UNIVERSITY OF OSLO
Department of Informatics

Efficient data collection using Android ADK in a high velocity, mobile environment

Master thesis

Gard Brandt Sandholt

May 1, 2012

Abstract

The purpose of this thesis was to investigate the use of a microcontroller and mobile phone to create a highly mobile data acquisition device for human movement. Another aim was to make the technology as transparent as possible and explore development possibilities.

Based on the description of use case scenarios the first set of system requirements were extracted. These system requirements lead to a set of technology requirements. A technology evaluation was conducted in order to find technology components for a prototype where the intended application were an internet connected center of pressure measurement application for an athlete in the ski jumping hill.

The results include a working prototype that demonstrated the functionality. The technology has a lot of potential and room for improvements in order to be as technology transparent to the athletes as possible. The results of tests showed that there was a need for further analysis in regards to the performance of the prototype and what requirements that are needed when applied in the ski jumping hill.

Acknowledgements

This thesis would not have existed without the help and guidance of several individuals and groups of people who have helped in one way or another and contributed to the preparation and completion of this thesis.

First and foremost I would like to thank my supervisor Professor Josef Noll of the University of Oslo & Movation for his help and guidance throughout this thesis. He has encouraged me to do a thorough technology analysis and I have gained invaluable knowledge from his supervision, points of view and examples.

My wife Mirjam for her patience and kindness. My son Joachim for his presence in my life. My parents and sister Ane for their support. I would not be the same without my friends and family who have given me the opportunity and freedom to seek out what I really want to do in life.

All the people working with technology who inspire me in one way or another to do what I love everyday. All the people who contribute to the DIY (Do It Yourself) communities around the internet who in one way or another inspire me everyday to do fun projects which enrich my life.

Lastly I would like to thank the University of Oslo and the Department of Informatics who gave me the possibility to pursue professional growth and has given me the opportunity to work in really nice facilities in the Ole Johan Dahl's building.

Glossary

AAP AndroidAccessory Protocol. 28, 43, 44, 63, 64, 68, 71

ADC Analog-to-Digital converter. 14, 17, 38, 41, 49–51, 61, 64, 75–79, 81

ADK Open Accessory Development Kit. 4, 6, 7, 24, 25, 29, 30, 37, 38, 40, 53, 58, 59, 76

ADT Android Development Tool. 60

ARM Advanced RISC Machine. 51

CDE Compact Dalvik Executable. 26

CPU Central Processing Unit. 13, 16, 17, 50, 76

DAQ Data acquisition system. 50

DIP Dual in-line package. 21

EDGE Enhanced Data rates for GSM Evolution. 2, 28

FlexiForce FlexiForce®. 23, 24

FSR Force Sensitive Resistors™. 13, 22, 23, 32, 35, 38, 39, 47–50, 52, 54–56

GPRS General packet radio service. 2, 13, 44, 52, 76, 77

GUI Graphical User Interface. 60, 70, 72, 78–80

I/O Input/Output. 41

IC Integrated circuit. 12, 14, 39, 40, 62, 75, 76

ICSP In Circuit Serial Programmer. 20

IDE Integrated development environment. 19, 26, 27, 59, 60, 66, 86

IOT Internet of Things. 1, 2

IP Internet Protocol. 32, 63, 66, 70, 71

ISA Instruction Set Architecture. 17

JVM Java Virtual Machine. 25, 26

LED Light-emitting Diode. 21, 26

MCU Microcontroller. 4, 6, 8, 9, 11–22, 24–28, 30, 32, 34, 36–44, 46, 47, 49–54, 56–64, 66, 68, 69, 74–79, 81, 83–86

NAT Network Address Translation. 32, 63

OS Operating System. 25, 27, 29

PC Personal Computer. 32, 54

PCB Printed circuit board. 19, 80, 81, 85

PWM Pulse-Width Modulation. 14, 20

RAM Random Access Memory. 18

RISC Reduced Instruction Set Computer. 16–18

SDK Software Development Kit. 25, 29, 30, 60

SMD Surface Mounted Device. 39

SMT Surface Mounted Technology. 39

SPI Serial Peripheral Interface bus. 14, 17, 22, 31, 40, 49, 51, 62, 64, 75, 76, 79–81, 83

TCP Transmission Control Protocol. 52

TI Texas Instruments. 17, 19, 21

UART Universal Asynchronous Receiver/Transmitter. 14, 20, 75

UDP User Datagram Protocol. 52

UMTS Universal Mobile Telecommunications System. 2, 13, 28, 44, 76, 77

USART Universal Synchronous/Asynchronous Receiver/Transmitter. 14, 17

USB Universal Serial Bus. 4, 9, 10, 13, 14, 19–22, 26, 28, 30–32, 37, 39, 40, 42, 43, 49, 51, 52, 57–59, 62–66, 68, 69, 76, 79, 81, 83

WLAN Wireless Local Area Network. 2, 13, 28, 44, 76, 77

WSN Wireless Sensor Networks. 4

Contents

Glossary **I**

Contents **III**

List of Figures **V**

1 Introduction **1**
1.1 Scenarios . . . 2
1.2 Problem Statement . . . 4
1.3 Scientific method . . . 5
1.4 Thesis structure . . . 6

2 Technology evaluation **8**
2.1 Key Features & method . . . 8
2.2 Requirements . . . 12
2.3 Key Technologies . . . 13
2.3.1 Microcontroller . . . 13
2.3.2 USB Host capabilities . . . 21
2.3.3 Sensors . . . 22
2.3.4 Mobile phone . . . 24
2.4 Middelware . . . 25
2.4.1 Programming environment . . . 25
2.4.2 Communication . . . 28
2.5 Application domain . . . 29
2.5.1 Mobile phone OS . . . 29
2.5.2 Client/Server/Peer-to-Peer . . . 32
2.6 Summary . . . 32

3 Functional architecture **34**
3.1 Intro and use-case specification . . . 34
3.2 Overview of capabilities & block description . . . 36
3.2.1 The four main blocks . . . 37
3.2.2 The three main interfaces . . . 41
3.3 Detailed specification of main blocks . . . 45

3.3.1 Sensors . . . 47
3.3.2 MCU . . . 49
3.3.3 Mobile phone . . . 51
3.3.4 Server and client control . . . 51
3.3.5 Summary . . . 52
3.4 Prototype system overview . . . 52

4 Implementation **53**
4.1 Prototype overview . . . 53
4.2 Development of the hardware . . . 54
4.3 Development of the software . . . 59
4.4 Hardware interaction . . . 61
4.5 Software interaction . . . 64
4.6 Summary . . . 66

5 Prototype demonstration and evaluation **67**
5.1 Demonstration . . . 67
5.1.1 Demonstration overview . . . 67
5.1.2 Run through . . . 70
5.2 Evaluation . . . 74
5.2.1 Requirements . . . 74
5.2.2 Functionality . . . 77
5.2.3 Performance . . . 77
5.2.4 Extensibility . . . 80
5.2.5 Evaluation summary . . . 81

6 Conclusion **84**
6.1 Conclusion . . . 84
6.2 Further development . . . 85

A Programming the Arduino **86**

Bibliography **I**

List of Figures

2.1 Solution proposal 1 . . . 9
2.2 Solution proposal 2 . . . 10
2.3 Solution proposal 3 . . . 11
2.4 Arduino Mega 2560 . . . 20
2.5 mbed NXP LPC1768 . . . 21
2.6 USB Host shield . . . 22
2.7 Interlink 402 . . . 23
2.8 FlexiForce® (FlexiForce) sensor . . . 24
2.9 Arduino Integrated development environment (IDE) . . . 27
2.10 Android ADK Demo Shield . . . 32

3.1 Selected block design . . . 36
3.2 Prototype (Taunus) system block topology . . . 37
3.3 Selected sensor, Interlink 402 . . . 38
3.4 Selected Microcontroller (MCU) prototyping board, Arduino Mega 2560 . . . 39
3.5 Selected mobile phone, Samsung Nexus S . . . 40
3.6 Mobile phone to MCU Universal Serial Bus (USB) physical interface . . . 43
3.7 Figure over the ski jump in-run segments . . . 46
3.8 Resistance at different force measurements . . . 47
3.9 Measurement of feat pressure, white is higher force and green is lower force 48

4.1 Force Sensitive Resistors™ (FSR) with soldered pins . . . 54
4.2 Voltage divider formula . . . 55
4.3 Voltage divider circuit . . . 55
4.4 Four voltage divider circuits . . . 56
4.5 Sensor circuit, back . . . 57
4.6 Sensor circuit, front . . . 57
4.7 MCU circuit . . . 58
4.8 USB host shield . . . 58
4.9 MCU circuit w/battery . . . 59
4.10 System interface overview . . . 61
4.11 Screenshot of the client controller server application . . . 65

5.1 Sensors mounted under shoe soles . . . 68
5.2 Arduino MCU prototyping board, USB host shield, sensor circuit perfboard 68

5.3 Android mobile phone connected to Arduino 69
5.4 Server application with sample data 69
5.5 Server application at initial launch 70
5.6 Android application screenshots 71
5.7 Server application with sample data 72
5.8 Sample of walking data 73
5.9 Detailed sample of walking data 73
5.10 Test subject of 80 kg testing the system 80

Chapter 1

Introduction

Computer network have been dominated by wired networks in the past. The last decade has shown an explosion of wireless networks, the size of internet and internet bandwidth. This has made it possible for computers all over the world to communicate and exchange data freely. These technologies has made it possible for what we see today, and what is becoming increasingly more popular everyday. One could call it the Internet of Things (IOT)[1].

The properties of the sensor devices is that they sense the environment environment around them, they use very little power and they are highly customizable. The sensor devices can be fitted with many types of sensors and communication devices. With these properties the sensor devices can sense its environment and communicate its collected data with its surrounding machines wirelessly. This enables a wide range of application and communication devices e.g. the mobile phone. Small devices like mobile phones now have full access to the internet, their coverage is increasing while their battery consumption is decreasing. Everyday the mobile phones have an impact on society, and the development so far suggest we will have better integrated internet on mobile phones in the future.

Further development will introduce data collecting sensor systems in more aspects of society than we see today. The possibilities are endless to where sensor networks can be deployed. Industries and domains like healthcare, automobile industry, sports and traffic is going to see a lot of development of these kinds of systems that will interact with our lives more and more in the future, and may provide us with useful information that will improve our lives.

Knowing the limitations of the sensors and sensor systems is increasingly important in order to construct and utilize the best solutions in any given situation. There are many different types of sensors for detecting all kinds of changes in the sensors surrounding environment. When the limitations of the sensors is known it is easier to determine where and when to use the sensor. This becomes crucial in environments where vital information is needed in order to perform the right diagnostics to a problem.

Microcontrollers can be found everywhere and in a large number of products and devices. Products like automobile control systems, medical devices, remote controls, power tools, office machines, appliances and toys. There are a wide variety of technologies for communication between sensor systems. They have different properties in terms of throughput, range and scalability. Technologies like Universal Mobile Telecommunications System (UMTS), General packet radio service (GPRS)/Enhanced Data rates for GSM Evolution (EDGE), and Wireless Local Area Network (WLAN) are found in mobile phone like the ones powered by the Android or iOS operating systems. They are very powerful in terms of use-case scenarios and can is used to connect to internet and share large amounts of data. When utilizing these technologies it is important to know their properties and their limitations.

While the evolution of sensor network integration continues the need for defining the limitations for these systems increases. This is natural as the demand for development of new applications with their own requirements increases and the need for these devices to perform as expected is increasing.

1.1 Scenarios

The scenarios in this thesis will focus on sport as an example where an integrated sensor circuit will contribute to an IOT application.

From the dawn of time mankind has always been competing against each other through the form of athletic games. All the way back from the ancient olympic games in ancient Greek until today where the competition forms and athletic games has evolved into a wide variety of sports and games. The athletic games has been the main entertainment field for human life on earth throughout our history.

Here in Norway today we are very interested in nordic skiing sports which consist of cross country skiing, ski jumping and nordic combined. These sports are based on cross country skiing which has been used for hunting and movement in the winter. Today skiing is a great part of the history especially for people living in scandinavia and its a well know recreational activity.

The main focus of this thesis is on the challenges a team faces when developing new and better methods for the athletes to develop and enhance their skills. And the limitation an athlete and their team face when they collect data from the teams participants. Most of the data collection of the athletes today are done manually by observing the athletes when they train and perform. Through video recording its possible to monitor and analyze the athletes and compare previous performance with the state of today as well as comparing the athletes against each other. So there is always a challenge to better the way the team develop their athletes and push the teams and their athletes limitations.

There are a number of challenges a team faces when they develop their training program. The following scenarios is some of the areas where sensor technology could prove effective.

Use of resources

If the teams were able to know how the athletes distributes all of their resources, they would be able to develop better programs and better the teams performance. By measuring and comparing the athletes against each other it would become clear how the athletes use their resources and who get the best results.

Heighten average results

The teams are interested in develop their average team result. If the whole team perform on a higher level the team spirit is greater and this might get the team more wins. By collecting more data from each of the teams individual athletes the team it would be possible to profile each athlete and compare the athlete with best performance to the others result in order to better the athletes overall average.

Distribution of power

The power distribution is one of most significant areas of athletic performance. It is crucial to save power for the right moment and to time when to take out the most of the power. The teams trainer needs to know how the bests athlete of the team distributes its power in order to find the optimal power distribution over the course. In order to outline and develop the team program the trainer can use the power distribution data to further develop the team.

Most effective power distribution

A challenge is to know which power distribution is the most effective. For instance in a sport like ski-jumping the power distribution through the push against the table is very interesting in terms of the outcome for the jump. Knowing which athlete that has the best power distribution could help the team develop methods for improving the whole teams power distribution and raise the level of the whole team.

Athletes center of pressure

The only way for a trainer to know how the athlete is balancing is to record video and analyze it to determine the balance of the athlete. This could be a lot simpler if it were possible to record the actual center of pressure of the athlete. This would contribute to a better understanding of what really goes on in the video and how the different athletes distribute their balance through their run.

The scenarios in this thesis focus on a ski jumper in motion, further analysis will lead to a prototype which will be able to summarize the information of a ski jumper in order to contribute to better measurement of the center of pressure and performance, subjects that there is close to zero knowledge of from the outdoor hills[2].

1.2 Problem Statement

Data collection systems could be used to monitor athletes in motion or high velocity to detect many types of data e.g. heart rate monitors that is used in sports today[3]. This would require the system to have some or all of the following high level properties. It would have to be lightweight with high processing power and with long battery life making it very mobile and it would have to be very customizable and cheap to develop and build.

If the scenario is to measure the center of pressure of athletes in motion then measuring the center of pressure data will have a specific set of requirements. The specific set of requirements will define the requirements for the key technologies. The challenge in choosing the right hardware components are reduced when the specific set of requirements are laid out and the limitations of the hardware components are known. The data collection unit would mainly consist of 4 main components. A custom sensor circuit consisting of one or more sensor types, a MCU which collects data from sensors and act as a USB host device to a mobile phone, a mobile phone which interacts with the MCU and a server host which sends command messages to the phone and receives the collected data from the mobile data collection system.

Google announced the release of the Android Open Accessory Development Kit (ADK) at the Google I/O 2011. The Android ADK technology is not new, it's not revolutionary nor does it make a dent in the universe. What the Android ADK is is a look into the future of Wireless Sensor Networks (WSN)[4]. The platform lets the users customize the Android accessory connected to the phone and similar solutions is available for the iPhone. This opens up a lot of interesting opportunities for creating many interesting applications. These kinds of applications would be highly customizable and enable the utilization of many different sensor types as well as being a highly mobile internet device.

A big challenge is picking the right hardware components and evaluating the components right. Evaluating the technologies is vital for a project like this to succeed. Knowing the hardware components, their usage area and their limitations would be the main goal of an evaluation of the technologies.

When the right components for the prototype are selected and tested for limitations, a functional prototype will test the components together. This lead to further evaluation of the components where new limitations may arise. Integration of the heterogeneities of the system should be agile and not take up to much time that would lead to delays in development. Some of the high level properties of the prototype will have be:

Highly Mobile
: The prototype needs to be able to move from one point to another and still perform as expected.

High Velocity
: The prototype need to be able to move in a high velocity environment like being

attached to an athlete racing on skis.

Weight scarce
The prototype need to be weight scarce in order to not to conflict with the athletes normal procedures.

The requirements of a highly mobile, high velocity and weight scarce device will lead to further analysis of a modern sensor collection device which will reflect the state of today where smaller and more powerful devices become cheaper and can be found in an increasing number of places.

The main goal of the prototype system is to demonstrate a cheap, lightweight and highly mobile way of data collection being able to measure the center of pressure of a ski jumper. The data collection can utilize a wide range of sensors but in this thesis the focus will be on collecting the center of pressure and ground reaction of a human being in motion[5].

By using processing power of a modern mobile phone and its internet connection is simpler and more efficient than building a custom data transfer system. A mobile device suits this purpose good because it is small, have high processing power, well documented and rich data libraries, internet connection and GPS to name a few interesting features. All these features make modern mobile devices great for testing the probability for using them for data collection in a mobile environment[6]. This technology can among a wide range of other things be used to monitor how the body is affected in high velocity movement. Studies is performed to gain better understanding of human movement science, however because of the practical difficulties of collecting data in the in-run of the ski jumping hill the data surrounding this subject is incomplete[7].

1.3 Scientific method

The scientific approach in this thesis is mainly based on establishing a use-case scenario where the main series of actions and events for the project is laid out. This forms the basis of the establishment for the rest of the requirements needed to conduct a thorough analysis for this thesis.

When the use-case scenario is established and defined the need for laying down a first set of requirements to the key features of the project. The first set of requirements will set the first expected minimal limitation of the project. Further development may conclude that the limitations of the system is breached and a reassessment of the requirements will be conducted.

When the first set of requirements are extracted from the use-case scenario the thesis outlines a list of key technologies which are based upon the use-case scenario. These tech-

nologies needs to be compared against alternate technologies in order to fulfill the first set of requirements. If no limitations for the core technologies are met the technology will be selected and put on the list of technologies.

After establishing a list of key technologies the evaluation of these technologies will begin. When performing a first evaluation of the key technologies the limitations of the technologies is in focus. The thesis will evaluate the limitations of key technologies in order to meet the first set of requirements.

When the evaluated technologies meets the first set of requirements and limitations the thesis will outline the core of the functional architecture. This is done by designing a prototype specification based on key technologies that is selected in respect to the set of requirements. This prototype will make it possible to conduct an experiment based on the use-case scenario.

The thesis will then outline how the prototype is established with focus to testing the functionality of the components during development. When limitations to one or more components is discovered, if the performance of the component does not meet the requirements, the prototype is redesigned and the functional architecture of the prototype is altered.

At last the prototype is demonstrated which is then the basis of an evaluation of the system. The evaluation of the prototype will evaluate whether the prototype meets the requirements and identifies the limitations, whether the components selected for the core architecture meets the minimal requirements for the use-case scenario. This will lead up to a brief conclusion of the project.

1.4 Thesis structure

This chapter gives an introduction to the thesis. It outlines the scenarios for a sensor system for monitoring an athlete in motion and the problems the system might encounter in order to supply teams of athletes with data, which may work as a supplement to their existing programs, and help them develop their skills and raise the average result of the teams.

Chapter 2

Chapter 2 describes the state-of-the-art technologies, hardware components, software middleware components and user application components which will make up a prototype for a sensor system which will assist an athlete with data collection. The chapter describe mobile ADK technologies in combination with MCUs combined into the server/client domain as well as an analysis of the limitations and evaluation of the individual components.

Chapter 3

Chapter 3 suggests a course of action for implementing the prototype which will be developed by the components that fit the prototype specification the best. The chapter suggests a component integration based on use-case scenarios which set the basis for the basic prototype requirements. The chapter also points out how the application user will perceive the system in use and how they are able to retrieve information from the system.

Chapter 4

Chapter 4 explains the prototype in detail and outlines the characteristics of the individual components. The chapter also explains how the individual components make up the entire prototype and how the prototype of the system is assembled. The chapter also compares the suggested methods of implementations provided in chapter 3 with the method of implementation that the prototype is based upon and developed.

Chapter 5

Chapter 5 gives a demonstration and evaluation of the prototype. The features of the prototype is presented in the demonstration and a general performance analysis and performance measurement is done. The result is evaluated in the evaluation part of the chapter and is evaluated in terms of their performance and limitations laid out in Chapter 2 where the individual components were described and analyzed for use in the prototype.

Chapter 6

Chapter 6 conclude the results and evaluation of the prototype and make a conclusion to the overall performance of this thesis prototype. The conclusion will reflect upon whether or not the ADK is a technology suited for high velocity data collection and what the results of this thesis. If the case is that the technology is suited for the high velocity data collection, a suggested course for further development is presented and how this could be executed.

Chapter 2

Technology evaluation

This chapter describe the state-of-the-art technology components in modern sensor technology systems and an evaluation of the individual components. The development of the prototype is based on decisions made after evaluating the technology based on the first set of requirements. First the system components are described and evaluated, these components are the ones that make up the MCU, sensor circuit and sensors of the prototype. Second the systems middelware components are described and evaluated. These components are software libraries and software that interact between the hardware and user application. Third and last part of the technology overview and evaluation is the application domain, the part of the system which the user interact with to control the system and get data out.

2.1 Key Features & method

This section outlines the first set of requirements of the system. The basic building block which the prototype will be built upon is evaluated throughout this chapter and selected based on the requirements of the prototype. Based on the use-case scenarios three solution models have been outlined in order to establish the first set of technology requirements.

