Cellular Devices In Mechatronic Applications

An evaluation from a sensor and communication perspective

DAVID BYSTRÖM NICLAS KEMPE

Department of Machine Design

MMK 2012:48 MDA 435
Masters' Degree Project
Stockholm, Sweden June 2012

Examensarbete MMK 2012:48 MDA 435

Mobiltelefoner i mekatroniska tillämpningar: En utvärdering från ett sensor och kommunikationsperspektiv

Godkänt	Examinator	Handledare
2012-06-29	Jan Wikander	Mats Hanson, KTH
	Uppdragsgivare	Kontaktperson
	ÅF AB	Peter Karlström, ÅF

Sammanfattning

En mobiltelefon är för många ett oumbärligt tillbehör som underlättar vårt moderna leverne. Med hjälp av den kan vi lätt få kontakt med vänner och bekanta via sociala nätverk eller så kan den användas som en portabel GPS, redo att användas när som helst nästan överallt på jorden. Dessutom är den för många synonym med en underhållningplattform. Man kan utan att överdriva säga att moderna s.k. smartphones är mycket mer avancerade och fokuserar mer på användbarhet i vardagen än deras föregångare gjorde. För att kunna åstadkomma denna ökade funktionalitet innehåller moderna smartphones en stor uppsättning hårdvarumoduler samt mjukvara som på olika sätt förstärker upplevelsen för användaren. Denna avhandling har som mål att utvärdera hur sensor- samt kommunikationshårdvaran i en modern mobiltelefon påverkar möjligheten att använda mobiltelefoner i mekatroniska tillämpningar.

I uppsatsen ingår en sammanställning över avhandlingar som på ett eller annat sätt använder sig av mobiltelefoner i mekatroniska tillämpningar vilken används som diskussionsunderlag när huvudfrågan utvärderas.

En utvärdering över eventuella trender i sensorprestanda hos mobiltelefoner utförs. Detta görs genom att sammanställa och kartlägga en rad olika datablad för sensorer som kan finnas i mobiltelefoner. Dessutom utförs en del tester med avsikt att undersöka avvikelser i samplingsfrekvens hos en accelerometer, ett gyroskop samt en magnetometer i en Android-baserad mobiltelefon.

Ett mobiltelefonbaserat säkerhetssystem för motocrossförare kallat “Crossafe” utvecklades för att dels stärka den teoretiska analysen samt för att förkovra värdefulla insikter i en ingenjörsmässig utvecklingsprocess som använder mobiltelefoner i mekatroniska tillämpningar.

Med den teoretiska analysen samt utfallet av säkerhetssystemet som bakgrund drogs slutsatsen att telefoner är lämpliga i mekatroniska applikationer med inte allt för högt ställda krav på prestanda och specifik funktionalitet. Det skall tilläggas att funktionaliteten dock kan utökas genom kommunikation med externa system genom telefonens kommunikationsteknologier.

Master of Science Thesis MMK 2012:48 MDA 435

Cellular Devices In Mechatronic Applications: An evaluation from a sensor and communication perspective

Approved	Examiner	Supervisor
2012-06-29	Jan Wikander	Mats Hanson, KTH
	Commissioner	Contact person
	ÅF AB	Peter Karlström, ÅF

Abstract

For most of us, a cell phone is a must-have gadget that makes our modern living a lot easier. It helps us stay in touch with friends and family via social networks, it can be used as a handheld GPS ready to use on-the-fly almost everywhere on the planet or as an entertainment platform allowing us to capture photos, playing games or listen to our favorite music. To say the least, modern smartphones are far more advanced and focuses more on everyday usability than their predecessors did only a couple of years ago. To be able to achieve all this functionality, modern cell phones are packed with hardware and software that enhances the end user experience in a number of different ways. This thesis aims to evaluate how the sensor and communication hardware in modern cell phones affects the feasibility of using cell phones in mechatronic applications.

A compilation of previous work that utilizes cell phones in various mechatronic constellations is made and used as a reference when assessing the main question of this thesis.

An evaluation of trends in cell phone sensor performance was made. This was done by compiling and mapping a number of different cell phones and their corresponding sensor data sheets after which conclusions regarding general trends in sensor performance is drawn. Further, testing evaluating variations in the sampling rate of the accelerometer, gyroscope and magnetometer in an Android based cell phone is performed.

A cell phone based safety system for motocross drivers named “Crossafe” is developed. From this, valuable insights could be acquired regarding different aspects of an engineering development process that utilizes cell phones in mechatronic applications.

Based on the theoretical evaluation and the results associated with the developed safety system, the cell phones was found to be suitable in mechatronic systems where requirements on performance and specific functionality are moderate. It shall be pointed out that the functionality can be extended by external systems communicating through the cell phones communication-interfaces.

Nomenclature

Notations

Symbols	Description
G	Gravitational unit, $1G = 9.82\ \frac{m}{s^2}$ at Stockholm, Sweden

Abbreviations

AAM	Active Appearance Model
AES	Advanced Encryption Standard
AHF	Adaptive High Frequency
ALS	Ambient Light Sensor
API	Application Programming Interface
BSI	Backside Illuminated
C2DM	Cloud to Device Messaging
CMOS	Complementary Metal Oxide Semiconductor
DSL	Digital Subscriber Line
DSSS	Direct Sequence Spread Spectrum
DTN	Delay Tolerant Network
EDGE	Enhanced Data Rates for GSM Revolution
EDR	Enhanced Data Rate
EME	Electro Mechanical Environment
GAE	Google App Engine
GPRS	General Packet Radio Services
GSM	Groupe Spéciale Mobile
HTTP	Hypertext Transfer Protocol
ITU	The International Telecommunication Union
JDO	Java Data Object
LAN	Local Area Network
NMT	Nordic Mobile Telephony
NFC	Near Field Communication
OS	Operation System
PDA	Personal Digital Assistant
PDF	Probability Density Function
PSTN	Public Switched Telephone Network
RFID	Radio Frequency Identification
RTOS	Real Time Operating System
SSP	Secure Simple Pairing
UCS	Universal Charging Solution
UWB	Ultra Wide Band
VTL	Virtual Trip Lines
WAP	Wireless Application Protocol
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network

Foreword

This thesis was done under supervision of Mats Hanson, KTH and Fredrik Mistander, ÅF. We would like to thank you for your patience, time and efforts in leading us towards the end goal, you were always willing to meet and discuss our problems regarding the project and to guide us in the right direction. We would also like to thank Peter Karlström, manager at software design, ÅF for his high ambition and will to support and help us during the project. Also Tor Ericson, manager system design, ÅF for founding the thesis project, without him this would not have been turned into reality. Last but not least we would like to thank Jan-Olof Svärd, federal physician at SVEMO for giving us information about Motocross and Enduro in general but also for giving us a more detailed picture of how accidents happen and what causes them.

David Byström & Niclas Kempe

Stockholm
June 2012

Contents

1 Introduction

1.1 Background

The number of mobile devices in the world has increased steadily the last couple of years. According to IDC (Feb 2012) approximately 1.5 billion handsets was sold in 2011 which is a 11.1% growth compared to 2010 which is also confirmed by mobileThinking strategy analytics (Feb 2012)[1]. Among these devices, approximately 32% of the shipped units was smartphones.[1] With the increasing market share of smartphones, the technical width of the hardware in cell phones has also increased significantly during the last decade. From solely a portable phone in the beginning of the 21st century, the modern cell phone has become so much more than just a communication device. In the late 2000s, the term "smartphone" quickly arose in cell phone commercials, further emphasizing the new, modern era of cell phones. Although a smartphone is regarded as something modern, the history behind the term actually began some twenty years ago, back in the nineties.

In the beginning of the 1990s the market for handheld devices was clearly disjuncted by so called Personal Digital Assistants (PDA) and common cell phones. PDAs provided a handheld platform with various organizational functions such as calendars, schedules and memos but also the ability to utilize local wireless networking and Internet browsing. Perhaps even more interesting was the fact that they interacted with the user via a touchscreen. The distinction between PDAs and cell phones would become more and more vague as the 1990s progressed; several cell phone manufacturers started to incorporate PDA functionality into their phones and vice versa.

In 1993, BellSouth and IBM announced the IBM Simon which according to a reviewer "looks and acts like a cellular phone but offers much more than voice communication" [2]. Cell phones in the beginning of the 90s are usually referred to as large bricks with small displays, scarce functionality and mechanic buttons. With its 36 x 115 mm LCD touch display and a number of pre-installed applications, Simon challenged all these conventions and steered the evolution of cell phones in a completely new direction.[3] The vision behind the IBM Simon was extremely progressive in the nineties and is up to date even today; develop usability for the end user. Simon was the first cell phone to incorporate lots of functionality originally found in PDAs and although the term "smartphone" was not yet popularised, the IBM Simon is a strong candidate for the epithet - "the worlds first smartphone".

The interesting observation during the 90s is that the cell phones began to be so much more than just a portable communication device; they became smarter and focused more on everyday usability for the user. Starting with the IBM Simon, some cell phones and PDAs that was manufactured during the nineties

[1] `http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats` Accessed 2012-06-04

[2] `http://research.microsoft.com/en-us/um/people/bibuxton/buxtoncollection/a/pdf/press%20release%201993.pdf` Accessed 2012-05-10

[3] `http://research.microsoft.com/en-us/um/people/bibuxton/buxtoncollection/a/pdf/simon%20review.pdf` Accessed 2012-05-10

can be seen as early predecessors to the modern smartphone. However, it would take the cell phone industry nearly a decade to introduce the term "smartphone" and a few more years to popularise the term.

In 2000 Ericsson launched the R380[4]. the first cell phone ever to be marketed as a smartphone. The functionality is similar to that in IBM Simon however the R380 being significantly faster and smaller. Further, the R380 features a built in modem and a WAP system for Internet access, a big difference from Simons PCMCIA slot and a giant leap in the direction towards modern smartphones.

Today, the increasing technical width of smartphones seems to know no limit. Manufacturers incorporates technologies that seemed futuristic when the R380 entered the market. In addition to the high tech hardware, sophisticated and easy-to-use operating systems makes it very easy to program applications that can run on virtually any cell phone using that operating system. For more advanced and optimized applications one can program even deeper down in the abstraction layers using native programming languages.

All of this makes cell phones a potentially powerful building block in various system configurations. However, qualitative research evaluating the potential of cell phones in mechatronic systems is scarce. Such a research could increase the interest of their potential, possibly giving rise to even more sophisticated cell phone based mechatronic applications than seen today. In 2011 a number of tragical accidents occurred in various motorcycle disciplines in Sweden. As preventive measures to future accidents, the Swedish motorcycle and snowmobile association (SVEMO) initiated a series of projects to increase the safety across the field of motorcycle sports.[5] Together with SVEMO the idea to implement a safety system on a cell phone was introduced and further realized in this master thesis.

1.2 Purpose

The main purpose with this thesis is to draw conclusions regarding cell phones' feasibility in mechatronic applications which is achieved by analyzing different technical aspects of modern cell phones such as their processors, sensors and communication potentials.

To gain valuable insights regarding the product development process of a cell phone application with requirements on sensor signal processing and communication infrastructure, an additional purpose of the thesis is to develop a smartphone based safety system to use in motorcycle related activities such as enduro. The system is to be implemented on one or more cell phones in the form of an application and aims to reduce the risk of accidents and to alarm external actors in the event of an accident.

[4] `http://www.ericsson.com/res/investors/docs/annual-reports-1970-2002/annual2000_understanding_en.pdf` Accessed 2012-04-01

[5] `http://www.svemo.se/sv/Sakerhetsgruppen/` Accessed 2012-04-01

1.3 Delimitations

As the main task of the thesis is rather elusive and the strategy with which to break down the problem can be pretty hard to grasp at a first glance, it was crucial that the working process was regulated with some delimitations.

The methodology followed throughout the main analysis was tailored in the early stages of the project such that it would fit the projects time budget. It was found ineligible to perform an exhaustive analysis due to the time constraints. This made the chosen methodology prone to delimitations. These limitations are obvious as the methodology is not capable of concerning every aspect there is to the problem which would be a non trivial task due to the complexity of the main question. When estimating general trends in the hardware performance of cell phones from theory, no limitations or exclusions with regards to operating system or manufacturer was intended to be made.

All programming has been performed on Android version 2.2, API level 8. This includes all the tests and all Android programming made in Crossafe. Any testing presented in the thesis has been programmed using standard Android API's. No programming has been made in Android native c/c++. Further, the conclusions drawn in this thesis doesn't reflect upon any possible effects that programming in native c/c++ could give rise to.

When analyzing the performance of the cell phones via testing, this was done exclusively on a HTC Sensation XE.

Regarding the developed cell phone application Crossafe, one major delimitation has been that it is to be seen as a prototype of a motorcycle safety system and by no means a commercial product. This delimitation saves valuable developing time as less effort has to be invested into testing the application. The goal however is to continue the development of the application when the thesis is complete and hopefully commercialize it in the future.

Crossafe was exclusively programmed using standard Android API's.

When developing the safety system application Crossafe, the software is exclusively written for and on Android based devices. If commercialized in the future, the goal is to port the application to other operating systems as well.

1.4 Method

The methodology used in this thesis can primarily be described as an evaluation of different trends in the hardware of cell phones, mainly assessing their sensor and communication potentials in mechatronic applications. Some testing oriented research is performed as to back up the more holistic and qualitative discussions performed continuously throughout this paper.

At first, a compilation of previous work that utilizes cell phones in various applications was performed. The results from this evaluation was used as a frame of reference when conclusions regarding cell phones feasibility in mechatronic applications was made.

An evaluation of trends in cell phone sensor performance was to be made. This is obtained by compiling and mapping a number of different cell phones and their corresponding sensor data sheets after which conclusions regarding general trends in sensor performance is drawn. To expand the width and reliability of the analysis, some minor testing regarding variations in sampling rate was written and executed on the accelerometer, gyroscope and magnetometer of a HTC Sensation XE.

A qualitative analysis of the different communication technologies supported by cell phones was to be done, the results of the analysis are then compared to how these technologies are used in mechatronic applications. By examining the differences and similarities in what functionality is supported it is possible to deduce in what ways a smartphone can be used. The possibility to extend the functionality of a smartphone with ones that are not supported is assessed to easier conclude the usability and limitation of the smartphones in mechatronic applications.

A motorcycle safety system was developed and realized in the form of an Android cell phone app. The results from this development process was used to further investigate and assess the possible use of cell phones in mechatronic applications.

The results are mainly presented in the form of discussions, assessing the most important conclusions that has been made during the course of the entire project. Furthermore, some interesting notes are made in the running text of this paper that are not considered as direct results but which should still be interesting when assessing the main question.

2 Frame of reference

When assessing the feasibility of mobile devices as sub parts or replacements of mechatronic systems, previous work will be considered to support the conclusions regarding their appropriate fields of use. Existing work within the subject is meager, though some publications were considered to be useful during this research. The different technologies found within modern cell phones has been covered by these articles, and in addition some minor evaluation of the performance and applications of both sensor- and different communication technologies has been made in these articles. However, no general assessment of the feasibility or suitability of cell phones in mechatronic applications has been made. By reading these articles, the probable use for mobile devices can be narrowed down to some suitable applications.

2.1 Related work regarding communication through mobile devices

2.1.1 Overhead hazard, construction worker tracking

In article [1] a proactive safety system is developed to prevent accidents on construction sites with the main focus on overhead hazards. The safety system is implemented through tracking in a three dimensional coordinate system. Several technologies that can be used for this has been evaluated and documented by others referred to in the paper, some of them are; RFID, WLAN, UWB and Indoor GPS. The suitability of each communication technology has been assessed. The authors in [1] utilized Ultra Wide Band communication, and a solution with virtual fencing around the hazardous areas. By using an algorithm based on the Jordan curve theorem it is possible to deduce whether a construction worker is inside an area circumvented by this virtual fence and thereby in danger. According to the paper, UWB was used since it performed well enough for indoor use and excellent outdoors where the line of sight between the tracked tags on the construction material and the receivers were unobstructed. The systems intended use was in the earlier stages of construction work, where open spaces are more common. The precision reached was generally around a few decimeters and in rare cases around 1 m.

2.1.2 Wireless sensor deployment for collaborative sensing with mobile phones

The authors of [2] have analyzed how deployment of wireless sensors can be done in an efficient and economic way. Wireless sensor networks (WSN) are used for monitoring physical or environmental conditions. When using WSN statistical reliability is in strong correlation to the amount of sensors used, this leads to a substantial cost when implementing large systems. One way of reducing costs is to utilize "Urban sensing", this paradigm has been developed for large scale sensing using communication infrastructure readily available. It is closely related to participatory sensing where users can choose to participate and if so, contribute to the data gathering by using their mobile phones or other sensor applications. [2] defines a new framework for minimizing wireless sensor nodes while maximizing the usage of participatory sensing without reducing the sensing quality. Mathematical models is used for determining sensing quality and for coping with the dynamic behavior of mobile sensing. Some of the related work of [2] consider usage of opportunistic networks but also DTNs - Delay tolerant networks.

