Aalborg University
7th semester project

Gait Sensor - For Monitoring Movement Before Fall

Participants:
Christoffer Lundager Nedergaard

Supervisor:
Sofus Birkedal Nielsen

7th of January 2016

AALBORG UNIVERSITET
STUDENTERRAPPORT

Instition for Electronic and IT
Fredrik Bajers Vej 7
9220 Aalborg E
Phone: 96 35 86 90
http://es.aau.dk

Title: Gait Sensor
Theme: Movement Monitoring Before Fall
Project period: 2015-09-01 - 2016-01-07

Project group: 710

Participants:

Christoffer Lundager Nedergaard

Supervisor:

Sofus Birkedal Nielsen

Number of copies: 2
Pages: 50
Number of appendixes: 2
Finished 7th of January 2016

Abstract:

This projects purpose was to develop a wearable sensory device focused on body segment movement and foot support measurement, gait patients wear while in a non-objective environment for clinician to obtain "real life" data for analysis in order to improve the specific clients gait rehabilitation.
A prototype has been developed using the Arduino DUE board, MPU-6050 MEMS, a Foot Sensor sponsored by Nordic NeuroSTIM and a SD-card for data storage. The Arduino DUE form the main device of the product along with the SD-card, the sensory is plugged into the Arduino using a purposely designed extension board. The idea being that the main device is mounted in the belt, the MEMS are mounted on different body segments and connected to the main device, the foot sensor being placed in the clients shoe like an additional insole. Data from the sensor is sampled at fixed intervals and stored unmodified on the SD-card in the .csv format, before the next interval starts to ensure that no data is lost.

This study is done by:

Christoffer Lundager Nedergaard Date

Preface

This study is written by 7th semester Electronics and IT student Christoffer Lundager Nedergaard in group 710 at Aalborg University as a part of the Bachelor of Engineering education. The supervisor of the project is Sofus Birkedal Nielsen (SBN). The project level is a 7th level Bachelor project counting 25 ECTS.

Reading Instructions

The project is divided into three parts, concerning different aspects of the project.

- Introduction
- System Design
- Closing

Figures, Listings, Equations and Tables all have unique numbers. All Figures have captions placed below it.

No CD is included

This study is done by:

Christoffer Lundager Nedergaard	Date

Contents

Part I

Introduction

1 Introduction

Basic human locomotion i.e. walking or gait as it is called in scientific works, is a normal activity in the majority of the worlds population. Gait involves the whole body, though some body segments are used more than others. This is especially the case in low speed moment such as in normal walking. This is due to the fact that when a human "walk" it is actually a controlled fall, that is continuously avoided and instigated.

Though human locomotion is part of everyday life, and can seem like the simplest of tasks, everyone experiences trips or falls several times during their life. Trips and falls are mainly experienced in the early and later years of ones life, where motoric skills are being developed and motoric skills gradually deteriorate respectively.

Studies have shown that 33% of adult over 65 years old falls every year [Hausdorff et al., 2001], and that 20-30% of them suffer moderate-to-severe injuries caused by those falls. Resulting in reduced mobility and independence and increase the risk of a premature death [Sterling et al., 2001],[Alexander et al., 1992].

In many cases where people fall and are injured they are enrolled into an rehabilitation program to help and speed up the recovery process. During this process risk assessments are made to determine progress of the client. This evaluation includes the "Up and Go" and "Turn 180°" exercises with an evaluation of several parameters such as difficulty and steadiness.

However researchers and scientists would like a more objective measure of the clients gait, including variability in stride length and time resulting in a gait speed. Also information on foot support time is relevant since it has proven to be predictive of a fall [Hausdorff et al., 2001], [Maki, 1997], [Hausdorff et al., 1997].

1.1 Problem Description

Sabata Gervasio an Adjunct at Aalborg University - Health Science and Technology has put forward a project proposal concerning a system for monitoring stroke patients at home, during daily activities resulting in more objective and natural measurements/recordings.

Similar studies have been made earlier (not limited to stroke patients) [Miyazaki, 1997] using on-body sensors such as accelerometers and gyroscopes, gathering information about body positioning, acceleration and velocity [Aminian, 2006], [Tong, 1999], [Kern and Granat, 2003]. Other projects use pressure sensors mounted in the insole of the shoe relaying information about stride time and weight placement.

Stroke patients are a massive economic burden on society, each year 1.3 million Europeans suffer from strokes. The economic burden lies within direct treatment, rehabilitation and basic care for the patient. In addition comes secondary injuries due to falls. Approximately 25 % of stroke patients end rehabilitation still having some degree of gait disability. Also hemiparetic stroke patient have increased risk of falling due to impaired postural control both static and dynamic.

The number of patient with hemiparesis secondary to strokes who suffer femoral neck fracture is two - four times higher than that in the general population and most fractures are caused by falls. Therefore to reduce the economic burden on society it is important to identify and target stroke patient in risk of falling in order to develop new rehabilitation training techniques.

Companies like Xsens and Shimmer already have products on the market capable of on-body motion monitoring. Their product are however quite expensive making them less attractive when dealing with preliminary research, multiple clients and/or large scale research.

1.2 Initiating Problem

Based upon the problem description above, the following problem set arise and form a basis of which the problem analysis is done:

Development of a low cost, low complexity gait data acquisition system for use in non-laboratory environments

1.3 Product Specific Requests

In order to get a clear picture of what the project/product should contain a meeting was held with Sabata. The following is Sabata Gervasio's rough idea for the system.

Usability requirements:

- Portable device
- Minimal donning/doffing effort
- Easy to use
- Minimal obstruction of movement
- Limited sensitivity to body placement
- Long power autonomy
- No/minimal data loss
- 50-200 Hz sample rate

Device parts (usage)

Data logger	Worn in belt or in a bag around the waist of the client.
Foot Sensor (n= 2)	Used to record stance of client and step duration.
IMU* (n= 7 - 10)	Used to record body movement, using accelerometers and gyroscopes.
Connectivity	Cabled or wireless system.
Power	Power for data acquisition between 8-12 hours
Data	Capable of holding 8-12 hours of data

** IMU: Inertial Measurement Unit*

The rehabilitation gait research is still on an early stage and data analysis is done in MATLAB, using kinematic models. Sabata would like data from the sensors in their RAW format as this ensures transparency when working with them.

1.4 Product Idea

The general idea of the product is that it should be easy to use and require a minimum of technical skill to operate. The clinician mounts the device and sensory on the client for testing and set up. The client then goes home, mounts the equipment onto one self and wears it while doing their daily activities.

1.4.1 Mounting the equipment

The main device is worn in the belt of in a bag attached around the waist. Sensors are mounted different places around the clients body using a Velcro bands to hold them in-place as per instructed by the clinician. The sensors are connected to the main device via cabling that may or may-not be fixed to the body around joints to minimize activity obstruction.

1.4.2 Using the equipment

To use the equipment is it simple turned "on" pushing a button and turned "off" when monitoring is done. When the device is turned "on" a green LED i lit up. If the battery level is to low the Green LED should start flashing, indicating that charging of the device is needed. If the no sensors are connected to the main device a red LED should light up. If the data storage is full a red LED should light up. Both sensor and data storage LEDs should be marked making the error easily recognisable.

1.4.3 What should be monitored/measured

The foot sensor is used to monitor the clients center of presure (CoP), stride, stance, swing and double support durations.

The accelerometers and gyroscopes are used to determine joint angle movements, position and acceleration of body segments.

2 Problem Analysis

To understand walking or gait better this chapter contains an analysis of what a biped gait is, its different phases and how they contribute significant information about how the client is walking including how analysis of such information can help eliminating falling.

2.1 Understanding Human Gait

Gait the basis of human locomotion. In normal biped gait the legs do most of the work. With most of the work done in the three major joints namely the pelvis and lower body, being - hip, knee, ankle. When moving at faster speeds than walking the upper body segments come into play, in order to maintain balance of the whole body. In research of gait simplified models of walking are widely accepted including models like the inverted pendulum [Mummolo et al., 2013].

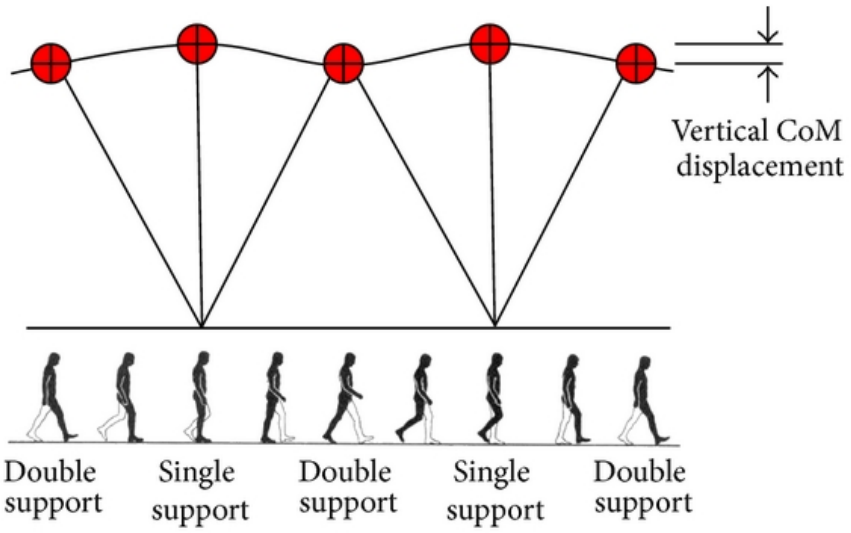

Figure 2.1. Inverted pendulum model for biped gait. Showing foot support phases, Vertical Center of Mass (CoM) displacement with the hip as CoM reference point [the scientific world journal, 2013].

Note that this model does not take individual joint movement into account. As shown in Figure 2.1 below the "inverted pendulum" both hip, knee and ankle movement is present when walking, resulting in different body segment speeds.

Generally gait is divided into two phases: Stance and Swing. These phases are determined by the foot support, the stance phase is when both feet touch the ground also know as double support (DS), the swing phase is when one foot is lifted mentioned as single support (SS). Gait at normal speeds consist of roughly 60% stance- and 40% swing phase.

Alternatively the gait cycle can be further divided into six or the newer eight sub-phases, shown in Figure 2.2, A: New gait terms, B: Classic gait terms, C: Completion of gait phase in %.

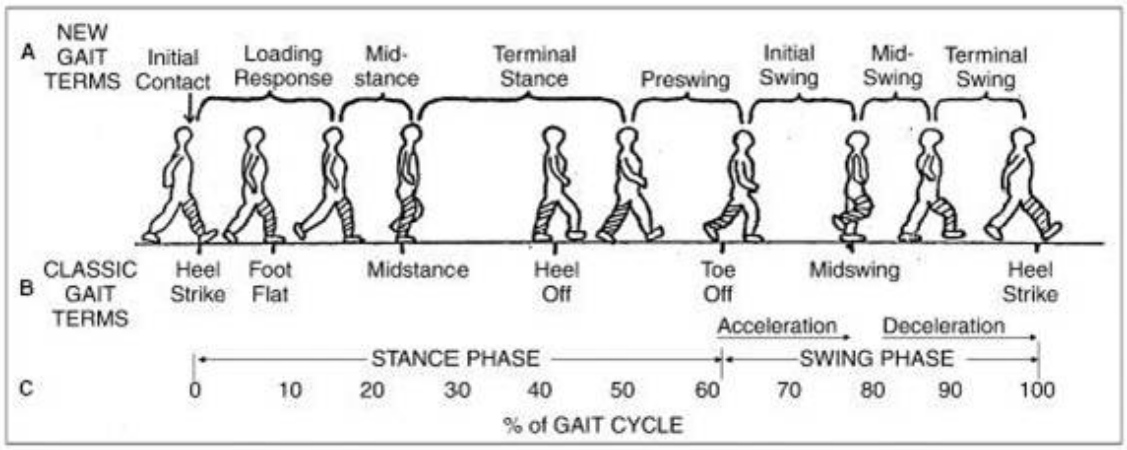

Figure 2.2. The eight gait phases shown including stance and swing indication [Physiopedia, 2015].

The different changes of joint angles are described in Figure 2.3. The fastest changes occurs during the swing phase for both the hip and knee, the ankle experiences the fastest change at the end of the stance phase.