Solution 1

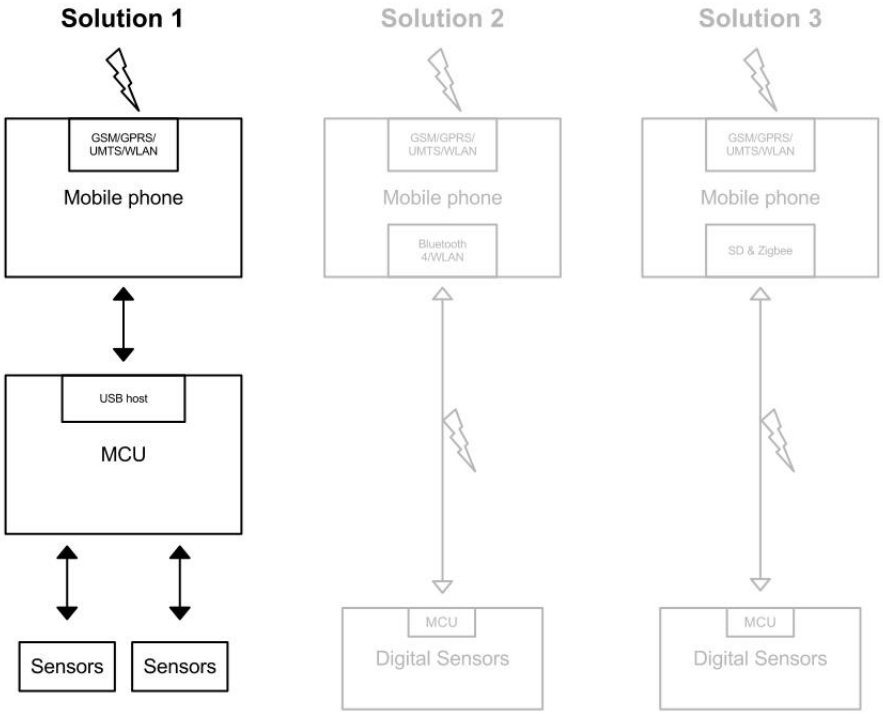

Figure 2.1: Solution proposal 1

The model in Figure 2.1 shows a possible solution where the sensors are connected to a MCU with USB host capabilities. The sensors are connected to the MCU through physical wires which carry the power and force signal. The MCU acts as an USB host entity and the mobile phone is the slave. The MCU receives force measurements from the sensors and forward the information to the mobile phone where data can be stored or forwarded through the mobile phones communication channels.

Solution 2

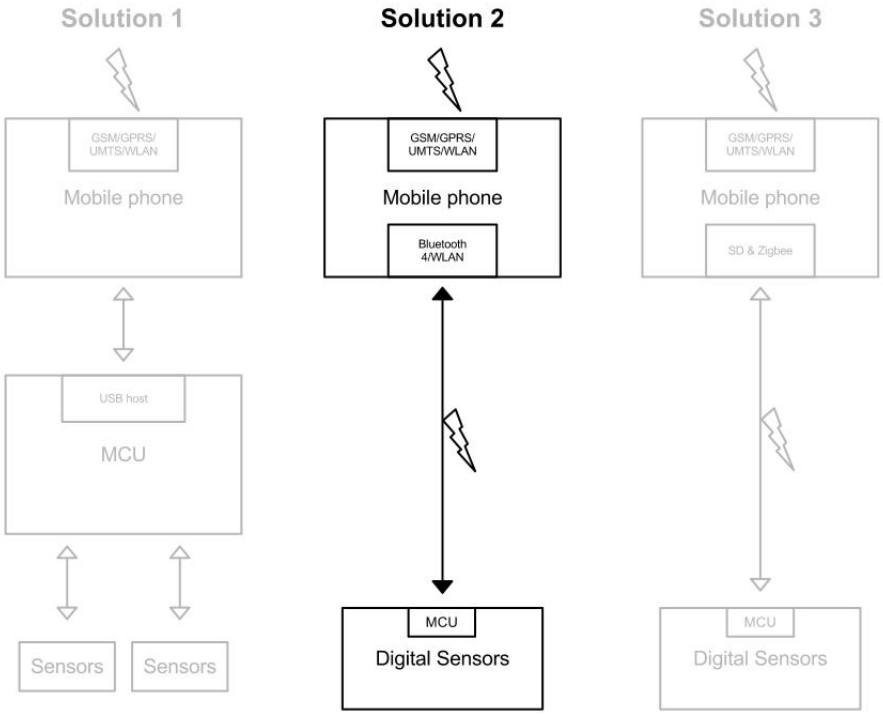

Figure 2.2: Solution proposal 2

The model in Figure 2.2 shows a possible solution where the sensors transmit wireless to the mobile phone. The sensors are powered through battery and transmits its data wirelessly through Bluetooth v4.0 or WLAN communication. This solution use the sensors wireless communication instead of being a USB host entity to forward its data to the mobile phone block unit.

Solution 3

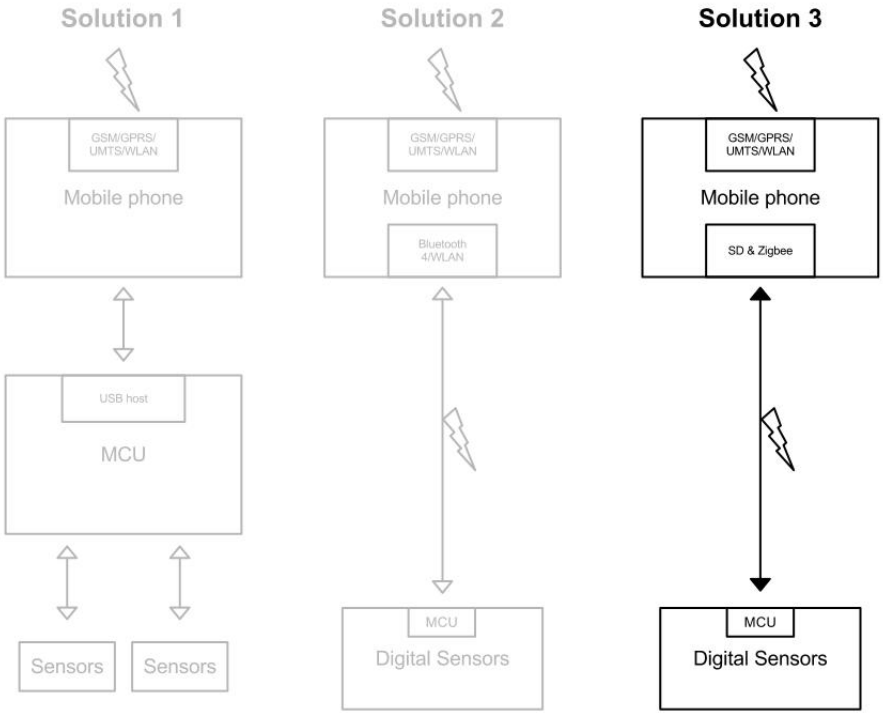

Figure 2.3: Solution proposal 3

The model in Figure 2.3 shows a possible solution where the sensors are wireless and transmit via Zigbee through a SD card. This solution is the same as the solution proposed in Figure 2.2 except with different communication. Since this solution also use wireless communication from sensors to mobile phone the challenge of keeping the sensor block of the system small and light enough is a bit greater than the solution proposed in Figure 2.1.

Based on the preliminary solution proposals, solution proposal 1 is selected because of its ease of development an extensibility. Further development may evolve into systems more similar to other solution proposals. Solution proposal 1 uses cabled connection between a mobile phone, MCU and the sensors. The next section will extract the first set of requirements based on solution proposal 1.

2.2 Requirements

Based on the proposed solutions a basic set of requirements can be extracted in order to determine which technology that is going to be evaluated. This section will outline the individual requirements for the blocks in the sensor system.

The data collection system has to be in the order of 200-300 grams (g), thus weight is tried to be minimized and considered in all components of the first set of technology requirements and in terms of future development.

Sensors requirements

The first set of sensor requirements includes:

Lightweight
In order to not interfere with the athletes equipment the sensors have to be in the order of 1 g and less.

Size
In order to fit into shoe soles the sensors have to be about 2 mm and thinner.

Force range
The sensors have to be able to give out readings for the amount of force per cm^2 of the sensors areal.

Power consume
They need to have as little power consumption as possible, in the order of 10 mA and less.

MCU requirements

The first set of MCU requirements includes:

Lightweight
The MCU needs to be lightweight, in the order of 10 g and less.

Size
The MCU needs to be smaller than 40 mm x 120 mm, small enough to not interfere with the athletes normal procedures.

Clock speed
The MCU needs to be fast enough to collect sensor data with more than 100 samples a second.

Number of ports
The MCU needs to have enough port for minimum 4 sensors.

Integrated circuit (IC) interfacing
The MCUs ability to interface between ICs.

Mobile phone requirements

The first set of mobile phone requirements includes:

Lightweight
The mobile phone needs to be less than 200 g.

Size
In order to not interfere with the athletes equipment the size must be less than 70 mm x 130 mm x 12 mm.

Central Processing Unit (CPU) speed
The mobile phone needs to be around 1 GHz and above, fast enough to collect the data received from sensors.

GPRS/UMTS/WLAN
The mobile phone needs to have wireless communication to the internet. GPRS is fast enogh to send commands in the order of 100 byte (B).

The system concept in this thesis is used in other systems. Polar has a system[1] which presumably use an accelerometer measures speed and distance to analyze the effectiveness of the run you take. Another system concept is the Nike+ iPod[2] which use an accelerometer to measure and analyze the effectiveness of your run. These two concepts are suitable for use in running and walking scenarios, if one were to do other data collection the concept would have to be customized.

2.3 Key Technologies

This section evaluates the key technologies, their properties and limitations, needed to assemble a functional force sensing sensor system. The capabilities and components in which the key technologies consist of are mainly a MCU, USB host capability and FSRs. This is the main hardware accessory of the system.

2.3.1 Microcontroller

A MCU is a small computer on a single integrated circuit containing a processor core, memory and programmable input/output ports. Programmable memory in the form of a NOR flash is often included.

When looking at MCUs the evaluation considerations is key for a successful project. The following evaluation considerations used for the MCU is:

Producer
The producer of the MCU is considered since each producer has their own features on the different MCUs.

[1] http://www.polar.fi/en/products/accessories/s3_stride_sensor, last accessed 23. Mar, 2012
[2] http://www.apple.com/ipod/nike/, last accessed 23. Mar, 2012

Architecture
The MCUs architecture is considered since the architecture affects power consumption and other features.

Processor core
The MCUs processor core and clock speed is considered since this affects how many calculations per second the MCU can do.

Programmable memory size
The onboard memory for the program is considered since it will hold the program running on the

Number of I/O ports
The sensors for this thesis project requires the MCU to have a minimum of four programmable I/O ports.

Type of flash memory
NOR flash or OPT ROM is two common types of onboard programmable memory.

RAM amount
RAM amount is considered since the available RAM has to support the required amount of running program.

Size
The size of the microcontroller is considered since it will affect the size of a production type of the prototype.

Cost
Cost is considered incase the production of a system like this thesis project would be produced.

Analog-to-Digital converter (ADC)
ADC is considered since the MCU needs to be able to convert analog signals to digital.

Pulse-Width Modulation (PWM)
PWM is considered although the prototype does not directly requires the PWM functionality it is a nice feature considering expansion of a specific prototype design.

Universal Asynchronous Receiver/Transmitter (UART)
UART and Universal Synchronous/Asynchronous Receiver/Transmitter (USART) capabilities are considered. Many microcontrollers and microcontroller equipment support UART so the feature is nice to have.

Serial Peripheral Interface bus (SPI)
SPI is used as a common interface to ICs. The IC can be a USB-host IC or a communications IC.

Interrupt
Interrupts are required for timers and other listening functionality on MCUs.

Processor bits
How many bits the processor handles is considered, 8-, 16- or 32-bit.

Clock frequency
The amount of clock frequency is considered since the MCU needs to collect data a certain amount of times a second.

Power consumption
The power consumption is crucial especially on battery driven circuits.

Programmability
The ease of programmability is important incase of development time, cost and learning curve.

The most important evaluation considerations for the MCUs are:

Producer
What type of MCUs are the manufacturers producing.

Prototyping board producer
What can the different prototyping board producers offer.

Prototyping boards
What are the most relevant prototyping boards.

Capabilities
What are the capabilities of the MCUs and prototyping boards.

MCU producers

The producers manufacture the MCUs and have different application goals in mind. Here are some of the most common producers and their common MCU features.

Microchip PIC

Microchip Technology[3] is an American MCU manufacturer which produce the PIC MCU line. PICs are popular with both industrial and hobbyists due to its low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools and serial programming and re-programming of on chip flash memory. Some PIC architecture features are:

- Harvard architecture, which separates the instruction and data memory
- Small number of fixed length instructions
- Most instructions are single cycle executions (2 or 4 clock cycles on 8-bit models)
- All RAM locations function registers as both source and/or destination of math and other functions
- A hardware stack for storing return addresses
- A fairly small amount of addressable data space (typically 256 B)
- Data space mapped CPU, port and peripheral registers
- The program counter is also mapped into the data space and writable

PIC MCU example:
The PIC24EP512GU810 is a 16-bit MCU with 512kB programmable flash memory.

Atmel AVR

The AVR architecture was developed by Alf-Egil Bogen and Vegard Wollan when they were students at Norwegian Institute of Technology (NTH) now Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. The AVR MCU technology was developed at Nordic VLSI where the two founders of Atmel Norway was working at the time. Atmel Norway was founded when the AVR technology was bought by Atmel Corporation in 1996 and has since developed to be the most important branch in the Atmel Corporation[4]. The term AVR is not defined however it is commonly accepted that AVR stands for **A**lf-Egil Bogen and **V**egard Wollan's **R**ISC processor[8]. Some AVR architecture features are:

- Reduced Instruction Set Computer (RISC) CPU architechture

[3]http://www.microchip.com/, last accessed 29. Apr, 2012
[4]http://www.tu.no/nyheter/ikt/article53873.ece, last accessed 12. Sep, 2011

- Mixed signal MCU, built in ADC
- USART, SPI
- Timers

Atmel AVR MCU example:
The ATmega2560 is a low power Atmel 8-bit AVR RISC-based MCU with 256kB flash memory. The MCU has 86 I/O general purpose lines and runs at 16MHz on 4.5-5.5 volts[9].

Texas Instruments (TI) MSP

The MSP mixed-signal MCU family is produced by Texas Instruments[5]. MSP is built around a 16-bit CPU and is designed to be low cost with low power consumption. There are six general generations of MSP430 processors. Some MSP architecture features are:

- 16-bit RISC CPU
- Ultra-low power, down to 120μA.
- Low power standby, down to 0.7μA
- Mixed-signal and digital technology

TI MCU example:
The TI MSP430 is a ultra low power 16-bit RISC-based MCU with 128B RAM and 2kB flash memory.

ARM MCU

ARM is a 32-bit RISC Instruction Set Architecture (ISA) developed by ARM Holdings[6]. ARM is suited for low power application due to its relatively simple architecture and as a result to this has become dominant in the mobile phone and embedded electronics market. Some ARM architecture features are:

- 32-bit RISC CPU
- Compact, low latency pipeline
- Integrated sleep state support
- Interrupt service routines in pure C

[5]http://www.ti.com/lsds/ti/microcontroller/16-bit_msp430/overview.page?DCMP=MCU_other&HQS=msp430, last accessed 29. Apr, 2012

[6]http://www.arm.com/, last accessed 29. Apr, 2012

ARM MCU example:
The ARM Cortex-M3 is a low power ARM 32-bit RISC-based MCU with 32KB Random Access Memory (RAM) 512kB flash memory.

One common feature is the low power consumption which is very important in embedded applications[10]. The four MCU producers are all great choices though the one selected depends on the use of MCU on a prototyping board which is evaluated next.

Prototyping board producer

The prototyping boards are preassembled Printed circuit board (PCB) boards based on MCUs. Some of the most common prototyping boards are evaluated here.

PIC Starter Kit

The producer of PIC MCUs develop preassembled prototyping platforms based on the PIC MCUs. Development environment and instructions is provided through Microchips web sites[7].

Arduino

Arduino is a preassembled MCU prototyping board based on Atmel's AVR MCUs[8]. It comes with development environment for writing software for the boards. Arduino can easily be used to develop interactive solutions that take input from a variety of switches and sensors and controlling a variety of lights, motors and other physical outputs. The Arduino projects can be stand alone solutions or communicate with software running on computers and mobile phones.

Launchpad

Launchpad is a preassembled MCU prototyping board based on the TI MSP430 MCU[9]. Launchpad includes all of the hardware and software needed to get started. It cost 4.30 USD and includes 2 programmable MSP430 MCUs, mini-USB cable, external crystal for increased clock accuracy and downloadable IDE.

mbed

mbed is a preassembled MCU prototyping board based on ARM architecture MCUs for use on solderless breadboard, stripboard and through hole PCBs. The mbeds comes with a built in USB programming interface which makes programming the MCU like using a USB Flash Drive. The compilation of code and making of the binary can be done through an online compiler which can be reached through mbeds web sites[10].

The Arduino and mbed prototyping boards are more interesting because of their internet community. When checking out the internet sites for information it becomes clear that they have a strong community which will be of much help during development.

[7] http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010053, last accessed 27. Apr, 2012

[8] http://www.arduino.cc/, last accessed 23. Mar, 2012

[9] http://processors.wiki.ti.com/index.php/MSP430_LaunchPad_(MSP-EXP430G2)

[10] http://mbed.org/compiler, last accessed 30. Mar, 2012

Prototyping boards

There are different types of prototyping boards from the different prototyping board producers. Below follows some of the prototyping board which is considered for this thesis.

PIC24E USB Starter kit

PIC24E USB is a preassembled prototyping board based on the PIC24EP512GU810 MCU. The board is produced by Microchip which makes the PIC MCUs and is designed to prototype USB host development using the PIC24E MCU family line. PIC24E contains an on-board programming/debugger, standard A USB and micro A/B connectors, three user-programmable LEDs and three push button switches.

Arduino Mega 2560

The Arduino Mega 2560 shown in figure 2.4 is a prototyping MCU board based on the Atmel AVR ATmega2560[11]. The prototyping board has 54 digital I/O pins of which 14 are PWM, 16 analog inputs, USB connection, 16MHz crystal oscillator, power jack, 4 UART hardware serial ports, an In Circuit Serial Programmer (ICSP) header and a reset button. The board contains everything that is needed to plug it into a computer and start developing. The Arduino prototyping boards come with a wide range of prototyping shields and libraries to take advantage of the shields. Some shields support XBee, Ethernet, Motor control of DC motors and so on.

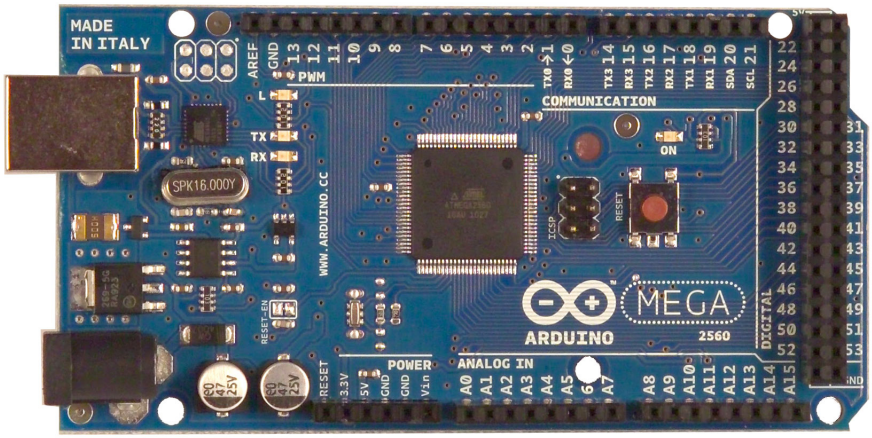

Figure 2.4: Arduino Mega 2560

MSP430 Launchpad

The MSP430 Launchpad is a preassembled prototyping board based on TI MSP430 MCU. The board includes a built-in flash emulator which connects to a development machine for in-system programming and debugging via the USB cable. It contains a Dual in-line package (DIP) socket capable of supporting up to 20 pins on any MSP430 value line. The board also has 2 programmable Light-emitting Diode (LED)s, 1 power LED and 2 programmable buttons.

mbed NXP LPC1768

The mbed NXP LPC1768 is a preassembled prototyping board based on ARM Cortex M3 which is suited for prototyping application for Ethernet, USB, and lots of peripheral interfaces and flash memory. In Figure 2.5 the commonly used interfaces and their locations the pins labled p5-p30 can be used as Digital I/O pins.

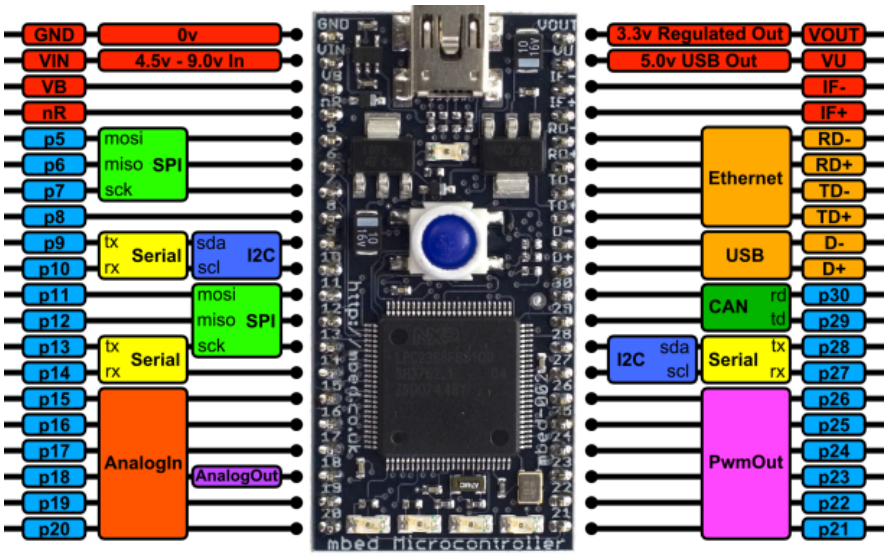

Figure 2.5: mbed NXP LPC1768

The evaluation of prototyping boards lead to the discovery of a very good community around the Arduino. The Arduino's standard form factor enables it to interface with a lot of prototyping shields and therefore seems like a good choice for the prototype. The next section will look at the USB host capabilities of the Arduino.