The authors of [2] also discuss the fact that most wireless sensors communicate via IEEE 802.15.4 standard (ZigBee etc) while mobile devices use 802.11 (WLAN) instead. A new architecture is assessed to accommodate the need for a bridge between the two standards. Other designs have utilized the fact that most sensors can be interfaced towards USB and then wirelessly connected to the mobile phones through BlueTooth adapters, though this is inconvenient and impractical when using Urban sensing. There are some additional properties with mobile phones to consider when using them in WSN's, sensing capabilities do

vary between brands and models. Obstacles in the environment such as hills, buildings and other signal blockers make connection intermittent. A mathematical model is used to predict the probability of a target device being detected by the node network.

2.1.3 Evaluation of traffic data obtained via GPS enabled mobile phones: The Mobile Century field experiment

Mobile sensing has been used within traffic monitoring for a couple of years as discussed by Juan C. Herrera et al. [3]. Before the smartphone boom, traffic monitoring was done by different techniques, the most common ones where by video cameras or RFID. Monitoring through video cameras was normally done through image recognition of registration plates, if two successful readings at two separate stations were done, the speed of the vehicle could be calculated. The other way, by RFID, is done in a similar way with stations registering the RFID transponder as it passes by. Both techniques are quite expensive to implement, especially in developing countries where resources for traffic monitoring is scarce. With the increasing usage of mobile devices, especially in developing countries, they are more than suitable for solving this issue, today basically every smartphone utilizes the GPS for positioning and navigation. When the article was written, experiments and testing had been done with success.

Today traffic monitoring through mobile devices are in common use, Google maps traffic feature is based on participatory sensing, where everyone willing to, can send their position and speed history automatically. The information is then presented live on Google maps with colored roads from green to red depending on traffic intensity. According to Juan C. Herrera et al. [3] the penetration rate[6] of mobile phones in the world is at 50%, where in developing countries the annual growth rate is greater than 30%, this was in 2008. According to an article[7] produced by ITU - The UN agency for information and communication technologies, produced in the end of 2011, 87% of all people around the world were using cell phones, India and China alone accounts for 30% of the global usage, this guarantees a bright future for mobile sensing. Before GPS's were common in cell phones, monitoring traffic was done with a triangulation technique through communication with cell signal towers [3]. Though this technique proved to inaccurate in comparison to GPS, especially when doing small scale tracking as with cars and traffic. The downside of tracking by cell phones is the privacy issue, even though information is encrypted it is still possible to distinguish a single ID and its movements according to Juan C. Herrera et al. [3].

The tracking of mobile devices via GPS can be done in two ways, either "Temporal Sampling" is used, this means that the mobile device transmits its data at even time intervals. The other way is by "Spatial Sampling" which means that the device sends its information at certain points of interest's or checkpoints, similar to a RFID transponder system, this method is more safe in terms of privacy since communication is only done when the phone is close to a checkpoint.

[6] Penetration rate is the proportion of mobile users in the world.

[7] `http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf` Accessed 2012-06-04

These checkpoints are called VTL's (Virtual trip lines).

Advantages of a GPS based system is cheap maintenance- and setup costs, according to the results in the article it is more accurate than solutions based on VTL's or loop detectors. Reliability and accuracy will most likely increase as the GPS technology advances in the future. The accuracy was proven to be increased with support from a loop detector system (RFID) but that also put demands on an infrastructure that can support the technology. The information acquired by the loop detector system and VTL's yield different results. Juan C. Herrera et al believe it is because of the way velocity is calculated in the both techniques, according to the authors; VTL-results are more likely to stay within one standard deviation of the mean velocity. As Juan C. Herrera et al [3] states, a traffic information system based on GPS need inverse modeling (modeling based on estimation theory) and data assimilation algorithms of its data sets to reduce potential errors and biased data.

Another important source of information are the specifications of modern phones, a comparison between different brands and price ranges is used to deduce general support with respect to sensor performance and communication versatility.

2.2 Related work regarding sensors in cell phones

One key to the increasing functionality in cell phones during the last couple of years is different needs to observe and sense the environment around the cell phone. Manufacturers started to incorporate different kinds of sensors in the cell phones to achieve new, innovative ways to interact with the users such as the popular feature "tilt-to-landscape-view" often seen in cell phones today.

Modern cell phones features an arsenal of different sensors, the most obvious being the microphone. Backside illuminated (BSI) image sensors makes it possible to capture high resolution photos and videos, even in low light conditions. Accelerometers provides a way to measure the accelerations that a cell phone is subdued to, making gesture based functionality such as "tilt-to-landscape-view" possible. Gyroscopes further improves the quality of motion interpretations, increasing responsiveness as well as reducing artifacts. Moreover the combination accelerometer/gyroscope makes it possible to obtain a 6-axis view of the cell phone's movement/orientation. Ambient Light Sensors (ALS) reduces power consumption as it makes it possible to adjust the screen light intensity according to the ambient light. Magnetometers can measure the direction and/or strength of a magnetic field and is often utilized in electronic compasses. GPS receivers triangulates the cell phone's position globally. Pressure sensors measure the atmospheric pressure which in turn gives information about the cell phone's altitude.

Previous work that discuss the use of cell phone sensors in mechatronic applications is scarce but there are some articles that was found to be of interest in this study. A lot of the work seems to focus on only one sensor, often suppressing the possible synergy effects of dealing with a platform that offers so much more.

Below follows a few short summaries of the articles that was found to be most related to our field of work and from which inspiration as well as new insights was collected. Each of these summaries is followed by a small discussion that enlightens why the articles was found to be of interest.

2.2.1 Activity Recognition using Cell Phone Accelerometers

The possibility to use cell phone accelerometers in more sophisticated applications is proposed by Kwapisz, Weiss and Moore [4]. In their article, human activity recognition using raw data from a 3-axis cell phone accelerometer placed in a person's leg pocket is tested and verified. Six different activities were to be recognized; walking, jogging, ascending stairs, descending stairs, sitting, and standing. To do this, data sets consisting of 10 second (200 sampling points) samples from 29 persons was collected and analyzed using three different classifications techniques: decision trees (J48), logistic regression and multilayer neural networks. By choosing the technique that induced the prediction model with the highest accuracy, Kwapisz, Weiss and Moore manages to achieve accuracies of over 90% for every activity except ascending/descending stairs which only reached about 50-60% accuracy.

In article [4] the authors effectively shows that it is possible to use accelerometers mounted in a cell phone to capture data sets of sufficient quality to be able to distinguish similar activities like "walking" and "jogging" from each other, using a mere sampling frequency of 20Hz. It is not specified whether the calculations for the different classification techniques was made by the cell phones themselves or by external processing power. However, judging by the overall methodology in their article, the calculations seems to have been made after the actual data collection, not in real time or with any requirements on timing which suggest that the calculations at least could be done by the cell phone.

2.2.2 Evaluation of Gyroscope-embedded Mobile Phones

In the latter years, manufacturers like HTC, Samsung, Apple and Nokia has begun to incorporate gyroscopes in their cell phones [8] [9] [10] [11]. Before that, the orientation of a phone was determined from accelerometer (providing pitch and roll angles) or accelerometer+magnetometer (providing yaw, pitch and roll angles) data. By fusing an accelerometer with a magnetometer the yaw angle can be determined by the direction of the magnetic field of the earth. However, a magnetometer is very sensitive to local noises or disturbances in the gravitational field and shall therefore not be considered fail safe. Additionally, accelerometers are error prone when used in angle measurements. In order to reduce these errors and obtain a more robust model for the angular movements of a cell phone, gyroscopes can be used as they actually measure angular velocity.

[8] `www.htc.com` Accessed 2012-06-15
[9] `www.samsung.com/us/mobile/cell-phones` Accessed 2012-06-15
[10] `www.apple.com` Accessed 2012-06-15
[11] `www.nokia.com` Accessed 2012-06-15

In the article by Barthold, Pathapati Subbu and Dantu [5] the possibility to use gyroscopes alone rather than the traditional accelerometer/magnetometer fusion is discussed. Problematics in the traditional approach is mentioned and experiments using the gyroscope in a Samsung Galaxy Nexus S is carried out. They show that the phones orientation can be determined at any time, even in magnetically interfered areas and when the phone is accelerating, using only a built-in gyroscope. Further, they performed some 15 experiments to determine the drift of the gyroscope which was found to be approx. $\pm 1^{\circ}/sec$. The drift of a gyroscope can be defined as the accumulating error when integrating the angular velocity signal to an angular signal, for a more detailed description please refer to section 4.6.1 - Sensor terminology.

In their article [5], Barthold, Pathapati Subbu and Dantu specifies the drift of the gyroscope to be $\pm 1^{\circ}/sec$ [5, p.1636], numbers that can be considered extremely high for a gyroscope as it would actually yield an angle with en error of approximately 180° after only 3 minutes. There are some possible reasons for this result. Firstly, the native drift compensation model for the gyroscope may be too naive, ignoring important factors that affects the drift. Secondly, the implementation of the gyroscope on the phone may be of poor quality; perhaps some guidelines from the gyroscope manufacturer has been ignored by the cell phone manufacturer. In any case, the interesting thing for our study is that their article indicates that the raw gyroscope signal from the cell phone could require additional processor power in terms of filtering models in order to reduce the drift. Furthermore, these filter models has to be obtained by other studies that characterizes the drift according to some parameters, similar to the procedure utilized by Barthold, Pathapati Subbu and Dantu. If the methodology used in their article is of sufficient scientific adequacy, and if the result that the proposed drift $\pm 1^{\circ}/sec$ can be generalized to other cell phones as well, this could mean that using the built-in gyroscopes to predict the angles of the orientation of a cell phones without more sophisticated filtering should be done cautiously.

2.2.3 Mobile phone platform as portable chemical analyzer

Cameras practically have become a mandatory part in cell phones during the last decade, especially during the last couple of years with the exploding popularity of social medias, allowing the users to share photos and videos on the fly. Up until recently the image sensors in cell phones have been of poor quality with high noise sensitivity and bad performance in low light conditions.

Antonio Garcia et al. [6] suggests an unusual application for mobile phone cameras when using one in a platform for chemical analysis. The platform utilizes a cell phone camera to capture images of a colorimetric chemical sensor in order to obtain certain characteristics that reflects the potassium concentration of the solution under study. They use a Nokia N73 cell phone which features a 3.2 megapixel CMOS image sensor. The images was subdued to a set of modified image processing operations, executed on the cell phone, in order to preserve battery life as well as to reduce the computation time. The results was found to be slightly worse than a reference platform but still better than many other platforms that wasn't implemented on a cell phone. Based on this, the platform based on cell phone cameras was found to be comparable

to conventional procedures based on image processing on off line computers according to Garcia et al.

Garcia et al. found that the 3.2 megapixels CMOS image sensor featured on the Nokia N73 was of sufficient quality to capture images processed directly on the cell phone to achieve results that was comparable to conventional methods. It is notable that this was possible using a sensor of a mere 3.2 megapixels when most modern smartphones features sensors with 5-8 or even 41 megapixels.[12] This suggest that modern cell phones should be able to be used in even more sophisticated soft real time image processing applications. Garcia et al. enlightens the very interesting concept of using a cell phone as an advanced stand alone system, utilizing some of its capability. This makes their article interesting to consider in our study.

2.2.4 Real-Time Facial Feature Tracking on a Mobile Device

Tresadern, Ionita and Coote [7] uses a cell phone to realize real time facial feature tracking in their article. This is done using the Nokia N900 which features a 5 megapixel image sensor and a processor running at 600MHz.[13] Through a modified energy efficient implementation of the AAM (Active Appearance Model) using a Haar-like feature basis, they manage to achieve performance similar to conventional methods. Further, Tresadern, Ionita and Coote examines the performance of some different models regarding accuracy and processor efficiency to tweak their model to be more suitable to use on mobile platforms.

Tresadern, Ionita and Coote shows that the front camera and processing power of a Nokia N900 is sufficient to use in a stand alone system for face recognition in real time. Taking into consideration that the processors in modern cell phones is substantially faster it should be possible to, given the same image frame size, implement even more advanced algorithms than proposed in their article without any loss in performance. A note should be made regarding the image frame size that was used in their article, it is specified that the front camera was used with a frame rate of about 25-30 frames per second[[7] , p. 287]. It it not specified how large the image frames were but other sources state that the front camera captures images of size 640x480. We think that this result further strengthens the cell phones position as a powerful mechatronic building block.

2.3 Android Introduction

Android is a software platform that includes an operating system and is mainly aimed towards handheld cellular devices such as smartphones and tablets. Android was announced in late 2007[14] as an open platform for mobile devices. In 2008, the first commercially used Android version 1.0 [15] was released and since

[12] `http://www.extremetech.com/extreme/119961-nokia-unveils-41-megapixel-808-pureview-smartphone-threatens-digital-camera-revolution` Accessed 2012-04-05

[13] `http://en.wikipedia.org/wiki/Nokia_N900#Processors` Accessed 2012-03-15

[14] `http://www.openhandsetalliance.com/press_110507.html` Accessed 2012-06-10

[15] `http://android-developers.blogspot.se/2008/09/announcing-android-10-sdk-release-1.html` Accessed 2012-06-10

then, several updated versions has been released. Each newer version contains bug fixes and added functionality and software tools. The most current Android version is the 4.0.4 version which was released in March 2012[16] . Today, the most common Android version is the 2.3 Gingerbread which is distributed in 65% of all Android devices[17].

The creation of an Android application starts by downloading the Android Software Development Kit (SDK) which contains the complete set of software libraries that can be utilized in the Android operating system. The system architecture of Android consists of a number of abstraction layers, all the way down from the hardware up to built in applications like the calendar or the home screen. In Figure 1, the major components that make up the Android operating system can be seen. Android uses a modified version of the open source Linux kernel which is used for core system services such as memory and process management, network stack and hardware drivers. This is the most distinguished low level layer in Android and serves as the first abstraction layer between the actual hardware and the rest of the software.

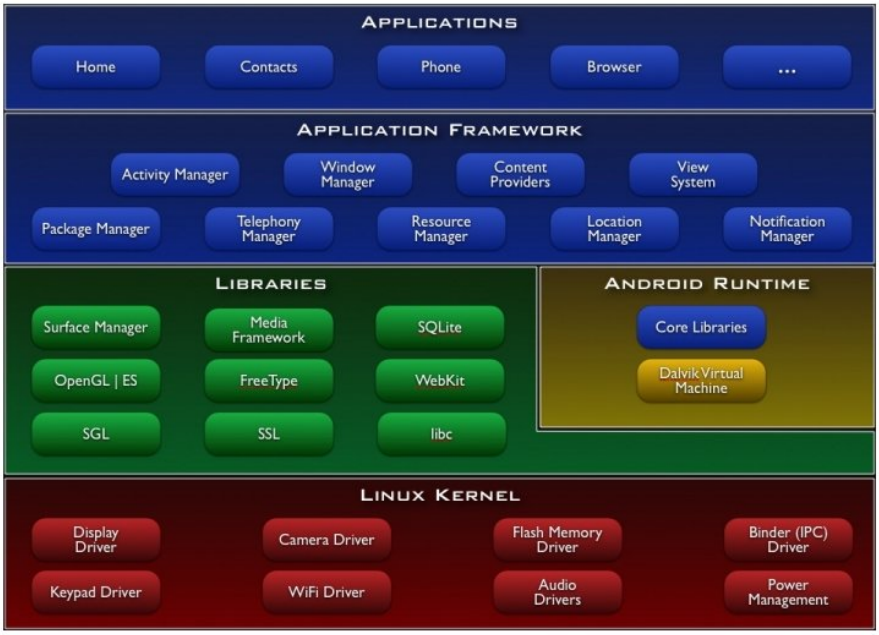


Figure 1: The major components of the Android system architecture.

The Android Runtime module consists of a set of core libraries that provides the functionality offered by standard Java libraries. To be able to run applications compiled by a Java language compiler, Android features a process virtual machine called the Dalvik Virtual Machine (DVM). The Dalvik Virtual Machine

[16] `http://www.theinquirer.net/inquirer/news/2164732/google-announces-android-404-gsm-nexus-galaxy-nexus-phones` Accessed 2012-06-10

[17] `http://developer.android.com/resources/dashboard/platform-versions.html` Accessed 2012-06-10

can run code that has been compiled by a Java language compiler and converted into the Dalvik Executable (.dex) format. This format is designed and optimized for systems that are constrained in terms of memory and processor speed. Each application in Android uses its own instance of the DVM.

Android features a set of core C/C++ libraries that is used by various parts of the system. Some examples of these are different graphic engine or media playback/recording libraries. These libraries expands the functionality imposed by the core libraries and are exposed to the developer via the Application Framework.

The Application Framework provides the developer with all the tools needed to access standard Android functionality. These standard APIs can be seen as the developers window down to the core functionality of Android. This framework is designed such that an application can communicate with other applications and reuse some of their functionality which promotes modularity of the applications. One example of a member of this abstraction layer is the SensorManager which grants access to the device's sensors. Another example is the Activity Manager which manages the lifecycle of applications and provides the navigation backstack.

