and M. Schaldach (1998). Hemodynamic assessment by evaluation of intramyocardial electrograms. In *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*.

Karakas, U. (2012, January). Lock-in technique for implantable cardiovascular sensors. Master's thesis, University of Freiburg.

Klabunde, R. E. (2005). *Cardiovascular Physiology Concepts*. Lippincott Williams and Wilkings.

Klabunde, R. E. (2011). Cardiac cycle. http://www.cvphysiology.com/Heart

Klein, G., D. Warkentin, K. Riff, and B. Lee (1998). *Minimally Invasive Implantable ECG Monitoring for Diagnosis of Atrial Flutter*. Medtronic Inc. United States Patent.

Kretzschmar, U. and C. Hernitscheck (2007, December). Using the usci i2c slave. Technical report, Texas Instruments Incorporated.

Kyriacou, P. (2006). Pulse oximetry in the oesophagus. *Physiological Measurement 27*, R1–R35.

Lobban, J. and C. Stafford (2008). Implantable ecg monitoring for diagnosis of atrial flutter. *EP Lab Digest*.

Maluf, N. and K. Williams (2004). *An Introduction to Microelectromechanical Systems Engineering* (Second ed.). Artech House Inc.

Marcinkevics, Z., M. Greve, J. Aivars, R. Erts, and A. Zehtabi (2009). Relationship between arterial pressure and pulse wave velocity using photoplethysmography during the post-exercise recovery period. *Acta Universitatis Latviensis 753*, 59–68.

Mendelson, Y., J. Duckworth, and G. Comtois (2006). A wearable reflectance pulse oximeter for remote physiological monitoring. In *Proceedings of the 28th IEEE EMBS Annual International Conference*.

Miller, J. C. and L. Monge (1946, May). Body temperature and respiration rate and their relation to adaptability in sheep. *Journal of Animal Science 5*, 147–153.

Mokwa, W. and U. Schnakenberg (2001). Micro-transponder systems for medical applications. *IEEE Transactions on Instrumentation and Measurement 50*, 1551–1555.

Murray, W. and P. Foster (1996). The peripheral pulse wave: Information overlooked. *Journal of Clinical Monitoring 12*, 365–377.

Potkay, J. A. (2007). *Long term, implantable blood pressure monitoring systems*. Springer Science + Business Media.

Rangayyan, R. M. (2002). *Bimedical Signal Analysis A Case-Study Approach*. John Wiley & Sons Inc.

Riistama, J., J. Vaisanen, S. Heinisuo, H. Harjunpää, S. Arra, K. Kokko, M.Mäntylä, J. Kaihilahti, P. Heino, and M. Kellomäki (2007). Wireless and inductively powered implant for measuring electrocardiogram. *International Federation for Medical and Biological Engineering 45*, 1163–1174.

Schuettler, M. (2009). Fertigung von hermetischen implantatkapseln am lehrstuhl bmt. interne Dokumentation des Lehrstuhls für Biomedizinische Mikro-technik der Universität Freiburg.

Schuettler, M., M. Huegle, J. Ordonez, J. Wilde, and T. Stieglitz (2010). A device for vacuum drying, inert gas backfilling and solder sealing of hermetic implant packages. In *32nd Annual International Conference of the IEEE EMBS*.

Schuettler, M., J. Ordonez, T. Santisteban, A. Schatz, J. Wilde, and T. Stieglitz (2010). Fabrication and test of a hermetic miniature implant package with 360 electrical feedthroughs. In *32nd Annual International Conference of the IEEE EMBS*.

Sensirion (2011, December). Datasheet sht25.

Song, Z., J. Jenkins, M. Burke, and R. Arzbaecher (2004). The feasibility of st-segment monitoring with a subcutaneous device. *Journal of Electrocardiology 37*, 174–179.

Texas Instruments Incorporated (2004, December). *MSP430x2xx Family User's Guide*. Texas Instruments Incorporated.

Texas Instruments Incorporated (2010, July). *MSP430™ Programming Via the JTAG Interface*. Texas Instruments Incorporated.

Texas Instruments Incorporated (2011, July). *MSP430F261x Mixed Signal Microcontroller*, Volume J. Texas Instruments Incorporated.

Texas Instruments Incorporated (2012, January). *Low-Power, 8-Channel, 24-Bit Analog Front-End for Biopotential Measurements*, Volume I. Texas Instruments Incorporated.

Note of Thanks

It would not have been possible to write this Master thesis without the help and contribution of the kind people, whom I would like to mention here.

I want first to thank Prof. Dr. Hans Zappe for giving me the opportunity to write my Master thesis on a very innovative and educative subject, at the Laboratory of Micro-optics. It was an invaluable experience for me to be welcomed and introduced to the OPT4LIFE project. I also wish to thank Prof. Dr. Peter Woias who agreed to participate in the evaluation this thesis as the second referee.

I would like to express my gratitude to my supervisor Dipl.-Ing. Michael Theodor. I can not imagine completing this thesis without his guidance and his valuable technical and theoretical support. I am very grateful to his helpfulness, attention and insight through all the steps of this thesis. I specially thank him for his infinite patience. I will never forget his sincerity and encouragement.

Special thanks are due to Dr.-Ing. Martin Schüttler and Dipl.-Ing. Juan Sebastian Ordonez from Laboratory for Biomedical Microtechnology, for their mentoring and collaborations in hermetic implant packaging.

I also wish to express my appreciation to Katharina Förster for her efforts and expertise at the implant surgeries.

I would like to acknowledge the financial, academic and technical support of the Laboratory of Micro-Optics and its staff. I want to express my gratitude to Dr. rer. nat. Andreas Seifert for his constructive suggestions and discussions on the OPT4LIFE project, and also for proofreading this thesis. My gratitude is also due to Dipl.-Ing. Bernd Aatz, without the help of whom I could not have managed to fabricate such an efficient implant package. I am very grateful to Utku Karakaş, Dominic Ruh and all my colleagues who were always willing to help.

Last but not the least, I want to thank to my family who have always been encouraging and supporting me. I would never have been able to achieve so much without them.