2.3.2 USB Host capabilities

Standard USB uses the master/slave architecture where the host acts as the protocol master and a USB device connected acts as the slave. In order to communicate between the mobile phone and the MCU the MCU has to be equipped with a USB host capabilities

so it can control the signals from the sensors and decide when to send them to the mobile phone. Since the USB connection located on the prototyping boards only has USB slave capabilities the USB host capabilities need to be added.

USB Host Shield

One solution for the MCUs that support SPI is the USB host shield shown in Figure 2.6 which gives the MCUs the capability of acting as a USB host thus making the MCU able to schedule the configuration and data transfer over the link. The USB host shield in Figure 2.6 is available as an integrated part of the Arduino Mega 2560 ADK version which is based on the Arduino Mega 2560 board.

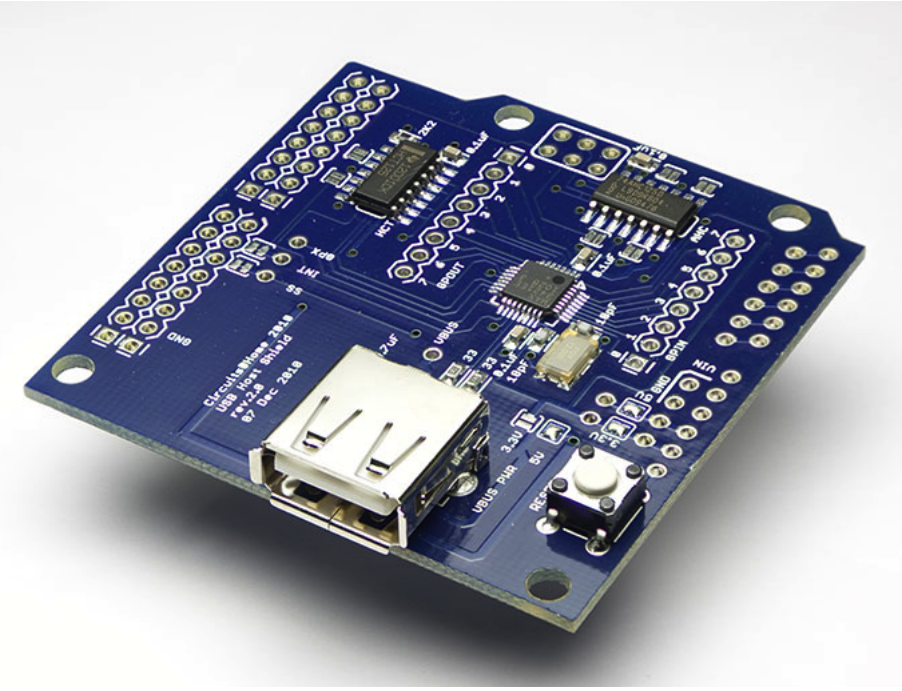

Figure 2.6: USB Host shield

The USB host shield in Figure 2.6 interface with the Arduino form factor and is selected for the prototype.

2.3.3 Sensors

FSR's are sensors that detect physical pressure and wight. FSR's are resistors and thus they are simple and easy to use. FSR's changes its resistive value in ohms Ω depending on the pressure given to the sensor. FSR's are not particularly good at measuring wight since every sensor vary in sensitivity of about 10%, however measuring the center of pressure is more suited for its capabilities. The FSR's resistance is infinite when no pressure is applied, the resistance goes down when pressure is applied.

Interlink FSR

The force sensors used in this project is called 'Interlink 402' and is shown in Figure 2.7.

Figure 2.7: Interlink 402

Interlink 402 properties:

Size
The diameter is 12,5 mm and it is 1mm thick.

Resistance range
Resistance range from infinite (no pressure) to 100 KΩ to 200 Ω (max pressure).

Force range
Force range from 0 to 100 N applied evenly over the active area.

Power supply
Interlink 402 uses less than 1 mA of current but it depends on supply voltage[11].

Tekscan FlexiForce

The FlexiForce sensors from Tekscan shown in Figure 2.8 is almost the same type of sensors as the FSR sensors. Some of the FlexiForce A401 properties is:

Size
The sensing area is 12.5 mm and they are 0.203mm thick.

Resistance range
Sensor resistance at no load is 5MΩ, referance resistance is 1kΩ to 100kΩ.

Force range
The force range goes from 0 to 110 N but can be adjusted through drive voltage.

[11]http://www.ladyada.net/learn/sensors/fsr.html, last accessed 27. Apr 2012

Power supply

Power supply should be constant. Max recommended current is 2.5mA.

Figure 2.8: FlexiForce sensor

The Interlink 402 sensors is selected because of its availability and the two sensors evaluated here seem to be close to each other in performance. However it would be preferable for further development of the prototype to test the FlexiForce sensors.

2.3.4 Mobile phone

This section evaluate mobile phones which can be used to utilize custom accessory.

iPhone

The iPhone has the ability to connect to MCUs via Redpark Breakout Pack for Arduino and iOS[12]. Without investigating the subject in full detail, there is a belief it would require a jailbreak of the iPhone if the need for a full background network application were needed, which is doable but not preferred.

Android

The Android powered mobile phones have a great advantage when it comes to system openness. The ability to force a service/application to run fully in the background is a great advantage for a remote controlled application. The Android phones that run Android version above 2.3.4 can also utilize the ADK which is made for interfacing between

[12]http://blog.makezine.com/2011/07/18/59-cable-lets-you-connect-iphone-to-arduino-no-jailbreaking/, last accessed 23. Apr 2012

the mobile Operating System (OS) and a MCU. A Google developed phone would be preferred since they have the ability to be updated with the latest software libraries for development by Google.

Samsung Nexus S The Nexus S is co-developed by Google and Samsung which makes the phone one of the Google phones which receives Android system updates as soon as they are ready contrary to other. The phone is capable of fast computation with its 512 MB memory and 1 GHz processor.

Samsung Galaxy Nexus The Galaxy Nexus is the newest, by time of writing, co-developed Samsung and Google android mobile phone which is bigger and heavier than the Nexus S although it is faster with its 1.2 GHz Dual-core and 1 GB memory.

The selected phone for the prototype are the Nexus S. This phone is a Google phone which means it can receive OS updates that can be important for development because of the ADK. It weighs 129 g and it is preferable with as little weight as possible. And lastly the phone has 512 MB memory and 1 GHz processor which makes it a quite powerful mobile device.

2.4 Middelware

This section evaluate the middleware layer where programming languages and libraries is used to interface the software of the different hardware components. This section lays the foundation for the application domain in the next section.

2.4.1 Programming environment

The programming environments, languages and libraries for developing the evaluated hardware components mentioned in this section is some of the key components for developing the prototype.

Java

Java is a rich programming language developed by Sun Microsystems and released in 1995. According to langpop.com[13], Java is the second most popular language in the world next after C. The language derives its syntax from C and C++ but does not offer the same low-level functionality and has a simpler object model which makes development easier. The language is compiled to bytecode and run on Java Virtual Machine (JVM). Since the language runs on a virtual machine the language is highly portable and can run on most systems including Windows, Mac OS, Linux and mobile phones without the need to recompile.

Java is the language which is used to develop Android applications and is the basis of the Android Software Development Kit (SDK). The Android OS for mobile phones

[13]http://www.langpop.com/, last accessed 23. Mar, 2012

use an alternative version of the JVM, Dalvik. Java applications are converted into .dex Compact Dalvik Executable (CDE) format, which is suitable for systems with low memory and processing power, before it is executed on the Dalvik VM.

C/C++

The C++ programming language is an extension of the C programming language. C++ adds classes to C and makes the language object oriented among other features. The Arduino development environment use C as programming language and imports various libraries written in C++ including the Wiring library, AndroidAccessory library and the USB_Host_Shield library.

The Wiring library sets up the main function into the Arduino sketches developed in the Arduino IDE. The simple main function calls two functions which is the functions one writes code that will be executed on the MCU. The simple example in code listing 2.1 shows how code written in C turn the onboard Arduino LED on and off at an interval of one second.

Listing 2.1: Simple Arduino sketch in C

```
int ledPin = 13;

void setup() {
        pinMode(ledPin, OUTPUT);
}

void loop() {
        digitalWrite(ledPin, HIGH);
        delay(1000);
        digitalWrite(ledPin, LOW);
        delay(1000);
}
```

The AndroidAccessory library use the USB_Host_Shield library to connect, communicate and check connection to the android device.

Wiring IDE

The Wiring development environment is an open source prototyping platform composed of a programming language and an IDE[14]. It is based upon the Processing IDE[15]. The Wiring development platform is the basis of which the Arduino IDE derives from.

Arduino IDE

The Arduino IDE is a cross-platform development environment written in Java. Arduino IDE is based upon the Wiring development platform and is customized to program the

[14]http://wiring.org.co/, last accessed 23. Mar, 2012
[15]http://www.processing.org/, last accessed 23. Mar, 2012

Arduino line of MCUs. Arduino hardware is programmed using the Wiring C/C++ syntax and libraries and a Processing based IDE shown in Figure 2.9.

Figure 2.9: Arduino IDE

The IDE is capable of compiling and uploading programs to Arduino hardware with a single click. This means that there are no need to edit a Makefile or program the ATmega MCU chip. The Arduino board comes with an Atmel ATmega MCU with a preinstalled bootloader which jumps to the location of the program uploaded by Arduino IDE on start. Every time new code is uploaded to the Arduino the board is reset thus running the bootloader which jumps to the new program in memory.

SQLite

SQLite database is integrated into the Android OS and is used for application data storage.

SQLite is a self-contained, server less, zero-configuration, transactional, SQL database library and is imported into the process that needs its functionalities. The codebase for SQLite is free software and is developed by an international team [16].

SQLite is a lightweight embedded SQL database and unlike most databases it has no standalone server process. SQLite reads and writes directly to a single disk-file. The SQLite's file-format is cross-platform meaning it can be used between 32-bit and 64-bit systems, and between big-endian and little-endian systems. Since SQLite does not have an external server process it is best to think of SQLite as a replacement for *fopen()* and not Oracle, MySQL, etc ...

SQLite is a compact library with a size of only 350KiB, with optional features omitted the size can be reduced to 200KiB. This property makes SQLite a very popular choice on memory constrained units like cellphones, PDAs and MP3 players. SQLite also has good performance in low-memory environments ...

SQLite is very carefully tested[17], with automated test suites and millions and millions of test cases. The SQLite library is therefore very reliable.

This section evaluated the programming environments, languages and libraries needed to develop the selected hardware components. Further evaluation will show what is selected for development and future development might introduce other tools for development.

2.4.2 Communication

This section will evaluate various communication standards used in the prototype system.

The AndroidAccessory Protocol (AAP) is used between the Arduino MCU and the Android phone allowing data to be transferred from the force sensors to the mobile phone. The mobile phone will either communicate with 2G (EDGE), 3G (UMTS)) or WLAN to an external computer client through the internet. This will require a protocol for message and file exchange between the mobile phone and a computer. This is required to start and stop the recording of data collection and data exchange.

USB

The communication between the MCU and the Mobile phone use a USB connection which the AAP use to exchange data between the MCU and phone. With the help of the USB host shield the MCU is the USB host and the Android phone is the USB slave.

[16] http://www.sqlite.org/crew.html, last accessed 23. Mar, 2012
[17] http://www.sqlite.org/testing.html, last accessed 23. Mar, 2012

2G (EDGE), 3G (UMTS), Wi-Fi

The technologies that the Android phone will use to connect to the internet is either 2G (EDGE), 3G (UMTS) or Wi-Fi depending on the availability. This is chosen automatically by the OS on the phone so there is no need to take this into account when designing the software for the system. The 3G coverage is pretty good and this is probably the technology that will be most used. One might want to test on Wi-Fi when developing locally but when testing the system out in the field the 3G and 2G will be used. 2G might be sufficient for the data amounts in the project to be sent over the network.

Conditions in crowded areas like arenas with many people might provide scarce network capabilities. The prototype need to have the capability of store & forward in order to function properly on 2G networks with low data rate capabilities.

2.5 Application domain

This section evaluates the SDK and ADK that is used for developing mobile applications needed for the prototype.

2.5.1 Mobile phone OS

Android is an OS for mobile devices such as smartphones and tablet computers. The company Android Inc. which was the initial developer of Android was purchased by Google in 2005 and the OS was unveiled 5 november 2007. Android consist of an OS kernel based on the multi-user Linux kernel. The middleware and libraries are written in C and applications backed by java-compatible libraries based on Apache Harmony. Each process runs on its own virtual machine, so an application is run in isolation from each other.

Android SDK

The Android SDK provides tools and APIs necessary to develop applications on the Android platform using the Java programming language[12].

The Android applications consist of 4 components:

Activities

Activities represent a single screen with a user interface. For example a Twitter application could have one activity to show all the new tweets, another activity to compose a tweet and another activity for replies. These activities is working together to form an application and each of the activities can be accessed from other applications. For example could the camera application get access to the Twitter compose activity to tweet about a photo.

Services

A service is a component that runs in the background and does not offer any user-interface. Services can perform long running operations or perform work for remote processes. For example a service can download content from the internet without blocking an activities user-interface or it might do work in the background while the user is in another application.

Content providers

A content provider handles the application data. Whether it is stored in the filesystem, SQLite database, on the internet or any other persistence technology. If an application permission allows it the application can read or write to the content provider. For example an application can be granted access to the the address book content provider for creating, reading, updating or deleting contacts. Content providers can also be private for the applications and not shared with any other applications.

Broadcast receivers

A broadcast receiver is responsible to respond to system-wide broadcast announcements. A system-wide broadcast can be that the screen has been turned off. Applications can also initiate broadcast and make applications aware of services or data being available for them to use. Broadcast receivers can display status bar notifications to alert users that a broadcast has occurred, however broadcast receivers are designed to do minimal work and instead instantiate services that does some work based on the event.

The Android SDK provide a wide range of functionality, the prototype utilize the Activity component of the Android SDK but further development might show the need for developing a service from the Services component.

Mobile Phone ADK

The Android ADK allows external USB hardware to interact with an Android-powered device in a special 'accessory' mode. This requires the connected accessory to be an USB-host, meaning it enumerates devices and powers the bus, and the Android phone acts as the USB-slave device. Android accessory is specifically design to support Android devices and use the simple Android accessory protocol which allows them to detect Android devices which supports accessory mode.

The main Android accessory hardware and software components:

USB MCU and USB host shield

An ADK board consist of an Arduino Mega 2560[11] based MCU combined with a Circuits@Home USB Host shield[13]. The ADK board can be equipped with *shields* which use the boards input and output pins. The board is controlled by running custom firmware written in C++ which defines the functionality and interaction with an Android device.

Android Demo Shield

The Arduino Demo Shield in figure 2.10 is a shield which comes with the original Android ADK. The ADK comes with a demo application which implements the use of the input and output components on the demo shield. These components include a joystick, LED outputs and temperature and light sensors. The components and design of hardware components is fully customizable and programmable.

USB Host Shield library

A library based on Circuits@Home USB Host shield library makes the ADK act as a USB-host. This allows the board to initiate transactions with USB devices.

Arduino sketch

An Arduino sketch defines the firmware that is run on the ADK and is written in C++. The firmware is programmed to interact with an Android device through the Android accessory library. The firmware is also set up to send data from the ADK board to the Android application and receive data from the Android application and outputs it to the ADK board and attachements.

Android accessory protocol

The Android accessory protocol is defined in the Android accessory library. It defines how to find connected accessory supported Android device, how to enumerate the bus and how to set up and communicate with the device.

Other libraries

The ADK boards also support other third-party libraries:

- CapSense library - capacitive sensor (human touch) library[18]
- I2C / TWI (Two-Wire Interface) library - Multi-master serial single-ended computer bus
- Servo library - Controlling servo motors
- SPI library - Synchronous serial data link

[18]http://www.arduino.cc/playground/Main/CapSense, last accessed 27. Apr, 2012

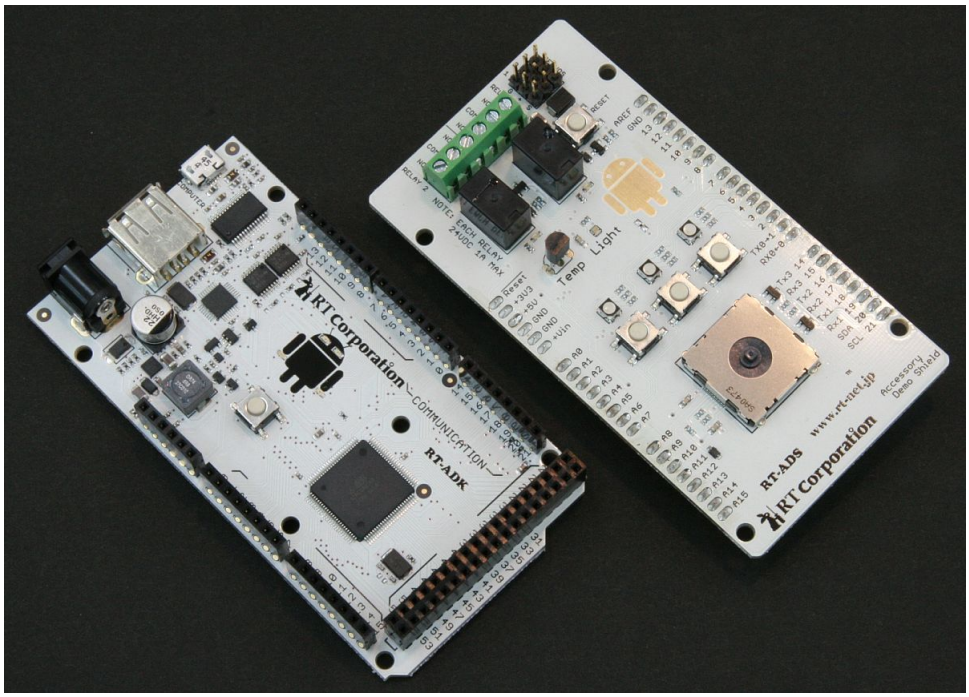

Figure 2.10: Android ADK Demo Shield

2.5.2 Client/Server/Peer-to-Peer

A fixed server is needed to address the server application running on a Personal Computer (PC). This server will be a point in the network topology in which the server application register its Internet Protocol (IP) address and client mobile connections lookup the server application IP. The server application will be separate of the server registration, the registration is needed to overcome Network Address Translation (NAT) translation within the mobile phone internet.

The drawback with having this registration server is that this introduces a single point of failure[14] for the system. If the registration server is down the whole system goes down. But the nodes in the network need to find each other in order to communicate and this requires the NAT of the mobile phone network to be bypassed.

2.6 Summary

This chapter evaluated the technology involved in order to make a mobile human movement data collection unit prototype.

The components selected for a prototype assembly is the Nexus S mobile phone, Arduino Mega 2560 MCU prototyping board, Circuits@Home USB host shield and the Interlink 402 FSRs. In addition to this there will be server application controlling the system.

The system assembly and detailed specification of the main blocks, as well as outline of the three main interfaces will be further evaluated in the next chapter.

Chapter 3

Functional architecture

This chapter revisit the use-case scenarios for the main challenges of this project. Then comes an overview of capabilities of the main blocks as well as a block description of the individual blocks. The chapter then moves on to a detailed specification of the main block which outlines the limitations of the prototype before the chapter is ended with a prototype system overview. Based on a preliminary evaluation of technologies this chapter will discuss the functional architecture of a prototype for a sensor system that measures an athletes performance and record the data. Further this chapter will look at solutions to different scenarios that the prototype might solve.

The goal of this chapter is to outline the capabilities of the prototype and afterwards be able to take one block out of the system and replace it with an upgraded block. When the MCU-block needs to be replaced it must be clear what kind of interface that is defined for the block to the rests of the system. In order to accomplish this, this chapter must define what the main interfaces between the individual blocks are.

Eventually further development depends on the ability to change blocks in the prototype model. The ability to make a block in the block diagram wireless is a goal and further making the system light enough to be used in competitions is desired.

3.1 Intro and use-case specification

In this section the main challenges for the selected concept are revisited. There are several benefits and issues regarding athletes and sensor data collection. Some of the challenges that these kinds of systems face today are simplicity, weight, cost and mobility.

The system would have to be simple enough for an athlete to use and the technological aspect would have to be fully transparent in order for an athlete to fully concentrate on performance and training.

As mentioned earlier the size and weight of the system would have to be low in order for an athlete to fully function as he would under normal circumstances. The smallest difference could make an athlete, who is dependent on good balance, not perform 100%.

The cost of the system monitoring the athletes has to be low in order for the teams of athletes to afford them. Athletes and teams are relying on sponsors and have very limited resources. The development of lighter, faster, smaller hardware devices will eventually make it possible for incorporating technology into athletes equipment thus bringing their sport forward.

In order for the athletes to carry the equipment with them the system they use will have to be mobile. The mobility of the equipment is dependent on the size and weight of the hardware which will affect the ability to keep the technology transparent to the athletes. The mobility of the system is crucial for the athlete in order to perform as good with the system as without the system.

A sensor system for monitoring an athlete would make a teams capable of collecting a lot of relevant information about an athletes current level as well as collecting data which could be used for comparison over time to show the athletes progress. The requirements for such a system would have to be low cost, simplicity and ease of use, light weight and small size that would make the system mobile. If a system with these requirements could be compiled, then the following scenarios is where it could be put to use.