At the top of the software layers are the native applications such as the calendar, the browser, the email client or the browser. These are all coded in Java and uses the same application framework available to the developers.

2.3.1 Android Native Development Kit NDK

Because of the abstraction layers, it is easy for a developer to get started when programming an application. However, these layers together with the Android application security model makes it hard for a developer to perform programming deeper down in the abstractions layers. The developer is bound to the functionality offered in the Application Framework layer when programming in Java.

For those edge cases when the standard API framework doesn't suffice, Android provides the Native Development Kit (NDK) which is a toolset that allows developers to embed components that makes use of native code in their applications. The Android NDK makes it possible to implement code written in native code languages like C and C++. This can yield a number of positive side effects to certain applications like increased performance. Note however that it is not guaranteed that an application becomes faster automatically by the use of the NDK. Coding in native C/C++ increases the complexity of the application significantly which should be taken into consideration when choosing whether or not to use native code.

Functionality that can typically benefit from being coded in native C/C++ are CPU intensive operations with small memory footprint such as signal processing or other mathematically intensive algorithms.

When coding in native languages, it is still possible to interface to the the API frameworks written in Java via the Java Native Interface (JNI). This allows

native code to take advantage of the convenience of the Android standard APIs and application frameworks.

When coding in native languages, it is still possible to interface to the the API frameworks written in Java via the Java Native Interface (JNI). This allows native code to take advantage of the convenience of the Android standard APIs and application frameworks.

Code written in native languages can be reused in other applications. This makes the use of the NDK an effective approach as to write your own performance increasing libraries that several applications can benefit from.

3 The Process

The methodology for deducing what role smartphones may play in mechatronic applications basically consisted of the following steps:

1. A quick analysis to pin out the technologies that doesn't seem suitable to use in mechatronic systems at all. Focus is put on trying to identify common trends in the performance of cell phones to be able to draw generalized conclusions about the possible inadequacy of some technologies. It was found that this approach could be a fast way to narrow down the analysis given that some technologies could be excluded from the following more thorough analysis.

2. A more thorough performance analysis regarding each of the technologies. Given the set of appropriate technologies from the initial analysis, the goal with this step is to collect data concerning the performance for each of the technologies that can be of interest when utilizing a cell phone in a mechatronic system. The results will be in form of the physical quantity, limitations in the units of measurement and important properties associated with each of the technologies. The restrictions imposed on the hardware and software will be assessed with regards to sensor- and communication technologies.

3. The last and major step is a more holistic discussion that weighs the preceding analyses together with other important characteristics of a mechatronic system, ex. cost, integrability, time to market etc., to deduce the adequacy of cell phones in mechatronic applications. Conclusions was to be made about the appropriateness of each of the technologies but more important from a viewpoint that consider the cell phone as one combined device. This part will also include a discussion about what "extras" you get by using a system based on, or consisting of smartphones.

To enhance the results, a practical implementation in the form of an Android based safety application was developed.

4 Theoretical Analysis

In order to broach our main question, it is necessary to really dig into the core of mechatronics; what characterizes a mechatronic system, what features could make cell phones appropriate to use in such systems? To be able to answer these questions in a way that is scientific decorous, the theoretical analysis was found to be more prone to a qualitative approach than a quantitative. Instead of deducing the important conclusions in the project from statistically obtained numerical data, more focus is put on a holistic discussion which weighs in a number of viewpoints on a mechatronic system that may or may not make a cell phone useful.

4.1 What is Mechatronics?

The common explanation of a mechatronics is that it is a discipline that transcends several others. "Mechatronics" as a word is a mix of "Mechanics" and "Electronics" which is quite accurate, though in reality it is a broader discipline than that. During the 80's the personal computers how we know them today started to increase in popularity. The computers was now a lot smaller, cheaper and more portable than before. The introduction of computers radically changed all engineering disciplines, firstly a new one arose; "Computers" but it also affected all the others. Today computers are one of the most important tools in all engineering disciplines, they are used in a wide range; from microprocessors to large distributed systems. Today a mechanical system very rarely comes without an electrical system and a computer, the same applies to control systems which are more often implemented digitally since computers make the development more easy but also increase the performance, monitor and alter the systems. Finally the integration between the big disciplines were so large that it itself became an engineering discipline. Figure 2 shows a Venn diagram of the different fields that make up mechatronics.

4.1.1 Desirable properties of a mechatronic system

Since it is a large field with a lot of different applications aiming at solving different problems the desirable properties are many, but in the context of sensor systems, embedded systems and control systems, performance is an important property. For sensor systems; sampling rate, drift and sensitivity all are measures of performance, while for embedded systems, real time performance are one. Control systems often are characterized by step responses, phase lags and oscillation. Often these are a part of a bigger system which introduces other challenges like concurrency (priority issues) and cooperation.

4.2 Cell phones in distributed- and embedded systems, an introduction

In recent years smartphones have started to take over the mobile phone market. They are packed with functionality that makes use of common sensors in

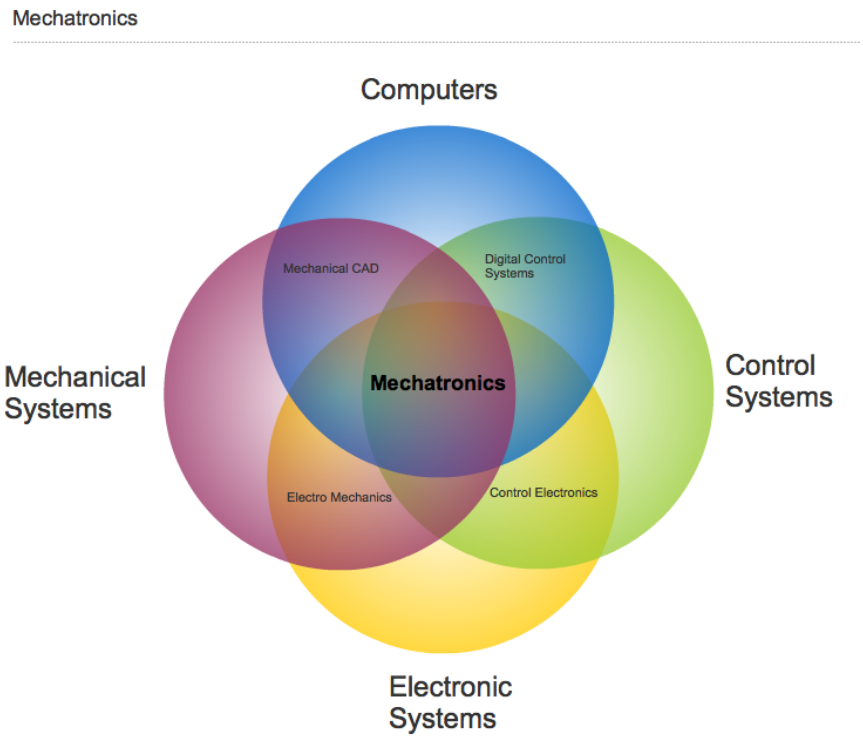


Figure 2: Venn-diagram of the disciplinces included in Mechatronics

mechatronic systems like accelerometers, GPS, proximity sensors etc. They also utilize and support the a wide variety of communication standards. It seems natural that they can replace or extend parts of an embedded system as they contain the hardware and software for several of the key subsystems, at least in terms of functionality. Let us look into this further, a regular embedded system may contain a number of different parts, see Figure 3.

Where gray features are functionality that can be accessed and used on a smartphone, blue are more common for an embedded system in general. It should be pointed out that the heart of an embedded system is a mechatronic system, and in turn, the embedded system is the heart of a larger system; in this thesis the smartphone. When studying the picture above it can be argued that an embedded system contain core parts of a mechatronic system in the sense of actuators, sensors and CPU but also communication methods not necessarily included in a mechatronic system in general, but a separate module, also a mechatronic system, just aimed at a different purpose. The smartphone (as an embedded system) can therefore be seen as a subset of mechatronic systems aimed at doing different tasks; sensing, actuation or communication with the result that the whole being greater than the sum of its parts. This can be seen as a hint that a smartphone might be used as a general mechatronic system capable of solving a variety of tasks, the question is just how good.

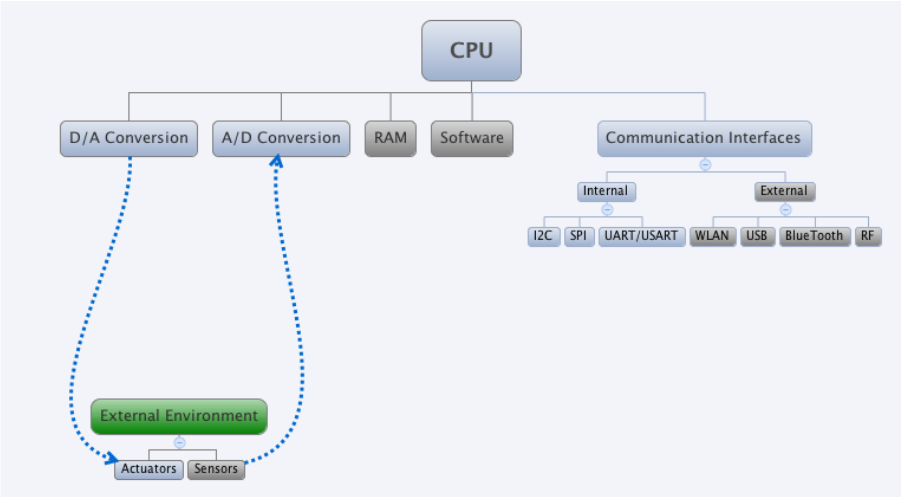


Figure 3: Components of a mechatronic system

4.3 How do mobile phones meet the requirements of embedded-, distributed- and mechatronic systems?

Above it was stated that smartphones and its subsystems do fulfill the properties of a mechatronic system; processor, sensors, actuators, communication hardware are all represented. In that matter alone it is a one-to-one comparison between phones and mechatronic systems since they are totally alike, if we were only to look at the functionality the analysis would be done. One of the reasons for this study is to consider whether performance and overhead both in hardware and software affects that relationship. One of the immediate differences a mechatronics engineer notice is that the smartphone already have the subsystems and modules connected and well integrated, no need for coding drivers, protocols or designing hardware as in a stand alone system made from scratch. What is much harder to say is whether this all-in-one-package with its overhead and functionality that comes out-of-the-box, affects performance, versatility or other not as intuitive or currently unseen properties. The stated question is not as easily answered as it might seem, certainly not when the question is asked without delimitations.

4.3.1 Operation systems and overhead

Overhead can easily be described as what is required to attain a certain goal, more overhead means more work to implement the same functionality. Operation systems is one example of overhead since the developer have to follow certain guidelines or tools to implement the functionality under different operation systems, though it is almost impossible to exclude this overhead since some sort of software to support the functionality is almost always needed, as will be described in this section.

The range of operation systems aimed for processors in mechatronic systems

are wide. Depending on the application, the operation systems need to be more or less advanced and sophisticated. The most simple ones might be OS's that are doing calculations in for instance control systems or simple interfacing of modules. These can not really be called operation systems in its right sense but in practice that is what they are. More advanced systems utilize high performance, real time operation systems to be able to perform according to requirements, in embedded systems there might be several processors or at least tasks that compete for the same resource and therefore the OS have to schedule what task can access that resource at that time all depending on the priority of that certain task (normally called Concurrency).

If a new system is to be designed, depending on purpose the system is designed for, a smartphone might be considered useful as the platform. Lets consider a sensor-node network with wireless real time transfer of the sensor data to a server/back-end. What the engineers firstly need to consider is how much time they have to spend learning how to develop for a certain smartphone OS versus the time needed to develop their own OS (or implementing a readily available one from the market), implementing the drivers and program the communication towards the back-end. If the engineers choice falls on the smartphones they will need to learn how to program for that specific platform, but they will have a real time operation system ready. If the sensors in the smartphone is enough to fulfill the requirements the drivers are already implemented, the thing that need to be done is the communication towards the back-end. The information on how to solve the communication is quite rich for most mobile platforms and will probably not be a big problem.

If on the other hand a mechatronic system is used, a readily RTOS can be used which can then be up and running in a short while, the sensor drivers need to be implemented which demands more time compared to the smartphone solution. Also the communication towards the back-end need to be done but that development time is probably quite similar as if done on a smartphone. If the system has functionality requirements far from that of what the smartphone OS can offer the operation system might need to be adapted or might not be suitable in the first place. If the functionality falls within that of a smartphone but only a fraction is needed, it is still a good starting point. The advantages of a smartphone based solution is that the system engineers can have the development environment up and running fast, the programming is often easy to get going if the developer have experience in programming object oriented languages. Internet is a rich source for implementation hints and information about the platform. In general it constitutes a good base for prototyping and experimentation. The downsides are that the OS is not easily configured for operation outside its specifications, this is where the advantages of a stand alone system comes in, more freely configurable but the cost is that it takes more time to get started and have a prototype up and running.

4.4 Analysis of the communication technologies in smartphones

4.4.1 Offered communication technologies in smartphones

As seen in Figure 2, smartphones offer support for high level communication protocols such as GSM, 802.11x (Wi-Fi) , 802.15 (BlueTooth), USB and to some extent NFC (RFID).

4.4.2 Why these technologies?

GSM

Groupe Spéciale Mobile is the second generation of wireless telephone technology (2G), a digital technology allowing transferring speeds of up to 473.6 kbit/s. Replacing the older analog first generation (1G). During the last decade, Internet has turned from a slow dial-up connection to the very fast connection it is today. Technical solutions like Internet through cable and xDSL made it possible to be constantly connected which has increased our demand for Internet access whatever we do and wherever we are. This also induced a need for simpler and lighter devices that can browse the web. Since we often carry our mobile phones with us it was a natural step that they would be fitted with this technology. When the user are not at home and need access to the web the mobile phone now make sure that it could be done. Now most of the basic things we needed a computer for can be done on the way to school and work on our mobile phones instead. Internet is based on communication by IP-packets and this led to the fact that also mobile phones needed to support this technology. Since mobile phones are meant to be portable, the IP-packet-based communication needed to be transformed into a wireless ditto compatible with mobile phones. This was done through various techniques beginning with GPRS.

GPRS

General Packet Radio Services is considered to be 2.5G, the 2.5 generation of wireless telephone technology, basically extended the functionality of GSM to include, by others; Internet applications for smart devices through wireless application protocol (WAP). Point-to-point (P2P) service: inter-networking with the Internet (IP). Point-to-Multipoint (P2M) service: point-to-multipoint multicast and point-to-multipoint group calls and MMS. GPRS made Instant Messaging and constant Internet access a reality. Today EDGE (Enhanced Data rates for GSM Evolution) is used which is similar to GPRS but with a higher rate of speed as one example, it is sometimes called (EGPRS).

3G

3G is the third generation mobile communication (3G). This is what is most common for mobile Internet both in smartphones and for computers and laptops.

In its most basic version it supports speeds of up to 2 mbit/s, the technology HSDPA - High-Speed Downlink Packet Access which is more common today supports speeds up to 14 mbit/s. HSDPA are sometimes called 3.5G or Turbo-3G. The latest revision is Evolved HSPA[18] [19] (HSPA+) support downloading speeds of up to 42 mbit/s. This is a technology aimed at doing the transition to 4G which is currently being deployed. As Internet through mobile and handheld devices grew popular the phones evolved more and more into what we now call "smartphones" with large screens to make browsing the Internet more easily.

WLAN

Wireless Local Area Network (LAN) also commonly called Wireless Fidelity (WiFi) is defined by IEEE Standard IEEE 802.11. It is the wireless counterpart to Ethernet and is very commonly used for connecting computers, both stationary and mobile ones with each other, also mobile phones, tablets and other devices that need connection to the Internet use WiFi for the same purposes. Normally the connection is done through access points like routers and gateways, several access points can be used to extend the range of the network. Wi-Fi made it possible to utilize the wireless Internet at home or at the office. It did not only increase the transfer speed but also limited the usage of the costly mobile Internet connection.

Bluetooth

Bluetooth is another common technology used in both mechatronic systems and mobile devices. As the USB-protocol it can be used for connecting peripheral devices like printers, headsets and for data transfer. It is commonly used in cars where it serves as the wireless communication link for wireless handsfree. When released in 1994 it was error prone which hindered some of the intended use for the technology, later on revision 1.1 and 1.2 was released. These new revisions solved the issues and implemented AHF - Adaptive frequency-hopping spread spectrum. AHF makes Bluetooth much less sensitive to interference from other wireless devices communicating at the 2.4 Ghz-band, this is performed by "hopping" between frequencies close to 2.4 Ghz 1600 times per second. The revision most commonly supported by cell phones today are 2.1 and 2.2 which were established in 2004 and is backward compatible with revision 1.1. With 2.1 Secure Simple Pairing (SSP) was implemented which simplified the pairing experience between devices[20] . The biggest change from earlier versions are Enhanced Data Rate (EDR) for faster transfer speed, the nominal speed supported is 3Mbit/s. Also Advanced Audio Distribution Profile is common from these revisions and is used for streaming high quality audio between Bluetooth devices. Bluetooth emerged as a way to couple hands free units and other modules to the mobile phones mainly to reduce the amount of wires needed for interfacing towards

[18] `http://www.gsma.com/aboutus/gsm-technology/hspa/` Accessed 2012-05-17

[19] `http://www.3gpp.org/HSPA` Accessed 2012-05-17

[20] `http://mclean-linsky.net/joel/cv/Simple%20Pairing_WP_V10r00.pdf` Accessed 2012-06-04

mobile phones and other devices. As the Bluetooth tutorial states: [21]

"Its key features are robustness, low complexity, low power and low cost. The technology also offers wireless access to LANs, PSTN, the mobile phone network and the Internet for a host of home appliances and portable handheld interfaces"

It is also used for transferring files and for sharing an Internet connection to other devices.