Assuming the gait speed for a normal functioning human is 5 km/h and a stride length of approximately 75 cm, further assuming a constant velocity during stance and swing phase enables us to roughly calculate the angular velocity of the different body segments. Stride length is defined as the distance of two successive placements of the same foot, also defined as one gait cycle.

$$v = 5\left[\frac{\text{km}}{\text{h}}\right] = 1.33\left[\frac{\text{m}}{\text{s}}\right] \tag{2.1}$$

$$\frac{0.75}{1.33} = 0.563\,[\text{s}] \leftarrow \text{Time of one gait cycle} \tag{2.2}$$

Using Figure 2.3 an estimated angular velocity of the hip, knee and ankle can be calculated.

$$\text{Hip:} \quad \frac{\frac{\Delta\text{Degrees}}{\Delta\%}}{\frac{s}{\%}} = \frac{\frac{28-(-15)}{80-60}}{\frac{0.563}{100}} = \frac{2.15}{0.00563} = 381.9 \left[\frac{\text{Degrees}}{s}\right] \tag{2.3}$$

$$\text{Knee:} \quad = \frac{\frac{60-18}{90-80}}{\frac{0.563}{100}} = \frac{4.2}{0.00563} = 746 \left[\frac{\text{Degrees}}{s}\right] \tag{2.4}$$

$$\text{Ankle:} \quad = \frac{\frac{5-(-10)}{60-55}}{\frac{0.563}{100}} = \frac{3}{0.00563} = 532.9 \left[\frac{\text{Degrees}}{s}\right] \tag{2.5}$$

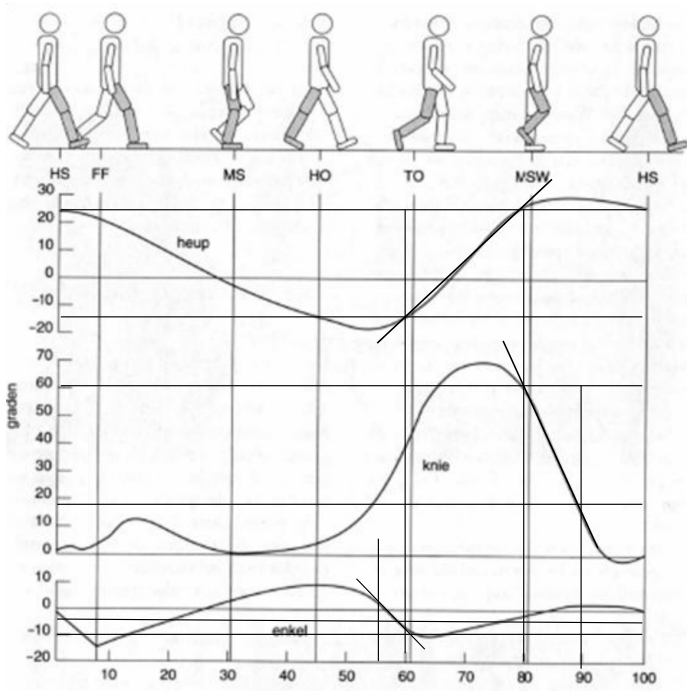

Figure 2.3. Angular joint movement during the different gait phases. HS: Heel strike, FF: Flat foot, MS: Midstance, HO: Heel off, TO: Toe off, MSW: Midswing. NOTE: X-axis [%], Y-axis [grader] [De Haagse Hogeschool, 2015]

2.2 Existing Solutions

There are several existing solutions made to monitor body movement. To mention a few Xsens and Shimmer are some of the companies supplying such hardware.

2.2.1 Monitoring Equipment using inertial sensors

Xsens:

Supplies two versions of sensory equipment specifically designed to monitor body movement. One of the solutions is made using straps to place wireless sensors on the body. The other is a lycra suit with built in wired sensory. The data output of the two solutions is 60 Hz and 240 Hz respectively both sampling 17 sensors. With a Battery time of 6 and 9.5 hours, and a wireless data transfer with a latency of 30 ms and 20 ms.

Shimmer

Supplies a kit consisting of multiple sensor bricks, the system is capable of livestreaming from 7 bricks to the docking station or simultaneous download of up to 60 docked bricks. Each brick is capable of monitoring one of eight things such as ECG, Heart rate and angular velocity.

2.2.2 Analysis Software using inertial sensors

Xsens

Xsens has developed their own software for 3D modelling using their Xsens inertial sensor technology. The software uses 3D orientation, acceleration and angular velocity from each of the maximum 10 sensors attached. The information is input to a segmented body model with added padding to make the model look more like a human. Tracker positioning include feet and legs and a recommended tracker on the upper body for trunk movement.

3 Requirements

Based on the Problem analysis and the meeting held with Sabata, requirements for the product are found. A description is made below explaining how the requirements are set up with the main requirements being usability, hardware and software.

ID	**Title**	**Description**	**Parent**	**Child**
Unique ID for Req.	Title of Req.	Description of specific Req.	Source / ID of origin	ID of consequence

3.1 Usability Requirements

These requirements are based on the meeting with Sabata and basic physiological knowledge.

ID	Title	Description	Parent	Child
U.1	Donning/Doffing	The device must be easy and quick to put on and take off.	section 1.3	NA
U.2	Sensor Mounting	The sensor must be mounted such that it does not inhibit the clients movement.	section 1.3	NA
U.3	Sensor Positioning	The sensors must have limited sensitivity to placement on the body.	section 1.3	NA
U.4	Power	The device must be powered by its own power supply to provide maximum autonomy.	section 1.3	NA
U.5	Data Storage	All sensor data must be stored, such that clinicians can use it for analysis.	section 1.3	NA
U.6	Data Access	The device data must be formated such that it is direct importable to analysis software.	section 1.3	S.4
U.7	Connectivity	The main device and sensors must be connected in such a way that it does not inhibit the clients movement.	section 1.3	NA

3.2 Hardware Requirements

These requirements form the basis for the hardware circuits and their function.

ID	Title	Description	Parent	Child
H.1	Sensors	The sensory shall consist of accelerometers, gyroscopes and a foot sensor.	section 1.3	NA
H.2	Data Storage	The data storage must be capable of storing minimum 8 hours of sampled data.	section 1.3	NA
H.3	Connectivity	The hardware must be capable of sampling data from at least 9 sensors (7 IMU's and 2 foot sensors).	section 1.3	NA
H.4	Data Sampling	The hardware must be capable of sampling the sensors at a rate of minimum 100 Hz.	section 1.3, 2.2	S.1
H.5	Power	The power supply must be capable of powering the device for at least 8 hours of use.	section 1.3	H.2
H.6	Stored Data	The data must be stored as unmodified values.	section 1.3	S.3

3.3 Software Requirements

ID	Title	Description	Parent	Child
S.1	Timing	The software must run on a timer controlled loop for precise sampling of at least 100 Hz.	H.4	NA
S.2	Connected sensors	The software must be capable of sensing how many sensors are connected and arrange data accordingly.	section 1.3	NA
S.3	Sampled Data	Sampled data must be stored unmodified.	section 1.3 & H.6	NA
S.4	Data Storage	Stored data must be stored in a uniform fixed format.	U.6	NA
S.5	Process Contraint	All data sampled from on loop run must be stored in the data storage before next run starts.	section 1.3	NA

Part II

System Design

4 System Design

An overview of the system design is created by describing the available materials, material interfaces, sensor modules and main device. Moving over to describe the parts more in depth, briefly the mechanical design idea including a sensor fixture idea, hardware design and software design.

4.1 Design

The system is designed as shown Fig. 4.1 using the Foot sensor and the MEMS to generate an input for the Processor to process and send to the Data Storage. The setup of signal path and parts incorporated is based on the requirements in chapter 3.

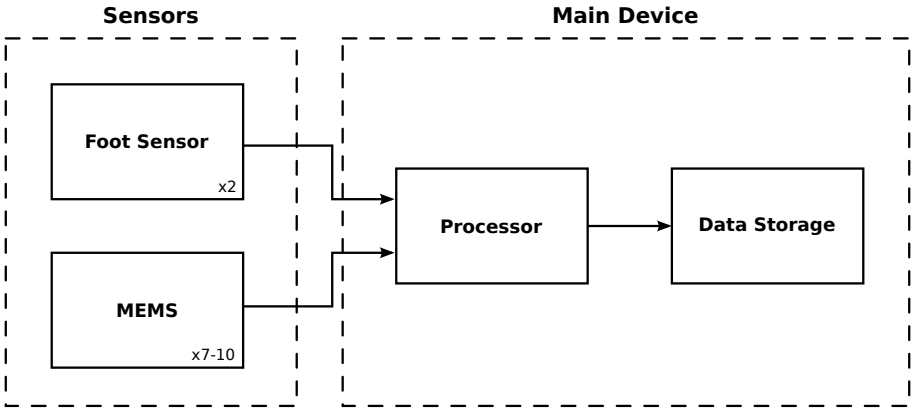

Figure 4.1. Basic system desing idea. Using two types of sensors, one data processing unit and data storage device.

*MEMS - Micro-Electro-Mechanical system

4.2 Available Prototyping Materials

To make a prototype of the system, some materials are needed. From Figure 4.1 it is seen that the system consists of two sensor types, one processor and a data storage or some sort.

Foot Sensor

The foot sensor is provided courtesy of Nordic NeuroSTIM. And is to be considered a Costumer Of The Shelf (COTS) product. The foot sensor consist of three force resistive pressure pads, one placed at the heel and the two others are placed at the lateral and medial forefoot.

The foot sensor is made such that it is inserted into the shoe on top of the insole and connected to the system via 4pin flat-pack connector. In the Sensor modules section more information is found on the foot sensor.

MEMS

Several Micro-Electro-Mechanical systems (MEMS) were available for the project. However they all used an analog output, which is good for most applications using micro processors as they often have multiple analog inputs (ADCs). In this application however the number of ADCs needed would quickly become a problem when making a simple prototype.

Therefore a MEMS using I2C for communication was ordered from a dealer. The MEMS chosen for the prototype is the MPU-6050 chip, as it includes: 3-axis- Accelerometer and -Gyroscope, each axis contained in a 16bit vector. The chip also includes its own Digital Motion Processor (DMP), which can be used to do off-processor calculations on the accelerometer and gyroscope data. As found later in the development process the DMP use and usage is rather poorly documented.

Processor

For simple prototyping two Arduino processor boards were available. The Arduino MEGA 2560 and the Arduino DUE board.

Feature	Arduino MEGA	Arduino DUE
Operating voltage	5 Volt	3.3 Volt
CPU Speed	16 MHz	84 MHz
Architecture	8 bit AVR	32 bit ARM Cortex-M3
Analog input	12	16
IO pins	54	54
Flash [KB]	256	512
SRAM [KB]	8	96
Programming	JTAG	JTAG / USB

Because of the Arduino DUEs 32 bit architecture and its 84 MHz clock speed, the Arduino DUE board was chosen for prototyping. Its 32 bit architecture is useful when considering the MEMS 16 bit data output for each of its accelerometer and gyroscope outputs.

Data Storage

For simplicity and storage size scalability a SD-card is chosen as the data storage. The SD-card interface is quite simple and uses a Serial Peripheral Interface (SPI) connection to communicate with the micro processor.

4.3 Interfaces

Using the available prototyping materials the interfaces between component "blocks" are fixed and will be described in the following sections.

MPU-6050 Board → Arduino DUE

The MPU-6050 board uses the communication standard I2C to communicate to other devices. The I2C connection uses a two wire topology with a line for a clock signal and a line for the data signal. Using only two wires limits the I2C to being a half-duplex communication line.

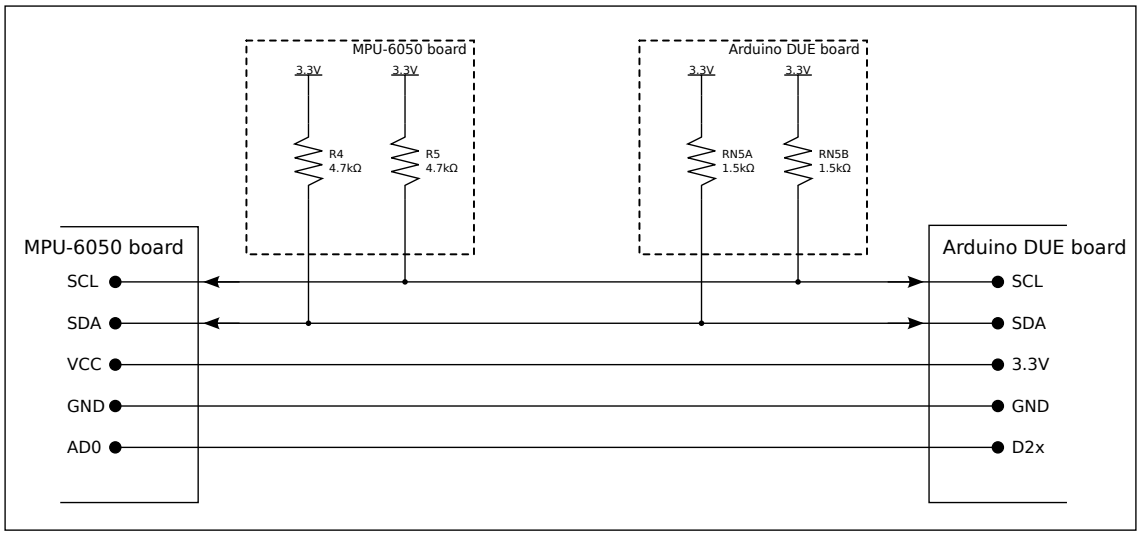

Figure 4.2. I2C interface between MPU-6050 sensor board and Arduino DUE board.

The I2C connection is an asynchronous serial communication line alike the UART. The data transfer speed of the I2C is hardware determined by the IC used. However the de facto standard for I2C communication has long been 100kbit/s i standard-mode and 400kbit/s i fast-mode, faster modes are available but will not be discussed because the Arduino DUE only supports up to fast-mode (400kbit/s).

In Figure 4.2 it is seen that both lines SCL and SDA are connected to pull-up resistors. This is necessary because the I2C clock and data output drivers are "open-drain" the result being that the I2C devices can only "sink" the lines so that there can be no bus contention e.i. when on device drives the line and another tries to pull it low possible causing short circuit and damage to devices.

The I2C protocol is based on a master to slave topology but is capable of having more than one master and multiple slaves connected to the same transmission line. As the MPU-6050 board does not have master capabilities it works as the slave and the Arduino DUE is the master.