How can the trainer know how the athlete use its resources?
By collecting sensor data over a period of time and comparing these data results against each other he trainer could get an indication on how the athletes use its resources. This could lead to further development of the athletes ability to perform because it would tell them when they waste energy during their performance.

How can the team develop better average results?
By measuring the best athletes average data variables like power distribution and center of pressure one could begin to build a better understanding of how the best athletes distribute their resources. These results can help the individual athletes to gain an understanding of how to better their performance thus raising the average results of the team.

How can the trainer know how the best athlete distributes its powers?
By placing FSR on an athlete one can make data recordings of power output from an athlete. These datas can be utilized for understanding how the best athletes distributes its powers as well as gaining an understanding of how athletes with poorer results waste their powers. In sports like cycling or cross-country skiing the distribution of power is crucial for the athletes to perform their best.

How can the trainer know which power distributions are most effective?
By measuring the power distribution of the athletes in a team, the trainer could compare the individual athletes results and performance with the data and effectively know which athlete that was distributing its power the most effective thus bringing this information to the team.

How can the trainer get the athletes center of pressure?
By placing a minimum of four force sensitive resistors in an athletes shoe soles. Two

for each foot, one at the front of the foot and one in the back of the foot. This would make it possible to measure the athletes center of pressure and how the balance is distributed throughout the coarse and performance. In a sport like ski-jumping measuring the center of pressure through the in-run to the table and the athletes push against the table is really interesting and would make it easier for the trainer to know how the individual athlete balance.

These use-case scenarios are the foundation of the design of the prototype which is outlined in detail in the rest of this chapter.

3.2 Overview of capabilities & block description

The block specification selected for the prototype is based upon the preliminary technology evaluation performed in the previous chapter and is described in Figure 3.1.

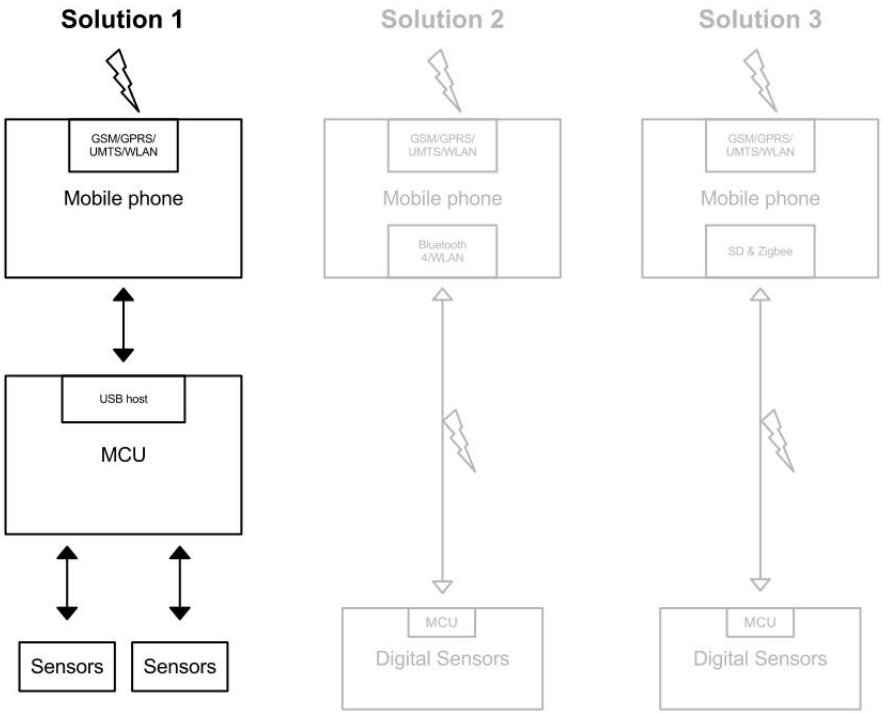

Figure 3.1: Selected block design

The block specification has one advantage over the two other alternatives and that is the separation of MCU and sensors. This makes the model more flexible in terms of block replacement. Although the model is easier and faster to develop because of the block specification the model may seam a bit clumsier then the other models because of the use of wires to the sensors. This could get in the way of the athletes ability to perform as

intended. How much impact this will have on performance is still unclear at this stage of development. However this is taken into consideration into the further evaluation and development of the prototype.

3.2.1 The four main blocks

Block specification in Figure 3.1 show 3 main blocks and one wireless connection to the outside world. The prototype project name is Taunus and in Figure 3.2 the outside world of the Taunus system is shown and the client/server block is visible. Figure 3.2 shows how a client controller will be controlling the smartphone and enable/disable data collection and control the data transfers.

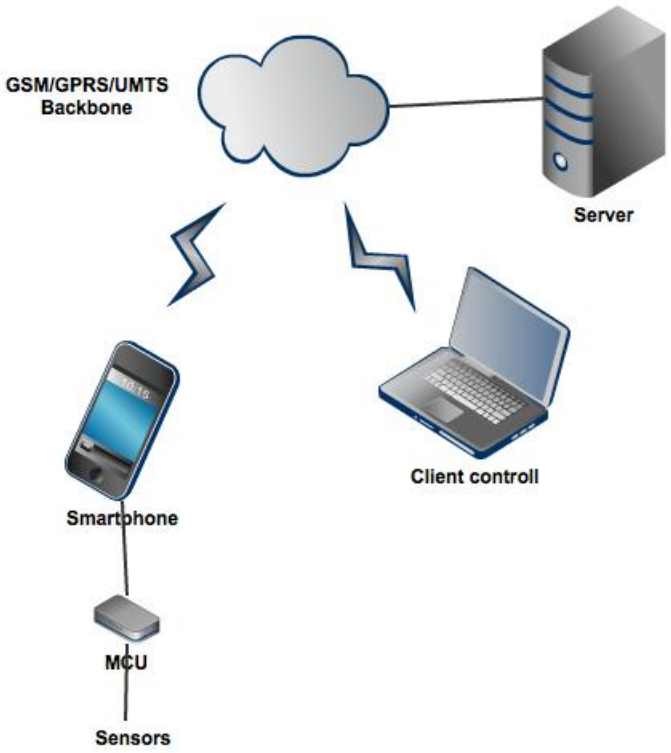

Figure 3.2: Prototype (Taunus) system block topology

Sensors The sensor block of Taunus consist of four Interlink 402 sensors, two for each foot. These are placed in front of each other which enables the reading the center of pressure of an athlete. The sensor block has one interface to the MCU.

MCU The microcontroller block chosen for the Taunus system consist of an Arduino 2560 Mega equipped with an USB-host shield which enables the Arduino to act as an USB-host device ideal for the ADK framework. The MCU has two interfaces, one against the sensors and one against the mobile phone.

Mobile phone The mobile phone chosen for the Taunus system is the Samsung Nexus S. The ADK reqires Android version 2.3.4 and higher which the Nexus S meets. The phone has two interfaces one against the MCU and one against the internet.

Client/Server The client controller runs a server program which enables client Taunus phones to connect. The client controller is a Java program capable of running on Linux, Mac OS or Windows PC. The client controller has one interface against the mobile phone.

Sensor

As mentioned previously the Interlink 402 sensors is the force sensors used in the Taunus system. The sensors are chosen because of their size, weight and usability.

Figure 3.3: Selected sensor, Interlink 402

The sensors are 1 mm thick which makes them ideal to put inside shoe soles or in other flat peripheral equipment and their diameter is 12,5 mm so the placement area can be adjusted after tests show that their positioning might be wrong.

The sensor weigh 0,26 g, an acceptable weight since the Taunus system has to be as light as possible in order to be as transparent as possible to the athlete and not interfere with the athletes normal routines.

The force range of the sensors are 100 N which correlates to about 10,2 kg. First assumptions indicate that this is not within the required force range, which is in the force range or weight of a human being or about 100 kg. However further analysis will conduct a force per square inch calculation which might indicate that the area under the foot where the sensors are placed receive less than 10,2 kg of force.

Since the FSRs, in their basic form, are resistors with changing resistance the sensors are simple to use and all that is required to measure the force is an ADC which is built in to most MCUs these days. Desired voltage is supplied to the FSR and the resistance

in the FSR change the voltage let through the resistor which is measurable at desired amounts of samplings a second.

MCU

As mentioned earlier the MCU selected for the Taunus system is the Arduino 2560 Mega. The board is a preassembled prototyping board with the MCU circuit preassembled and ready for development. This board is chosen because of its interfacing support to ICs, community support and ease of programmability.

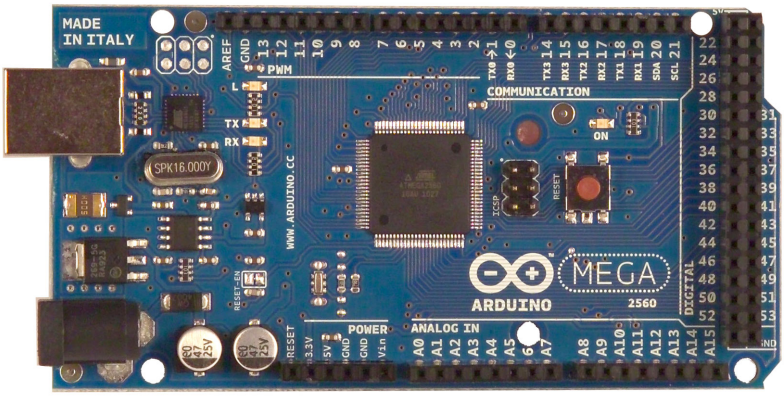

Figure 3.4: Selected MCU prototyping board, Arduino Mega 2560

The board is equipped with an AVR ATmega2560 MCU which is preprogrammed to receive a written program for the MCU without using external hardware. This makes development fast and it is very easy to try out different approaches and readjust the code, and it operates on 5V which is very common and is therefor simple to integrate with a lot of existing sensors and integrated circuits which supports the same voltage.

The board board is 53 mm x 110 mm which is fine for a prototype but the electronics needed for the prototype can be fitted to a small area if Surface Mounted Technology (SMT) is used. Although the MCU is a Surface Mounted Device (SMD) the rest of the board is really big and future development would utilize SMT for the best integration of components. In Figure 3.4 one can see a silver oval component marked SPK16.000Y, above this component is a small black squared IC. The black IC is an AVR ATmega8U2 SMD MCU which is used for USB to serial conversion and is an example of how small a MCU could be in future development.

The weight of the board is 40 g which is acceptable for a prototype since that is not sufficient to be too distracting or interfere with an athletes normal proceedings. The main weight problem in the prototype lies in the telephone and it will be the main weight issue for this system if not lighter mobile phones is invented.

The board can interface with the SPI which many ICs use, one such IC is the MAX3421E USB host controller. The IC is included on the Circuit@Home[1] USB-host shield project which has the form factor of the Arduino. This makes it very easy to interface the MCU as an USB-host shield device thus making it able to take control over the USB and interface with the mobile phone.

Mobile phone

As mentioned earlier the mobile phone selected for the prototype is the Samsung Nexus S. The phone is selected because of its size, processing speed, interfacing capabilities and android adk capabilities.

Figure 3.5: Selected mobile phone, Samsung Nexus S

Because of its 129 g and 63 mm x 123.9 mm x 10.88 mm the Nexus S is a good alternative for a system thats moving around a lot.

With a processing power of 1 GHz the mobile phone has more than enough processing power to perform data collection from a MCU.

The phone is equipped with USB micro-B which is the hardware interfacing to the MCU.

In order to use the ADK functionality the phone needs to be able to run Android version 2.3.4 and upwards which the Nexus S does.

[1]http://www.circuitsathome.com/arduino_usb_host_shield_projects, last accessed 13. Apr, 2012

Server and Client control

The server is in control over the client phone and sends commands to the phone. The commands tell the phone to either record data and transmit or stream data directly and so forth.

The server application for the prototype will be an application written in Java because of the ease of development and cross-platform support. Testing the software can be carried out on Linux, Mac OS or Windows, an internet connection is what is needed.

In the next section the interface of the server application is described, there an outline of the client/server protocol is described.

3.2.2 The three main interfaces

In order to be able to upgrade a block in the block diagram, interfaces between the blocks has to be defined. There are four main blocks and between them, three interfaces has to be designed. The interfaces is either hardware Input/Output (I/O) defined or software protocol defined.

MCU and sensor interface

The selected sensors connects to their individual input ports on the prototyping board. Their drive voltage output is what can be recorded by the MCU and converted into a digital signal using an ADC.

The ADC are a built in feature into the ATmega2560 MCU. In Figure 3.4 the ports marked with a capital A in front of them are configured to receive analog input. The ports A8, A9, A10 and A11 will be used to read the prototype sensor data values, four ports for the four sensors.

Listing 3.1: Sample from Arduino library analogRead reads analog data from pin number

```
int analogRead(uint8_t pin) {
        uint8_t low, high;

#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
        if (pin >= 54) pin -= 54; // allow for channel or pin numbers
#else
        if (pin >= 14) pin -= 14; // allow for channel or pin numbers
#endif

#if defined(ADCSRB) && defined(MUX5)
        // the MUX5 bit of ADCSRB selects whether we're reading from
            channels
        // 0 to 7 (MUX5 low) or 8 to 15 (MUX5 high).
        ADCSRB = (ADCSRB & ~(1 << MUX5)) | (((pin >> 3) & 0x01) << MUX5);
#endif

        // set the analog reference (high two bits of ADMUX) and select the
        // channel (low 4 bits).  this also sets ADLAR (left-adjust result)
        // to 0 (the default).
```

```
#if defined(ADMUX)
        ADMUX = (analog_reference << 6) | (pin & 0x07);
#endif

        // without a delay, we seem to read from the wrong channel
        //delay(1);

#if defined(ADCSRA) && defined(ADCL)
        // start the conversion
        sbi(ADCSRA, ADSC);

        // ADSC is cleared when the conversion finishes
        while (bit_is_set(ADCSRA, ADSC));

        // we have to read ADCL first; doing so locks both ADCL
        // and ADCH until ADCH is read.  reading ADCL second would
        // cause the results of each conversion to be discarded,
        // as ADCL and ADCH would be locked when it completed.
        low  = ADCL;
        high = ADCH;
#else
        // we dont have an ADC, return 0
        low  = 0;
        high = 0;
#endif

        // combine the two bytes
        return (high << 8) | low;
}
```

The sample code in Listing 3.2 show how the analogRead function is implemented in the Arduino/Wiring library. This is done by setting the ADCSRB register with values from MUX5 registers bit values to which port/channel to listen to and ADMUX register with port number bit values. Then the conversion is started with the *sbi()* function call and ADCL register is read before ADCH register which collects a 10-bit value reading containing the converted digital value.

By using the Wiring/Arduino global function call analogRead on the ports A8-A11 the analog to digital conversion is described above and the digital values are returned as int's.

Mobile phone and MCU interface

The interface between the mobile phone and the MCU is on the physical side standard USB with Standard A USB (MCU-Host shield interface) to Micro-B USB as shown in Figure 3.6.

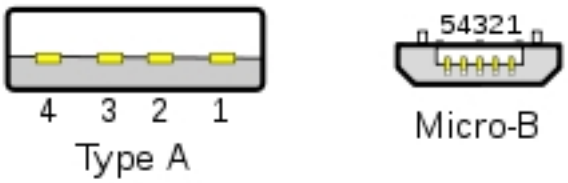

Figure 3.6: Mobile phone to MCU USB physical interface

The USB-host capabilities of the MCU is what makes it possible to communicate via the AAP. The USB-host accessory controls the bus and must adhere to the AAP, which defines how an accessory detects and sets up communication with an Android-powered device. In general, an accessory should carry out[2]:

1. Wait for and detect connected devices
2. Determine the devices accessory mode support
3. Attempt to start the device in accessory mode if needed
4. Establish communication with the device if it support the AAP

Wait for and detect connected devices
The Android accessory should continuously check for connected Android-powered devices and when connected determine if device supports accessory mode.

Determine the devices accessory mode support
The accessory should check the product IDs of the product it tries to connect to. The ID should match Google's ID (0x18D1) and product ID should be 0x2D00 or 0x2D01.

Attempt to start the device in accessory mode if needed

1. Send 51 control request ("Get Protocol") to figure out if the device supports Android accessory mode.
2. If the device returns a proper protocol version, send identifying string information to the device.
3. When the identifying string are sent, request the device start up in accessory mode.

[2]http://developer.android.com/guide/topics/usb/adk.html, last accessed 17. Apr, 2012

Establish communication with the device if it support the AAP

If the accessory mode is detected, the accessory can query the device's interface descriptors to obtain the bulk endpoints to communicate with the device.

In the next chapter the detailed information on how to implement the AAP into the MCU firmware is described.

Mobile phone and server interface

The mobile phone and server interface consist of 2 interfaces. The mobile phone communications interfaces and the software protocol interface. The mobile phone communications interfaces are the GPRS, UMTS and WLAN which all can be utilized. The software protocol is sent on top of these technologies and is defined below.

The command protocol list:

Number	Command	Description
101	Start	The start command is used in combination with other commands to indicate that the indicated functionality should be started.
102	Stop	The stop command is used in combination with other commands to indicate that the indicated functionality should be stopped.
201	Record	The record command is used in combination with the start or stop command to indicate start or stop of the recording of sensor data.
202	Stream	The stream command is used in combination with the start or stop command to indicate start or stop of the streaming of sensor data.
203	Send	The send command is used in combination with the start or stop command to indicate start or stop of sending recorded sensor data.
301	Connection	The connection command is used in combination with the start stop command as a hello message with the server application to indicate a new connection or to stop the connection.
302	Ping	The ping command is used for checking connection, if clients is still connected.
303	Pong	The pong command is used for acknowledging the ping command to indicate that the connection is still up.
401	Sensor data	The sensor data command indicate an incoming sensor data value.

Examples of use:

Example	Description	Summary
101:301	Start connection	The 101:301 command indicate a hello command from the client.
101:201	Start record	The 101:201 command indicate that the mobile phone shall start record any receiving sensor data.
102:201	Stop record	The 102:201 command indicate that the mobile phone shall stop record any receiving sensor data.
101:203:401	Start sending sensor data	The 101:203:401 command indicates that the server should get ready to receive recorded sensor data.
401:001:516	Sensor data	The 401:001:516 command indicate that a sensor value from sensor 1 is received with a sensor value of 516.
102:203:401	Stop sending sensor data	The 102:203:401 command indicates that the server should stop receiving recorded sensor data.
102:301	Stop connection and disconnect	The 102:301 command indicates the the server should disconnect the client connection.

The client application sends *101:301* hello server to indicate a new connection with the server. The server application send the command *101:201* over the internet connection to indicate that the mobile phone should start recording the sensor data, and send the *102:201* command to indicate that the phone should stop record. The server receives the sensor data in the format of *type:sensor:value* from the phone and displays the data in a line chart. Before data is sent to the server from the client phone the *101:203:401 start:sending:sensordata* command is sent to the server. This enables the server to receive recorded data. The *102:203:401 stop:sending:sensordata* disables the servers receiving mode and data is displayed on the server. Lastly the client shutdown the connection by sending *102:301* stop connection command message.

The next section conducts a detailed specification of the main blocks.

3.3 Detailed specification of main blocks

This section contains a detailed specification of the main blocks. The prototype will be an application for the ski jumping hill where the application needs to collect data in a certain amount of time in high velocity. The main goal of this section will be to establish the required boundaries for the selected blocks. The ski jumping hill can be divided into

segments where different things happen in the separate segments. This section will then outline what the blocks are able to do in each segment.

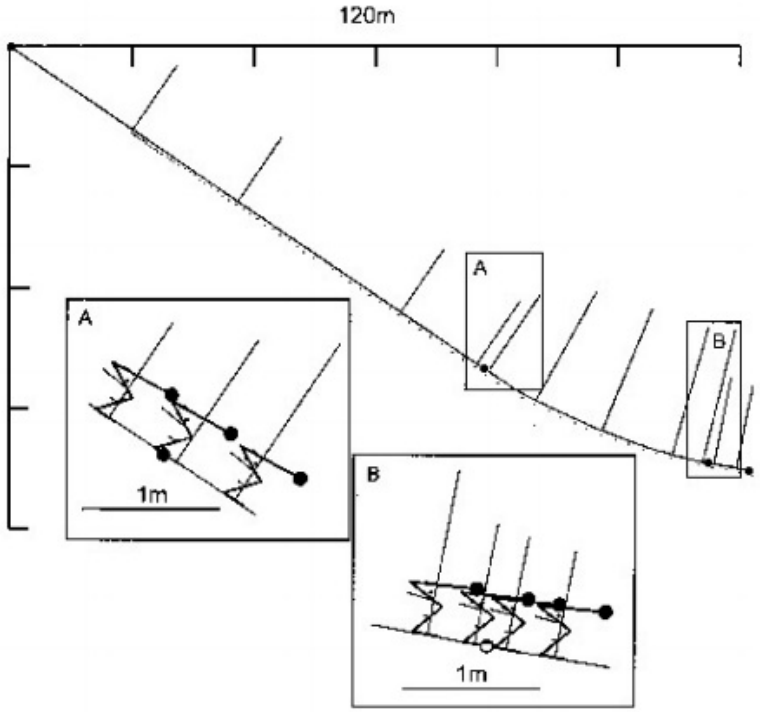

Figure 3.7: Figure over the ski jump in-run segments

In Figure 3.7 the ski jump in-run is divided into tree segments. Segment one is from the start position and a straight of 95 meters, segment two is the 18,35 meters curvature from the straight to the jumping table and the third and final segment is the take-off table of 6,65 meters, 120 meters in total. Depending on the drag and resistance a ski jumper will cross these three segments in 5-6 seconds. Segment one will require less samples per second because of lower speed contra the take-off table where the speed of the ski jumper is approximately 93,5 km/h.