USB

Universal-Serial-Bus is widely used in mobile devices today, it is one way of connecting peripherals, in general mobile phones support USB 2.0 but the latest revision is 3.0 see Figure 4[22]. Several different versions have been developed since the first revision. The newest mobile devices per today supports USB 3.0 with a maximum capacity of 5 Gbit/s. This revision also increased the amount of current that the host device can source, making USB 3.0 able to charge battery based peripheral devices and also power them only through the USB-cable, this minimizes the amount of cables used and boost the portability of the peripherals. From USB 2.0 the sourced current was large enough to power screens, even more so with USB 3.0 where screens up to 24" in diameter can be powered solely through the USB-connection. USB-connections come in several types, the most common ones in mechatronic systems are type A and B, while mobile phones almost exclusively uses Micro B.

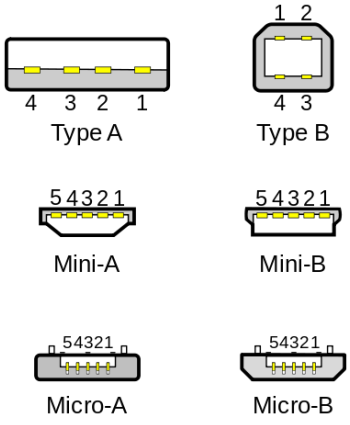


Figure 4: Common USB connections

[21] http://www.tutorial-reports.com/wireless/bluetooth/introduction.php Accessed 2012-05-17

[22] http://upload.wikimedia.org/wikipedia/commons/c/cb/Types-usb_new.svg Accessed 2012-05-17

It is also used for Internet tethering, transferring files. It has been used for several years through proprietary connectors at several phone companies but are now more common in microUSB. In the lastest revision (3.0) the amount of current a USB host is able to source has increased allowing devices that need more power able to use USB as an interface for communication and charging.

The International Telecommunication Union (ITU) approved the Universal Charging Solution (UCS) as a standard on the 22 of October 2009 which practically means that the microUSB-interface now is used as a standard for charging mobile devices[23]. When developing for Android, a real device can be used for debugging-purposes and for running the code when it is during development, USB is used for flashing the application .apk-files onto the device. When debugging, live log data are streamed to the development environment, very useful when debugging.

NFC

Near Field Communication (NFC) has recently been implemented in mobile devices, it has, though been of common use for communication and identification in mechatronic systems for years. In mechatronic it is called Radio Frequency Identification (RFID). The technology is based on induction, by subjecting the antenna to a magnetic field, the induced current is enough to propel the radio circuit and transmit its message. Recently the swedish transportation company Scandinavian Airlines (SAS) introduced their NFC-based "Smart Pass" to simplify the pre-flight procedures at airports for frequent-flyers[24]. Also Citibank India introduced its "A-little-world" banking solution to simplify banking errands.[25] [26]

4.5 Differences between communication techonology in mechatronic systems and smartphones

4.5.1 What communication-technologies do the "Mechatronic" processors offer?

Basic mechatronic systems implemented to solve simple control problems or module-interfaceing where most calculations are light are normally based on 8-bit processors for minimizing cost and ensure simplicity. When more power or advanced features are needed up to 32-bit processors are often used. These processors are done by several companies, for example Atmel and Freescale, following is a quick view at the communication methods offered by a set of processors ranging from 8-32-bits by the producer Atmel.

[23] `http://www.itu.int/newsroom/press_releases/2009/49.html` Accessed 2012.06-04

[24] `http://www.sas.se/shared/allt-om-resan/Infor-resan/Smart-Pass/` (Only available in swedish) Accessed 2012-06-04

[25] `http://www.nfcworld.com/2010/03/11/33051/citis-bangalore-trial-offering-cardholders-phone-subsidies-can-kickstart-nfc-transaction-volumes` Accessed 2012-06-04

[26] `http://www.alittleworld.com` Accessed 2012-06-04

Atmel ATmega16 8-bit processor

The ATmega16 together with ATmega8 constitutes the most basic microprocessors ATmel offers, it utilizes a clock rate at 16 MHz. The peripheral features consist, by others of:

- TWI (Two-wire Serial Interface, also known as I^2C)

  I^2C constitutes of a two-way serial bus developed for inter-IC communication, requiring only two lines, a Serial Data Line (SDA) and Serial Clock Line (SCL). Later revisions support transfer speeds from 400 kb/s in Fast Mode, up to 3.4 Mbit/s in High Speed Fast Mode. Benefits include low power consumption, low cost, a wide power supply voltage support and temperature operating range.[27]

- USART/UART (Universan Synchronous/Ascynchronous Receiver/Transmitter)

  Is the main protocol for serial communication in embedded and mechatronic systems, it transfers parallell data to serial on the transmitting side, and back again on the receiving side. Commonly used together with the RS-232 standard in mechatronic applications.[28]

- SPI (Serial Peripheral Interface)

  In similarity with USART a serial transmission protocoll with the distinction of a master/slave way of communicating, in contrast to USART (UART) not able to run in asynchronous speed. In comparison to TWI/I2C it supports higher transfer speed with the cost of more I/O-pins needed, in similarity with TWI it uses a source clock line named SCKL for syncronizing the signals. Two data lines called MISO and MOSI (Master Input Slave Output and the opposite) and a SSn - Slave Select signal used by the master to select what slave to talk to.[29]

The ATmega16 also includes other important features for communication development such as internal calibrated RC Oscillator, external and internal interrupt sources, usable for receiving and transmitting messages. Also sleep modes for power management when the communication is idle. 32 Programmable I/O Lines can be used for GPIO.

Atmel 32-bit AVR UC3 - AT32UC3

One of the high end microprocessors currently available from Atmel is the AT32UC3. The reason for examining a 32-bit processor is the fact that they are currently among the most advanced microprocessors, and they are offering a rich set of features from a mechatronic point of view.

With a maximum operating frequency of 50 MHz it is faster than the ATmega 16, it also features a wider variety of functionality. The processor offer the same

[27] `http://www.nxp.com/documents/user_manual/UM10204.pdf` Accessed 2012-05-23

[28] `http://ww1.microchip.com/downloads/en/devicedoc/usart.pdf` Accessed 2012-05-23

[29] `http://www.byteparadigm.com/kb/article/AA-00255/22/Introduction-to-SPI-and-IC-protocols.html` Accessed 2012-05-23

functionality as the ATmega16 but also some key features that is similar to the functionality offered by smartphones:

- USB On-The-Go On-The-Go is a supplement to the USB 2.0 Standard, a device can perform as both master and slave instead as of in Standard USB where a device must be the host and the other a client, only the host can initiate data transfers.
- Ethernet 10/100 Ethernet MAC

  Compatible with the IEEE 802.3 Standard including; adress checker, statistics and control registers, receive and transmit blocks, DMA interface.

One big difference between the higher level communication techniques offered by the AT32 and the more basic one compared to the ATmega16 is the fact that they are more suited for interfacing towards modern peripherals, USB are almost exclusively used for this purpose, Ethernet support also simplifies the connectivity towards Internet, it can be transferred to wireless communication using the 802.11 standard and simplifies connection towards wireless devices. The more basic communication protocols (USART, SPI, I2C) are more suited for internal communication in embedded or distributed systems since they are more easily interfaced. Though it should be mentioned that there are modules for transferring serial communication to ethernet and vice versa.

4.5.2 Common processors in smartphones

ARM-based processors [30] [31]

The processors used in mobile phones today are to an extent of 98% based on the ARM-architecture. The smartphone companies often use the architecture as a base in there SoC's (System on a Chip) which simplified means the CPU, this is done to give them more freedom to choose components and use their own design solutions but still let the cores run on the ARM-architecture. Qualcomm is one producer that uses the ARM-architecture for their processors, though they don not use the reference design (which they are rather alone in not doing). They produce four different types of processors called "Snapdragon", beginning with S1 and ending with S4, where a higher number corresponds to a more advanced processor.

The processors that goes under the S1-class is used in phones for the mass market and are commonly used in more economic oriented smartphones. The early S1's-chipsets included the QSD, it was later replaced by the MSM and AQP processors whit the distinction that AQP did not support wireless technology, this leads to MSM being a more popular choice as a processor in smartphones. The Snapdragon processors from class S1-S3 supports mainly wireless technology intended for telephony and Internet communication such as 2G, 3G, 4G. While the most expensive and fastest S4-processors also includes Wi-Fi according to 802.11 b/g/n standard.

[30] `http://www.neowin.net/news/guide-to-smartphone-hardware-17-processors#/news/guide-to-smartphone-hardware-17-processors?page=2` Accessed 2012-05-23

[31] `http://news.cnet.com/ARMed-for-the-living-room/2100-1006_3-6056729.html` Accessed 2012-05-23

4.5.3 What communication technologies are offered on a mechatronic system level?

Mechatronic systems in general are very modular, almost any technology can be implemented, it comes down to how much effort there is to put in to it. As mentioned in the technical reference there are numerous way of communicating with and between mechatronic systems. Zigbee, Xbee, WiFi, RF. RF is actually a collection of communication technologies, ranging from low- to high frequencies and utilizing different hardware and software for supporting error detection, timing, authentication and all other aspects of wireless communication, the RF-modules available can be very cheap and offer performance enough for low-speed, close-range communication.

ZigBee

ZigBee is a communication standard and protocol based upon IEEE 802.15.4 aimed at low-power, low-cost applications. It supports mesh networking and operates in 2.4 Ghz, 900 Mhz and 868 Mhz. The ZigBee protocol features:[32]

- Support for multi network topologies such as point-to-point
- Point-to-multipoint and mesh networks
- Low duty cycle - provides long battery life
- Low latency
- Direct Sequence Spread Spectrum (DSSS)
- 128-bit AES encryption for secure data connections
- Collision avoidance, retries and acknowledgements

XBee

is an implementation of the ZigBee protocol, the modules comes in a few different versions, the most basic one is the standard, low-cost 1mW XBee-module and at the higher end of the scale are the 100 mW XBee PRO. The modules come with different types of antennas, there are three versions: chip antenna, wire antenna and one that can be fitted with an SMA external antenna, this antenna can be used with the 100mW XBee PRO and yields a line-of-sight range of up to 24 km.

RF

RF is the general name for wireless communication within mechatronic systems, but when speaking in terms of RF-modules the 315-434 Mhz modules are often meant. These modules do not come with the high level protocols that most of

[32] `http://www.digi.com/technology/rf-articles/wireless-zigbee` Accessed 2012-05-23

the ZigBee- and Wi-Fi-based modules come with. Often the communication-protocol need to be implemented separately, mainly through UART-based serial communication. The range is limited and the connections are often prone to disturbances, efficiently reducing the effective range.

Ethernet

Ethernet is a common way to connect a mechatronic system to Internet or LAN, it is available through relatively cheap modules. Normal communication speeds are 10 or 100mbit/s.

4.5.4 Differences and similarities

The only thing limiting what a mechatronic processor can support is what basic communication-protocols it offers to communicate with each module, some modules comes with software and hardware implementing the communication protocol leaving only the work of reading the digital data from a GPIO-pin. Others leave the decoding and encoding to the software running on the processor and therefore need to be implemented separately. The positive aspects of being able to choose how to implement the communication protocol is that it can be altered outside the boundry should it be implemented on a smartphone. Making it able to meet the requirements of energy critical systems or systems demanding high safety through its own encryption techniques for instance.

The similarity between the systems as a whole is the fact that a lot of the functionality are implemented by extending the functionality beyond that of the processor, since the processors are very similar in what tasks they perform, except for the most extreme mobile phone processors like the Snapdragon S4 where Wi-Fi and other wireless features are already integrated, though this is not common for the mass-market versions. The mobile phone is already a highly integrated mechatronic and distributed system with a CPU handling everything from modules like camera, modems, communication to the screen, user interface to software. It also indicates that the systems are interchangeable and gives the phone a chance to play the role of a dedicated mechatronic system, especially in applications where its communication technologies are required. The last statement is true as long as it concerns a mechatronic system not putting extreme weight on requirements such as energy savings, safety or real-time-operations, since a dedicated system only is limited by time and money they can be more specialized in how to operate and therefore fulfill requirements not reachable by a smartphone based solution.

4.6 Evaluation of sensors in smartphones

Sensors plays an important role in almost any mechatronic application. They provide engineers with a “window” to the real non-digital world when the system needs information about different local physical quantities such as temperature, pressure, acceleration, compass heading etc.

Many of the new and innovative ways in which the modern cell phone is used requires technology to sense and "feel" its environment. The popular "tilt-to-landscape-view" functionality uses information about what forces are acting on the phone or how the phone is being twisted to determine when to switch the display mode from portrait to landscape view. Navigation map applications uses GPS technology to determine your global position when you want navigation assistance or perhaps just want to know where your friends is at. Even the trivial feature that turns off the touchscreen when you move the phone to your ear to answer a phone call "feels" the proximity of your cheek by using information from proximity sensors.

Although sensors are a practically mandatory part in any modern cell phone, the set of sensors that different manufacturers incorporates in their phones can differ.

To frame this discussion in favor of the main question of this thesis it is important to recall the process described in section 3. The section is opened with a brief summary to pin out the sensors that was and wasn't found to be suitable for a deeper analysis. This is followed by a short intro to sensor terminology. The section is ended by presenting and discussing each of the sensor types that can be found in a cell phone and critically assess the trends we found in sensor performance. The results from our tests are also shortly presented and elaborated upon.

Summary

In this summary, the different kinds of sensors that was found in one or more cell phones are presented below. During the first step of the analysis, the following sensors was found to be suitable for a deeper analysis:

- Accelerometer
- Gyroscope
- Magnetometer

Due to conclusions regarding some of the sensors' inadequacy to perform in mechatronic applications, their inherent simplicity or the intricacy in performing a thorough analysis the following sensors was excluded from the deeper analysis:

- GPS
- Image Sensor (i.e for the camera)
- Microphone
- Barometric pressure sensor
- Light sensor
- proximity sensor

4.6.1 Sensor terminology

Applications that incorporate sensors are confined by different performance specifications associated with the sensors. The resolution of a sensor is often specified in bits which originates from the binary representation of a sampled value in the AD converter. It indicates the number of quantization levels a sampled value can undertake. Thus, the smallest quantization step in the acceleration signal that an 8bit accelerometer with the range 3G can output is given by $\frac{2 \cdot 3G}{2^8} \approx 23mG$.

The resolution is tightly coupled with the sensor sensitivity and accuracy. The sensitivity can be defined as the smallest change in input signal that can be registered in the output signal. The accuracy of a sensor is defined as the maximum deviation of a sampled value compared to the actual real value.

Offset is sometimes referred to as the bias and is a static error that is most easily shown by reading the sensor output in a state where the sampled values should read zero. The statistically constant deviation from zero is referred to as the sensor's offset.

The specifications mentioned above is likely to find in a sensor data sheet. Generally, higher resolution/sensitivity/accuracy and lower offset means a more expensive, high end sensor. Although these specifications may not induce critical restrictions on the host application it is interesting to discuss whether cell phone manufacturers chooses to incorporate expensive, high end sensors or cheaper low end sensors. Chances are that engineers have cost optimized the selection of sensors with regards to the function they will execute at the end user. This has the obvious effect that sensors that isn't used in ways that demands high resolution/sensitivity/accuracy or low offset will be chosen of poor quality. If such trends can be found it would have great impact on the workflow in this thesis as these sensors could be excluded from the coming more thorough analysis.

The range of a sensor defines an interval, typically $\pm max_{measuredquantity}$ of the sampled values outside which they are saturated to the value $max_{measuredquantity}$.

4.6.2 Android Sensors

Sensors in Android can be accessed via the class "SensorManager" that uses the Android system sensor service. An instance of SensorManager can be obtained by calling the function getSystemService() with the argument SENSOR_SERVICE. In order to start sampling a sensor, a SensorEventListener has to be created and registered. A SensorEventListener receives the "event" that occur when a new sensor sample is available from the sensor service. Such an event contains information about the sensor such as the timestamp for the sample, the sampled values and which sensor that generated the event. A single SensorEventListener can be shared among different sensors or used individually for only one sensor. To register a listener for a sensor, the SensorManager specific method registerListener() must be called with the arguments: the name of the SensorEventListener, the type of the sensor (i.e. TYPE_ACCELEROMETER or TYPE_GYROSCOPE etc.) and what sampling rate mode that should be used (i.e. SENSOR_DELAY_FASTEST, SENSOR_DELAY_NORMAL etc.). In

such a way, several different sensors can be registered and sampled in a single program.