In Figure 4.2 it is seen that both the MPU-6050 board and the Arduino DUE board have pull-up resistors mounted. This is fine as long as the I2C driver is capable of "sinking" the voltage below the logic voltage level for a "LOW". From the Texas Instruments Application Report [Arora, 2015] the minimum pull-up resistor value is calculated as:

$$\mathrm{R_P(min)} = \frac{(\mathrm{V_{CC}} - \mathrm{V_{OL}(max)}}{\mathrm{I_{OL}}} \tag{4.1}$$

In the datasheet [Inv, 2012] for the MPU-6050 chip $\mathrm{V_{OL}}$ and $\mathrm{I_{OL}}$ is found as 0.4 Volt and 3 mA respectively. The Arduino DUE [ATM, 2012] sports the same capabilities only one calculation is needed.

$$\mathrm{R_P(min)} = \frac{3.3 - 0.4}{0.003} = 966.7\,[\Omega] \tag{4.2}$$

When connecting three sensors to the BUS the pull-up impedance falls below 966.7 Ω to about 766.3 Ω. Because only a single pull-up resistor is really needed the pull-up resistors on the MPU-6050 boards could be removed or their value changed to be more fitting.

The pull-up resistor is also tied to a time constraint. The signal must have a rise time of less than 300 ns in fast-mode, this rise time is determined by the pull-up resistor and the combined BUS capacitance. From the Texas Instruments Application Report [Arora, 2015] the maximum pull-up resistance is calculated as

$$\mathrm{R_p(max)} = \frac{\mathrm{t_r}}{0.8473 \cdot \mathrm{C_b}} \tag{4.3}$$

Both the MPU-6050 and the Arduino DUE datasheet [Inv, 2012] [ATM, 2012] list $\mathrm{t_r} = 300\mathrm{ns}$ and $\mathrm{C_b} = 400\mathrm{pf}$

$$\mathrm{R_p(max)} = \frac{300E-09}{0.8473 \cdot 400E-12} = 885.16\,[\Omega] \tag{4.4}$$

Because the lowest value of the pull-up resistor is 966.7 Ω the circuit is not capable of driving a BUS capacitance of 400 pf with a maximum rise time of 300 ns. Rearranging the formula the maximum capacitance is found

$$\mathrm{C_b(max)} = \frac{\mathrm{t_r}}{0.8473 \cdot \mathrm{R_P(min)}} = \frac{300\mathrm{E}-09}{0.8473 \cdot 966.7} = 366.262\,[\mathrm{pf}] \tag{4.5}$$

Thus in order to have a proper I2C bus running at 400 kbit/s the BUS capacitance needs to be measured and pull-up resistor values are chosen based on this measurement.

Initial ideas for cables connecting the main device to the sensors was using a USB cable, being available in different lengths and having a sturdy connector it seemed like the ideal cable for the job. However it turns out that a standard 1 m USB cable has a too large cable capacitance. With the lowest capacitance value at 100 pF between D+ and D-, Shield → 5V or 0V measuring 340 pF, shield → D+ or D- measuring 180 pF and 5V or 0V → D+ or D- 140 pF, and only being capable of connecting one sensor to the main device, an USB cable is not suited for I2C interface.

Having a look at a standard flat cable it is specified to have a capacitance of only 45 $\frac{pF}{m}$ lead to neighbour lead. In addition the flat cable is capable of being a multi connector wire, thus only one cable is required per leg making it the ideal prototype cable.

Foot Sensor → Arduino DUE

The Foot sensor supplied by Nordic NeuroSTIM consist of three force resistive pressure pads. The resistors are connected to a common ground or source and three separate pins on a four pin connector.

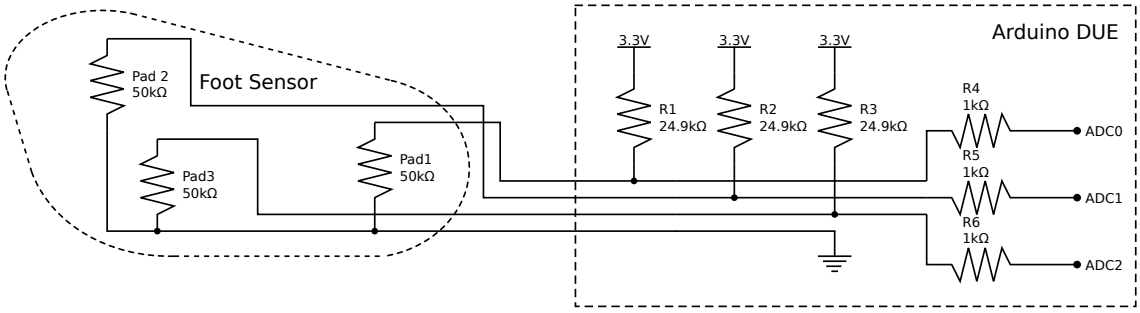

Figure 4.3. Foot sensor connected to the Arduino DUE.

In Figure 4.3 The common pin is connected to Ground (0v) while the other pins are connected to pull-up resistors and an ADC input resistor. The ADC is a cyclic pipeline 12-bit Analog-to-Digital converter and will be discussed later in section 4.5.

Arduino DUE → Data Storage

Arduino Due uses its SPI interface to communicate with the Data Storage (SD-card). A basic SPI communication is made using four wires:

- Master Out Slave In (MOSI)
- Master In Slave Out (MISO)
- Serial Clock (SPCK)
- Slave Select (nSS)

The SPI connection is a full-duplex meaning it is capable of receiving data while transmitting data. The connection is a synchronous data bus meaning it runs off a clock signal controlling the BUS speed as seen in Figure 4.4.

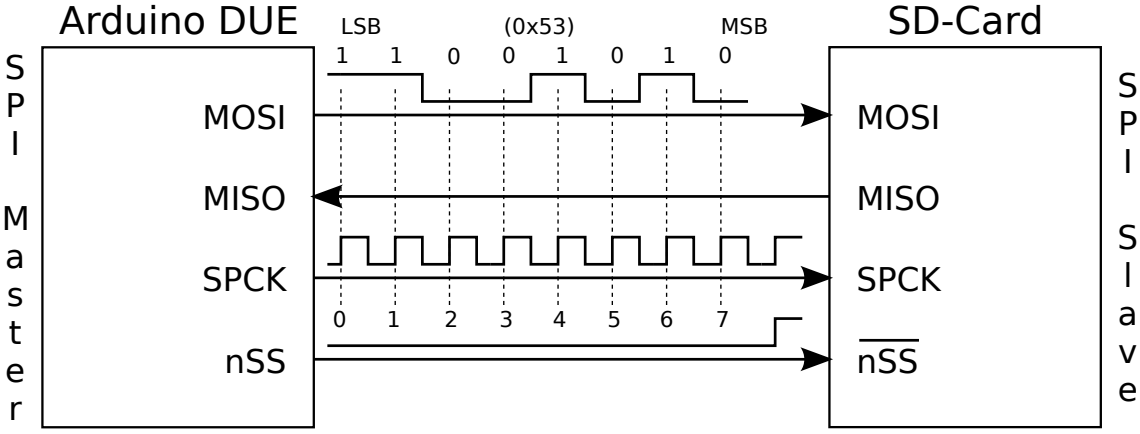

Figure 4.4. SPI interface between the Arduino DUE and SD-card.

Like the I2C the SPI BUS can have more than one slave device, unlike the I2C connection the SPI slave device need a slave select (nSS) line to active it (n denoting slave number). The clock signal is only controlled by the master, this means that when a slave device needs to send data to the master. The master has to know in advance and keep the clock signal running. This is usually no problem since the SPI mostly is connected to sensors onto which the master asks for data, thus knowing that an answer will be given.

The SPI connection supports clock speed ranging from kHz to several MHz. Communicating with a Flash memory it is typically the flash memory that limits the data rate. Using an SD-card, predefined data rates are given from the manufacturer usually around 10 Mbit/s. In some cases the SPI is capable of detecting transmission errors, and then lowering the clock frequency.

4.4 Sensor Modules

The system contains two sensor modules namely the Foot sensor and the MPU-6050 (accelerometer and gyroscope).

Foot Sensor

The Foot Sensor is provided as a COTS product by Nordic NeuroSTIM. It is a very simple product, a foot shaped sensory containing three force resistive pads. Shown in Figure 4.3 the three resistors have an resistance of 50 kΩ when no force is exerted on it. When force is applied to the pad the resistance drops to around 1 kΩ. As explained the three resistors are connected to a common pin, leaving their other ends connectible to a resistor network as shown in Figure 4.3 or similar circuits.

MPU-6050

The MPU-6050 provides inertial sensing capabilities through its gyroscopes and accelerometers. Figure 4.5 shows orientation of the different axis according to chip orientation.

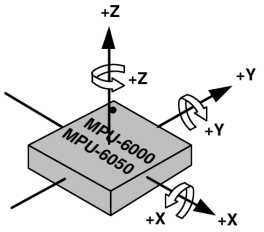

Figure 4.5. Orientation of MPU-6050 axes, showing gyroscopic and accelerometer sensitivity. Picture from MPU-6050 product specification [Inv, 2012].

It is seen how normal (readable) orientation of the chip corresponds to the "right hand rule" in a Cartesian coordinate system. (Thumb = X-axis, Index = Y-axis, Middle = Z-axis).

To better understand how the MPU-6050 functions, it's block diagram is shown in Figure 4.6. The MPU-60x0 (both 6000 and 6050) contain three separate accelerometers and gyroscopes, each with an 16-bit Analog-to-Digital converter handling the data output.

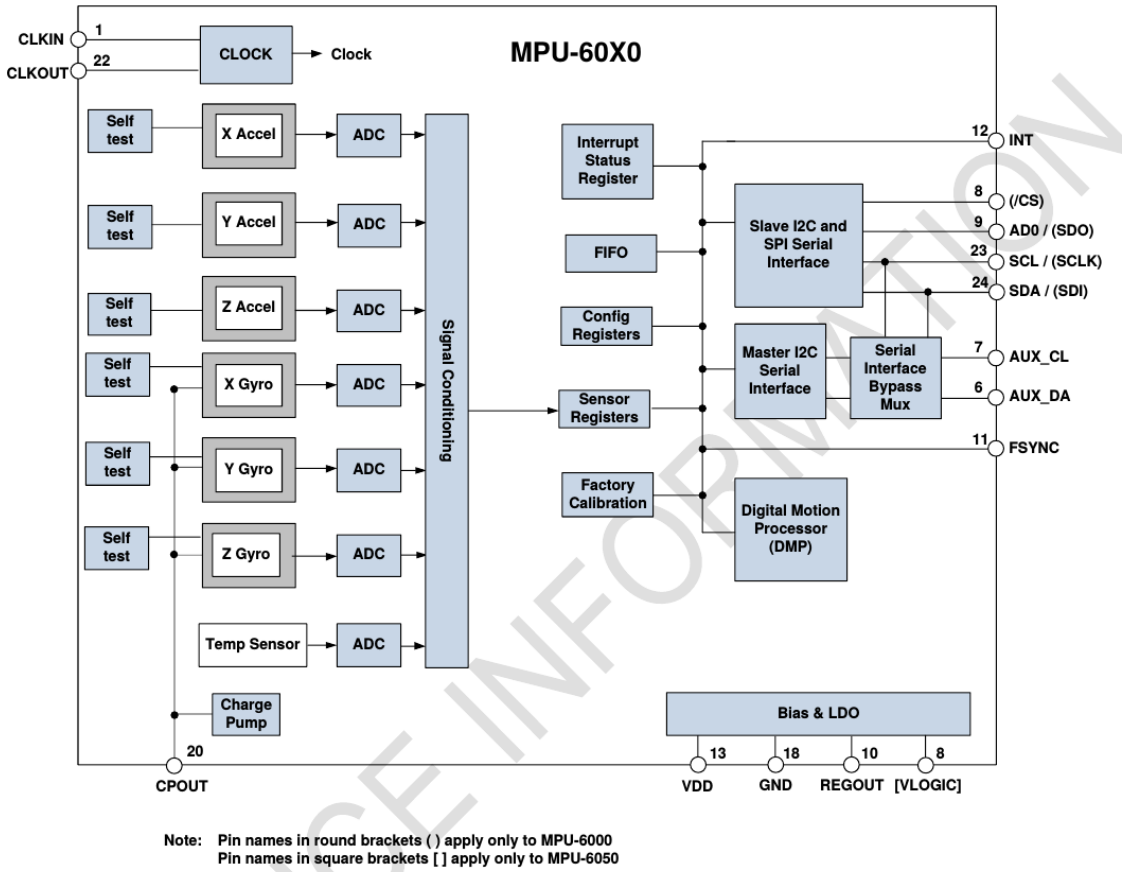

Figure 4.6. Block diagram of the MPU-6050 internals. Picture from MPU-6050 product specification [Inv, 2012].

The gyroscope features three independent vibratory MEMS rate gyroscopes [Inv, 2012]. Detecting rotation in the X-, Y-, Z-axes. When the sensor is rotated it is picked up by an capacitive pickoff. The signal is then amplified, demodulated and filtered producing a voltage representing the angular rate. The gyroscopes have an programmable scale range of ±250, ±500, ±1000 or ±2000 °/sec and a programmable sample rate from 8 kHz to 3.9 Hz.

The accelerometers has separate proof masses and is sampled by dedicated sigma-delta ADCs. Movement of the sensor displaces proof mass which is picked up by capacitive sensors, detecting the displacement differentially. As with the gyroscopes the accelerometer also has a programmable scale range of ±2g, ±4g, ±8g or ±16g. It is stated that when the accelerometer is placed on a flat surface it will measure 0g in the X and

Y axis and +1g on the Z axis [Inv, 2012]. Unlike the gyroscope the accelerometer has a fixed sample rate of 1 kHz.