The ski jumper parameters used will be an average ski jumper of 60 kg mass and height 1.75m with an average foot area of $63{,}5cm^2$.

In order to conduct a detailed max performance analysis for the MCU and sensors of the in-run, the sample rate and resolution needs to be considered in order to outline the prototype systems limitations.

The sample rate is important in terms of wanted number of samplings. If the athlete use 5,41 seconds in the whole in-run, that corresponds to 5410ms. And if the wanted number of samplings or required number of samplings is 1 per ms or 5410 samples, then 4000 samplings does not meet the sampling rate requirement.

The resolution is important when the degree of measurement is in focus. If the athlete is skiing down the in-run with shifting balance and weight on the sensors and the voltage variation in the sensors shift 0.1V per 100g. If the system is required to measure voltage variations of 0.1V but is only capable of reading voltage variations of 0.5V, then the system does not meet the resolution requirement.

A more detailed analysis of the sampling rate and resolution is conducted in the subsequent sections.

3.3.1 Sensors

The athletes equipment can be mounted with several types of sensors, and in the case of measuring the center of pressure, from four similar FSR to any number of the same sensor. Four sensors is minimal to measure the center of pressure, two for each foot, one in front of the center of pressure and one behind the center of pressure of each foot. The sensors is applied with equal amount of voltage all the time through the in-run. The number of samplings per second is up to the MCU to decide but the sensors will be able to give readings throughout the whole in-run whether it would be preferable to sample every 0.1ms, 0.01ms or 0.001ms.

Figure 3.8 shows how resistance of the FSR change when force is applied to the sensors. The sensors is an infinite resistor when no pressure is applied, the resistance goes down when the pressure increases. The graph indicates the approximate resistance at different force measurements made in Newtons (N). Notice that the graph is not linear and the resistance quickly jumps from infinite to 100KΩ.

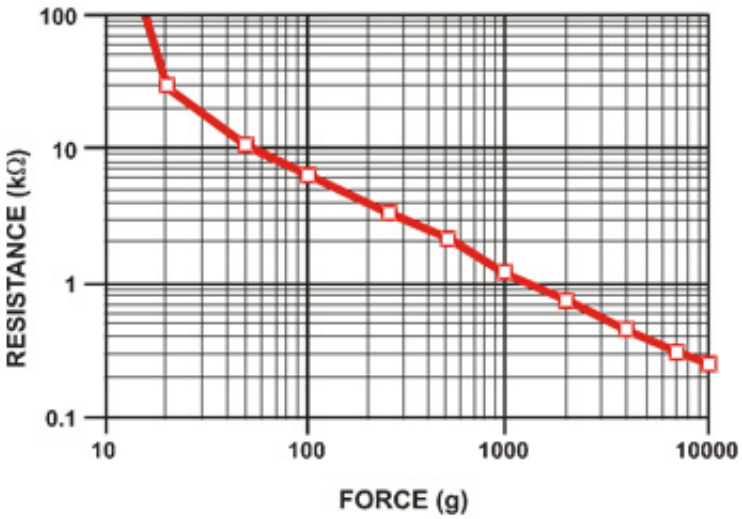

Figure 3.8: Resistance at different force measurements

The force range of the FSRs is from 0 to 100 N. 1 N is equal to the amount of force required to accelerate a mass of one kilogram at a rate of one meter per second squared. On the earths surface 1 kg of mass exerts a force of approximately 9.81 N.

$$N = kg * \frac{m}{s^2}$$

The average ski jumper mentioned earlier exerts a weight W

$$W = 60kg * 9.81N = 588.6N$$

588.6 N force to the ground. This is much more than of the FSRs 100 N force limit. However the ski jumpers force to the ground is divided onto two legs which exert a Weight/Leg W_L, $W_L = W/2$

$$W_L = 588.6N/2 = 294.3N$$

294.3 N on each foot underneath the jumper.

As a first estimate the pressure zone P_z is the force over the total pressure area $P_z = W_L/pressurearea$

$$P_z = 294.3N/63.5cm^2 = 4.63N/cm^2$$

Since the FSRs total area is $1.56cm^2$ the

$$P_z = 1.56cm^2 * 4.63N = 7.23N$$

and is well within the range of the FSRs 0 to 100 N. Of course this is only an estimate since the acceleration of the body towards the center of the earth is not divided equally on the whole foot blade.

Figure 3.9: Measurement of feat pressure, white is higher force and green is lower force

Figure 3.9 indicates that roughly 10% of the area contribute to 60% of the weight, further calculations gives $W_L = 294.3N * 0.60 = 176.58N$. If the total area under the foot which is receiving force is adjusted from $63.5cm^2$ to $6.35cm^2$ *divided by* 2 *zones* the P_z becomes

$$P_z = 176.58N/(6.35cm^2/2) = 55.62N/cm^2$$

$$P_z = 1.56cm^2 * 55.62N/cm^2 = 86.76N$$

86.76 N of pressure on each sensor are probably closer to what could be expected from the prototype readings if the sensors are placed in the high pressure areas indicated by white in Figure 3.9. This gives the ski jumper a pressure headroom P_{hr} of $P_{hr} = 100N - 86.76N = 13.24N = 1.35kg$ to inflict each of the four sensors, a total of $P_{hr} = 13.24\%$ of W.

3.3.2 MCU

The main objectives of the MCU is to collect analog input from the sensors and then using an ADC, convert the analog input to a digital representation. The digital representation of a sensor input is then relayed through the USB host shield to the Android powered device.

The capabilities of the MCU lies in its sample rate, resolution and transfer rate through the USB host shield. The SPI interface which the MCU communicates with the USB host shield can achieve speeds of 128 Mbits however this is hardware dependent. The SPI on the ATmega2560 can achieve 4 Mbits. The USB host shield contains all of the digital logic and analog circuitry necessary to implement a full-speed USB which can achieve 12 Mbits.

Resolution

The MCU selected is the AVR ATmega2560 and has an ADC of 10 bit resolution. This means that the resolution is 2^{10} or 1024 (0 to 1023) of the reference voltage. The reference voltage used is 5V on the Arduino prototyping board which means that the smallest detectable amount of voltage variation is 5/1023 or 4.9mV (0.0049V). But that is not the limitation of the ATmega2560, the voltage that the ports on the Arduino is tied to is 5V however this is adjustable. The Arduino can be tied to 1.1V through software which gives a resolution of 1.1/1023 or 1.1mV (0.0011V)[3]. There is also the ability to utilize the AREF pin which can be tied to 0.5V which again enables the ability to detect a voltage of 0.5/1023 or 0.49mV (0.00049V).

The reference voltage supplied to the sensors in the prototype system in this thesis will be the standard 5V which gives a voltage variation of 5/1023 or 4.9mV which corresponds to 0.098 N or 9.97 g as the smallest detectable amount of force unit from the FSRs.

[3]http://arduino.cc/en/Reference/AnalogReference, last accessed 20. Apr, 2012

Sample rate

In order to sample the FSRs there is a need for a Data acquisition system (DAQ). A DAQ is a system which collects data at given time intervals. If the DAQ sample values at 1000 samples a second then that is to 1000 voltage readings a second, and the time spacing between the samples is 0.001 second (1ms). Since the time spacing is at a given time interval, the time spacing is known and there is no need to collect the time spacing at each sample. But the Arduino which is being used in the prototype is not initialized as a DAQ so the Arduino does not sample at precise intervals, although it can be setup to do so. When the Arduino is calculating or looping there is a certain amount of CPU time doing those tasks. Since the Arduino/Wiring library is set up the way it is data collection becomes more of an afterthought for the Arduino. It is possible to set up timers before and after analogRead to measure the Arduino's performance.

At the highest speed the ski jumper have a speed of 93.5 km/h. This corresponds to meters per second $\frac{m}{s}$

$$\frac{m}{s} = 93,5km/h = 25,97m/s$$

25.97 m/s. If the wanted number of samples is 100 samples per meter a second $S_{\frac{m}{s}}$ in the last 6.65 m of the in-run, in the ski jumping hill, where the speed is at its highest the MCU needs to be able to sample

$$S_{\frac{m}{s}} = 2600S/25.97\frac{m}{s} = 100.1$$

2600 samples a second.

If the debugging information is written to the serial library included in Arduino the sampling rate goes down. The amount varies depending on which baud is used on the serial transfer. Using 9600 to 115200 baud may affect the sampling rate by 10 times.

AnalogRead which reads the voltage pin reading and runs it through the ADC takes about 120 microsecond which gives a sampling rate of 8300 samples a second. 8300 samples a second is probably more than enough for the prototype but this is not the limitation of the Arduino Mega.

On the ATmega2560 the ADC clock is set to 16MHz and divided by a pre-scale factor. It turns out that in the wiring.c file which is one of the setup files for the Arduino the pre-scale is set to 128 (16MHz/128 = 125KHz). A conversion takes 13 ADC clocks so the sample rate is about 125KHz/13 or 9600Hz. If the pre-scale is set to something else like 16, then this would give a sample rate of 77KHz.

The documentation for the Atmel AVR ADC[15] specifies:

> The ADC accuracy also depends on the ADC clock. The recommended maximum ADC clock frequency is limited by the internal DAC in the conversion circuitry. For optimum performance, the ADC clock should not exceed

200 kHz. However, frequencies up to 1 MHz do not reduce the ADC resolution significantly. Operating the ADC with frequencies greater than 1 MHz is not characterized.

So using a pre-scale of 16 would give an ADC clock of 1 MHz and a sample rate of 77KHz without much loss of resolution. The 77KHz is theoretical and the limitation of the ATmega2560 is probably somewhere near 56KHz which should be within reach. This is well above what is required for the prototype where the order of 4000 to 8000 samples a second should be more than sufficient. Setting the pre-scale to 16 can easily be done by adding the following code example:

Listing 3.2: Setting pre-scale to 16 to get about 56KHz sampling rate

```
// defines for setting and clearing register bits
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

// set prescale to 16
sbi(ADCSRA,ADPS2);
cbi(ADCSRA,ADPS1);
cbi(ADCSRA,ADPS0);
```

3.3.3 Mobile phone

The Nexus S connected to the MCU by its USB connection. The phone is equipped with a USB 2.0 Hi-Speed connection with a theoretical data rate transfer of 480 Mbits. The phone is the USB slave of the MCU so the USB host shield sets the speed to 4 Mbits which is the maximum SPI transfer rate from the Arduino board to the USB host shield. Any delay in the SPI transfer would slow down the number of samplings the system gets because data is directly transferred through the MCU to the mobile phone.

The mobile phone is equipped with an Advanced RISC Machine (ARM) 1GHz Cortex A8 (Hummingbird) processor which is very low power and fast. The processor can handle a Hi-Speed USB transfer rate so the phone handles 4 Mbits over USB with ease.

3.3.4 Server and client control

The main objectives of the client controller and server application is to monitor and control the mobile phones in the system. The ability to control a device is limited to start/stop streaming data or start/stop recording data. The streaming of data is only suitable over a Wi-Fi connection because of the responsiveness.

The data is sent over Transmission Control Protocol (TCP) so every packet is required to come through. This is not suitable for streaming over 3G network where it would be better with User Datagram Protocol (UDP) data sending to get the best responsiveness, but this could prove unusable if to many data packets were lost. This this problem leads to the functionality where the client controller are able to send short commands. These commands are the start/stop recording/streaming commands. Start recording command starts the recording on the clint phone and when a stop recording command is received by the phone the recording is stopped and the sensor data is sent to the server application client controller.

This makes the system more reliable even at very poor connections and data rate restrictive networks like the GPRS network data rates.

3.3.5 Summary

This chapter looked at the functional architecture of the prototype system. Revisiting the use-case scenarios outlined the demands for the blocks in the system. Then the chapter presented an overview over the main blocks and their capabilities. The main blocks also include interfaces between them which were described before starting the detailed specification of the main blocks. The detailed specification of the main blocks looked at the individual blocks in the system and outlines their limitations and analyzed them against the requirements of the block. This was important in order to outline the systems limitations and if the individual blocks of the system can perform as expected in a prototype environment.

3.4 Prototype system overview

The system design is now ready for implementation and the selected overview design is outlined in Figure 3.2. As mentioned earlier the mobile phone, a Samsung Nexus S, is expected to perform without problems due to its high processing power and built in Hi-Speed USB capabilities.

The mobile phone connects to the USB host shield by Circuits@Home, which is expected to have a speed of about 4 Mbits. Presumably enough to send the sensor data to the phone at a sample rate of 4000-8000 samples a second.

The MCU is the Arduino Mega 2560 was chosen for its ease of programmability and widespread support and as outlined in the detailed specification is capable of sampling at theoretical 77KHz which is a lot more than needed.

Finally the FSR sensors selected proved to be a little bit trickier to find the limitation for its use. The sensor measures 100N and preliminary calculations in the detailed specification showed that, if the sensors are placed in high pressure areas under the foot that 86.76 N could be expected when an athlete of 60 kg were standing on the sensors with an inflict headroom of total 12.5% on all sensors.

The next chapter looks at the implementation of the prototype system.

Chapter 4

Implementation

This chapter will describe how to complete the data collection, how the hardware sensors connects to a MCU and how they connect to an Android phone. This chapter will also explain how the software, MCU software is connected to the Android mobile software and how the external client software remote control the mobile phone. In general this chapter explains how the hardware and software components of the prototype is designed.

The prototype description is divided into two parts:

Part I

The first part describe the development of the hardware and software components and is divided into two subparts, hardware development and software development.

The first subpart describe how the hardware sensor circuit is designed, work and connected to the MCU in order to apply power to and data readings from the sensors.

The second subpart describe the development of the Arduino software, Android software and Desktop server application.

Part II

The second part describe how the prototype components interact with each other and is divided into two subparts, hardware interaction and software interaction.

The first subpart describe how the hardware components is connected and how they interact with each other.

The second subpart describe how the software components interact in order to collect sensor data and transfer to the desktop application.

4.1 Prototype overview

"Taunus" is the project name for the prototype and is the system created by the author to prototype and test the Android ADK technology for a high velocity data collection system. The project will use force resistive sensors to measure the center of pressure of

an athlete in motion and record the data collected and transfer them to en external client through the Internet.

4.2 Development of the hardware

The hardware part of this system consists of the sensors, Arduino Mega 2560 MCU, Android Nexus mobile phone and standard PCs.

Sensor circuit

The sensor circuit is the hardware part of the prototype that is the most customized. It has 4 FSR that detects physical pressure. The FSR can be connected either way because they are basically resistors and thus non-polarized. The first test of connecting the FSR was done connection one FSR directly to a breadboard.

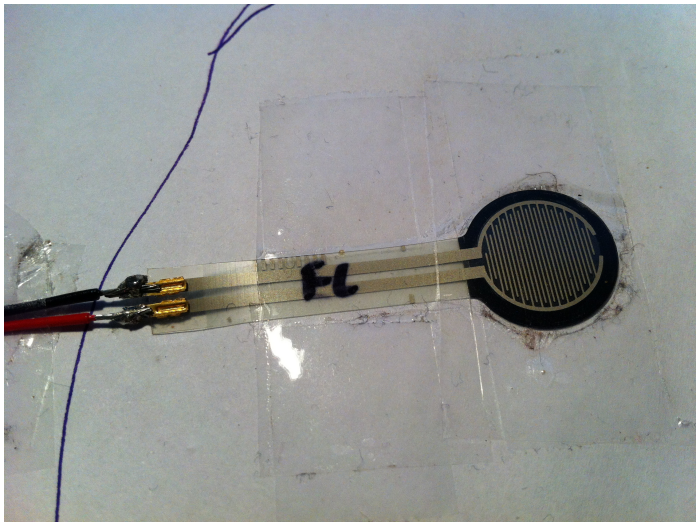

Figure 4.1: FSR with soldered pins

Though it is not recommended to solder the wires directly on the FSR pins, it would be the best solution if the sensors were to be mounted into shoe soles. Soldering, as opposed to using some kind of alligator clip, would have the smallest lumps thus making it the better alternative when placing them into shoe soles.

The sensors is supplied with 5V of power on one pin and the other pin to the ground. Before the ground pin there is a connection to an analog input pin on the MCU. This creates a voltage divider circuit, shown in Figure 4.3 and makes it possible to measure the voltage that comes across the FSR which ranges from 0V to about 5V, roughly the same as the amount of voltage supplied to the FSR.

$$V_{out} = \frac{R2}{R1 + R2} * V_{in}$$

Figure 4.2: Voltage divider formula

In the voltage divider formula Figure 4.2 the V_{out} is the ANALOG Voltage shown in Figure 4.3. The ANALOG Voltage is the result of applying V_{in} Figure 4.2, +5V Figure 4.3, across R1 and R2 Figure 4.2, FSR and R Figure 4.3.

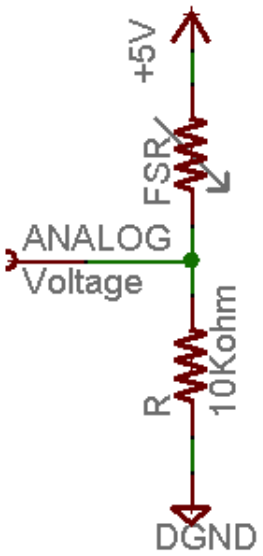

Figure 4.3: Voltage divider circuit

Using the voltage divider formula to calculate expected output voltages when different pressure is applied. The following example use the values

$$V_{in} = 5V$$

$$R2 = 1k\Omega$$

$$R1 = 1M - 200\Omega$$

The expected voltage when no pressure V_{np} applied

$$V_{np} = \frac{1k\Omega}{1M + 1k\Omega} * 5V = 0.00499V$$

Expected voltage when medium pressure $V_{mp} = 45N$ is applied, FSR 1kΩ

$$V_{mp} = \frac{1k\Omega}{1k + 1k\Omega} * 5V = 2.25V$$

The expected voltage with maximum pressure $V_{xp} = 90N$ applied

$$V_{xp} = \frac{1k\Omega}{200 + 1k\Omega} * 5V = 4.16V$$

The sensor circuit consists of four voltage divider circuits which is connected in parallel. The 5V applied to the circuit is distributed equal to the four sensors as shown in Figure 4.4. The parallel circuit creates equal voltage to the four sensors, the four sensors have different resistance depending on the force applied thus creating a variable output voltage. The different resistance is illustrated in Figure 4.4 where two sensors is given 200Ω, maximum pressure applied, and the other two is given 500Ω, slightly less force applied. The maximum output voltage is 4.17V when a 1k resistor is used before the ground. As an alternative 10k resistors could be used to get the voltage range from 0 to 5V.

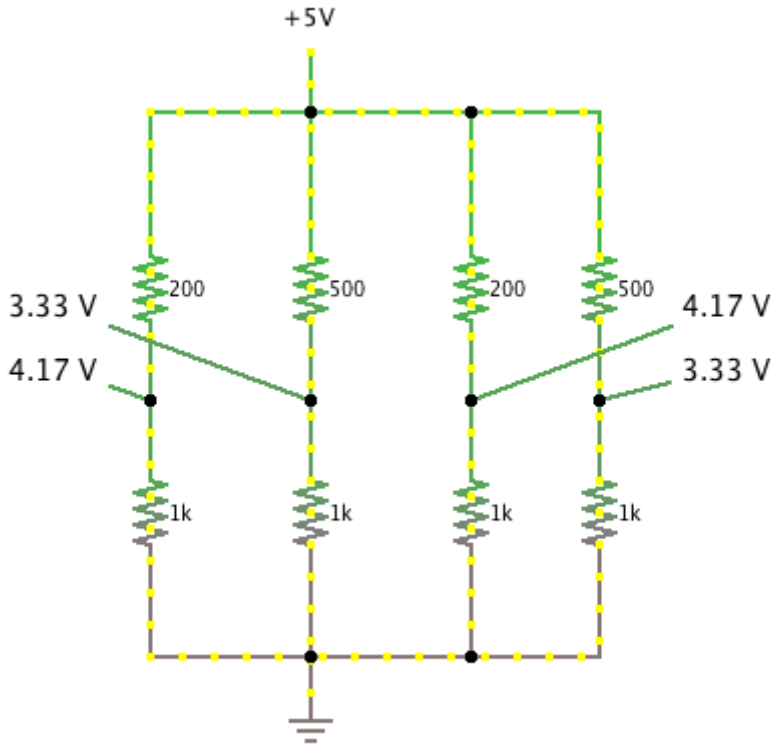

Figure 4.4: Four voltage divider circuits

The image in Figure 4.5 shows the back of the sensor circuit where the red wires transfer 5V of power to the FSRs. The black wires transfer the voltage that the FSRs do not resist. The remaining amount of voltage can then be read on the pins that are connected to analog input sockets on the MCU. The 1k resistors lead to ground.

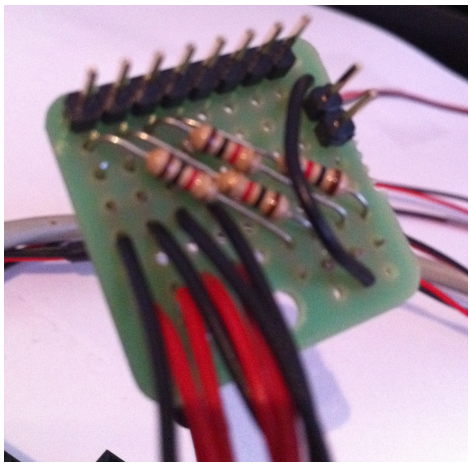

Figure 4.5: Sensor circuit, back

The image in Figure 4.6 shows the front of the sensor circuit. The green board is a very simple single sided perfboard for soldering components to and creating small prototype circuits[1]. In one direction every three holes are connected with a 20mil trace. The board has a standard 0.1" (2.54 mm) spacing which is the same as the Arduino prototyping platform thus making it easy to connect.