The overhead of Android is a blessing for projects where rapid prototyping is important. APIs with high abstraction makes it very easy to combine the different hardware modules in a cell phone into new, innovative applications. On the other hand, the overhead can sometimes lower the transparency to the hardware layer for a programmer. Using sensors in the higher abstraction layers of Android suffers from a big drawback; one can't set the sampling rate of a sensor to be a specific value. Instead, one can choose from four different rates: "UI", "Normal", "Game" or "Fastest". Through very quick testing, it was found that the "fastest" mode was indeed the fastest and most robust which was the sampling rate mode that was used in every test and program in the whole project.

4.6.3 Common sensors

Below follows a summary of each of the sensor types mentioned above that can be found in cell phones. Their function is described as well as how they are commercially used in cell phones. In addition, conclusions is drawn regarding performance limitations that could affect their usefulness in mechatronic applications negatively.

Accelerometer

An accelerometer measures absolute acceleration $[\frac{m}{s^2}]$ relative free fall along one, two or three orthogonal axes. This means that an accelerometer subdued to free fall registers $0\frac{m}{s^2}$ along each of its axes. Consequently, an accelerometer placed with its z-axis orthogonal to the center of the earth would register 1G along the z-axis. They are used in a broad range of different applications. In everything from cars and industrial supervision to gaming consoles, engineers have found use for them. The extensive use of accelerometers typically originates from various demands of knowing what forces that are acting on an object. These forces could manifest themselves as vibrations in the object or simply a spatial motion of the object. In the context of cell phones, accelerometers are used mainly to enhance the end user's GUI and gaming experience. The common feature "tilt-to-landscape-view" are often implemented using accelerometers to determine the direction of gravity and consequently change from portrait to landscape view when the user tilts the phone. It is sometimes also used in different games where the user can tilt the phone to maneuver for example a space ship or a ball through a maze.

Accelerometers was found to be the most common sensor in smartphones, with the exception of microphones and cameras. Looking at the table in appendix A one can see that the accelerometers found in cell phones is of a pretty decent standard. It should be noted that the LIS3-series of accelerometers from STMicroelectronics is over represented in the sample of cell phones under study. They are obviously used by both Samsung and Apple who together makes up for approximately 40% of the cell phone market share today according to IDC

(Feb 2011). [33] We argue that this over representation of the LIS3-series therefore should be considered as at least statistically indicative of the market as whole.

The range of the accelerometers was found to be dynamically selectable and strictly between $\pm 2G$ and $\pm 16G$ which should be enough for a wide variety of applications.

It was found that the resolution varies somewhat but the trend indicates that the most common resolution is 16 bit.

The same trend can be seen regarding the sensitivities which are generally pretty high. Furthermore, output noises was generally just over 200 $\frac{\hat{A}\mu g}{\sqrt{Hz}}$.

In Android 2.2, the accelerometer is normally accessed via the SensorManager and a standard SensorEventListener.

Gyroscope

A gyroscope measures angular velocity $[rad/s]$ alongside one, two or three orthogonal axes. As opposed to what many people may think, a gyroscope cannot directly measure the angle of an object. By integrating the angular velocity, the change in angles of an object can be extracted. However, this is true only in theory, practically it is a lot more complicated. The inevitable bias error described in section 4.5.1 makes it impossible to get a stable, robust reading of the angle by integrating the raw angle velocity signal due to the fact that the static bias error would be accumulated in each step of the integration. This would make the angle error grow linearly with time which popularly is called the bias drift. Instead, more or less sophisticated algorithms that cancels the static bias drift prior integration can be implemented to reduce these effects.

Using accelerometers to determine the tilt in different cell phone applications is prone to be erroneous. Should the phone at all times be held completely still while being tilted, they would be able to determine the tilt angle at all times. However - accelerometers measures acceleration and nothing more. This means that the acceleration signal of, lets say a clockwise tilt of a phone looks the same as the signal of a right, slightly upward directed acceleration of the phone. Without going into any further details, this could give rise to certain false positives should the phone be moved in that way. The way some cell phone manufacturers started to deal with these problems was to fuse the accelerometer signal with the signal from a gyroscope. Fusing two sensors refers to the technique where the signals from the two different sensors are combined in a filter to create an output signal that is improved compared to if the sensor signals would have been used separately. In this case, accelerometer- and gyroscope signals are fused together to create an improved and less erroneous signal of the current angle of the phone. Such a fusion is often called a 6-axis fusion and refers to the combined 3 axes of the accelerometer and 3 axes of the gyroscope. Gyroscopes are rarely seen in older smartphones and are a bit less common than accelerometers in modern smartphones. This is probably due to the fact that

[33] `http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats` Accessed 2012-05-20

manufacturers just recently has become aware of the fact that gyroscopes can be fused together with an accelerometer to enhance these features.

In Appendix A it is obvious that the STMicroelectronics gyroscope L3G4200D is tremendously over-represented. The direct effect of this is that the performance trends drawn in this thesis regarding gyroscopes will be heavily biased.

The measuring range of cell phone gyroscopes was found to be dynamically selectable between $250rad/s$ and $2000rad/s$. The resolution was found to be 16bits and that the sensitivity ranges from 8.75 to 70 $mrad/s$. We argue that the performance specifications of the gyroscopes found in cell phones suggest that their most appropriate field of work is applications where high angular velocities needs to be monitored or registered. Due to the relatively wide range of the gyroscopes we argue that they are not optimal to use in applications where precision when monitoring slow rate angular velocities is desired.

In Android 2.2, the gyroscope is normally accessed via the SensorManager and a standard SensorEventListener.

Magnetometer

A magnetometer measures the strength of magnetic fields T along one, two or three orthogonal axes. As one Tesla is of a relatively large magnitude, magnetometers often ranges in the magnitude of nT. As a reference, the Earth's magnetic field is about $31uT$ at the equator[34]. This makes three axis magnetometers capable of detecting the strength and direction of the Earth's magnetic field which is utilized in cell phones in so called e-compasse. Just as conventional non-electronic compasses, magnetometers are sensitive to disturbances in the magnetic field and should be used with caution.

One problem with the sensor fusion between an accelerometer and a gyroscope is that there is no way to actually determine the yaw angle in a robust manner. Some manufacturers of embedded sensors have started to deal with this problem by creating a 9-axis fusion of signals from a 3-axis accelerometer, a 3 axis gyroscope and a 3-axis magnetometer[35] which could further increase the precision with which the orientation of a phone is determined.

The results presented in Appendix A shows that the Asahi Kasei AK8975 magnetometer is used in every cell phone that we could confirm to be equipped with a magnetometer. As the data sheet for the AK8975 was unable to provide sufficient information about its sensitivity, sampling rate and output noise, we were unable to draw conclusions regarding any trends in these specifications. The range and resolution however was found to be and 13bits respectively. We argue that this is sufficient for a number of applications. We also argue that the contribution that magnetometers brings in a 9-axis sensor fusion is of great interest and importance when evaluating their potential role in mechatronic applications.

[34] `http://www.wisegeek.com/how-strong-is-the-earths-magnetic-field.htm` Accessed 2012-05-01

[35] `http://www.phonearena.com/news/9-axis-motion-sensor-fusion-promises-unprecedented-precision-for-future-smartphones-and-tablets_id22482` Accessed 2012-03-12

In Android 2.2, the magnetometer is normally accessed via the SensorManager and a standard SensorEventListener.

GPS

Perhaps not popularly known as a sensor but which falls under that category in this thesis is the GPS module. A GPS module uses the Global Positioning System which is a free satellite based service available for everyone to use. It provides GPS modules with information about their global position aswell as time information everywhere on the planet where the modules has a free line of sight to at least four GPS satellites.

GPS modules exists in more or less all modern cell phones and are used in various navigation and map applications. Social networks allow users to quickly share and "check in" their position to let friends know where they're at. Due to their already extensive use in cell phones and the advantage of constantly be able to locate a phone's position, we argue that GPS modules will be one of the key elements to be used in future cell phone applications.

In Android 2.2, the GPS is normally accessed via the LocationManager and a standard LocationListener.

Image sensor (camera)

Image sensors are actually large arrays of light sensitive diodes known as "photosites". The size of an image sensor is often specified by the amount of photo sites they hose which in camera commercials is often referred to as "pixels". They are used in digital cameras instead of photo films used in older cameras and works in approximately the same manner.

Together with compact optics, image sensors constitutes the building blocks for the digital cameras seen in cell phones. For about ten years ago, cameras in cell phones were rarely seen in other than high end phones and even in that case the image sensors were of poor quality. Today, image sensors have become considerably cheaper and of a much higher quality due to improved manufacturing techniques.[36] Digital cameras are presumably one of the most popular features and one of the key arguments when marketing new cell phones. Mechatronic applications utilizing cameras will most certainly involve some kind of image processing algorithm which often are very processor intense. Tresadern, Ionita and Coote (2011) suggest an implementation of a face recognition algorithm on a cell phone. We argue that the results from their study is of a sufficient standard for us to be able to draw the conclusion that cell phone processors are fast enough to be used in real time image processing. Note however that this conclusion is free from assumptions regarding the sizes of the images used in the computations.

Tresadern, Ionita and Coote implemented the algorithm on a Nokia N900 with a 600Mhz processor using a video stream with a frame size of 640x480 pixels

[36] `http://www.imagesensors.org/Past%20Workshops/2009%20Workshop/2009%20Papers/021_paper_fontaine_trends.pdf` Accessed 2012-06-10

and a frame rate of 25-30 fps. It is common that modern cell phones can record videos with full HD quality, i.e. with a frame size of 1920x1080 and a frame rate of 30 fps. Such a stream contains about 8 times the amount of pixels compared to a 640x480 stream which very roughly corresponds to the amount of extra calculations should the same algorithm be implemented on the separate streams. Combined with the fact that processors in cell phones today are only between 2-3 times faster than the N900 it is reasonably to believe that a similar face recognition program to that suggested by Tresadern, Ionita and Coote would run slower on a modern full HD video stream compared to an older 640x480 stream. We argue that this increase in the amount of calculations should be taken into consideration when implementing image processing algorithms on a HD video stream in a portable cellular device.

In Android 2.2, the camera is normally accessed via the Camera service and a picture can be taken by calling takePicture on a Camera object. A video stream can be recorded and saved programatically by configuring the MediaRecorder class.

Barometric pressure sensor

A barometric pressure sensor measures atmospheric pressure [Pa]. Although they are pretty rare to be seen in cell phones, barometric pressure sensors have started to appear more frequently in the last couple of years. They are used in some applications to determine a phone's height over the sea[37] or simply as a digital barometer.

Due to their rareness, only one of the cell phones that is examined in this thesis was found to have a barometric pressure sensor installed. Because of this, no conclusions regarding trends in cell phone pressure sensor performance could be drawn.

None of the test phones was found to be equipped with a pressure sensor. This rendered us unable to perform any tests that could have at least given us a few indications regarding sampling time and different statistical signal measures.

In Android 2.2, the barometric pressure sensor is normally accessed via the SensorManager and a standard SensorEventListener.

Microphone

A microphone is an acoustic sensor that registers and converts the air pressure fluctuations originating from sound waves into an electric signal.

Microphones are self-evident parts of cell phones but perhaps not in mechatronic applications. One of the fields that they are used is in machine monitoring where they can be used to detect periodic imperfections in for example a cog wheel by using for example Fourier theory.

[37] `http://developer.android.com/reference/android/hardware/SensorManager.html#getAltitude(float,%20float)` Accessed 2012-06-10

During the mapping of sensors in cell phones, no manufacturer supplying any information at all about what microphones they are using in their cell phones could be found. Further, in the scope of this thesis, there was not enough time in order to perform any tests or analysis regarding frequency curves, polar patterns or sensitivity of the microphones found in the test phones either.

In Android 2.2, an audio stream can be recorded using the microphone and saved programatically by configuring the MediaRecorder class.

Light sensor

A light sensor measures the intensity of incoming light in Lux. In cell phones, light sensors are commonly used to actively regulate the screen illumination intensity depending on the environment's light intensity. This is a clever way to reduce the power consumption of the screen.

Although they are extremely common in cell phones, it was impossible to find any information about what kind of light sensors that are used in cell phones. However, some minor testing could be performed on the test phones. Basically two main implications could be done; the sampling rate of the light sensors seems to be approximately 1Hz and the resolution seemed to be rather low.

In Android 2.2, the light sensor is normally accessed via the SensorManager and a standard SensorEventListener.

Proximity sensor

A proximity sensor measures the orthogonal distance to [m] or the presence of an object. Proximity sensors are commonly used in cell phones with touch screens. They prevent unintentional screen inputs by for example the ear or the cheek when the user answers a phone call.

In Android 2.2, the proximity sensor is normally accessed via the SensorManager and a standard SensorEventListener.

4.6.4 Sensor sampling time tests

Some tests regarding possible inconsistencies in the sampling time was made on some of the sensors. Two test cases was considered. The first concerns the variance in sampling rate that occur as a result of possible Android overhead when sampling the accelerometer, gyroscope and the magnetometer separately. The second test case considers the same variance in sampling time when the three sensors are sampled at the same time. These programs was tested in all of the four Android specific samplig rate modes: Normal, UI, Game and Fastest separately. We argue that by testing how the sampling time differs when all three of the sensors are active, conclusions can be drawn regarding variances in sampling time for a possible 9-axis sensor fusion between the three sensors.

Android sensor events was described i earlier chapters. Such a sensor event contains a number of different information posts, one beeing the timestamp at which the event occured. The timestamp uses a built in clock with a resolution of nanoseconds which is scaled down to millisecond in the test programs.

4.6.5 Summary

Two test programs was written. The first collecting a number of sensor readings from each of the three mentioned sensors separately. The second program collected a specified number of sensor readings from all of the three sensors at the same time. The sampled data from these programs was used in a Matlab script that fitted the time differences between each consecutive sample to normal distributions of the probability density functions for the different sampling times. Quite big variances in the consistency of the sampling times could be seen when the sensors was sampled

From the standard deviations of these PDFs, quite big variances of the sampling times could be seen for some of the sensors which suggest that that the Android overhead should be accounted for in applications.

4.6.6 Test program

The test programs was written for each sensor type such that any effects of the sampling rate was minimized. The program flow can be seen in Figure 5.

At the "Program startup", necessary Android specific details such as registering listeners and drawing the GUI is performed. Further, a StringBuffer that will contain all sensor readings is initialized to prevent allocation of it in run time which may cause the program to run slow. When the button in pressed the sampling activates and continues until n reaches 10000 iterations. The StringBuffer is then converted into a huge string and saved to a file after which the program exits.

Test results and discussion

It was found that the standard deviation sigma, , deduced from the fitted PDFs varied very much between the sensors when they were sampled individually. Sensors can be sampled individually by registering only one SensorEventListener to the Android SensorManager. In contrast to the variance in standard deviation, the mean of the sampling time was found to be extremely consistent for all the sensors. Looking at Figure 6 it is obvious that the sampling time of the accelerometer varies the least with a mere standard deviation of around 1 ms and in some rare cases 2 ms. In contrast to this, the gyroscope yields sigmas in the span 5 to 7 ms which is considerably higher. The sampling time was found to vary the most for the magnetometer, from 4ms up to about 18ms. When the sampling rate mode "UI" was tested, the sampling rate for the magnetometer seemed to be more consistent and stable if compared to the results in the other modes. No good conclusion as to why this behavior was observed

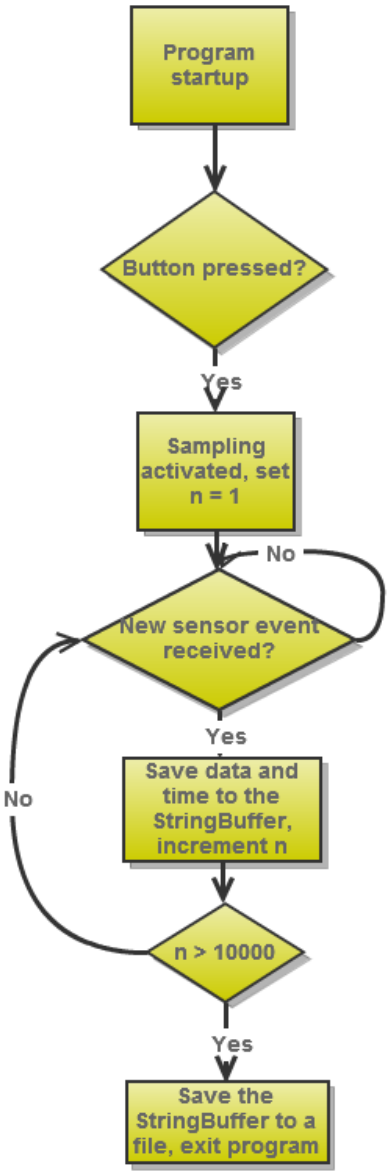


Figure 5: Flow chart of the programs used to test the sampling time of the sensors.

could be drawn but the disparity was at least consistent during every test in the UI-mode. In Figures 7, 8 and 9 the resulting PDFs for a test case where the sampling mode "Fastest" was chosen can be seen.