The output from the gyroscope and the accelerometer are filtered by a digital lowpass filter with programmable cut-off frequency.

DLPF_CFG	Accelerometer (F_s = 1kHz)		Gyroscope		
	Bandwidth (Hz)	Delay (ms)	Bandwidth (Hz)	Delay (ms)	Fs (kHz)
0	260	0	256	0.98	8
1	184	2.0	188	1.9	1
2	94	3.0	98	2.8	1
3	44	4.9	42	4.8	1
4	21	8.5	20	8.3	1
5	10	13.8	10	13.4	1
6	5	19.0	5	18.6	1
7	RESERVED		RESERVED		8

Figure 4.7. Digital lowpass filter table from MPU-6050 product sheet [Inv, 2012].

All six sensors 16-bit values are stored in the Sensor Register accessible for the I2C interface and Digital Motion Processor (DMP). The Sensor Register stores the newest measurements, X-, Y- and Z-axis in 8-bit registers, most significant Byte first for each axis.

In the Config Registers, clock source, device reset, accelerometer and gyroscope standby are all set. The clock source can either be an internal 8 MHz oscillator, a Phase Lock Loop controlled clock based on either gyroscope axis or an external square wave clock source of 32.768 kHz or 19.2 MHz. In the product specification it is noted that when running the gyroscopes, selecting the gyros as a clock source provided a more accurate clock [Inv, 2012].

The Slave I2C and SPI Serial Interface handles the I2C communication lines (NOTE: SPI interface is only available in the MPU-6000.). The I2C device address is 0x68 or 0x69 depending in logic level of the AD0 pin.

4.5 Main Device

The main device is based on the Arduino DUE which uses Atmels 32-bit SAM3X8E CORTEX-M3 ARM processor.

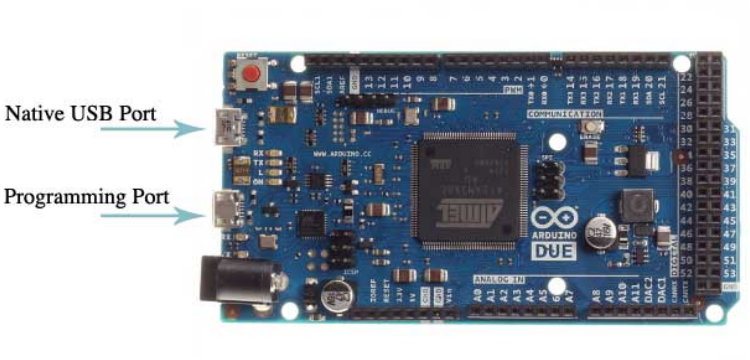

Figure 4.8. Product picture of the Arduino DUE board from the Arduino website showing progamming and native USB port.

The Arduino DUE board is easily connectible with peripherals using the pin headers placed along the boards edge, making it very flexible during prototyping. As the SAM3X8 processor support more features than necessary only the features used in the prototype like interfaces to sensors and data storage are discussed.

Top half of the internal block design of the SAM3X8 CPU for full block diagram see Appendix B.1

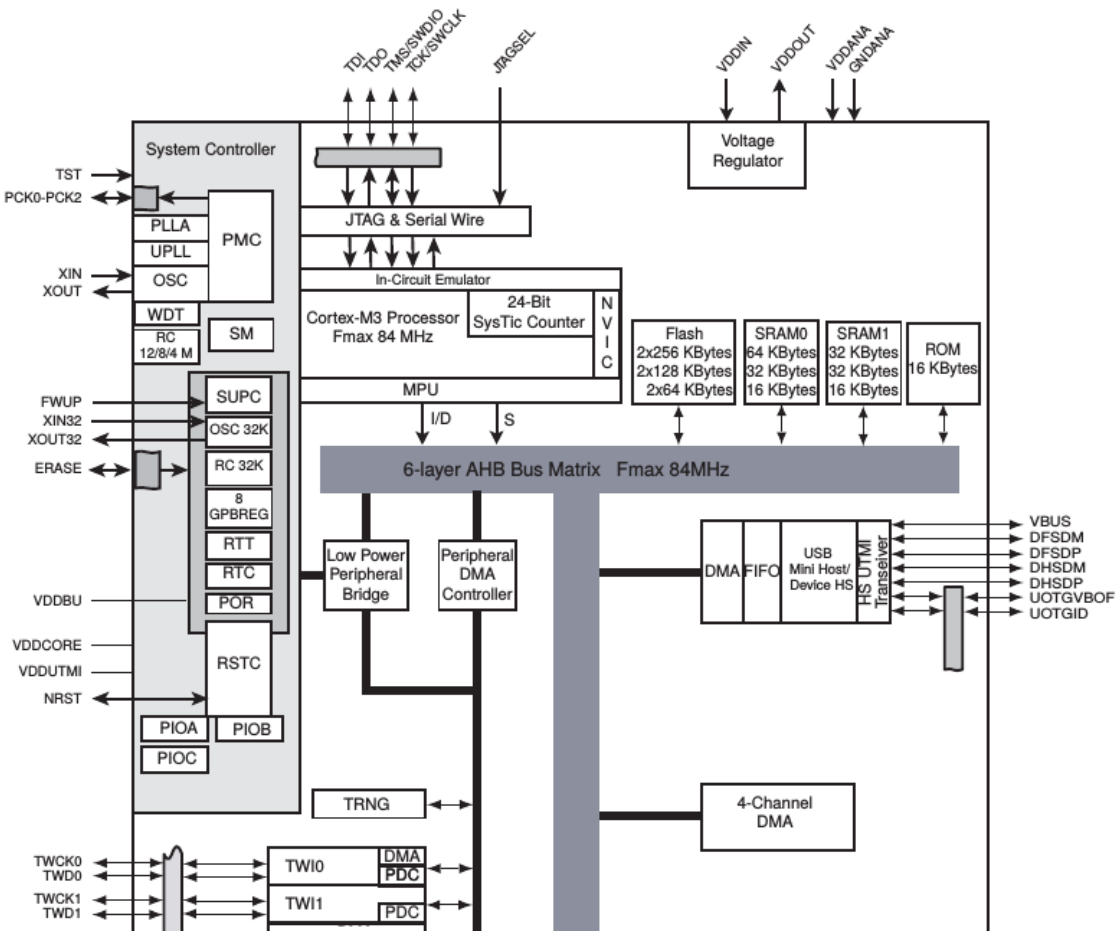

Figure 4.9. Top half of the SAM3X8 block diagram, showing the ARM architecture [ATM, 2012].

This ARM processor uses the Harvard architecture, unlike AVR processors which uses the Harvard architecture exclusively some ARM processors uses the Von Neumann architecture. The Cortex-M3 processor uses two seperate data bridges, namely the Advanced Peripheral Bridge (APB) and the Advanced High-performance Bridge (AHB). The APB is a low speed bridge connecting the UART, ADC, TWI, PWM, DAC and CAN peripherals, while the AHB is a high speed brigde connected to the SPI, SSC, HSMCI peripherals (Abrevations can be found in the SAM3X8 datasheet [**?**]).

I2C Arduino DUE interfaec

Using the I2C interface and ADCs the APB is used. Looking first at the I2C interface the Power Management Controller (PMC) must be handled to turn on the clock signal for the TWI peripheral.

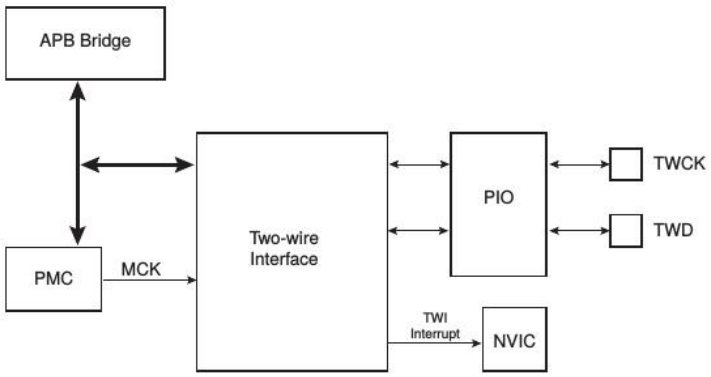

Figure 4.10. Block diagram of the TWI interface [ATM, 2012].

As seen in Figure 4.10 the APB sends data to the TWI which in-turn handles the proper output pins through the Parallel Input/Output Controller (PIO). Interrupts are send directly to the Nested Vector Interrupt Controller which is directly connected to the processor seen in Figure 4.9.

TWCK and TWD are for an I2C interface renamed SCL and SDA respectively.

Analog-to-Digital interface

Taking a look at the SAM3X8E Analog-to-Digital converter it has a 12-bit resolution and a 1 MHz max conversion rate e.i. Sample frequency. Using an integrated multiplexer it has 16 seperate analog inputs controlled by the PIO.

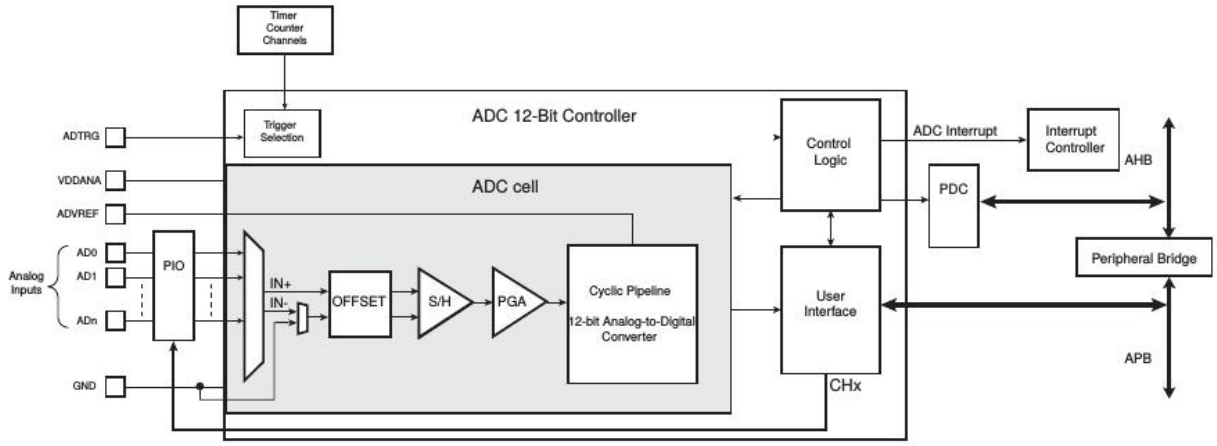

Figure 4.11. Block diagram of the SAM3X8E Analog-To-Digital converter [ATM, 2012].

The ADC cell receives its input directly from the PIO to the multiplexer. The multiplexer is programmable such that the output can be single ended or fully differential. The input can be offset if single ended input i selected otherwise, offset is bypassed, The Sample/Hold (S/H) operator ensures the signal is stable for conversion. The PGA block is a programmable gain amplifier and can be set to 1/2, 1, 2 and 4 times. The input signal is sampled by a Cyclic Pipeline 12-bit ADC like the one shown in Figure 4.12

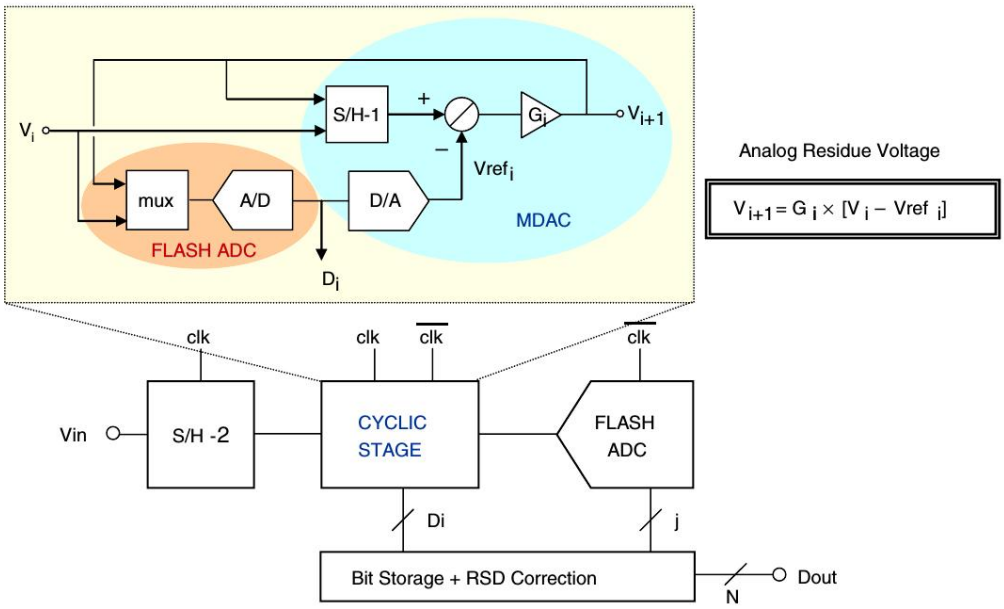

Figure 4.12. Function diagram of a cyclic pipeline ADC [ATM, 2011].

Being a Cyclic ADC it runs off the clock in a cycle like maner, the Cyclic Stage (block) in Figure 4.12 is used multiple times during a conversion. From the SAM3S4 ADC Application Note (also applicable for SAM3X8) [ATM, 2011] it is found that a full 12-bit conversion takes 20 ADC clock cycles thus the ADC clock needs to be 20 MHz to result in a 1 MHz sample rate.