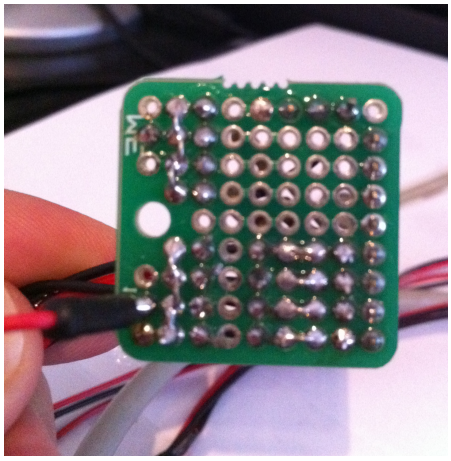

Figure 4.6: Sensor circuit, front

MCU board

The image in Figure 4.7 is of the fully assembled MCU. It has an USB host shield and the sensor circuit connected to it. Power for the sensor circuit is drawn from the 5V pin on the USB host shield which is the 5V pin from the Arduino relayed through the USB host shield.

[1]http://www.reprise.com/host/circuits/perfboard.asp, last accessed 27. Apr, 2012

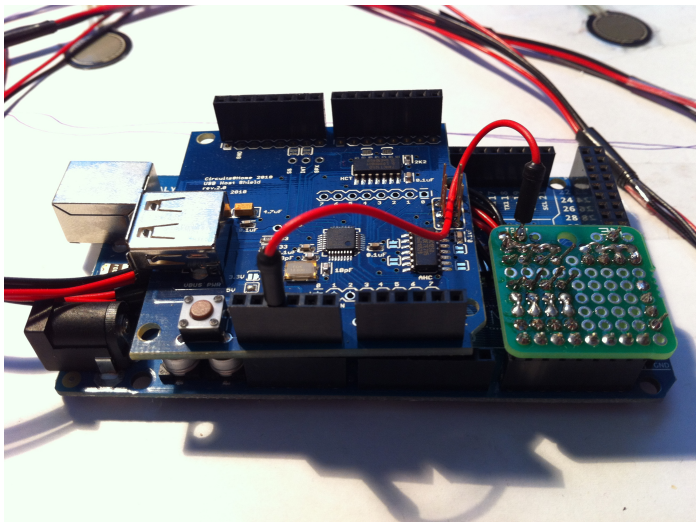

Figure 4.7: MCU circuit

The USB connection on the Arduino is an USB slave connection. The Android ADK requires an USB host connection on its accessory, so the Android phone can act as the USB slave. The USB host shield on top of the Arduino in Figure 4.8 is required for the Arduino to act as an USB host. The USB host shield can be integrated onto the Arduino board, this is done in the Arduino Mega ADK version[16]. The Arduino Mega 2560 used in this prototype requires the USB host shield in order to be the USB host. The shield is mounted easily to the Arduino board because the pin layout is the same.

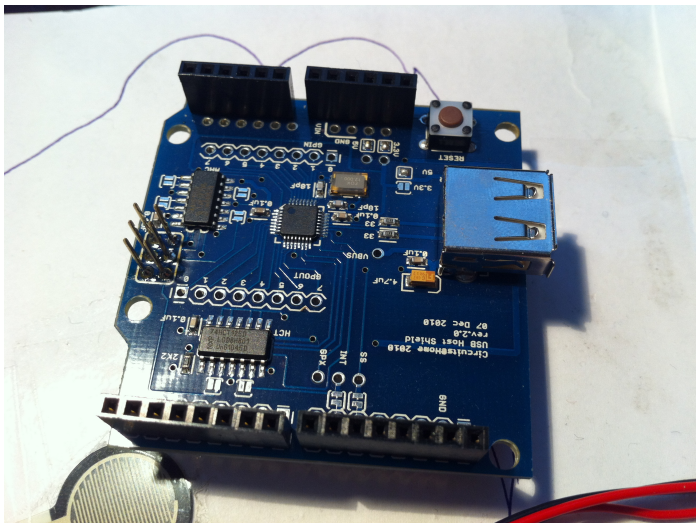

Figure 4.8: USB host shield

The size of this prototype is a lot larger than what could be done with this MCU circuit. As can be seen in Figure 4.9 the battery pack is 2/3 of the MCU size. The MCU

can use any type of battery pack as long as it outputs between 6-20V. The recommended voltage use is between 7-12V, if 6V is used the board may be unstable because this could supply the board with less than 5V making it unusable. If more than 12V is supplied then the voltage regulator may overheat and damage the board. It would be an advantage to have as small as possible battery pack, but for this prototype a 9V battery is used and with the battery casing the prototype becomes relatively big.

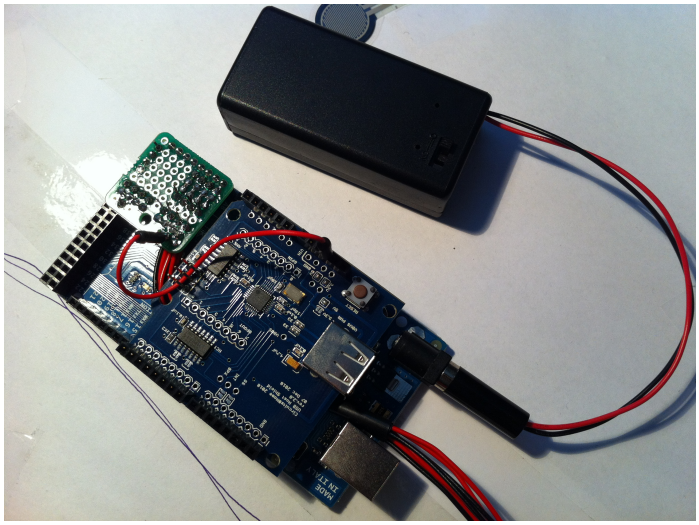

Figure 4.9: MCU circuit w/battery

The Android mobile phone is connected to the Arduino MCUs USB host shield through a type-A to micro USB cable which is a standard USB cable with a *Micro-B to Standard A* USB interface. The USB Micro-B interface is being accepted by nearly all the cellphone manufacturers in the world and is becoming the standard charging port for all newer mobile phone devices.

4.3 Development of the software

This section will describe the software components that are necessary to implement a prototype.

Arduino IDE

Programming the Arduino MCU requires the Arduino IDE[2]. The specific version used in the development of the prototype is the Arduino 0022. The code used to program the Arduino follows below. It requires the USB host shield library[3] and AndroidAccessory[4]

[2]http://arduino.cc/en/Main/Software, last accessed 24. Apr, 2012
[3]https://github.com/felis/USB_Host_Shield, last accessed 24. Apr, 2012
[4]http://developer.android.com/guide/topics/USB/ADK.html , last accessed 24. Apr, 2012

library to be installed.

Detailed description of how to program the MCU is includes in Appendix A.

Eclipse IDE

Eclipse is an IDE by the Eclipse - The Eclipse Foundation open source community[5] and is used to develop the Android application for Taunus.

The Android Development Tool (ADT)[6] is a plugin for Eclipse IDE is an interface between Android SDK and Eclipse IDE. It is installed as an Eclipse Plugin and makes it possible to compile and run software directly on Android powered equipment from Eclipse.

Detailed code for the application is included on a CD that comes with this thesis and on GitHub[7].

NetBeans IDE

NetBeans is an IDE by Oracle previously Sun[8]. The NetBeans IDE is used to develop the server application. NetBeans IDE has a built in Graphical User Interface (GUI) builder which is easy to use and speeds up development.

Detailed code for the server application is included on a CD that comes with this thesis and on GitHub[9].

[5]http://www.eclipse.org/, last accessed 18. Apr, 2012
[6]http://developer.android.com/sdk/index.html, last accessed 18. Apr, 2012
[7]https://github.com/sangar/Taunus, last accessed 23. Apr, 2012
[8]http://netbeans.org/, last accessed 18. Apr, 2012
[9]https://github.com/sangar/TaunusServer, last accessed 23. Apr 2012

4.4 Hardware interaction

This section explains the hardware interaction of the project, from sensor input ADC to data sent over network to the server.

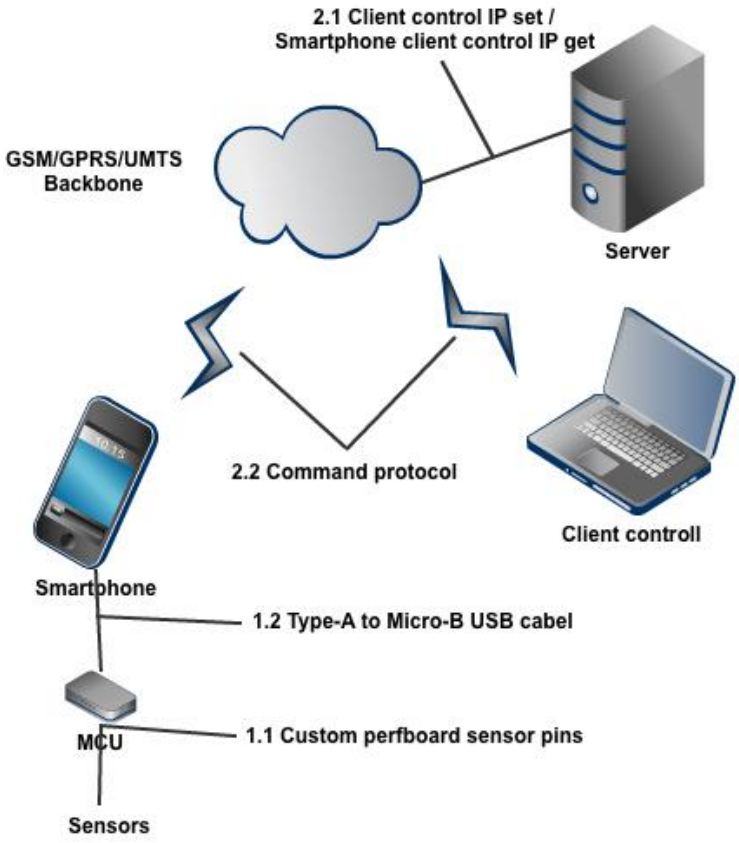

Figure 4.10: System interface overview

Sensor circuit

As mentioned earlier the ADC of the MCU ATmega2560 used in this project is built into the MCU hardware. The Arduino board runs on 5V current which is put into the custom sensor circuit perfboard and data is read from the voltage divider, 1.1 in Figure 4.10.

MCU

On the MCU software side the ADC is called by setting specific values to specific registers on the MCU. This allows for the collection of digital representation of the analog input from sensors. Figure 4.1 shows and comments how the MCU register values is converted into an integer representation of Ansi C int.

Listing 4.1: Analog to digital registers on ATmega2560 (Simplified)

```
int analogRead(uint8_t pin) {
        uint8_t low, high;

        if (pin >= 54) pin -= 54; // allow for channel or pin
           numbers

        // the MUX5 bit of ADCSRB selects whether we're reading
           from channels
        // 0 to 7 (MUX5 low) or 8 to 15 (MUX5 high).
        ADCSRB = (ADCSRB & ~(1 << MUX5)) | (((pin >> 3) & 0x01)
           << MUX5);

        // set the analog reference (high two bits of ADMUX) and
            select the
        // channel (low 4 bits).  this also sets ADLAR (left-
           adjust result)
        // to 0 (the default).
        ADMUX = (analog_reference << 6) | (pin & 0x07);

        // start the conversion
        sbi(ADCSRA, ADSC);

        // ADSC is cleared when the conversion finishes
        while (bit_is_set(ADCSRA, ADSC));

        // we have to read ADCL first; doing so locks both ADCL
        // and ADCH until ADCH is read.  reading ADCL second
           would
        // cause the results of each conversion to be discarded,
        // as ADCL and ADCH would be locked when it completed.
        low  = ADCL;
        high = ADCH;

        // combine the two bytes
        return (high << 8) | low;
}
```

USB Host Shield

The data between MCU and Android mobile phone is run through the USB-host shield. The host shield is driven by the Max3421e IC and is programmed through the SPI interface which is imported through the USB_Host_Shield library that takes care of the SPI

interfacing. The interface between the USB Host Shield and Android phone is the Type-A to Micro-B USB cable, 1.2 in Figure 4.10.

Android phone

The Android phone connects to the USB host shield and is the USB slave in relationship to the USB host shield. If the Android equipment runs Android version 2.3.4/3.1 or higher the AAP running from MCU firmware will try to configure the Android device to accept the Android accessory and start the accessory application.

Network communication

The NAT inside the mobile telephone network makes is impossible to run a server application from a mobile phone and connect to it using standard IP routing. This problem was solved by a 3rd party server. The server application has to register its IP address with the registration server, so that the Android mobile phone can look up the server applications IP address without having to enter it into the phone physically, 2.1 in Figure 4.10. In Figure 4.2 is a simple Python script which registers the IP address of a request calling *?mode=set* on the script, *?mode=get* gets currently stored IP address.

Listing 4.2: Server application script

```
#!/usr/bin/python3.2

import os
import sys
import cgi

def main():
        print("Content-type: text/html\n\n")
        print()

        mode=''
        request = cgi.FieldStorage()
        if request.getvalue('mode'):
                mode = request.getvalue('mode').lower()

        env = os.environ
        if mode == 'set':
                f = open('ipstorage', 'w')
                f.write(env['REMOTE_ADDR']);
                f.close()
        elif mode == 'get':
                f = open('ipstorage', 'r')
                ip = f.read()
                f.close()
```

```
                print(ip)
        elif env:
                print(env['REMOTE_ADDR'])

if __name__ == '__main__':
        main()
```

4.5 Software interaction

This section describes how the software interaction is conducted. How the MCU firmware the data is collected through ADCs and sent through the SPI interface, USB host shield and AAP. This section also describes how the data is collected on the Android phone through the AAP and how this is absorbed by the Android application.

MCU

The ADC conversion is displayed in code listing 4.1. The communication between the USB host shield is done through the USB_Host_Shield written by Oleg Mazurov[10]. The USB_Host_Shield library is utilized by the AndroidAccessory library that is responsible for communicating with the Android application via the AAP.

The AndroidAccessory library consist of two files, one header file and one C++ source file. The header file defines four public methods and one constructor shown in code listing 4.3.

Listing 4.3: AndroidAccessory library header file

```
AndroidAccessory(const char *manufacturer,
                 const char *model,
                 const char *description,
                 const char *version,
                 const char *uri,
                 const char *serial);

void powerOn(void);

bool isConnected(void);
int read(void *buff, int len, unsigned int nakLimit = \gls{usb}
   _NAK_LIMIT);
int write(void *buff, int len);
```

The accessory needs five 0 terminated strings within 255 characters to identify the accessory. This is to be able to differentiate which application should be started on the Android equipment. When an AndroidAccessory object is created the *powerOn()* method

[10]https://github.com/felis/USB_Host_Shield_2.0, last accessed 18. Apr, 2012

is called on setup and the *isConnected()* methods checks for any USB devices devices. When an USB device connects, the read and write methods can be utilized.

Android

On the Android equipment side the Android application open a file descriptor to the USB accessory. This file descriptor is the pipe to the sensor system were data is written and read.

Server application

The client controller server application sends commands to the Android mobile phone, 2.2 in Figure 4.10. These commands start/stop the streaming of data or start/stop recording of data. Figure 4.11 shows a screenshot of the server application. The application contains a chart library which enables the application to display data in a linechart as seen in Figure 4.11. The library used is called JFreeChart[11].

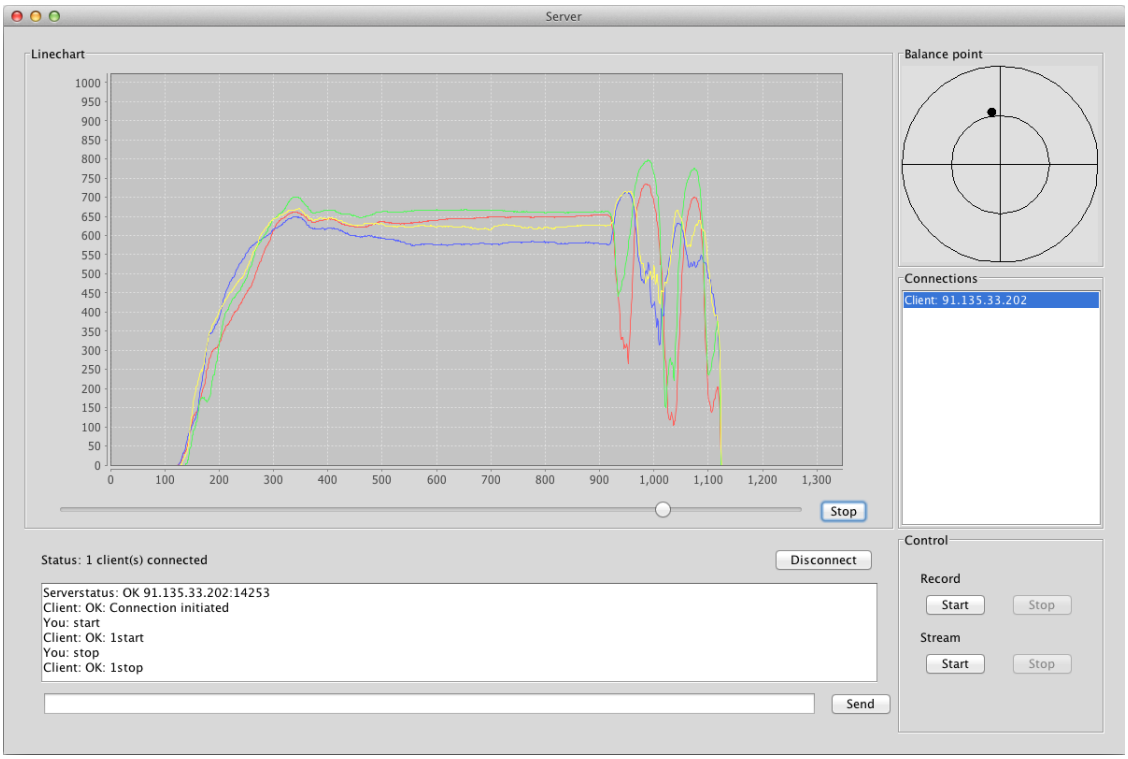

Figure 4.11: Screenshot of the client controller server application

[11]http://www.jfree.org/jfreechart/, last accessed 30. Apr, 2012

4.6 Summary

This chapter set the stage for all components used, these components include the sensors custom circuit, the MCU with the USB host shield, the Android mobile phone, the IP registration server and the client controller server application.

The sensor circuit showed that the voltage divider formula could predict what the voltages could be expected at different values from the sensors.

The Arduino IDE was used to establish MCU application which converts analog signals from the sensors to digital representations and send the data through the USB host shield to the Android mobile phone.

The Eclipse IDE was used to establish a mobile application which listens on a file descriptor for data from a MCU. The mobile application checks the registration server for the IP address of the client controller and connects to the client controller server application if available. The application further awaits for instructions from the client controller for which task to perform.

The NetBeans IDE was used to establish a client controller server application which registers its IP address with the registration server before listening for connecting clients. When clients connect the client controller can start sending command and instruct the clients about what to do.

This chapter covered the development cycle, the run cycle is further described at the start of the next chapter.

Chapter 5

Prototype demonstration and evaluation

This chapter will demonstrate the prototype features, and how the prototype components interact in practice. The chapter finishes with an evaluation of the prototype system which look at how the prototype did perform in practice.

5.1 Demonstration

This section gives an overview over the individual components that the prototype system consist of through a demonstration overview. Then a detailed run through of the system where a complete how-to is conducted in order to outline all of the systems functions.

5.1.1 Demonstration overview

This section gives an overview of the four main components in the prototype system.

Figure 5.1: Sensors mounted under shoe soles

The sensors are places under the foot of the designated athlete. The prototype of this thesis use two sensors for each foot, which makes it possible to measure if an athlete is balancing in front of the foot or back of the foot. The prototype is not measuring force, only the relationship between front of foot and back of foot.

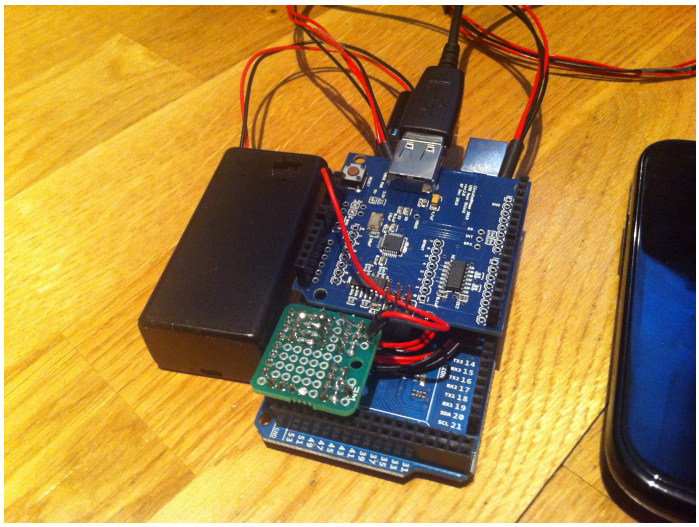

Figure 5.2: Arduino MCU prototyping board, USB host shield, sensor circuit perfboard

The MCU is connected to the sensors and relays the sensor data through the USB host shield after converting the analog sensor signals to digital representations. The USB host shield sends the data collected to the Android equipment over the AAP and USB interface.