It is obvious that there is a huge difference in the determinism of sampling times between each of the sensor types. This is a rather interesting observation as it suggests that Android itself could have some sort of prioritization order between the three sensors. This in turn could give rise to the nondeterministic behavior observed in the tests. However, no good could be drawn as to why these effects were observed. In any case, we argue that these results suggest that certain applications where strict sample rate is desirable should take these effects into consideration.

	Sensors sampled individually					
Sampling Rate	Accelerometer		Gyroscope		Magnetometer	
FASTEST	*mu (ms)*	*sigma (ms)*	*mu (ms)*	*sigma (ms)*	*mu (ms)*	*sigma (ms)*
	20,18	1,47	20,00	6,08	20,17	9,10
	20,06	0,89	20,00	5,50	20,06	9,13
	20,03	0,76	20,05	5,80	20,11	9,42
	20,00	0,60	20,00	6,10	20,18	10,00
Average	20,07	0,93	20,01	5,87	20,13	9,41
GAME						
	20,10	1,35	20,00	6,50	20,10	9,05
	20,16	1,55	20,10	4,18	20,20	9,10
	20,00	0,98	20,04	6,00	20,14	9,93
	20,02	1,04	20,12	6,40	20,14	8,79
Average	20,06	1,19	20,09	5,53	20,16	9,27
UI						
	70,00	1,40	70,05	3,54	69,90	2,80
	70,00	2,26	70,05	4,18	69,90	3,12
	70,50	2,10	70,10	4,91	69,90	2,82
	70,00	0,80	69,90	6,00	69,50	5,32
Average	70,17	1,72	70,02	5,03	69,77	3,75
NORMAL						
	200,00	0,83	200,00	5,40	200,00	17,70
	200,00	0,60	200,00	4,90	200,00	17,50
	200,00	1,20	200,00	3,60	200,00	13,50
	200,00	0,58	200,00	6,65	200,00	12,60
Average	200,00	0,79	200,00	5,05	200,00	14,53

Figure 6: Test results when the sensors was sampled individually where mu is the sampling time mean and sigma is the sampling time standard deviation derived from the fitted normal distribution.

The same tests was performed when the three sensors was sampled in parallel to each other. This was done by registering multiple SensorEventListeners to the Android SensorManager. This yielded a number of interesting observations. Looking at Figure 10, one can see that the sampling time mean and standard deviation is more or less identical between the different sensors. In addition to that, the average sigmas is considerably higher for the accelerometer and gyroscope in Fastest and Game mode compared to when they were sampled individually. In UI And Normal mode, the standard deviations was found to be considerably lower for the gyroscope and magnetometer compared to when they were sampled individually. This is an example of an extremely inconsistent and nondeterministic behavior for any system and should be taken into consideration in applications that demands that these sensors are sampled in parallel. All in all, the Normal mode actually yields the most robust behavior where the standard deviation is relatively low around 2.65 ms. On the other hand, the sampling period is 200 ms means that an application with sampling mode Normal activated will have to have a very limited bandwidth.

No good explanation could be found as to why this behavior was observed when the sensors was sampled in parallel. The interesting observation here is

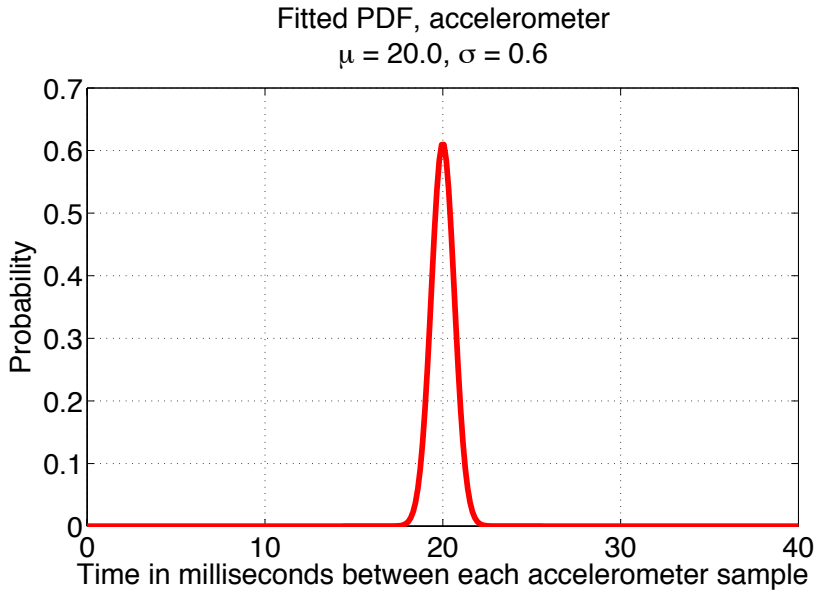


Figure 7: Fitted PDF of the differencies in sampling time between each accelerometer sample when the sampling rate mode "Fastest" was chosen.

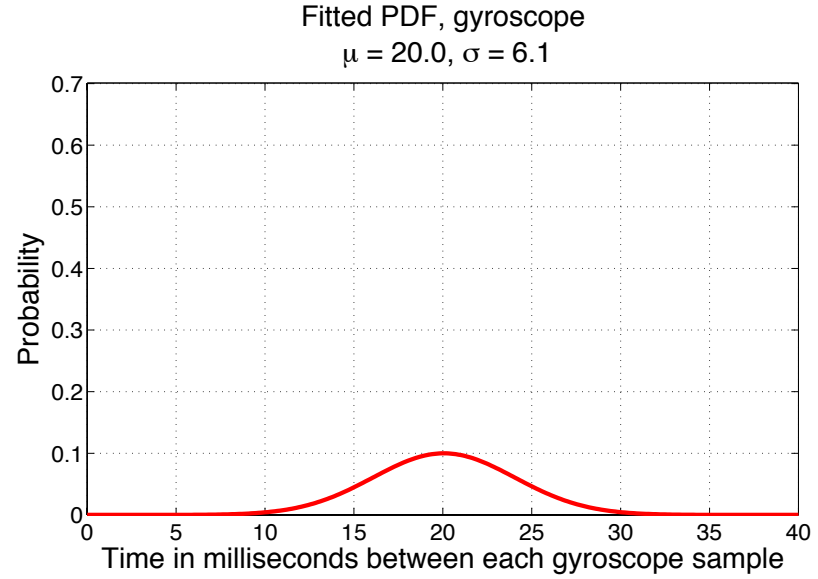


Figure 8: Fitted PDF of the differencies in sampling time between each gyroscope sample when the sampling rate mode "Fastest" was chosen.

that Android seems to alter the sampling time of the sensors such that their individual sensor event occur more or less at the same time, something that we argue should be taken into consideration when sampling multiple sensors.

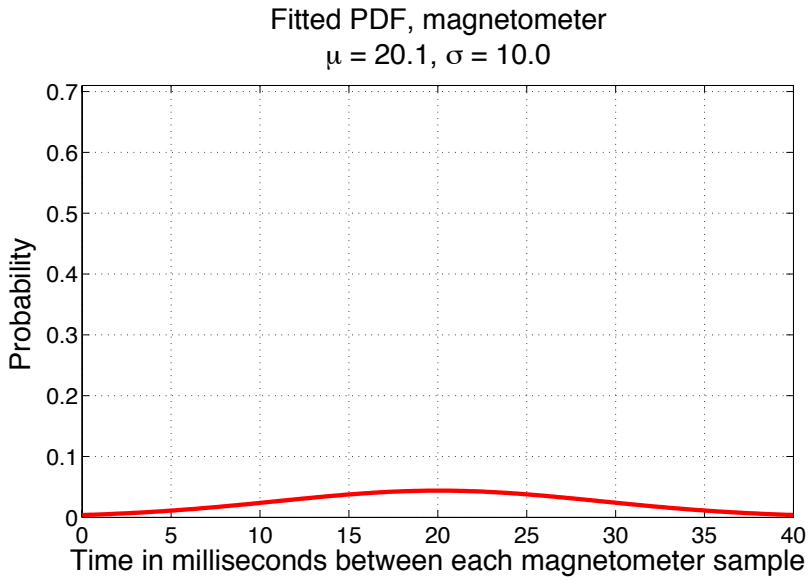


Figure 9: Fitted PDF of the differencies in sampling time between each magnetometer sample when the sampling rate mode "Fastest" was chosen.

	Sensors sampled in parallel					
Sampling Rate	Accelerometer		Gyroscope		Magnetometer	
FASTEST	*mu (ms)*	*sigma (ms)*	*mu (ms)*	*sigma (ms)*	*mu (ms)*	*sigma (ms)*
	20,10	9,54	20,10	9,50	20,10	9,50
	20,70	9,62	20,70	9,82	20,70	9,80
	20,10	9,67	20,10	9,74	20,10	9,70
	20,10	9,61	20,10	9,71	20,10	9,62
Average	20,25	9,61	20,25	9,69	20,25	9,66
GAME						
	20,70	11,40	20,70	11,42	20,70	11,21
	21,60	11,50	21,60	11,49	21,60	11,50
	20,90	11,43	20,90	11,46	20,90	11,42
	20,90	11,49	20,90	11,51	20,90	11,52
Average	21,03	11,46	21,03	11,47	21,03	11,41
UI						
	70,10	3,14	70,10	3,10	70,10	3,15
	70,10	3,10	70,10	3,10	70,10	3,13
	70,00	3,20	70,00	3,17	70,00	3,19
	70,30	3,16	70,30	3,15	70,30	3,15
Average	70,13	3,15	70,13	3,13	70,13	3,16
NORMAL						
	200,20	2,65	200,20	2,63	200,20	2,65
	200,30	2,67	200,30	2,67	200,30	2,66
	200,00	2,59	200,10	2,61	200,10	2,62
	200,30	2,70	200,30	2,66	200,30	2,71
Average	200,20	2,65	200,23	2,64	200,23	2,66

Figure 10: Test results when the sensors was sampled in parallel where mu is the sampling time mean and sigma is the sampling time standard deviation derived from the fitted normal distribution.

5 Practical Implementation - Crossafe Safety System

5.1 Problem Background

During the last couple of years, several accidents with fatal outcome have occurred during motocross events. The Swedish motocross and snowmobile association SVEMO, have been working on developing safety solutions for reducing the risk of these accidents. One of the proposed solutions was an active electronic safety system placed on the drivers which evolved into h!the Crossafe system. By analyzing the accidents and discussing these with SVEMO's federal physician some general causes as to why these accidents turned out fatal could be seen. This analysis was mapped towards the performance and specifications that a modern smartphone has. It was considered that most fatal accidents occur when drivers falls in front of other drivers and hence beeing run over. After evaluating the performance of smartphones it was clear that it would be very hard to prevent accidents where the distance between the drivers are very short like in motocross for example. However, the severity of drivers falling in front of other drivers are not just dangerous in motocross. In enduro (cross country driving) the same problem arise even though the distances are longer between the drivers. If one driver falls and is not able to get out of the way the same situation will arise possibly exposing the driver for a very dangerous situation. Further, as the time between crashes are significantlly longer in enduro than in motocross, it was found that the performance of smartphones would be sufficient to develop a safety system that can be used in enduro.

Practical Implementation

In the theoretical reference frame chapter, a number of different articles utilizing cell phones in mechatronic applications was presented. In the scope of this thesis, Crossafe is used as a reference project to supplement these already mentioned applications. The value that Crossafe brings to the analysis comes partly from the technical width of the application that uses sensor signal processing and sophisticated communication infrastructure. But what is more is that valuable insights could be embraced regarding the different aspects of an engineering development process that utilizes cell phones. We argue that these insights would not be possible by simply studying previous work in the field. We also argue that these insights supports the use of cell phone's in prototype development in particular.

Scenarios

When determining the requirements and locking in the functionality two user scenarios were identified as the main ones. It was clear quite early that the solution based on smartphones were not suitable on the motocross track where the distance between the drivers are short, and the time for reactions to prevent accidents are milliseconds to a couple of seconds. Instead focus were put on identifying where this system could be used. The key criteria was that more time was needed for the system based on smartphones to react upon accidents and inform other drivers or inform medical personnel about the accident. Both

functionalities were discussed but the main priority was put on giving the fallen driver quick medical assistance and in second hand hint other drivers coming from behind that a driver is lying in the track, possibly unconscious. The focus was put on giving the driver fast medical attention where after these two scenarios were identified:

"Supervised-mode", where the drivers are under supervision by the competition management and medical personnel, typically during a race or supervisioned practice.

"Non-supervised-mode", where the driver are alone or practicing with friends. The accompanying friends together with relatives are the first ones able to assist the driver, and they in turn are able to call for medical assistance.

Smartphone based security-system

In the introduction it is stated that more and more people in the world gain access to telephony communication and cell phones. In northern Europe the penetration rate is very high. This makes the smartphones an interesting choice of platform for building this system. Since basically everyone got one the distribution of the system-platform is done, the only thing that needs to be done is to install the application. The application can be uploaded to the virtual markets that smartphones utilize today for sharing applications and software, the system then becomes available for download to all users. As stated in the theoretical analysis the smartphones have sensors able to determine a wide variety of different local physical quantities. They also have the ability to communicate through Internet or via the telephony network which makes them suitable as sensor-nodes able to tell other nodes about their status. The concept of sensor-nodes is the basic idea behind the Crossafe safety system. Since the problem suited very well as a way for proving the usability of smartphones in mechatronic applications it was decided together with ÅF and SVEMO that a proof of concept would be delivered. This system would solve both the safety issue for motocross drivers but also act as a brilliant example of how the smartphones can be utilized within this sort of applications normally implemented through stand-alone systems.

Functionality Requirements

A number of documents was written that specified what functionality the safety system should have. These were developed in conformity with what the stakeholders expected from the application and was iterated over a couple of times during the project, it should be mentioned that focus was on designing an architecture. For a complete reference of these documents, please refer to Appendix B

5.2 System Description

5.2.1 Functionality in brief

The basic idea with Crossafe is to reduce the time from when a driver falls to when external actors are alarmed about it. These actors could be other drivers, training partners, race organizers, the race doctor or simply the fallen

drivers relatives. By reducing this time, a number of positive effects can be expected. First and foremost, if other drivers can be alerted about that another driver has fallen, this could prevent accidents where they crash into the already fallen driver. Further, if the race doctor is alerted directly about the position of where a driver has fallen, the time until he can rescue the driver should decrease significantly.

The Crossafe concept features some different key functionalities. The crash detection algorithm senses when a driver has crashed by using the cell phone's gyroscope and accelerometer. The cell phone can communicate with our self developed cloud solution based on the Google App Engine which in turn can communicate with all the other phones that has the application installed. Should a driver crash, the cell phone of other drivers can collect information about who has crashed, when he crashed and where he crashed which is valuable information should they decide to help the fallen driver. The position of the crashed can easily be viewed in the phone's map application.

The concept was developed on the Android platform, implemented through an application running on the phone. In brief the safety-system is based on the criteria that every driver carries a smartphone with them while driving. The drivers are prompted to login with their google account when starting the application. The google account mail address together with each device id (which is fetched by the application) is used for identification throughout the whole system. When the driver is registered the program searches for an application specific file called the "helmet card" which holds important personal information about the user. If no helmet card is found, the user is prompted to supply one before the application starts. The user is now free to navigate through the GUI and can start the "Safety Mode" by simply pressing a button. In safety-mode the application are constantly monitoring the sensor-input from the accelerometer and gyroscope where an algorithm is able to determine if a driver falls.

When the crash algorithm detects a fall it is very hard to deduce the severity of the crash, the driver might be able to move out of the way by him-/herself. Chances are that the crash algorithm has detected a false crash and triggered a false alarm. To prevent unnecessary alarms the driver will be prompted to acknowledge whether it was a false alarm or not. If the driver does not respond within a certain time, a message will be sent with two purposes: mainly requesting medical assistance but also hinting other drivers that a driver has fallen and that there is a potential risk for him lying in the track. The message contains information about the fallen driver id, position, timestamp for both the message and when the position was read. The message is received by a webserver running a database, this is where the information is stored. When the message is received on the server-side a push-notification (C2DM - Cloud To Device Messaging) is sent to all other devices/drivers notifying them about the new crash message that can now be downloaded. The latest message can be downloaded showing who has sent the message, at what position it was sent and the position of the crash can be viewed through the android native application Google Maps. This also shows the position of the fallen driver and the person viewing the map, this is useful to quickly be able to assist the driver. There is also a possibility to add information about relatives or other persons that might be interested in

knowing a drivers position in the case of an accident. This is done in Enduro today by using so called helmet cards which contains information of the id of the driver and hes corresponding relatives. The application basically allows a user to store the same information digitally. This is useful when practicing alone or to notify other drivers not able to run the application. The notification is done by sending a SMS-message to the persons/relatives listed in the helmet card file and is therefore also compatible with all cell phones able to receive SMS.