The ADC clock is set by using a prescale value of 0-255, $ADC_{clock} = \frac{MCK}{PRESCAL}$ MCK

being the master clock and PRESCAL the prescale value set in the ADC Mode Register.

SPI Arduino DUE interface

The SPI is as before mentioned a synchronous serial data interface for external peripherals connected to the processor. In Figure 4.13 it is shown how the SAM3X8E SPI interface is connected to the inter-peripherals of the SAM3X8E

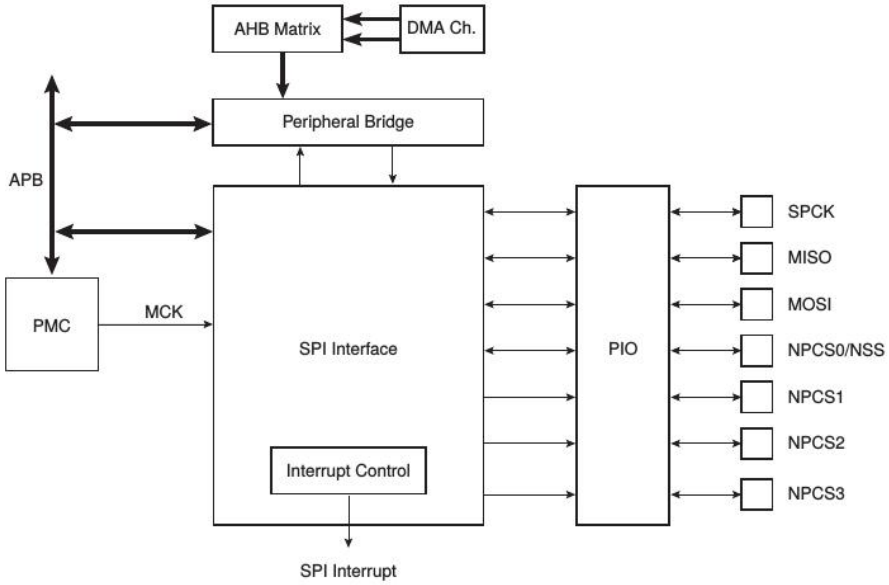

Figure 4.13. Block diagram of the SAM3X8E SPI interface [ATM, 2012].

Like the I2C interface the SPI interface receives its clock signal through the PMC. Unlike the ADC and I2C this interface is connected directly to the Advanced High-performance Bridge which runs at a higher speed. This is done to ensure the high data rate needed to use the full potential of the SPI port. Interrupt control is done sending signals to the NVIC in Figure 4.9

Timer Counter Arduino DUE interface

The Timer Counter TC does not affect the other interfaces as such, however to easily create a fixed and controllable sample rate it is used for interrupt control. The TC split into three 32-bit channels which can be programmed independently, making them capable of doing frequency measuring, event counting, pulse generation, delay timing and others.

In Figure 4.14 it is seen that five TIMER_CLOCKn signals connected to the TC block, these signals are internal clock signals routed through the PMC to the TC (i.e. enabled through the PMC register). Also a set of three external clock sources (TCLKn) can be chosen from the PIO which also controls the input/output TIOAn and TIOBn. Each channel has its own interrupt line to the NVIC.

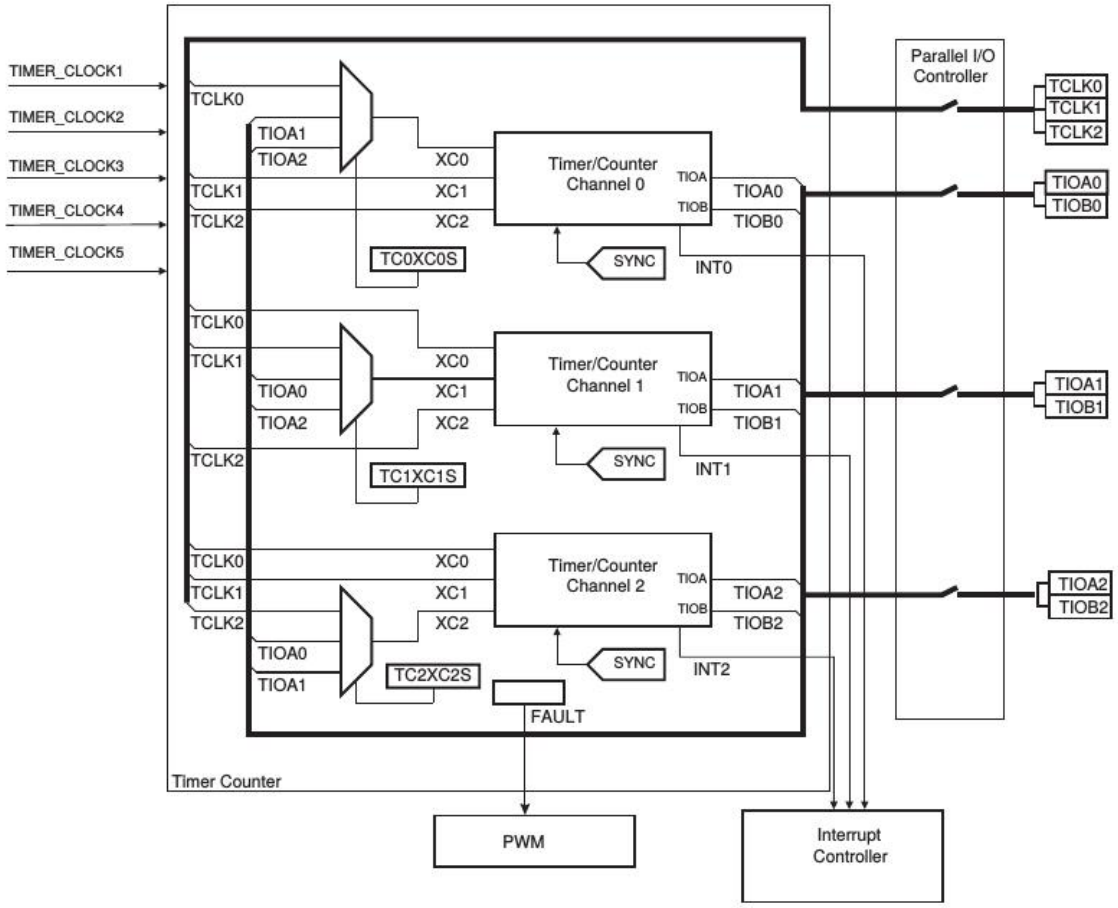

Figure 4.14. Block diagram of the SAM3X8E Timer Conuter [ATM, 2012].

The FAULT output is connected to the PWM controller and can be used to determine if one counter running faster than another, e.g. if two motor speeds are measured and one is running faster than the other the FAULT signal is triggered and the PWM controller make a correction.

4.6 Mechanical Design

The mechanical design of the product is to be done in a way such that it is as easy to use as possible. Living up to the usability requirements set in section 3.1. One example of a mechanical design is given for the MEMS Fixture in appendix A.1. This design incorporates a non wireless solution, where the MEMS is held in place using a velcro strap, fed through the small “hinges” on either side of the small box.

The signal cable is then pluged into the box from the top, so that is does not bulge as a bend would do if it had to be plugged in from the bottom. With this fixture it is possible to mount it “upside down”. The fixture also provides a solution which is not body part specific as the velcro strap holding it in place should be adjustable, so that it fits most body segments.

4.7 Hardware Design

For the prototype the Arduino DUE is connected to a Vero-board onto which connectors to sensors and the SD-card extender board is soldered. It turns out that the connectors on the Arduino DUE board are aligned such that it fits the vero board spacing, only one exception is found and that is the connector row closest to the "Native USB Port". Not using this connector it mattered little during the hardware design phase.

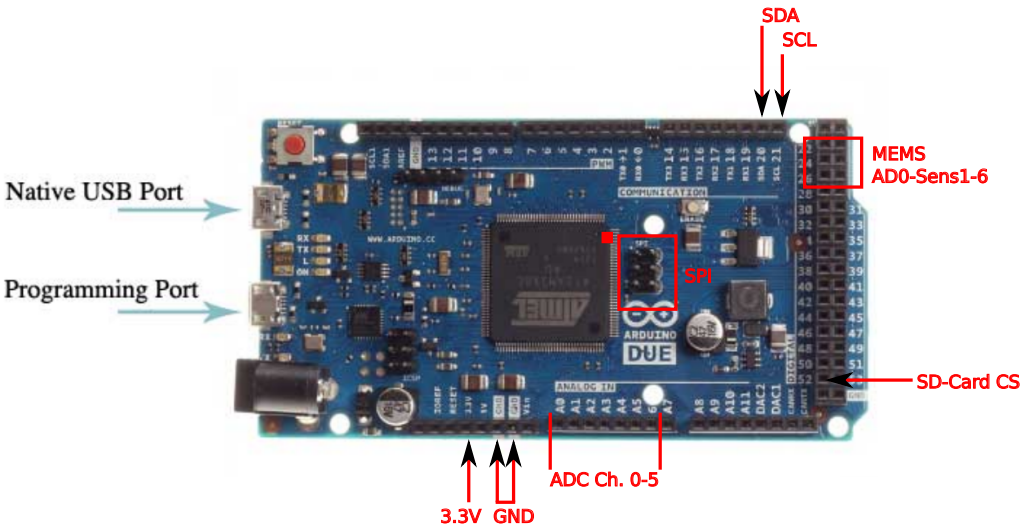

Figure 4.15. Connections used on the Arduino DUE.

Having chosen to use the MPU-6050 board no extra circuit design is needed to interface with the Arduino DUE. The SD-card adapter is also ready for use out of the box. The Foot Sensor from Nordic NeuroSTIM need some additional circuit design i.e. as seen in Figure 4.3. The resistors are mounted on the vero board and connected as shown in Figure 4.3, additional resistors are mounted for the second foot sensor, and are connected to pin ADC3-5.

Connection from the Extender board is done using a ten lead flat cable as it the required size for connection of three sensors and a foot sensor, thus the client only need to wear a single cable down each leg.

The extender board is wired using wire wrap, connection diagram is shown in Figure 4.16.

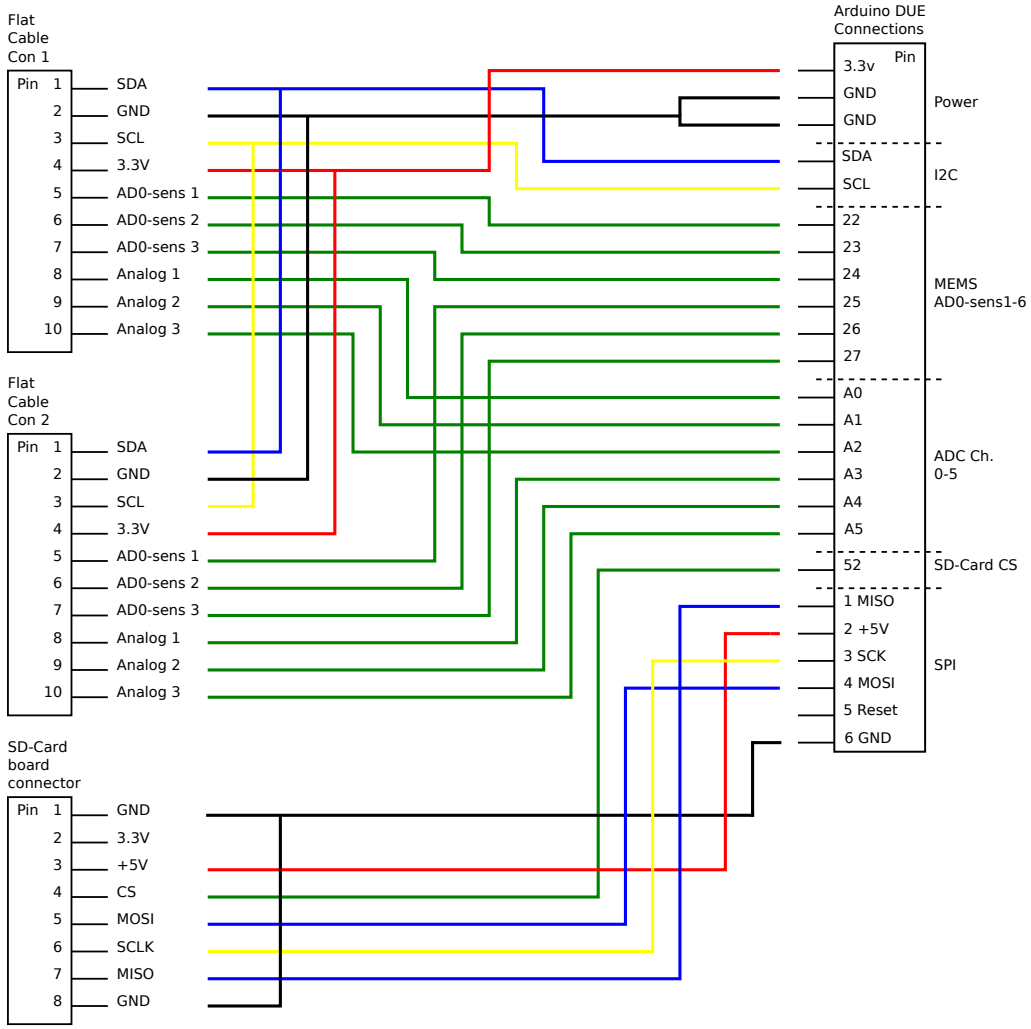

Figure 4.16. Extender board connections between the Arduinu DUE and the peripheral connectors. Wire colors resemble colors used on actual hardware (NOTE: Black GND = White wire, ADC resistors not shown see Figure 4.3).