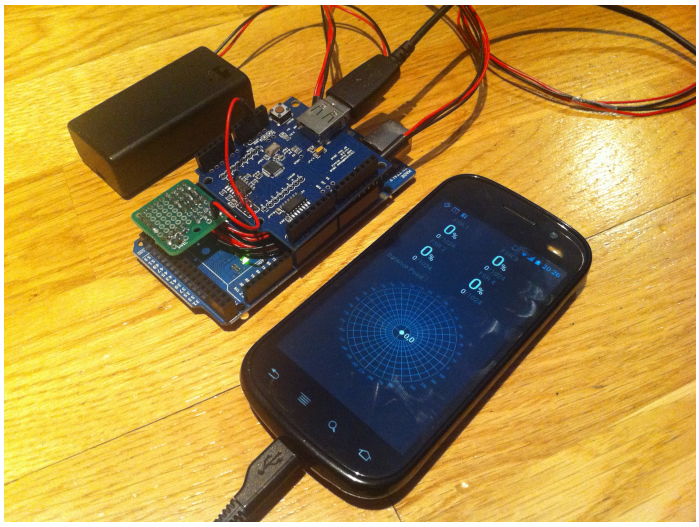

Figure 5.3: Android mobile phone connected to Arduino

The mobile Android equipment receives the sensor data from the MCU accessory through the USB cable and displays the average of the four sensor on its screen and either records the data before sending it to the server or directly streams the data to the server.

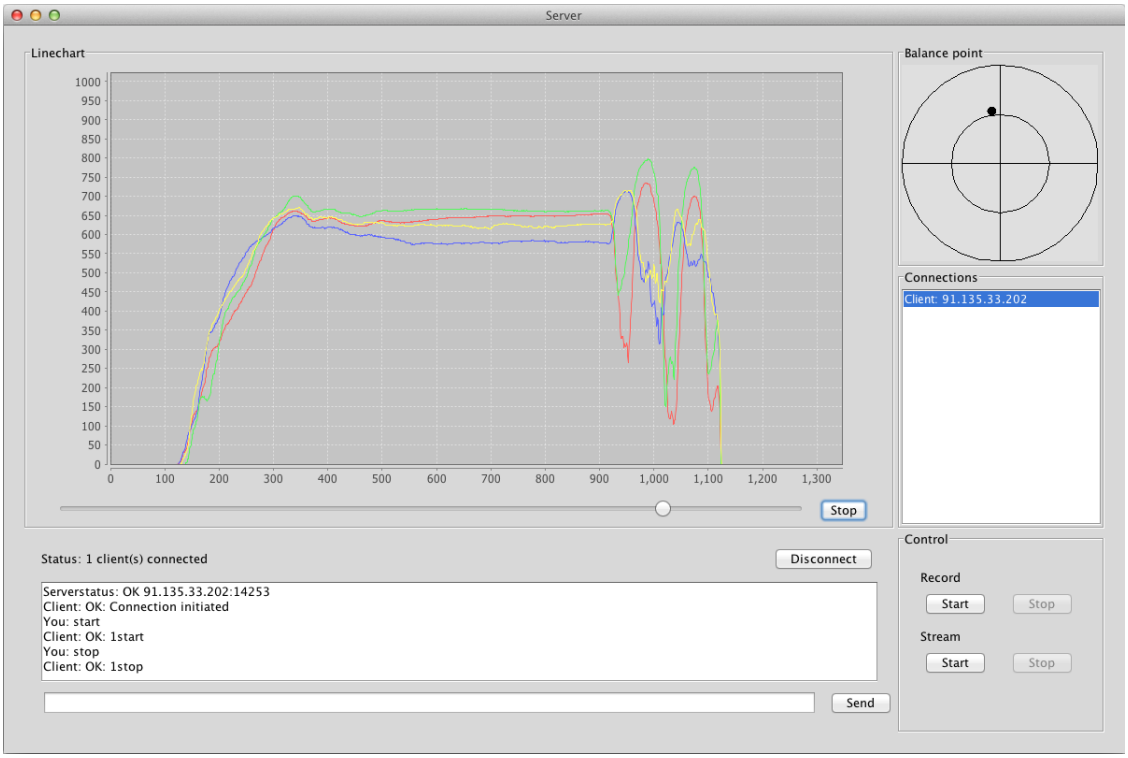

Figure 5.4: Server application with sample data

The server application, shown in Figure 5.4, keeps track of all the Android equipment in the network. The server application sends commands to the Android application, the commands are short messages to tell it to either start or stop recording of data, start or stop streaming of data or disconnect from the network. When the server tells the client to record data, the client will record data until it receives a stop recording command and the client will send all of its recorded sensor data to the server application. The server application will then receive the data and display the recorded sensor data.

5.1.2 Run through

This section demonstrate how the sensor prototype system works, how the blocks interact and pass data between each other in a detailed how-to. The demonstration shows how the data is passed through the system, from the sensors all the way to the server application which displays the data.

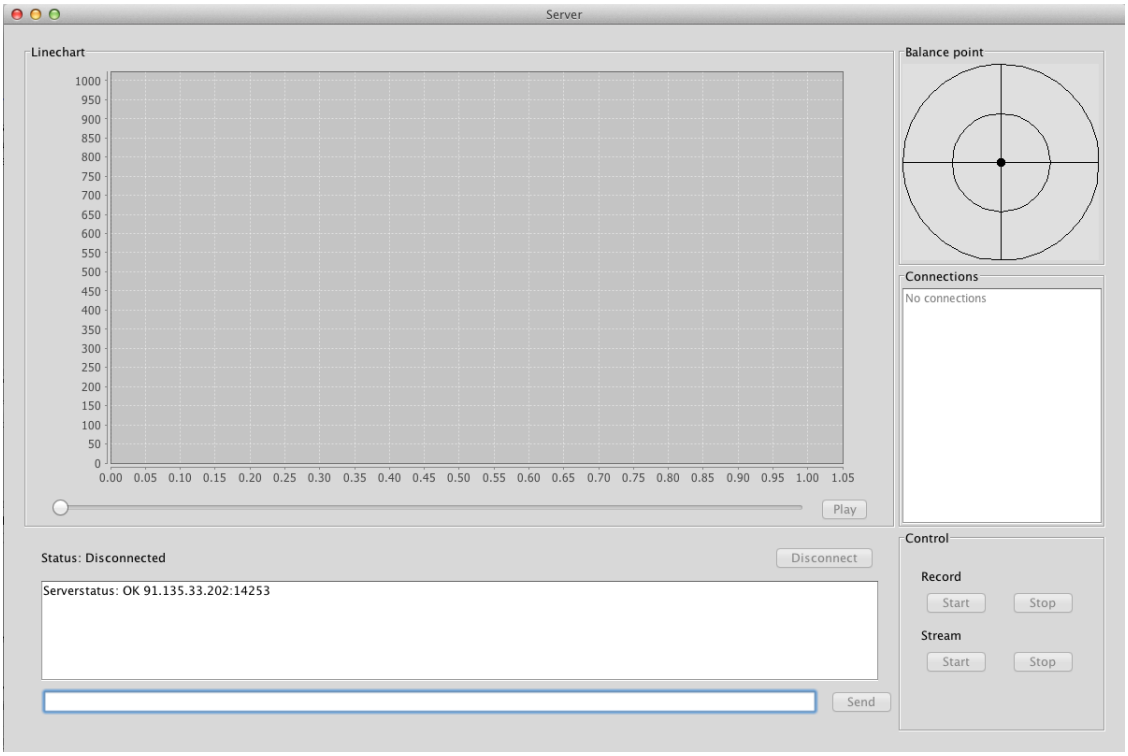

Figure 5.5: Server application at initial launch

The system is initiated by starting the client controller server application shown in Figure 5.5 . The server application then displays its GUI, registers its IP address with the registration server before starting the server socket and listening for client to connect.

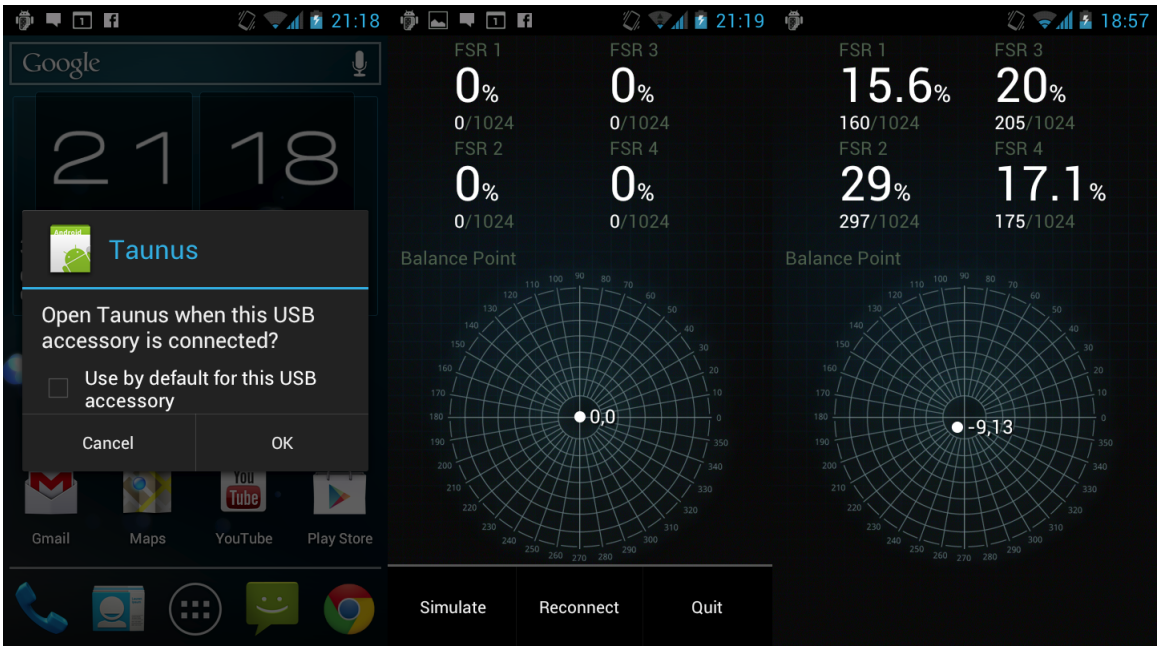

Figure 5.6: Android application screenshots

The Taunus Android application is initiated by the AAP when the accessory connects shown on the left in Figure 5.6. The user is prompted about starting the application for the specific Android accessory. The middle of Figure 5.6 shows the menu which enables the user to send simulation data to the server, connect or reestablish the connection to the server or quit and close connection to the server. The right view in Figure 5.6 shows how data from the four sensors is shown on the mobile phone and the how the center of pressure is displayed as an average of the four sensor values.

When the Taunus application starts it requests the server applications IP address from the registration server, if the IP address is obtained from the registration server the Taunus application connects to the server applications socket and send *101:203 start:connection* message to the server. The connection is up when the server accepts the Taunus application connection and receives the hello message.

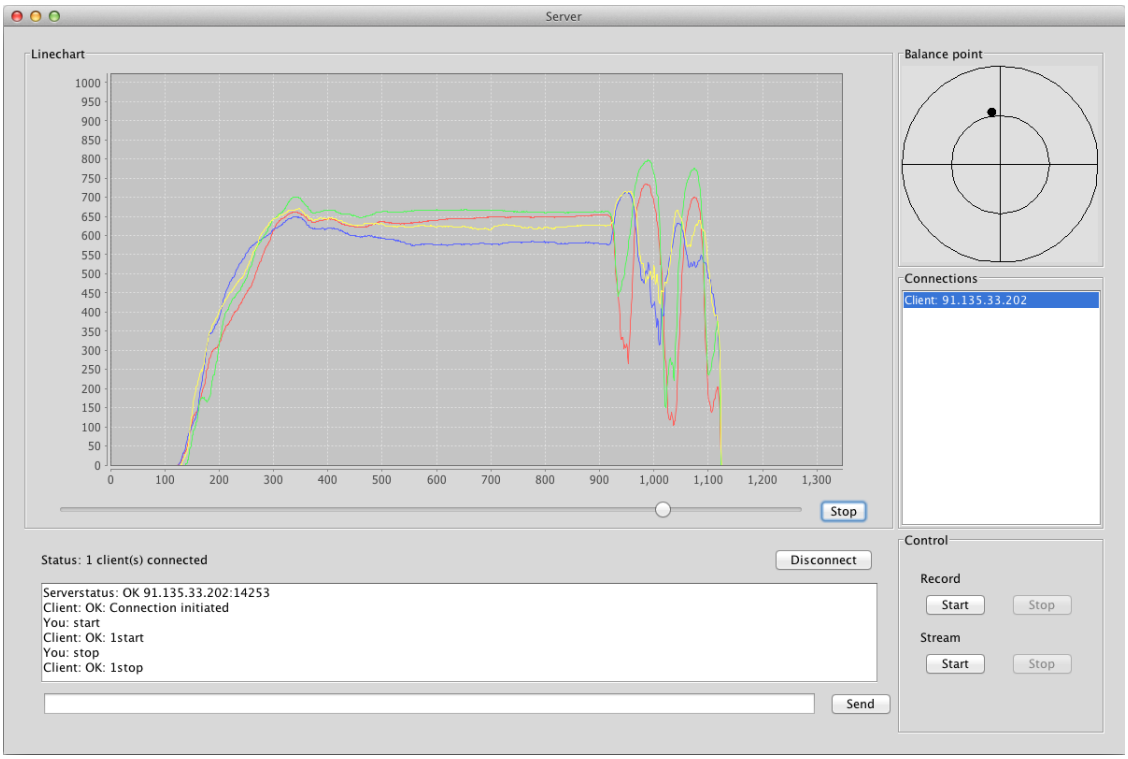

Figure 5.7: Server application with sample data

When the server application accepts a client connection the server keeps track of connected devices in a connection list, as seen in the right middle panel of the GUI of Figure 5.7. The selected connection is the one that is controlled by the control panel, shown in the lower right panel of in Figure 5.7 of the server GUI.

The control panel can start/stop recording and streaming of sensor data. If streaming is started, the sensor data is shown and the data is updated live in the line chart view of the GUI, the data is forwarded by the phone until the stop data streaming command is received.

When recording of data is started, the phone stores the data until the stop recording message is received and the data is forwarded to the server application and displayed in the line chart view of the GUI.

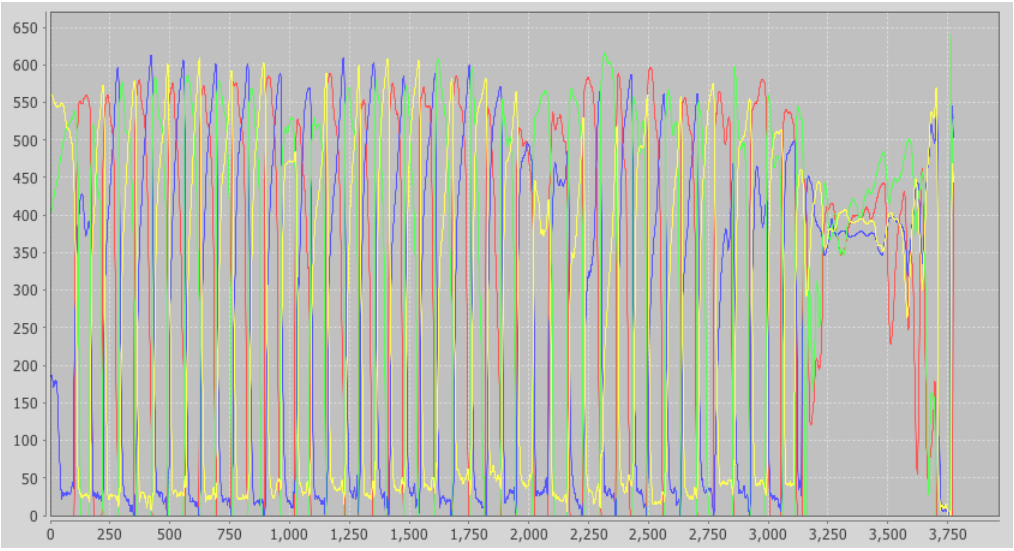

Figure 5.8: Sample of walking data

Figure 5.8 shows a sample of recorded walking data. It is possible to zoom the data in the line chart view, a sample of this is shown in Figure 5.9. In addition to this the line chart view can be saved as an image.

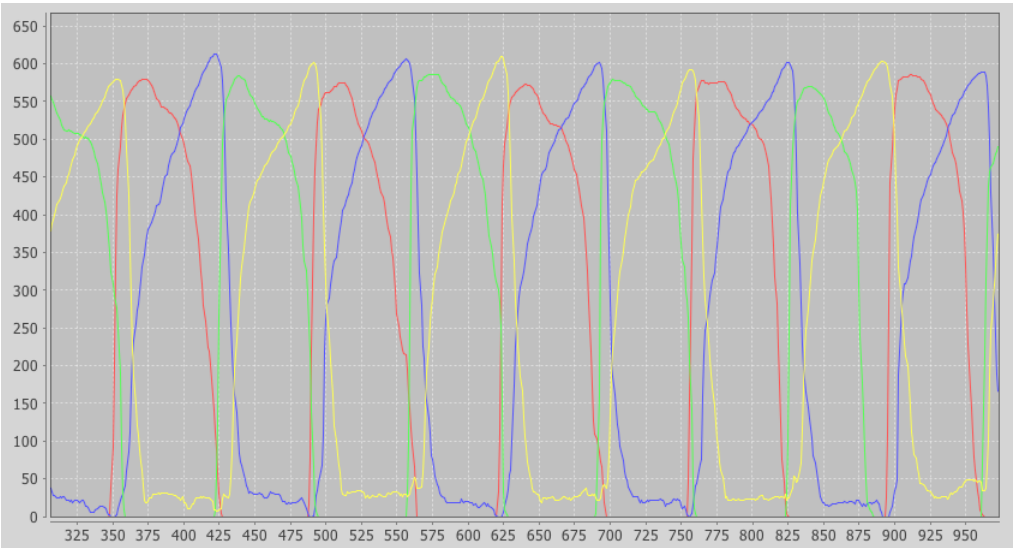

Figure 5.9: Detailed sample of walking data

5.2 Evaluation

This section evaluate the prototype system implementation. This section is divided into the following subsections:

Requirements
The introduction introduced the system parameters and technical parameters. Some of these parameters have been pointed out with numbers, example giving the maximum weight of the whole system in the order of 300 g. Thus the numbers can be used to evaluate against the requirements.

Functionality
Evaluation in terms of the overall functionality, collect, store and forward.

Performance
Evaluation in terms of the performance, maximum data amount, transmission speed, storage capacity, sampling rate and resolution.

Extensibility
Evaluate how much more could get out of the system.

5.2.1 Requirements

In the beginning of chapter 2 of this thesis, the first set of requirements were extracted. The first set of requirements were extracted for the sensors, MCU and mobile phone. The system requirements were outlined with a ski jumping application in mind and development of a system for use in a high velocity mobile environment like in the ski jumping hill. The technical parameters required the system to be able to sample center of pressure, collect sensor data on a mobile device and send the data to a computer for analysis. The system needed to be light weight so weight was considered for all blocks of the prototype.

Sensors

The sensors were supposed to be place under the foot soles of athletes being monitored. In order to conduct a center of pressure test, it would require 2 sensors for each foot and a total of 4 as the minimum. The system was going to measure the average between four points in order to given an estimate on the center of pressure. This would require the sensors to be able to measure the weight of an athlete, so the force range of the sensors were considered. And finally the sensors were going to be utilized in a mobile environment which means the battery consumption of the sensors were considered.

The following requirements were laid out for the sensors:

Lightweight
In order to be used in a highly mobile application and not interfere to much with the ski jumpers equipment the first set of requirements estimated that the sensors should weigh 1 g of less.

Size

The size requirements required the sensors to be thin, and the first set of requirements extracted the requirement of 2 mm and thinner.

Force range

The force range requirement was the the sensors would have to be able to give out readings of the amount of force per cm^2 by an athlete of about 60 kg.

Power consume

The power consume requirement were estimated to be 10 mA or less, low in regards to battery consumption.

The sensors in the assembled prototype system is the Interlink 402. Four of this type of sensor were used in order to calculate an average center of pressure in the tests. They were selected because they are thin and small with a thickness of 1.25 mm as its thickest, which is within the required 2mm, and their total area of $1.56cm^2$. This combined with their lightness of under 1 gram made them a good choice for the prototype. Their force range is described with a limit of 100N which seemed to be to narrow but calculations showed that the sensors, with their diameter combined with calculations of the spread of the footprint of a human, were able to detect forces within the weight of a human being of at 60 kg if they were placed in high pressure zones under the foot. The calculations showed that the athlete of 60 kg could inflict about 13.5% of his bodyweight before maximizing the sensors. And the power consumption is very low, about 1mA which is acceptable[1].

MCU prototyping board

The MCU were supposed to be placed on the body of an athlete and interface between the sensors and the mobile phone. In order for the MCU to collect the values from sensors the MCU would have to have at least four ADCs. The MCU would have to be fast enough to sample the required amount of samples, initial requirement of 100 samples per second. The MCU would have to be able to interface with ICs with SPI or UART interfacing in order to extend its functionality. And the MCUs power consumption would be evaluated since this were to be a mobile sensor application.

The initial requirements were:

Lightweight

The MCU had to be lighter than 10 g.

Size

The MCU for the first prototype would need to be smaller than 40 mm x 120 mm.

Clock speed

The MCU would have to be fast enough to collect more than 100 samples a second.

[1] http://www.ladyada.net/learn/sensors/fsr.html, last accessed 24. Apr, 2012

Number of ports
The MCU needed to have four or more ADC ports.

IC interfacing
The MCU would have to have the ability to interface with other ICs.