5.2.2 Authorization and identification

Driver ID

During the problem definition and research phase it was clear that each driver needed to be identified through a unique ID. When a driver falls it is possible for other drivers and medical personel to see who the fallen driver is, this can be used to prepare the medical effort by knowing what blood type or what general medical condition the driver has. The ID is currently connected to the google user account used when logging in on the application. Google accounts are well integrated on most of Google's platforms making it an ideal solution in this case. Since google accounts are needed to utilize the features of an Android-based smartphone the probability of a user not having a google account is slim. Firstly this makes the connection towards the application simple with no need to create new accounts or other things that waste time when a driver just want to get out on the track, another synergistic effect is probably that more drivers will use the application instead of just skipping it since they haven't got any account and couldn't bother creating one. Thirdly there will be no problem with unidentified drivers making it possible to prepare the medical effort at an early stage.

Helmet Card

To simplify the process of identifying the driver in case of an accident, a RFID-tag is often used. This tag is placed onto the helmet and can be read with a RFID-reader, it contains basic information about the driver and relatives. This concept was ported into the application as it was considered important to be able to contact any relatives notifying them about the accident. This is also useful when a driver is practicing alone since a relative can be automatically contacted if the driver doesn't respond to the "false alarm"-acknowledge. The user is prompted to add personal information like; name, phone number, social security number, last but not least information about what relative to be contacted with name and phone number. How this information is used is later described under "Requesting medical assistance".

Communication within the system

To be able to provide the functionality mentioned above the devices must be able to communicate with each other. The communication is done over TCP/IP,

mainly through HTTP-communication. Following is a technical description concerning the communication.

Webserver/Back-end

When designing the system layout concerning web-server and backend for the application, several designs were discussed, pros and cons of different solutions were considered. Since the safety system is at a concept-stage and the time available for developing the system was quite short, it was necessary that the backend-solution needed to be fairly straight forward to setup. If it was possible, an all-in-one solution offering front-end, backend and database capabilities would be perfect. After some consideration Google App Engine (GAE) was chosen. GAE was released in 2008 and provides an automatically scaleable PaaS - "Platform as a Service", meaning that the consumer develops the software and deployment using tools from the provider. The provider on the other hand provides servers, network and storage. Google App Engine makes it easy for developers to create a web application or just using it for back-end to a mobile application. The servers are automatically scalable which means that resources are automatically ramped up as the amount of users and connections grows, this simplifies the development since the customer don't have to consider scaleability during development, the downside to this is that the programming need to be done in some predefined ways. Google charges for the GAE service when the application has grown sufficiently large, the costs increase depending on the amount of users, traffic etc. The free-of-charge limit though is relatively high and does normally not apply when trying out the service or creating small applications. Google app engine is a great choice when the time for developing a back-end or front-end of a web application is short. The tools available are easy to use with rich online documentation. The Eclipse development environment supports a plugin provided by Google to support parallel development of an Android connected web application.

In the Crossafe-application, Google App Engine acts as a back-end implementing a http servlet for communication and a "Java Data Object" (JDO)-based database called datastore which is included in the GAE toolset. See Figure 11 on page 49 for an overview.

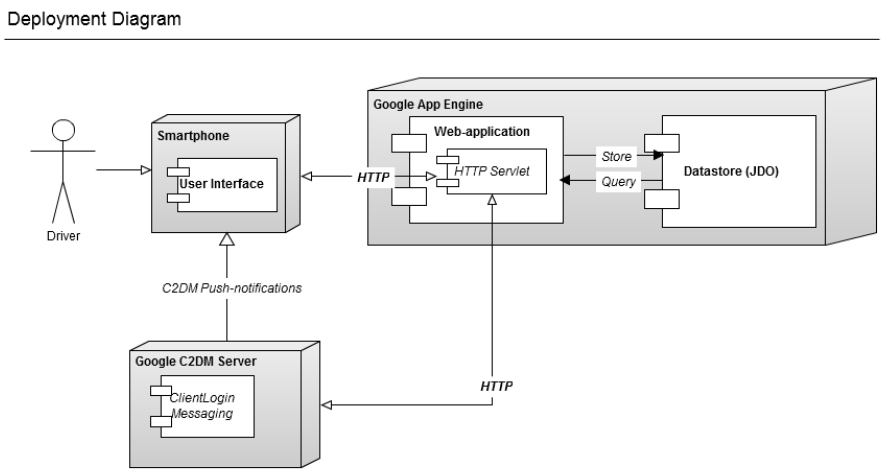


Figure 11: Deployment Diagram of the system

Requesting medical assistance and warning others

When the driver falls and are unable to help himself a message is automatically sent, the webserver receives the message over http communication and stores it in a database. It contains information about driver id, at what time the message was sent and at what position the driver is located at. This message is received by the other drivers to warn them, also giving them the possibility to see where the driver is and possibly aid him. If the driver has already received help this can be used as a hint to drive carefully when passing the scene of the accident to reduce the risk for colliding with medical personnel who possibly are residing in the track. This can also, as stated above, be done with a SMS text message that is sent to the relatives and family members registered in the helmet-card. This can be used to inform the relatives that medical assistance is needed if the driver are driving alone, or just to inform them that an accident have occurred on the race track, even though medical assistance is already on its way. A flowchart describing the program logic for sending a message when a driver falls can be seen in Figure 12 on page 50.

The message sent when a driver has crashed are called “M1-messages” internally, their structure can be seen in Table 1 on 49.

Functionality	**Type**	**Example**
Driver Id	String	john.doe@gmail.com
Position	String	59.6323425, 18.2345236
Message timestamp	Long64	Tue May 15 13:36:48 CEST 2012
Position timestamp	Long64	Tue May 15 13:36:48 CEST 2012

Table 1: M1-Message structure

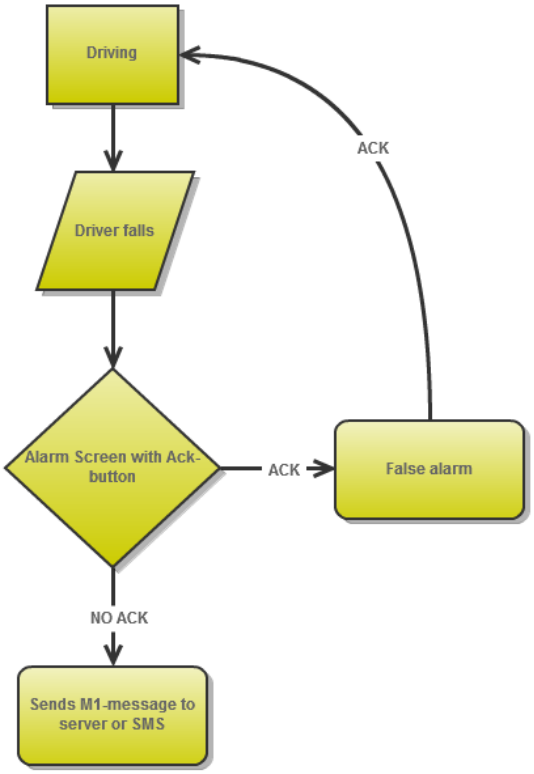


Figure 12: Flowchart describing the logic for sending a message when a driver falls.

The message data is sent through a HTTP-Post message, a servlet on the server-side receives the message and stores the relevant information in the database. See the system communication overview in the next chapter for details.

C2DM / Push-notifications

When the message is received. the information about what driver has fallen and the position of the fall is registered in the database. All devices using the applications (i.e. all drivers) will be informed that a driver has fallen, this is done through a technology called Cloud To Device Messaging or Push-notifications. By using a role-account for the application-server (normally a gmail adress) the web-application can act as a sender of these push-messages. The app-server contacts the servers at google that handles C2DM-push-messaging and requests a ClientLogin Authentication Token for this role-account. This is done through a http-post message. The app-server is now ready to send push-notifications to the connected devices.

The fallen drivers phone sends a M1-message (help-message), the message is received by the app-server and the information is stored in the database. The app-server then, in turn, contacts the C2DM-servers and tell them to push a notification to each device-registration ID, registered in the database. This is also done through a http-post-message. A describing picture of this process can be seen in Figure 13 on page 51 below, which is followed by a step-by-step explanation.

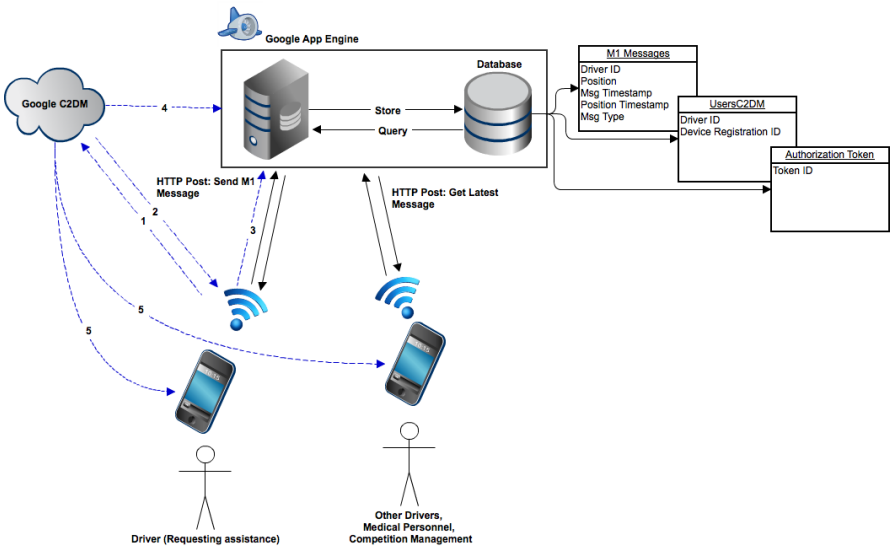


Figure 13: System overview with C2DM-messages in dashed, blue arrows

1. The user is prompted to login to the application with a google account, a registration-request is sent to the C2DM servers.

2. The response to that request contains a registration ID, unique to that instance of the application.
3. This registration ID together with the Driver ID is then passed on to the app-server storing it in the database. The app-server contacts the C2DM Server to receive an authentication-token so that it is eligible for sending C2DM pushes.
4. When a driver falls, a M1-message (help-message) is sent (not shown in the picture), this message is registered on the server, it will then contact the C2DM-servers and make them push a notification to each device.
5. All devices connected receive a notification about the new information on the server, in this case that a driver has fallen, it is then possible to view the driver id and position on the "View latest message"-tab.

This is a much better solution than having the client devices polling the server for new data, if no data is available the communication is done in vain. It is more efficient to have the server telling the client devices when new data is available. This reduces the amount of data that need to be sent which is a great advantage since transferring data over a mobile Internet connection is expensive or are limited by a quota. The downside of using C2DM is that google do not promise that the message will be sent to the device, they do not promise it will be received instantly either. Other limitations include a limited amount of messages that can be sent from one application (even though the quota is fairly high) and how many messages that can be sent to one device. As long as the usage is within reasonable levels this is not limiting. The message size is limited to 1024 byte, this is because it is meant to be used as a push-notification protocol, not a full featured communication protocol. The aim is to use the push-messages to tell the clients that new data are available on the server side, the clients then connect to retrieve the new data.

View position, assisting a fallen driver

When a new message is received, each device can view that latest message by pressing the "View latest message"-tab. The tab shows the information specified in Table 2 on page 52.

Functionality	**Type**	**Example**
Driver Id	String	john.doe@gmail.com
Position	String	59.6323425, 18.2345236
Show on map	Activity-call	Button: "Show on map"

Table 2: "View latest message" - tab information

To see the position on the driver on google maps, it is possible through the "Show on map"-button included in the view-window. This makes navigation easier and simplifies the medical effort.

Communication to external actors

To be able to send text messages to the relatives specified in the helmet card, a test messaging class was written. The class is able to dynamically create text messages specifying who has fallen and at what position. Further, a test message parser is implemented that extracts information from a text message, given that the message is written on a predefined form. This is not used in the current implementation of Crossafe due to lack of testing but should be a nice feature in future releases.

5.2.3 Sensorimplementation and algorithms

Sensing a fall

To be able to register if a driver falls, an algorithm using accelerometer and gyroscopic sample data was implemented. The approach of the algorithm is based on some minor some assumptions of the mechanics of a human body subdued to a crash. A human body on a motorcycle is assumed to be more or less stationary in an upward position of the torso. While riding the bike, the body is assumed to still be more or less hold in an upward position while exposed to different accelerations. When a crash occurs, the body is assumed to twist at a relatively high rate in either direction while falling off the bike or when hitting the ground. These twists are then assumed to be followed by movements that results in low accelerations on the axis parallel to the "toe-to-top" direction. These assumptions imposes restrictions on how the phone should be mounted to the body when using the application. These restrictions was found to be necessary both in order to lower the amount of false positives generated but also to lower the development time of the algorithm which was of great importance for the project. In Figure 14 on page 54 , a picture showing the axes of the coordinate system that Android sensors are using can be seen.

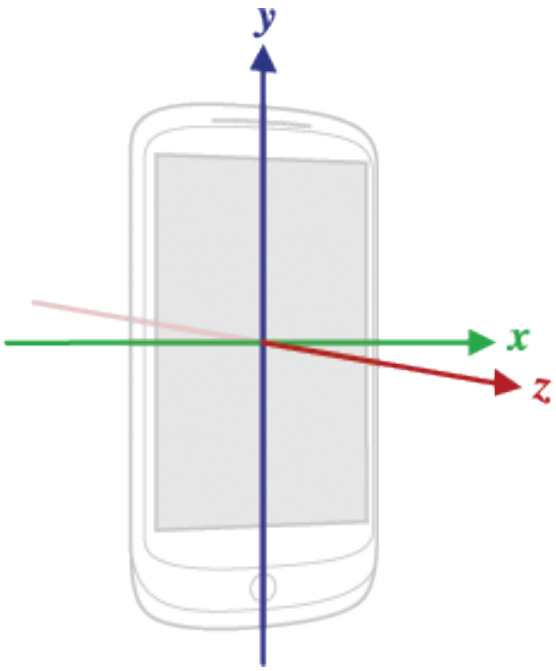


Figure 14: Overview picture showing the direction of the coordinate axes that Android sensor data is based on [39]

With these directions postulated, the program requires that the cell phone is mounted on the torso with the y-axis pointing upwards in order for the algorithm to work properly.

The fall detection algorithm is implemented using a number of different states and three moving average filters acting on the y-axis of the accelerometer- and the x and z axes of the gyroscopic data. The flowchart of how the fall detection operates can be seen in Figure 15 on page 55.

Saving sensor data

To retrospectively be able to process the accelerometer and gyro data on an external system, the sampled accelerometer and gyro data is saved to a file using a file writer class. This allows for examples trainers to for example measure the performance of a motocross driver in different parts of the course using algorithms that would be too processor intense to be performed in real time on a cell phone. The file names are dynamically generated on the form "sensor type"_"date"_"time that the sampling started" such that different occasions can be distinguished from each other. The class that deals with the file streams in the application also makes sure that the size of the files never exceeds a certain size by rescaling the files in run-time. This prevents the files from growing too large, filling too much space on the user's phone.

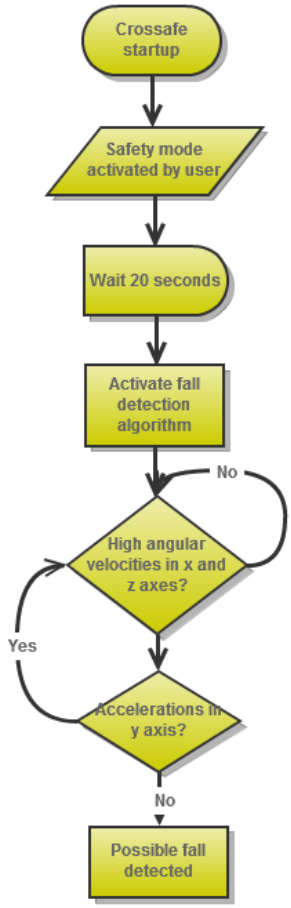


Figure 15: Flowchart of the different states in the fall detection functionality.