Connections to the flat cable are made intentionally this way, with the SDA line only having one neighbour minimizing cable capacitance for this output. A GND wire is placed between the SDA and SCL minimizing cross-talk between signal, but increases the capacitance "seen" by the SCL driver due to the two AC ground plane created by GND and 3.3V.

AD0-sens wires are used to select the MPU-6050 chip. The I2C device address of the MPU-6050 is either 0x68 or 0x69 depending on the logic value of its AD0 pin. Connecting AD0 to AD0-sens the device address can be set allowing more than two sensor using the same I2C connection. All six sensors share the same I2C connection, this is possible by only setting the AD0 pin logic LOW on the desired chip and all other AD0's logic HIGH. Thus the desired chip hold the device address 0x68 and all other 0x69, this does restrict software to never call a device on address 0x69, as multiple devices would in theory answer. Reading all MPU-6050 devices requires a software controlled multiplexer of sorts.

The MPU-6050 board is not directly connectible a flat cable so an extension board is made for each MEMS.

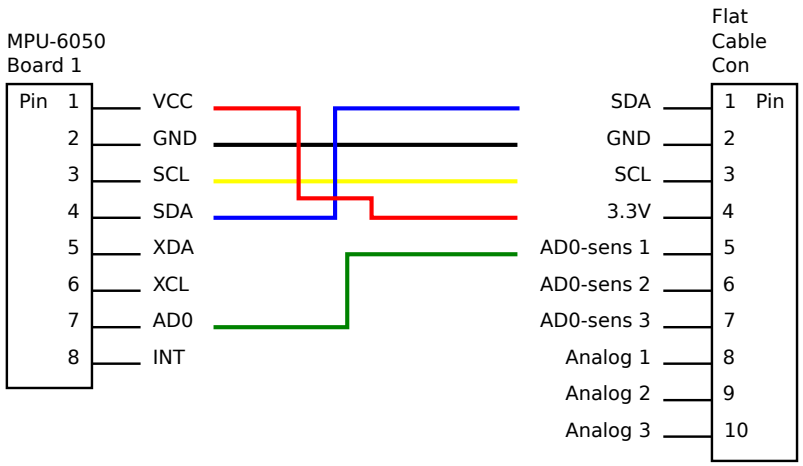

Figure 4.17. Extender board connections between the flat cable and the MPU-6050 board. Wire colors resemble colors used on actual hardware (NOTE: Black GND = White wire).

Note: Figure 4.17 showing MPU-6050 Board 1, AD0-sens 1-3 connection is shifted respectively to the MPU-6050 Board number. Placement of the sensors on the flat cable is interchangeable, but interchanging sensors from one leg to the another should be done with caution making sure that no two identical MPU-6050 boards are connected to the same flat cable.

The Foot Sensor is connected at the end of the flat cable completing the cable.

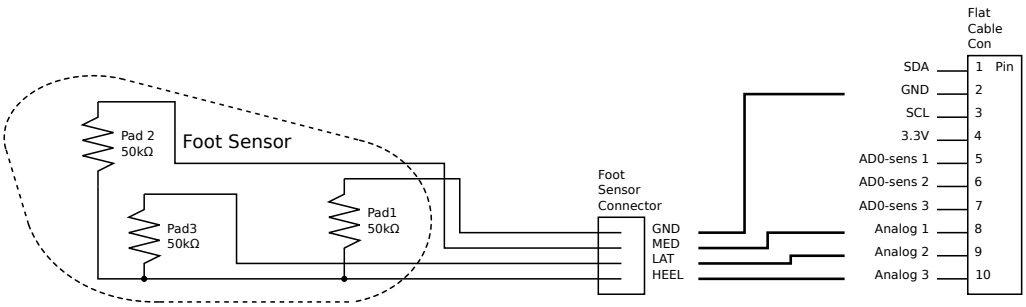

Figure 4.18. Flat cable connection to the Foot Sensor.

4.8 Software Design

The software design in the prototype is done using the Arduino IDE and Arduino C language, this is only done to make a quick test bench. The real product software should be written in proper C langauge following a software standard such as the EN 62304 standard for medical devices.

Software Flow

The program flow and its function is determined by partly the system use, system idea and partly by the software requirements. The main function of the software is to requesting sensor data and storing the answer in the data storage. Considering the software constraints S.1, S.5 set in the software requirements section the following solution of sequential execution structure was chosen. Added to this decision comes requirement S.3, hence the software only need to do fetch and store operations in a fixed time.

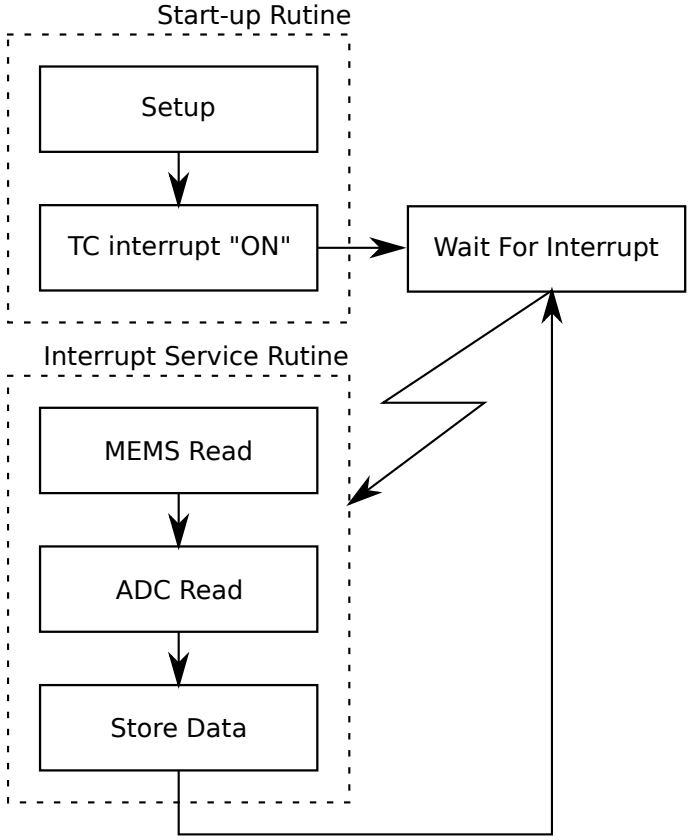

Figure 4.19. Basic software flow diagram showing "Start-up"- and "Interrupt Service" Rutine.

The Start-up Rutine is only run once after powering up the processor, once the Start-up Rutine is done the Timer Counter Interrupt is enabled and control the software execution. The only processes run on the processor is run within the Interrupt Service Rutine (ISR), and when done returns to the "Wait For Interrupt" state.

Start-up Rutine

After powering up the processor, the "Start-up Rutine" is ran. This Rutine contains setup of:

- Digital output pins in the PMC
- Starting the I2C interface

- Setup of connected MPU-6050 sensors
- starting the SPI interface
- Setting the SPI to full speed
- Finding an available file name
- Opening a file on the SD-card with the available file name
- Write Data Header
- Enable Timer Counter clock in PMC
- Configure Timer Counter
- Enable Timer Counter Interrupt

Digital output setup

The digital outputs are used to control the AD0-Sens pins and the SD-Card Chip Select (CS). The pins are set as outputs using the following code

```
// digital pin setup
  pinMode(mems1_pin, OUTPUT);
  pinMode(mems2_pin, OUTPUT);
  pinMode(mems3_pin, OUTPUT);
  pinMode(mems4_pin, OUTPUT);
  pinMode(mems5_pin, OUTPUT);
  pinMode(mems6_pin, OUTPUT);
// SD-Card chipselect pin setup
  pinMode(chipSelect, OUTPUT);
```

Where mems1_pin is assigned the pin number to be set as output on the Arduino DUE board.

MPU-6050 setup

Setting up the MPU-6050 sensors are done one at a time. Using a "for" loop running for the amount of sensors chosen by the "n" variable. An "if" statement is used to select the individual sensors based on the "for" loops cycle. Writing to the "REG_PIOA_SODR" or "REG_PIOB_CODR" is a direct way of setting a pin HIGH or LOW respectively, doing it this way only takes 100 ns compared to 10 μs using the Arduino C method.

```
// I2C sensor setup -- initialisation
    for(uint8_t i=0; i < n; i++)
  {
    if(i==0){
      // only code the the two connectable sensors e.i. on port 1
      REG_PIOA_SODR |= 0x00000001 << 14;
      // sets pin PA14 high - arduino pin 23
      REG_PIOB_CODR |= 0x00000001 << 26;
      // clears pin PB26 - arduino pin 22
    }
    else if (i==1){
      REG_PIOA_CODR |= 0x00000001 << 14;
      // clears pin PA14 high - arduino pin 23
      REG_PIOB_SODR |= 0x00000001 << 26;
```

```
      // sets pin PB26 - arduino pin 22
    }
     Wire.begin();
     Wire.beginTransmission(memsaddr);
     Wire.write(0x6B);
     // Power management of register 1
     Wire.write(0);
     // Writes into Power register 1 to clear sleep
     // - wakes up MPU-6050
     Wire.endTransmission(true);
     // Terminates setup of MPU-6050 chip
  }
```

The "Wire.begin();" creates the I2C start condition on the SDA line (namely SDA going LOW while SCL is HIGH). "Wire.beginTransmission(memsaddr);" outputs the I2C device address (0x68) on the I2C bus calling a device on that address. Next up is moving the I2C devices internal address pointer to the Power management register (0x6B) and writing 0x0, clearing the MPU-6050 devices sleep mode and turning it on. The "Wire.endTransmission(true);" ends the transmission setting the stop condition (SDA going HIGH while SCL is HIGH).

SPI setup

The SPI setup is done as a backgound feature using the "sd.begin();" function, it inputs the pin number used for CS (52), and the desired SPI speed. Using SPI_FULL_SPEED set the bus to 42 MHz.

```
// SD-card setup -- initialisation
  if(!sd.begin(chipSelect, SPI_FULL_SPEED))
  {
    sd.initErrorHalt();
  }
```

If the "if" statement is true an error message is written to the SD-card.

SD-Card setup

To ensure that no data is overwritten if the device if restarted during a session. In order to do this a check is made to ensure that the file name is not allready existing on the SD-Card.

```
// Find an unused file name.
  while (sd.exists(fileName)) {
    if (fileName[BASE_NAME_SIZE + 1] != '9')
    {
      fileName[BASE_NAME_SIZE + 1]++;
    }
    else if (fileName[BASE_NAME_SIZE] != '9')
    {
```

```
      fileName[BASE_NAME_SIZE + 1] = '0';
      fileName[BASE_NAME_SIZE]++;
    }
    else
    {
      error("Can't create file name");
    }
  }
```

Using the "while" statement ensures that every possible name is tried cycling through and increasing the file name value until it is 9 characters long. If not file name is found an error message is written to the SD-card.

```
// Open SD-card file - write error file.open if fails
  if (!file.open(fileName, O_CREAT | O_WRITE | O_EXCL)) {
    error("file.open");
  }
```

When a name is found the file is created on the SD-card using the "file.open();" expression. If the file fails to be created and opened an error message is written on the SD-card.

Preparing the file for data a dataheader is written to the file using the function "dataheader(void);".

```
void dataheader(void) {
// milis,AdX1,AdY1,AdZ1,GyX1,GyY1,GyZ1,AdX2,...
// ,heel1, Med1, Lat1, Heel2,
  file.print(F("milis"));
  for(uint8_t y=0; y < n; y++)
  {
    file.print(F(",AcX"));
    file.print(y,DEC);
    file.print(F(",AcY"));
    file.print(y,DEC);
    file.print(F(",AcZ"));
    file.print(y,DEC);
    file.print(F(",GyX"));
    file.print(y,DEC);
    file.print(F(",GyY"));
    file.print(y,DEC);
    file.print(F(",GyZ"));
    file.print(y,DEC);
  }
  for(uint8_t e=0; e < n; e++)
  {
    file.print(F(",MED"));
    file.print(e,DEC);
```

```
    file.print(F(",LAT"));
    file.print(e,DEC);
    file.print(F(",HEEL"));
    file.print(e,DEC);
  }
  file.println();
}
```

This function writes a .csv one line header, starting with milis, then the MPU-6050 sensor data and then the two foot sensor ADC values. The last write done creates a line change. Every value is seperated by a comma thus the .csv file name extension.