The MCU prototyping board selected used in the prototype is the Arduino Mega 2560 combined with the Circuits@home USB host shield. The USB host shield is used to interface with the mobile phone as an USB host and is utilized via SPI communication. This solution was selected because of the requirements from the Android ADK requirements. The weight of the MCU board is 36 g plus the weight of the USB host shield of about 15 g the whole MCU prototype block weighs about 51 g which is above the requirement of 10 grams and lighter, but since the system requirement was that the whole system should be under 300 g the 10 g were less significant since the mobile phone weighs as much as 129 g. And the size is within the required limitation but when adding the required battery pack, of about 50 grams, for power the size limit is breached and the weight limit is further breached. The MCU collects four sensor readings with ADC as expected and performs presumably satisfactory.

Mobile phone

The mobile phone in the system were supposed to be carried by an athlete so the weight requirement would have to be low, initially under 200g. The mobile phone should not interfere with the athletes normal procedures so the size were considered and the estimate were below, 70mm x 130mm x 12mm. In order for the mobile phone to be fast enough to handle the data samplings from the MCU the requirement of the CPU on the mobile phone were 1GHz and above. The mobile would also require different communication technologies which could be used for command and data transfer. Both high speed WLAN and mobile internet would be desired.

The initial requirements of the mobile phone were:

Lightweight
The mobile phone needs to be less than 200grams.

Size
In order to not interfere with the athletes equipment the size must be less than 70mm x 130mm x 12mm.

CPU speed
The mobile phone needs to be around 1gHz and above, fast enough to collect the data received from sensors.

GPRS/UMTS/WLAN
The mobile phone needs to have wireless communication to the internet. GPRS is fast enough to send commands in the order of 100 B.

The selected mobile phone used in the prototype system is the Nexus S. The phone has an acceptable weight of 129 g which is according to the first set of requirements. And its dimensions is 63 mm x 123.9 mm x 10.88 mm which is also acceptable. The processing power is 1 GHz and it performs good at tests of the prototype system. Finally the requirements for communication ability is also acceptable since the phone can communicate with internet with GPRS, UMTS and WLAN though it is not possible to rely on the network even if there are small data amounts due to congestion. If the system would be used in a stadium with lots of people the network could be crowded and down as a result.

5.2.2 Functionality

This section evaluates what works and what does not work in the prototype system.

The prototype is mobile but not highly mobile. Highly mobile would mean that the system is not in the way for the athletes normal procedures. The MCU unit is to big and has a lot of sharp pins on it. This would not work well if mounted on an athlete especially if the athlete were a ski jumper with a tight suit.

The functionality of the prototype is simple. The sensors collect a force reading under each foot and their value is picked up and sampled by the MCU which relays the data to the connected mobile phone. The mobile phone is controlled by a remote server via internet and data is sent back to the server from the mobile phone. The computer server application displays the data and can playback the sensor data in order to view the athletes center of pressure during the test. The server application display the four individual sensor readings in a line chart as well as displaying them in an average value view. This average value view shows a point inside of circles, like a bullseye, and the point is moving around based on the average value from the four sensors thus making it easy to see what the athletes center of pressure is.

Future development could alter the system functionality, the mobile phone could be replaced by a communications device which would require some other form of communication than internet. But the good thing about internet is that its everywhere. But then again a custom communication device would be limited to the area of the athlete and would have to be moved along with the teams.

5.2.3 Performance

The performance of the system is ok, it does what it should do, but what can be obtained from the system.

The system is initially set up to sample each sensor every 10ms with the ADC prescale of 128. The sample rate is adjustable with the prescale, so the the ADC clock C runs at 16MHz divided by a prescale factor P. Thus when the prescale is set to 128

$$C/P = 16MHz/128 = 125KHz$$

and the ADC conversion takes 13 cycles to complete, the sample rate S is

$$S = 125KHz/13 = 9615$$

or about 9600Hz as the sample rate.

The prototype is fast and responsive in the initial setup but there was a hope of getting more samples out of the system. The MCU can be tuned to sample about 56k samples a second. The initial calculations showed that if 100 $S\frac{m}{s}$ at the highest speed were the wanted sampling rate, the number of samplings would have to be 2600 a second.

Tests showed that 475 samples a second were maximum. There were hopes of getting a much higher sampling rate since the MCU initially is capable of at least a sampling rate of 56KHz. This is what the system performs but the system can be adjusted to perform at higher level than the prototype setup. Further analysis is required to determine the exact number of wanted samples a second at high speed, and to determine where the exact limitation in the system are.

The values in the preceding test result tables are

Prescale The prescale factor which adjusts the sampling rate.

Delay Delay between the each sampling, giving the mobile time to work

Samples The number of samples gathered

Time The amount of time of sample gathering

GUI response How the mobile phone GUI responded during test

Performance results for initial setup:

Prescale	Delay	Samples	Time	GUI response
128	10 ms	250	10 sec	Optimal
128	1 ms	1700	10 sec	Optimal
128	0.01 ms	4750	10 sec	Slow
128	0 ms	4750	10 sec	Slow

By Setting the prescale to 32 the sample rate is changed from

$$C/P = 16MHz/32 = 500KHz$$

which divided by 13 cycles gives

$$S = 500KHz/13 = 38461$$

or about a sample rate of 38KHz.

Performance results for prescale set to 32 setup:

Prescale	Delay	Samples	Time	GUI response
32	10 ms	250	10 sec	Optimal
32	1 ms	1700	10 sec	Optimal
32	0.01 ms	4750	10 sec	Slow
32	0 ms	4000	10 sec	Slow

Changing the sample rate did not make much difference on performance, the results are pretty similar. How about changing the prescale value to 16

$$C/P16MHz/16 = 1MHz$$

$$S = 1MHz/13 = 77KHz$$

and see what the results become with the ADC of the ATmega2560 running at its highest sample rate.

Performance results for prescale set to 16 setup:

Prescale	Delay	Samples	Time	GUI response
16	10 ms	250	10 sec	Optimal
16	1 ms	2000	10 sec	Optimal
16	0.01 ms	4250	10 sec	Slow
16	0 ms	4000	10 sec	Slow

The results show pretty similar results when changing the scaling factor for the sample rate, the result is a maximum performance of 475 samples a second. Based upon the fact that the sampled data is sent to the MAX3421E host controllers chip through SPI before being sent through the USB cable to the mobile phone. And that the prescaling performance tests were very similar, the limitation of maximum number of samples a second presumably lies in the SPI transfer between the MCU and the mobile phone and not the prescale factor or measurement rate of ADC readings. If the limitation of number of samplings lies in the SPI transfer then it would be an idea to optimize the AndroidAccessory library and the USB host shield library to get better performance.

Another issue was the slow Android GUI. When sampling polls were down to 0.01ms and below the Android GUI was slow. This could indicate that the Android phone was not fast enough to collect the data. But since the results recorded on the server application was so similar in the different tests and the data rates from the USB host shield is nowhere near the Hi-Speed USB 2.0 transfer of 480 Mbits, that the Nexus S supports, that the limitation presumably lies in the SPI transfer. This could also be because of the way Android handles its graphical display, as outlined in a forum contribution [2]. Further analysis is required based on these assumptions. The results so far is not sufficient to say exactly what causes are or aren't responsible for these results.

[2]http://forums.appleinsider.com/t/137684/former-google-intern-explains-why-ui-lag-occurs-more-often-in-android-than-ios, last accessed 25. Apr, 2012

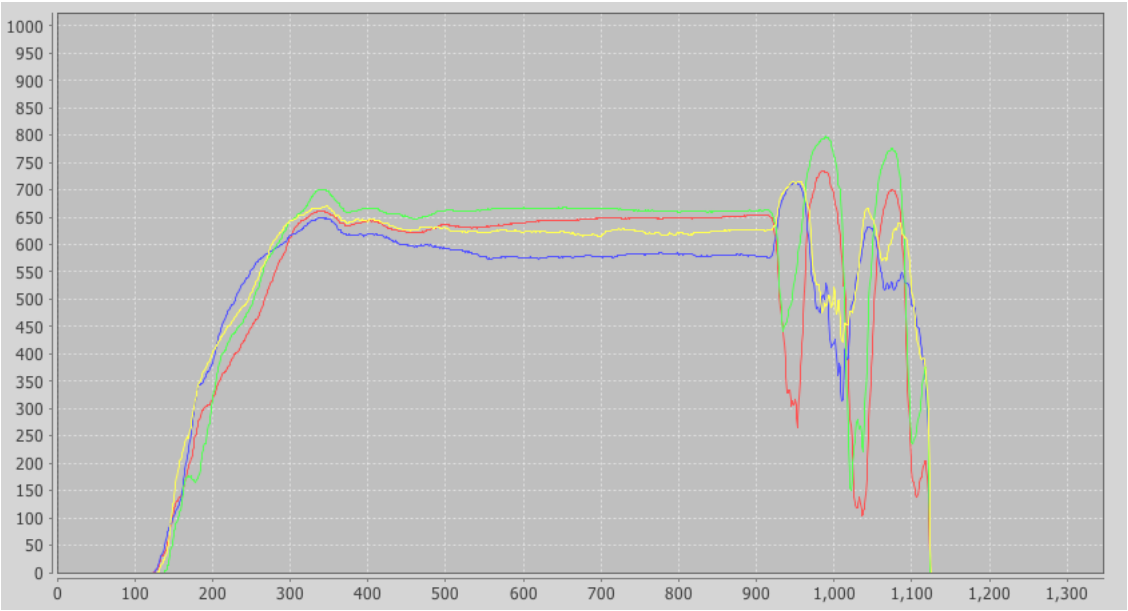

Figure 5.10: Test subject of 80 kg testing the system

Figure 5.10 shows data readings from a test subject of around 80 kg. The test subject is doing an imitational ski jump where none of the sensors flatten out at their max peak. One of the sensors reaches the value 800 of 1024 which indicate that the limitations for the sensors is not met. The cause for this may lie in the shoe insoles in which the sensors are mounted under. They are made out of shock absorbing material and may distribute the pressure under the soles in an unpredictable way.

Further analysis is required to prove the limitations of the sensors and how pressure is divided under the insoles which the sensors are mounted under.

5.2.4 Extensibility

This section looks at how much more could get out of the prototype system. The system could be extended by different components like, adding the number of sensors, making the sensors wireless, increase performance by dropping the GUI, design a custom PCB for the prototype or look at the SPI transfer. The highest priority of extending the project would be to make the the sensors wireless. The next priority would be to make a custom PCB for the prototype before looking at the SPI transfer.

Number of sensors

The number of sensors in the prototype system is only four, the minimum amount of sensors required to read the center of pressure of an athlete. It could prove useful to use up to 5 sensors for each foot, making it possible to create an image of how the pressure is divided onto the foot blade. There could also be an idea to maybe find sensors which can

give a full reading of the foot pressure, with many hundreds or thousands of points giving a detailed view of the pressure under the feet. This would require a complete redesign of the system, and it would likely become a different type of application. It would be a good application for measuring and making custom insoles data sheets.

Wireless sensors

The system could extend to use wireless sensors which would eliminate the use of wires and make a great contribution to the goal of making the technology transparent. This could provide the same application with more complexity thus extending the development time. This would also require a MCU, battery and a radio transmitter or a radio transceiver in each shoe sole.

Custom PCB

A custom PCB would also be a great contribution to making the technology more transparent. Making a custom PCB is one of the top priorities for further development because the size of the prototyping board is one of the main reasons why testing was not done in an outdoor jumping arena.

Different MCU

The system could easily work with another MCU as long as it fills all the requirements of ADC, USB host capabilities, cost and ease of programability. An assumption is that a different MCU could have different data transfer rate through the SPI interface which presumably might increase the sampling rate. The most logical replacement for the MCU prototyping board is the mbed which was evaluated for this project. The mbed is based on a much more powerful MCU that run with a 96 MHz clock compared to 16 MHz of the ATmega2560.

5.2.5 Evaluation summary

This summary looks at five different subjects of evaluation. This summary will provide two tables of sufficiency as to how sufficient the overall functionality of the system requirements and technical requirements are.

The degree of evaluation used are

Not good The requirement does not fulfill and are of high priority for further development

Moderate The requirement is only partly fulfilled and is prioritized for further development

Ok The requirement is fulfilled and improvements are done in further development

Quite good The requirement is fulfilled and performance is good

Excellent The requirement work excellent and high above expected

In addition to these degrees a plus (+) or minus (-) sign can be used to indicate if the degrees is not exact, that it approaches another degree.

Compliance table of system requirements

Degree	Mobility	Functionality	Performance	Extensibility
Not good				
Moderate	X		X	
Ok		X		
Quite good				X
Excellent				

Based on the table above it is easy to get a sense of the compliance of system requirements.

Mobility The mobility of the system is moderate indicating that the size and weight of the system is areas to improve in further analysis and development of the prototype.

Functionality The functionality of the prototype is ok indicating that the system does collect the center of pressure for an athlete and are able to send the data over internet.

Performance The performance of the system is moderate indicating that the expected sampling rate of the system was lower then measured although the system is functionally performing.

Extensibility The extensibility of the system is quite good indicating the system prototype is a good platform for further analysis, development and evaluation.

Compliance table of technical requirements

Degree	Sensors	MCU board	USB host	Sampling rate	Mobile phone
Not good					
Moderate	X		-	X	
Ok	+	X	X		
Quite good					X
Excellent					

Based on the table above it is easy to get a sense of the compliance of technological requirements.

Sensors The sensors are marked moderate+ indicating there are room for improvements. There was a hope they would have more performance headroom but they work sufficiently under testing.

MCU board The MCU is marked ok indicating that the performance, size, cost and ease of programmability was sufficient. There is almost a minus there because of size and sample rate but they are sufficient for a testing prototype.

USB host The USB host capabilities are marked ok- because there is an assumption that the sample rate is suffering from data transfer rate in the SPI from the MCU. But the test show that it works according to the requirements.

Sampling rate The sampling rate is marked moderate and if 100 $S\frac{m}{s}$ at high speed were required it would be marked not good. The initial calculations showed that 2600 samples a second were required to get 100 $S\frac{m}{s}$.

Mobile phone The mobile phone is marked quite good indicating that size, performance and functionality is good. There mobile phone is heavy with its 129 g but there is nothing that can be done to improve this unless the component is completely replaced.

The next chapter will finish with the thesis conclusion followed by thoughts around further development.

Chapter 6

Conclusion

This is the final chapter of this thesis and includes the conclusion as well as structure a foundation for further development. The conclusion is a look at the problem statement from chapter 1 reflected against the overall satisfaction of the thesis and prototype development.

6.1 Conclusion

The main objective of this thesis has been to gain a deeper theoretical knowledge about the human movement science technology. The first part of this thesis evaluates the technology needed in order to make measurement equipment for athletes with high mobility and technology transparency in mind. In this field of study there is close to zero knowledge because the wireless technology has emerged the last decade.

The second objective of this thesis were to establish a prototype that was measuring the human movement of an athlete. The technology evaluated have mobility and weight in consideration and consist of a mobile telephone, a MCU and force sensors. Part of the main objective was to develop the prototype and form a foundation to enhance the human movement science.

This thesis also evaluated and tests the limitations of the system, not only to meet the requirements outlined for the system but also in terms of what was possible to get out of the system. This lead to an evaluation of the system which showed that the systems initial functionality were satisfactory, but the sampling rate of the MCU block was not performing as expected based on analysis.

The measuring of an athletes center of pressure application shown in this thesis, illustrates how the technology can be utilized. Further development would provide more technology transparency to the system which would make it easier for the athlete to perform unaffected by the technology thus making the system more mobile and the technology more transparent.

6.2 Further development

This section discuss further development. This section is split in two separate subsections, commercial and block. The commercial development discuss possible uses for the technology in commercial applications and thoughts about bigger solutions. The block development talks about how to develop the prototype system further in order to get a more technology transparent system.

Commercial development

The technology describe in this thesis could be extended with different kinds of sensors thus making it possible to develop a wide range of applications. Application like other forms of human movement science, healthcare, monitoring systems, safety systems and consumer electronics[1].

In regards to an exercise platform it one could establish a database system with a web interface and other clients where performance could be followed over time. The data collected could be analyzed and evaluated to follow the development and the areas for improvement.

Block development

Further development of the blocks would lead to designing a custom PCB for the MCU block. This would reduce MCU block size by 80-90%.

The next development phase would also be to make the sensor block wireless which would also require a custom PCB with MCU and radio transceiver. This would introduce at least two more MCUs to the system, one transceiver for the main MCU block and two transceivers or transmitters for each of the sensor blocks. This will increased complexity and increase development time but it would make the system more technology transparent and easier to mount on equipment or athletes.

[1] http://www.sparkfun.com/news/578, last accessed 26. Apr, 2012

Appendix A

Programming the Arduino

Use Arduino IDE to load this code and program the Arduino MCU.

Listing A.1: Arduino firmware without USB and AndroidAccessory library code

```
#include <Max3421e.h>
#include <Usb.h>
#include <AndroidAccessory.h>

#define ONBOARD_LED  13

#define SENSOR1  A8
#define SENSOR2  A9
#define SENSOR3  A10
#define SENSOR4  A11

// Taunus is the current working title of the master thesis
   prototype
AndroidAccessory acc("Gard Sandholt",                  //
   manufacturer
                     "Taunus",                         // model
                     "Taunus Arduino Board",           //
                        description
                     "1.0",                            // version
                     "http://bash.gasamedia.com",      // uri
                     "0000000012345678");              // serial

void setup();
void loop();

void init_sensors() {
    pinMode(SENSOR1, INPUT);
    pinMode(SENSOR2, INPUT);
```

```
    pinMode(SENSOR3, INPUT);
    pinMode(SENSOR4, INPUT);
}

//uint16_t s1, s2, s3, s4;
void setup() {
    Serial.begin(115200);
    Serial.print("\r\nStart");

    init_sensors();
    acc.powerOn();
}

static byte count = 0;
void loop() {
    byte msg[3];

    if(acc.isConnected()) {
        if (digitalRead(ONBOARD_LED) == LOW) {
            digitalWrite(ONBOARD_LED, HIGH);
        }

        int len = acc.read(msg, sizeof(msg), 1);
        if (len > 0) {
          // received msg from Android
        }

        uint16_t val;
        switch(count++ % 0x4) {
            case 0x0:
                val = analogRead(SENSOR1);
                msg[0] = 0x1;
                msg[1] = val >> 8;
                msg[2] = val & 0xff;
                acc.write(msg, 3);
                break;

            case 0x1:
                val = analogRead(SENSOR2);
                msg[0] = 0x2;
                msg[1] = val >> 8;
                msg[2] = val & 0xff;
                acc.write(msg, 3);
                break;
```

```
            case 0x2:
                val = analogRead(SENSOR3);
                msg[0] = 0x3;
                msg[1] = val >> 8;
                msg[2] = val & 0xff;
                acc.write(msg, 3);
                break;

            case 0x3:
                val = analogRead(SENSOR4);
                msg[0] = 0x4;
                msg[1] = val >> 8;
                msg[2] = val & 0xff;
                acc.write(msg, 3);
                break;

        } // end switch

        count = count % 256;
    } else {
        // write default values to arduino
        digitalWrite(ONBOARD_LED, LOW);
    }

    delay(10);
}
```

Bibliography

[1] A. Jara, M. Zamora, and A. Skarmeta, "An architecture based on internet of things to support mobility and security in medical environments," in *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, pp. 1 –5, jan. 2010.

[2] M. Virmavirta and P. Koml, "Measurement of take-off forces in ski jumping," *Scandinavian journal of medicine & science in sports*, vol. 3, no. 4, pp. 229–236, 1993.

[3] A. Jeukendrup and A. Van Diemen, "Heart rate monitoring during training and competition in cyclists," *Journal of Sports Sciences*, vol. 16, no. S1, pp. 91–99, 1998.

[4] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[5] S. Babiel, U. Hartmann, P. Spitzenpfeil, and J. Mester, "Ground reaction forces in alpine skiing, cross-country skiing and ski jumping," *Science and skiing*, pp. 200–207, 1997.

[6] P. Jayaraman, A. Zaslavsky, and J. Delsing, "Sensor data collection using heterogeneous mobile devices," in *Pervasive Services, IEEE International Conference on*, pp. 161–164, IEEE, 2007.

[7] G. Ettema, S. Braten, and M. Bobbert, "Dynamics of the in-run in ski jumping: A simulation study," *Journal of Applied Biomechanics*, vol. 21, no. 3, p. 247, 2005.

[8] U. S. of Computer Science and Engineering, "General AVR info." `http://www.cse.unsw.edu.au/~pcb/avr/avr.html`, 2011. [Online; accessed 12-September-2011].

[9] Atmel, "8-bit Atmel Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash." `http://www.atmel.com/dyn/resources/prod_documents/2549S.pdf`, 2011. [Online; accessed 07-September-2011].

[10] H. Lekatsas, J. Henkel, and W. Wolf, "Code compression for low power embedded system design," in *Proceedings of the 37th Annual Design Automation Conference*, pp. 294–299, ACM, 2000.

[11] Arduino, "Arduino - ArduinoBoardMega2560." `http://arduino.cc/en/Main/ArduinoBoardMega2560`, 2011. [Online; accessed 07-September-2011].

[12] G. Inc., "What is Android? | Android Developers." http://developer.android.com/guide/basics/what-is-android.html, 2011. [Online; accessed 15-September-2011].

[13] Circuits@Home, "USB Host Shield 2.0 for Arduino." http://www.circuitsathome.com/products-page/arduino-shields/usb-host-shield-2-0-for-arduino/, 2011. [Online; accessed 26-September-2011].

[14] R. A. Kirkman, "Evaluating single point of failures for safety reliability," *Reliability, IEEE Transactions on*, vol. R-28, pp. 259–263, aug. 1979.

[15] Atmel, "Characterization and Calibration of the ADC on an AVR." http://www.atmel.com/dyn/resources/prod_documents/DOC2559.PDF, 2012. [Online; accessed 22-April-2012].

[16] Arduino, "Arduino - ArduinoBoardADK." http://arduino.cc/en/Main/ArduinoBoardADK, 2012. [Online; accessed 20-February-2012].