Timer Counter setup

The Timer Counter setup starts with enabling the clock signal to the Timer Counter desired. Then the Timer Counter is configured by setting which Timer Counter to configure (TC), its channel (1), enabling Waveform operation mode, selecting register compare mode and the timer clock input signal.

```
pmc_enable_periph_clk(ID_TC7);    // enable peripheral clock TC7

// Timer setup function

TC_Configure(TC2, 1, TC_CMR_WAVE | TC_CMR_WAVSEL_UP_RC
| TC_CMR_TCCLKS_TIMER_CLOCK3);
//  timer counter, channel, waveform mode,
//  counter run up then reset, divider choosen timer clock3 (32)
TC_SetRC(TC2, 1, 26250);
// timer counter 2 channel 1 counts to 26250, resets and repeats
TC_Start(TC2, 1);
// MCK / 32 = timer_clock3 -> 84 MHz / 32 = 2.625E6 -->
// for 100 Hz interupt 2.625E6 / 100 = 26250 reset value
// TC_Start(TC2, 1);

// enable timer interrupts on the timer
TC2->TC_CHANNEL[1].TC_IER=TC_IER_CPCS;
// IER = interrupt enable register
TC2->TC_CHANNEL[1].TC_IDR=~TC_IER_CPCS;
// IDR = interrupt disable register

// enabling nested vector interupt controller
NVIC_EnableIRQ(TC7_IRQn);
```

Having configured the Timer Counter its register for register comparison is set to 26250, having chosen the TIMER_CLOCK3 running at 2.625 MHz. Thus for a 100 Hz interrupt the register value needs to be $\frac{2{,}625{,}000}{100} = 26250$ When the compare value is set the timer is started, however without enabling the interrupt signals no interrupt will occur. Setting

the Timer Counter Interrupt Enable Register and Clearing the Timer Counter Interrupt Disable Register enables interrupts. In addition to this the NVIC also need to be enabled, and specifying the ISR to run when an interrupt occurs.

Interrupt Service Rutine

When an interrupt occurs the processor leaves its "Wait For Interrupt" state and enters the "Interrupt Service Rutine". This rutine contains the following process:

- Clear interrupt flag restarting the timer
- Read all MPU-6050 sensor data
- Read all ADC channels
- Write Data to the SD-card
- Check that data is written to the SD-card

Timer reset
When the ISR is entered the Timer Counter status is read in order to re-enable the interrupt function.

```
// Get the status to clear it
// and allow the interrupt to fire again
TC_GetStatus(TC2, 1);
```

Read MPU-6050 data
Receiving data from the MPU-6050 MEMS is done much like during the setup, each sensor is called individually and its register cursor is moved to the desired register. When the cursor is moved a total of 14 bytes is requested also called a burst read. This is done by calling the device with a read bit instead of a write bit in the address followed by a master "ACK", the slave device then send the byte pointed to by the register cursor. Upon byte receive the master does another "ACK" and next byte is sent. When the last byte is received the master does a "NACK" and stop condition.

```
  for(uint8_t i=0; i < n; i++)
  {
  if(i==0){
     REG_PIOA_SODR |= 0x00000001 << 14;
     // sets pin PA14 high - arduino pin 23
     REG_PIOB_CODR |= 0x00000001 << 26;
     // clears pin PB26 - arduino pin 22
  }
  else if(i==1)
  {
    REG_PIOB_SODR |= 0x00000001 << 26;
    // sets pin PB26 - arduino pin 22
    REG_PIOA_CODR |= 0x00000001 << 14;
    // clears pin PA14 high - arduino pin 23
```

```
  }
  Wire.beginTransmission(memsaddr);
  Wire.write(0x3B);
  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(memsaddr,14,true);
  // request a total of 14 registers

  AcX[i]=Wire.read()<<8|Wire.read();
  // 0x3B (ACCEL_YOUT_H) & 0x3C (ACCEL_YOUT_L)
  AcY[i]=Wire.read()<<8|Wire.read();
  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ[i]=Wire.read()<<8|Wire.read();
  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp[i]=Wire.read()<<8|Wire.read();
  // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX[i]=Wire.read()<<8|Wire.read();
  // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY[i]=Wire.read()<<8|Wire.read();
  // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ[i]=Wire.read()<<8|Wire.read();
  // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
  }
```

Read ADC Channels

To read the ADCs channels the Arduino C function "analogRead();" is used, using the "u" variable as input to determine channel and to define where the functions output is stored.

```
// ADC read os foot sensors
  for(uint8_t u = 0; u < 6; u++)
  {
    data[u] = analogRead(u);
  }
```

Write data to SD-card

Before writing the data acquired the time stamp "Time" is incremented by ten, representing that 10 ms has passed since last sample. The time stamp is written to the SD-card along with all other data using a "for" loop to cycle through the values. The write cycle is finished by writing a line change. To ensure data is written to the SD-card a sync and write error check is done - if either has occurred an error message is written to the SD-card.

```
 Time += 1UL*10;
// Write all the Data to the SD-card
 file.print(Time);
  for(uint8_t t=0; t < n; t++)
  {
    file.write(',');
```

```
    file.print(AcX[t]);
    file.write(',');
    file.print(AcY[t]);
    file.write(',');
    file.print(AcZ[t]);
    file.write(',');
    file.print(GyX[t]);
    file.write(',');
    file.print(GyY[t]);
    file.write(',');
    file.print(GyZ[t]);
  }
  for(uint8_t r=0; r < 6; r++)
  {
    file.write(',');
    file.print(data[r]);
  }
  file.println();

  if (!file.sync() || file.getWriteError()) {
    error("write error");
  }
```

Part III

Closing

5 Closing

5.1 Acceptance Test

In the previous chapters the design of a gait data logger has been developed, based on requests and a simple body movement analysis. Usability, Hardware and Software requirement have been set in chapter 3 these requirement will be discussed and tested to see if they are met.

Usability Requirements

ID	Title	Description	Parent	Child
U.1	Donning/Doffing	The device must be easy and quick to put on and take off.	section 1.3	NA

Test Method:

In order to test this requirement a role play is setup using volunteers i.e. people who have not been involved in the development of the product possible creating a bias. The role play should be video recorded and review by the volunteers in order for them to "OK" the video material and in order for them to relive the experience, after the video review. A dialog should be initiated by a clinician having some knowledge of the product.

Result:

Testing was not done due to product only being on early prototype stage.

U.2	Sensor Mounting	The sensor must be mounted such that it does not inhibit the clients movement.	section 1.3	NA

Test Method:

See Test Method of U.1

Result:

See test Result of U.1

U.3	Sensor Positioning	The sensors must have limited sensitivity to placement on the body.	section 1.3	NA

Test Method:

Sensors are placed on a person and the device is turned on, the person then walks in a specific pattern. When the person has walked the pattern the device is turned off and the sensors are moved/adjusted. The device is turned on again and the walk pattern is repeated. After completing a series of sensor placements the data gathered is compared either directly or using a model.

Result:

Testing was not done due to product only being on early prototype stage.

U.4	Power	The device must be powered by its own power supply to provide maximum autonomy.	section 1.3	NA

Test Method:

As the device is intended to ship with its own power bank or hold an internal battery - no test is carried out

Result:

The device is intended to come with its own power bank

U.5	Data Storage	All sensor data must be stored, such that clinicians can use it for analysis.	section 1.3	NA

Test Method:

Initial gait analysis is done using MATLAB and data is stored using the .csv extension. Running the MATLAB command CSVREAD('filename',1,0) to see if it can read the file.

Result:

Using the CSVREAD('filename',1,0) result in Gait00.csv being read into the workspace of MATLAB - Thus data is stored in a readable format that can be used for analysis.

U.6	Data Access	The device data must be formated such that it is direct importable to analysis software.	section 1.3	S.4

Test Method:

See test method of U.5

Result:

See test Result of U.5

U.7	Connectivity	The main device and sensors must be connected in such a way that it does not inhibit the clients movement.	section 1.3	NA

Test Method:

See test method of U.1

Result:

See test Result of U.1

Hardware Reuirements

ID	Title	Description	Parent	Child
H.1	Sensors	The sensory shall consist of accelerometers, gyroscopes and a foot sensor.	section 1.3	NA

Test Method:

By inspecting the hardware used its components are found.

Result:

The early prototype consist of 3-axis- Accelerometer and Gyroscope. To measure foot support the Nordic NeuroSTIM foot sensor is used.

H.2	Data Storage	The data storage must be capable of storing minimum 8 hours of sampled data.	section 1.3	NA

Test Method:

The sensors were connected to the main device and it was turned on and left over night - being checked after approximately 12 hours.

Result:

Checking the Gait01.csv files end milis entry being 21,598,340 milis.

TEST NOTE: the test was running 100 Hz but only incrementing the Time variable by 5.

$$h = \frac{counter_{value} \cdot \frac{1}{Sample_{rate}}}{increment \cdot 3600} = \frac{21,598,340 \cdot \frac{1}{100}}{5 \cdot 3600} = 11.999 \tag{5.1}$$

H.3	Connectivity	The hardware must be capable of sampling data from at least 9 sensors (7 IMU's and 2 foot sensors).	section 1.3	NA

Test Method:

Connecting sensors to the device and turning it on running it for a minute, turning the device off and reading the SD-card checking that it contains 7 sets of IMU data and 2 sets of Foot sensor data.

Result:

This test is not completed due to the prototype stage, however it is possible to sample two IMU and two Foot sensors at 100 Hz.

H.4	Data Sampling	The hardware must be capable of sampling the sensors at a rate of minimum 100 Hz.	section 1.3, 2.2	S.1

Test Method:

Measuring the AD0-sens 1 pin on the extender board the interrupt frequency is found, as this controls the sampling of the sensors it result in the sensor sample frequency.

Result:

Having connected an oscilloscope probe to the AD0-sens 1 pin the following oscilloscope image was obtained.

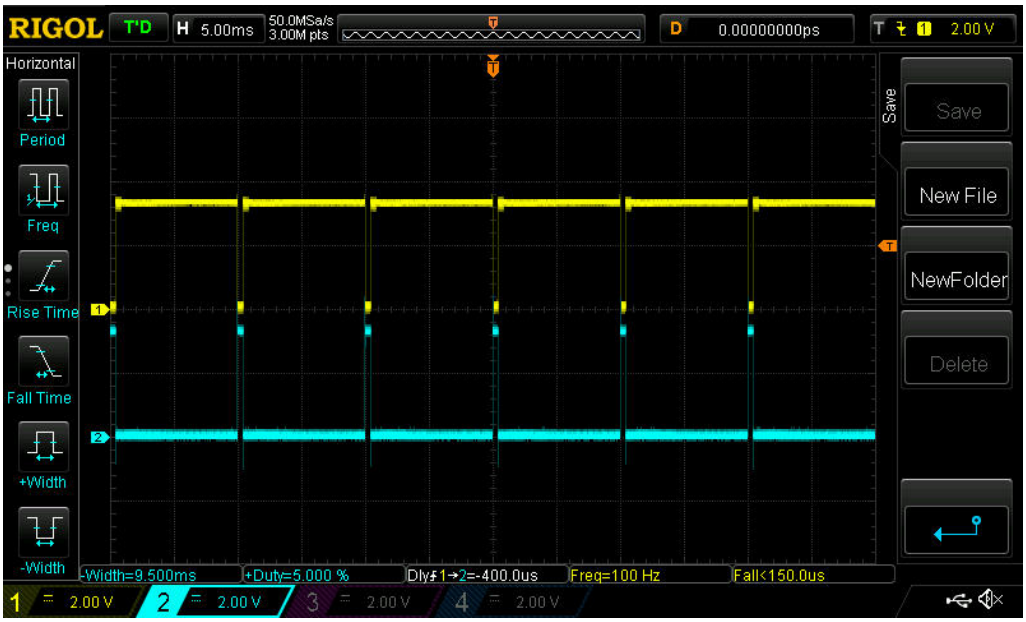

Figure 5.1. 100 Hz sample rate, time base 5 ms, scale 2 v, Channel 1 (yellow) is connected to AD0-sens 1 pin, when the signal drops the MEMS is activated i.e. having address 0x68.

H.5	Power	The power supply must be capable of powering the device for at least 8 hours of use.	section 1.3	H.2

Test Method:

Connect the sensors to the device and turn it on, leave the device for 8 hours, checking up on it in 5 minute intervals.

Result:

Testing was not done due to product only being on early prototype stage.

H.6	Stored Data	The data must be stored as unmodified values.	section 1.3	S.3

Test Method:

Program the device to output the binary value of a sensor reading to on of the processors digital port, measure the value and compare it to the one stored on the SD-card

Result:

This test is not complete. The value stored on the SD-card is ASCii coded, the value should be unmodified only changing its format.

Software Requirements

ID	Title	Description	Parent	Child
S.1	Timing	The software must run on a timer controlled loop for precise sampling of at least 100 Hz.	H.4	NA

Test Method:

See Test Method H.4

Result:

See Test Result H.4

S.2	Connected sensors	The software must be capable of sensing how many sensors are connected and arrange data accordingly.	section 1.3	NA

Test Method:

Connect all sensor to the device and turn it on for a minute. Turn off the device and

remove one sensor, turn the device back on for a minute. This process is repeated until (and including) no sensors are attached. Having run the device without sensors add one sensor and repeat the process until all sensors are reattached to the device. Check the data logged onto the SD-card for data arrangement

Result:

This test is not done because this feature is not yet available to the early prototype.

ID	Title	Description	Parent	Child
S.3	Sampled Data	Sampled data must be stored unmodified.	section 1.3 & H.6	NA

Test Method:

See Test Method H.6

Result:

See Test Result H.6

ID	Title	Description	Parent	Child
S.4	Data Storage	Stored data must be stored in a uniform fixed format.	U.6	NA

Test Method:

Checking the data log files generated during Test Methos S.2, Checking that time stamp i written in column 1 followed by the MEMS data form the connected MPU-6050 boards, then the data from the Foot sensors.

Result:

Reading the .csv file generated it is verified that the data is stored in a fixed format.

```
milis ,AcX0,AcY0,AcZ0,GyX0,GyY0,GyZ0,AcX1,AcY1,AcZ1,GyX1,GyY1,
5,1480,-16448,-5728,-280,-73,-271,1232,-16632,-4568,-453,-80,
10,1348,-16428,-5868,-275,96,-162,1280,-16452,-4384,-391,-30,
15,1420,-16448,-5684,-279,99,-177,1220,-16508,-4428,-462,-6,
20,1376,-16416,-5824,-285,105,-175,1304,-16548,-4332,-421,-32,
25,1420,-16468,-5676,-286,105,-168,1324,-16636,-4436,-413,-36,
30,1416,-16496,-5800,-253,99,-187,1252,-16556,-4420,-430,-41,
35,1424,-16460,-5560,-294,93,-173,1244,-16564,-4384,-408,-48,
40,1464,-16392,-5780,-285,121,-186,1268,-16444,-4308,-432,-63,
45,1464,-16416,-5700,-265,139,-192,1344,-16528,-4360,-415,-53,
```

NOTE: This is only part of the .csv file, the original file contains more columns. Data displayed does not come from Test S.2

ID	Title	Description	Parent	Child
S.5	Process Contraint	All data sampled from on loop run must be stored in the data storage before next run starts.	section 1.3	NA

Test Method:

This is tested using a modified version of the product software, having added exstra control of an external digital pin. Setting the pin HIGH when entering the ISR and setting it LOW when all data is stored. Measuring the "on" time, if it is shorter than the sample time the test is passed.

Result:

This test is not completed.

5.2 Conclusion

This project titled *Gait Sensor* themed and aimed at producing a medical device for use in gait research. The initiating project problem being:

Development of a low cost, low complexity gait data acquisition system for use in non-laboratory environments

A meeting was held with Sabata Gervasio the original proposer of the project. Sabata Gervasio had some specific requests as to the hardware used in the project and based on these and a simplified analysis of the human gait cycle a prototype was developed.

To design a Gait sensor using the hardware Sabata specified an electronic system is made. Making a prototype capable of the task set the following equipment was used:

- Arduino DUE development board
- IMU - MPU-6050
- Nordic NeuroSTIM Foot Sensor
- SD-card extension board for Arduino development boards

The prototype is based on the Arduino DUE functioning as the platform onto which the prototype is build. Building an extender board to use with the Arduino DUE board enabled me to connect the sensor cables. The sensor cables being purposely build to fit the gait sensor project scope of measuring lower body segmental movement.

The prototype developed during the project period is capable of sampling two MPU-6050 IMUs and six ADC at a sample rate of minimum 100 Hz and storing the data on a SD-card before the next sampling occurs. Handling a total of 26.4 kbit per second.

When the sensors are connected and a FAT16 or FAT32 formatted SD-card is inserted in the SD-card extension board, the Arduino DUE is powered up running a setup rutine

after which sensor sampling starts. Because the data is stored just after each sampling, the device is simply turned off when monitoring is done. The SD-card now contains the sampled data each sample tied to a specific time in a comma separated file using the extension .csv directly importable into MATLAB or another editor.

5.3 Further Development/Research

Developing further on this prototype adding more MEMS and optimizing the software using a proper C language, and designing PCBs making it more sturdy would take the prototype along way. Optimizing the device would make it more ready for role play where its mechanical design is tested. The sensors interface considered it is properly better to change them to a different model like the MPU-6000, with greater data rates. As research progress some sensor measurements might be omitted due to redundancy from other values or lack of information gain.

A wireless sensor solution could also be considered it does however require, that each sensor include its own power supply, but considering today's Embedded technology a single cell battery should be capable of powering the device for one monitoring session. Making the system wireless a Wifi connection could be use full for uploading data directly to the clinician, enabling data analysis to be done on an early stage,

As this is being developed as a research tool considering how to keed the cost down per unit, could also be analysed to make a cost benefit analysis.

A lot more time can be sped on developing this project into a finished product, helping gait research.

Bibliography

BH Alexander, FP Rivara, and ME Wolf. The cost and frequency of hospitalization for fall-related injuries in older adults. American journal of public health, 1992.

K. Aminian. Computational intelligence for movement science. IGI Global, 2006.

Rajan Arora. I2c bus pullup resistor calculation (slva689.pdf). Texas Instruments Application Report, 2015.

Analog-to-Digital Converter in the SAM3S4. ATMEL., 2011.

AT91SAM ARM-based Flash MCU. ATMEL., 2012.

De Haagse Hogeschool. Bergwandelen Het Onderzoek. url: http://www.eduweb.hhs.nl/¿bergwandelen/onderzoek.htm, 2015.

JM Hausdorff, HK Edelberg, SL Mitchell, L a" Goldberger", and JY Wei. Increased gait unsteadiness in community-dwelling elderly fallers. Archives of physical medicine and rehabilitation, 1997.

JM Hausdorff, D "a Rios", and HK Edelberg. Gait variability and fall risk in community-living older adults: a 1-year prospective study. Archives of psysical medicine and rehabilitation, 2001.

MPU-6000 and MPU-6050 Product Specification Rev. 3.3. InvenSense Inc., 2012.

N Kern and MH Granat. Multi-sensor activity context detection for wearable computing. EUSAI, 2003.

B. Maki. Gait changes in older adults: predictors of falls or indicators of fear. Journal of the American Geriatrics society, 1997.

S Miyazaki. Long-term unrestrained measurement of stride length and walking velocity utilizing a piezoelectric gyroscope. IEEE Transaction on Bio-medical engineering, 1997.

Carlotta Mummolo, Luigi Mangialardi, and Joo H. Kim. Quantifying dynamic characteristics of human walking for comprehensive gait cycle. Journal of biomedical engineering, 2013.

Physiopedia. Gait - Phases of the gait cycle. URL: http://www.physio-pedia.com/Gait, 2015.

Daniel A. Sterling, Judith A. O?Connor, and J Bonadies. Geriatric falls: injury severity is high and disproportionate to mechanism. Journal of Trauma-Injury & Critical care, 2001.

the scientific world journal. three-dimensional gait analysis can shed new light on walking in patients with haemophilia. url: http://www.hindiwa.com/journal/tswj/2013/284358/, 2013.

K Tong. A practical gait analysis system using gyroscopes. Medical engineering & physics, 1999.

A MEMS Fixture drawing

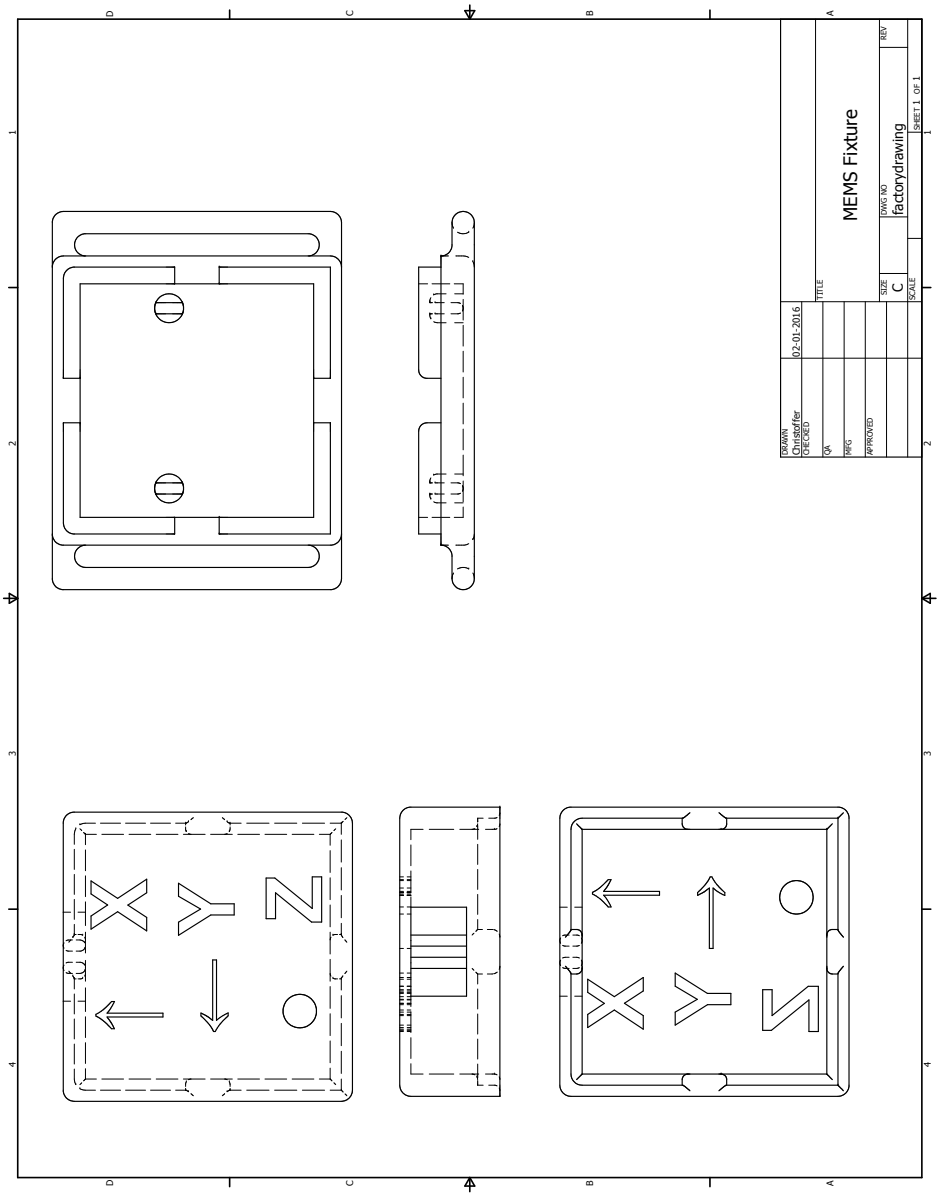

Figure A.1. Mechanical drawing of MEMS Fixture

B SAM3X8E block diagram

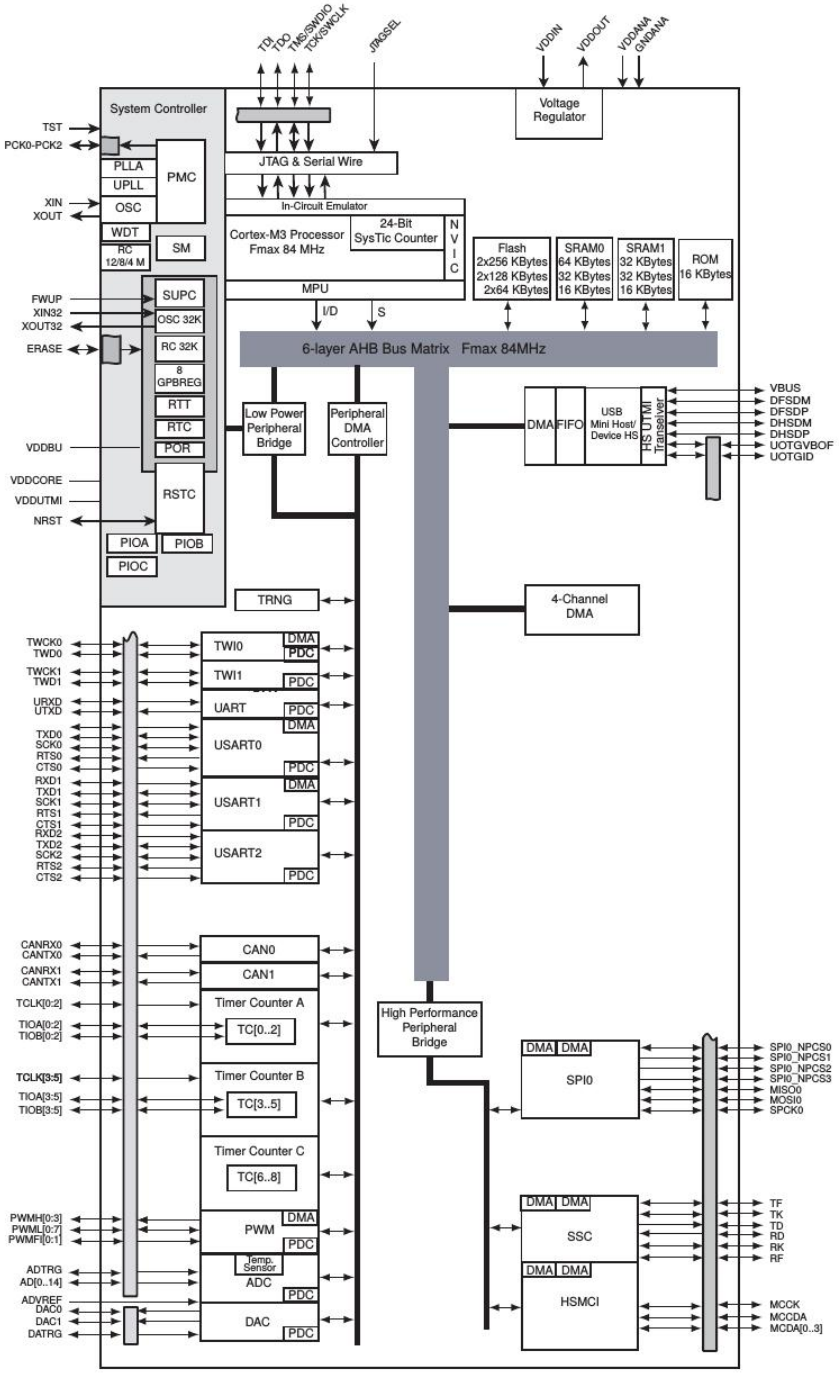

Figure B.1. Block diagram of SAM3X8E