Moving average filter

As discussed in section 4.6.6 the sampling rate of the Android sensors varies somewhat. This was taken into thought but it was not considered a problem when developing the fall detection algorithm. Since the chosen strategy didn't imply any substantial frequency dependency in the algorithm, it was found that a moving average filter implementation would suffice. The filter acts on the subset $\{n - a_{sensoraxis_i}, n\}$ of accelerometer and gyroscopic data where a sensor axis i is the constant window length of the filter acting on the corresponding sensor axis i. The moving average equation used in the program is on the form

$$MA = \sum_{n-a_{sensoraxis_i}}^{n} x_n \tag{1}$$

where the most recent n:th sample x_n is added continuously to a queue where as the oldest active sample $x_n - a - 1$ is withdrawn. Should the output of the filter at any time fulfill the relationship

$$MA > a_{sensoraxis_i} \cdot b_{sensoraxis_i} \tag{2}$$

it would be interpreted as an overshoot of the predefined limit b sensor axis i associated with the corresponding sensor axis i. The decision of the multiplication in equation 2 on page 56 saves processor power as no averaging division has to be performed in equation 1 on page 56. Having different limits a and b for each axis allows tailoring of which kind of motions that will generate an overshoot. This is a very important feature as the mechanics involved in a crash varies depending on what kind of crash it is. In this way, the same basic application could be used in other fields than motorcycling just by changing the parameters a and b. We argue that this implementation is sufficiently sophisticated in proportion to the overall program as well as the time spent developing it.

6 Results

From the analysis and practical implementation presented in this thesis it was found that a modern cell phone could play the role of a generalist in a mechatronic application. It is shown that they can perform in applications where the requirements on sensors and communication does not require extreme performance. Smartphones have become a relatively cheap platform with a lot of sophisticated hardware and operating systems that can be programmed using freeware IDE's. Even though the mobile phone cant do demanding tasks, or tasks with requirements that extend outside what the phone can possibly deliver, it does not mean that it cant be used within that system.

The biggest difference between processors intended for smartphones compared to those in embedded and mechatronic systems are not in what communication technologies they offer. The ones intended for smartphones was found to include modems for communicating through one or several of the technologies 2G/3G/4G while processors in embedded systems contain support for communication between processors and between modules extending the systems functionality.

Most smartphone processors utilizes GPU (Graphical Processing Units) for hardware support when calculating graphics, this is outside the scope of this thesis but it is an important note that this is where the main difference is. By studying the existing work within the field of smartphones in mechatronic applications, especially from a communication point-of-view, they have been found to be used as sensor-nodes communicating through one of the wireless technologies available, be it Bluetooth, WLAN or Mobile-data-connection.

Embedded systems often requires real time communication and execution of time critical tasks, this can be done by implementing a RTOS - "Real-Time Operating System" for scheduling tasks and coordinate sensor and actuator operation with communication. Most mobile phones already uses its own RTOS with libraries and interfaces ready to be used. As stated the most common use case is using the phone as a sensor or data node, which also indicates that this might be the most suitable purpose for using smartphones; easily setup communication nodes, able to transmit any kind of data. "

7 Discussion and Conclusions

7.1 Discussion

During the span of mobile development from the mid 70's until today, mobile phones have become more feature-rich, offering Internet connection, calendars, music players etc. But it can be questioned whether its usability has improved in the same way. When it comes to communication the biggest drawback is the constant need for high speed Internet access if every feature of the phone is to be utilized. High transfer rates demands high frequencies, since high frequency communication yields a shorter range of communication for the same level of transmit power, a smartphone is less able to properly function in rural areas. The high communication rates also demand more power as most other modules in a smartphone, yielding a much shorter battery time compared to just a couple of years ago. Back in the 70's phones communicated on a much lower frequency, the nordic-developed system NMT, communicated at 450Mhz and offered a yet-to-be beaten range when it comes to cell phones. NMT was in use until it was suspended in 2007 by the sami-people living in scandinavia an russia, since this was the only way of communicating by cell phone in the most remote areas. By looking at the evolution of mobile phones it seems like the sami people will have to wait for an alternative to the NMT since the development of smartphones are heading in the opposite direction, with large screens and faster processors to cope with the increasing demand for mobile entertainment.

As mentioned above, the mobile nature of smartphones, its ability to access high speed Internet and easy-to-use development kits makes it an ideal platform for example prototype development, it is also a good platform for isolated systems or systems solving temporary tasks, like temporary traffic observations or as a mobile surveillance system with its camera and Internet-connection. Mobile phone development tend towards solving several of the tasks that used to be done by stand alone systems, some examples are GPS, wireless connections in all its flavors, texting, telephony etc. This evolution is likely to continue with more functionality getting stuffed into the smartphone as technology advances. The aim of all important every day stuff we use contained under one hood or in one device might not be far away either, soon the wallet and keys might all be stored within the smartphone, banking applications both through Internet and through NFC are already implemented, using NFC for opening your house or car is probably not far away either. Safety applications might also be a future area for mobile devices, the smartphone is with us where ever we go. The phone is capable of sensing the movement and tilt of the phone which can be used for sensing situations where the user might be in danger. Just being able to quickly and silently call for help might be the difference between life and death. Elderly people could change their safety wristband with an alarm button into a small smartphone based device where a communication line can be used for speaking to medical assistance, this can be used for listening to radio or keeping track of what time it is at the same time. This was just to name a few, there are almost an infinite amount of applications where smartphones can be used outside the sphere of entertainment and telephony.

The maximum sample rate that can be set when using common APIs in Android

was found to be 50Hz. This limitation in sampling rate could be one of the ways that the Android overhead manifest itself and could be a convenient way for Android to make sure that the sampling service wont occupy the processor too much. However, this behavior of Android imposes some restrictions on applications using sensors. According to the Nyquist theorem, the sample rate in an application needs to be at least twice the highest frequency in the system to prevent aliasing of the frequencies in the sampled signal. This means that; applications using Android sensors where frequency information of the sensor signal have to be intact have to either apply a low pass filter to the signal or limit the frequencies in the system to to a maximum of 25Hz. We argue that this should be taken into consideration if using Android sensors in mechatronic applications. However, applications like the one described by Kwapisz, Weiss and Moore (2010) that doesn't rely on frequency recomposition should still be able to benefit from cell phone sensoring.

Another interesting note about how the operating system of cell phones restricts processor power is that the majority of functionality in smartphones are accessed through its screen. The aim is therefore to make sure that the user is able to access all functionality through the screen. Looks and feel are important which means that the processor need to be fast enough to offer a pleasant user experience without choppy interfaces or long access times. The GUI blocks a lot of cpu power from other applications which should be taken into consideration when developing more cpu demanding mechatronic cell phone applications.

It was noted that an already common application is to use cell phones as sensor nodes. If a sensor network need to broadcast sensor data to an external server, and it is implemented through a dedicated mechatronic system, a communication interface has to be chosen, which includes both choosing and implementing hardware but also the software, this can be very time consuming. When using mobile phones however this is already done, the needed software libraries are readily available. This could lower the development cost for projects using cell phones as sensor nodes and should affect the usability of cell phones in mechatronic applications positively.

Another aspect that come into play when assessing the main question is where the development will take us, it is inevitable to not consider what the future will look like and how mechatronic systems will be used in the future. The smartphones together with social media have made us able to tell friends and relatives where we are and what we are doing no matter what time and place through geo-location. Many services have utilized this to make it easier for us to find the closest restaurant or petrol station. As mentioned in the article about traffic monitoring [3] it is also used for showing traffic intensity on the roads. At a glance it might seem as the smartphones turn more and more into information nodes able to tell other nodes about its surroundings. To take it further, the smartphones today are parts in an enormous sensor network consisting of millions of nodes. The future for mobile phones at this point are heading towards larger screens and better media capacity. Constantly improved Internet access and integration towards cloud technology are turning smartphones into connection devices where information are accessed remotely, which in turn yields spreading of information easier.

7.2 Conclusions

Cell phones can be used in mechatronic applications that involves requirements on sensor data processing and communication infrastructure. The limit of their versatility is mainly governed by the cell phone's processor power, sensor availability and quality, battery longevity and different environmental aspects such as Internet accessibility, communication interfaces to other parts of the system, temperature, vibrations, humidity, EME etc. The smartphones serves as cost-efficient platforms for prototype development or setup of temporary systems. The connection abilities together with its sensors makes the smartphones useful as sensor-nodes, able to deliver data for statistical surveys whether it be about traffic flow or make applications where the users can compare their athletic performance.

8 Future work

There are some aspects of the theoretical analysis that could be improved. The testing of the sensors could be made more rigorously and unified. By using for example a test suit, each of the sensor types accessible from common Android APIs could be subdued to the same tests. This would increase the stringency of the evaluation. However, it could be argued whether this would actually increase the value of the evaluation by an amount that is proportional to the work load. Further, sensors like the microphone and the camera should have its own tests designed for them as they are quite different compared to the other sensors. Interesting measures of the microphone could for example be a frequency response curve, a graph over its polar pattern or the sensitivity, just to name a few.

Another improvement could be to investigate any possible gains in performance when coding some tests in Android native c/c++. One huge drawback with the common Android APIs is that the sampling frequency seems to be limited at 50Hz. Chances are that it is possible to overrun this and similar overhead specific limitations when coding in native c/c++. However, it should be noted that coding in native c/c++ is cumbersome if compared to common java based Android programming which should prolong any development in Android.

The theoretical evaluation concerning smartphones in mechatronic systems can be improved by comparing development cost, implementation time and a more detailed prediction of when a system successfully can be based on a smartphone or a dedicated system when it comes to length of operation, time-to-market or project budget. A more thorough study of in what areas the smartphones are currently being used could reveal other not-so-obvious areas where their potential use might be as great, this because it is easier to conclude where they are not used if it is clear where they are used. The perfect result of the practical implementation was an application mainly providing a safety system, but also a tool that would make training more fun. During the brainstorming sessions in the beginning of the project a lot of functionality were discussed, some were discarded but a lot was kept until a few weeks from the end where a final lock-in had to be done concerning what functionality to keep. Since this safety system was developed as a concept, focus was put on making a stable, well performing system architecture, since experience within the field of Android-application-programming and back-ends for mobile device applications where quite scarce, a large effort had to be put in analyzing different solutions and system designs. It was necessary that a back-end could be setup fast enough to support a first concept, when it was up and running focus could be put on developing the functionality of the Android application. The back-end combined with the Android application is the back-bone of the system, further functionality can then just be added to make the system more feature rich. Some of the functionality that had to be discarded was driver groups, these groups can serve as a fun way of practicing with friends. The plan was to give drivers the ability to create groups that others can join, the drivers within the group are then able to share max-, min- and average speed, lap times, driven distance or just position. This can be combined with the ability to store achievements and present a performance curve giving the drivers the ability to see the progress of their training. Further

functionality to share the achievements through social media such as Facebook, twitter or Google+ would improve the competitive element of the system. Further, an analysis can be done in efficient ways to inform the driver, a distinction also has to be done between safety critical information like fallen drivers in the track or just non critical information like average speed or distance driven. The algorithms for sensing a fall can be done more sophisticated to reduce the risk of false alarms and to properly sense a fall, this would demand analysis and testing of the physics behind how a driver falls. As mentioned above, the system architecture is done to support further functionality, to increase the safety aspect the GPS can be used to sense whether a driver is heading towards another driver lying in the track, efficiently hindering a collision that could put both drivers in more danger.

References

[1] Alessandro Carbonari et al. A proactive system for real-time safety management in construction sites. *Automation in Construction*, 20(6):686–698, 2011.

[2] Jiangchuan Liu Zheng Ruan, Edith C.-H. Ngai. Wireless sensor deployment for collaborative sensing with mobile phones. *Computer Networks*, 55(15):3224–3225, 2011.

[3] Juan C. Herrera et al. Evaluation of traffic data obtained via gps-enabled mobile phones: The mobile century field experiment. *Transportation Research*, Part C(18):568–583, 2010.

[4] Jennifer R. Kwapisz. Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, 12(2):74–82, 2010.

[5] Ram Dantu Christopher Barthold, Kalyan Pathapati Subbu. Evaluation of gyroscope-embedded mobile phones. *IEEE International Conference on Systems, Man, and Cybernetics*, Oct 2011:1632–1638, 2011.

[6] Antonio et al. Garcìa. Mobile phone platform as portable chemical analyzer. *Sensors Actuators: B. Chemical*, 156(1):350–359, 2011.

[7] M. ; Cootes T. Tresadern, P. ; Ionita. Real time feature tracking on a mobile device. *International Journal of Computer Vision*, 96(3):280–289, 2011.

A Sensor table

		Samsung Galaxy Note (N7000)	Samsung Galaxy S2 (GT-I9100)	Samsung Galaxy Nexus (GT-I9250)	HTC Evo 4G	Apple iPhone 4	Apple iPhone 4S	Motorola Droid RAZR
Released		Q4 2011	Q2-3 2011	Q4 2011	Q2-3 2010	Q2-3 2010	Q4 2011	
Processor		Samsung Exynos 4210 1.4 GHz dual-core	Samsung K3PE7E700B 4 Gb mobile dram PoP w dual-core 1.2 GHz processor	TI 4460 OMAP microprocessor, 1.5 GhZ dual core ARM-based processor	1 GHz Qualcomm QSD8650 Snapdragon processor	Apple A4 1GHz	Apple A5 dual cores @ 800MHz-1GHz	TI OMAP4430 1.2GHz
Accelerometer		**STMicroelectronics LIS3DH**	**STMicroelectronics LIS3DH**	**Bosch BMA220**	**Bosch BMA150**	**ST Microelectronics LIS331DLH**	**ST Microelectronics LIS331DLH**	**STMicroelectronics LIS3DH**
	Range (G)	±2/±4/±8/±16 dynamically selectable fullscale	±2/±4/±8/±16 dynamically selectable fullscale	±2/±4/±8/±16 dynamically selectable fullscale	±2/±4/±8 dynamically selectable fullscale	±2/±4/±8 dynamically selectable fullscale	±2/±4/±8 dynamically selectable fullscale	±2/±4/±8/±16 dynamically selectable fullscale
	Resolution (bit)	16	16	6	10	16	16	16
	Sensitivity (mG/LSB)	1/2/4/12	1/2/4/12	62.5/125/250/500	3.9/7.8/15.6	1/2/3.9	1/2/3.9	1/2/4/12
	Sampling frequency (Hz)							
	Output noise density (µg/√Hz)	220	220	2000	500	218	218	220
Gyroscope		**STMicroelectronics L3G4200D**	**STMicroelectronics L3G4200D**	**InvenSense MPU – 3050**	**No**	**STMicroelectronics L3G4200D**	**STMicroelectronics L3G4200D**	**STMicroelectronics L3G4200D (questionable source)**
	Range (°/s)	±250/±500/±2000 dynamically selectable fullscale	±250/±500/±2000 dynamically selectable fullscale	±250/±500/±1000/±2000	N/A	±250/±500/±2000 dynamically selectab	±250/±500/±2000 dynamically selectable	±250/±500/±2000 dynamically selectable fullsc
	Resolution (bit)	16	16	16	N/A	16	16	16
	Sensitivity (m°/s/LSB)	8.75/17.50/70	8.75/17.50/70	7.5/15.3/30.5/61	N/A	8.75/17.50/70	8.75/17.50/70	8.75/17.50/70
	Sampling frequency (Hz)				N/A			
	Output noise (°/s/√Hz)	0.03 @ 50Hz	0.03 @ 50Hz	0.01 @ 10Hz	N/A	0.03 @ 50Hz	0.03 @ 50Hz	0.03 @ 50Hz
Primary image sensor		**Yes**	**Yes**	**Samsung S5K4E5**	**Yes**	**OmniVision OV5650 (presumed from available specifications)**	**Yes**	**Omnivision OV2B3BA**
	Sensor elements (Megapixels)	Unknown	8	5, 1.4 µm pixel size	Unknown	5, 1.75 µm pixel size	8Mp	8 MP, 1.4 um pixel pitch
	Type	Unknown	Unknown	CMOS (BSI)	Unknown	CMOS (BSI)	CMOS	CMOS (BSI)
Magnetometer		**Asahi Kasei AK8975**	**Asahi Kasei AK8975**	**Yes**	**Unknown**	**Asahi Kasei AK8975**	**Asahi Kasei AK8975 (not confirmed)**	**Asahi Kasei AK8975**
	Range (µT)	±1200	±1200	Unknown	Unknown	±1200	±1200	±1200
	Resolution (bit)	13	13	Unknown	Unknown	13	13	13
	Sensitivity (nT/LSB)	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown
	Sampling frequency (Hz)			Unknown	Unknown	Unknown	Unknown	
	Output noise (nT/√Hz)	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown
Pressure sensor		**Unknown**	**Unknown**	**Bosch BMP180**				**Bosch BMP180**
	Range (kPa)	Unknown	Unknown	30...110	**Unknown**	**No**	**Unknown**	30...110
	Resolution (bit)	Unknown	Unknown	16 to 19	Unknown	N/A	Unknown	16 to 19
	Sampling frequency (Hz)	Unknown	Unknown	Unknown	Unknown	N/A	Unknown	
	RMS noise (Pa)	Unknown	Unknown	0.03-0.06	Unknown	N/A	Unknown	0.03-0.06
Light/proximity sensor		**Unknown**	**Unknown**	**Yes**	**Unknown**	**Yes, light sensor confirmed**	**Unknown**	TAOS CT405
	Range of effect (m)	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	
	Sensitivity (Lumen)	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	
Source								
		http://www.chipworks.com/en/technical-competitive-analysis/resources/recent-teardowns/2011/12/inside-the-samsung-galaxy-note-n7000/	http://www.ifixit.com/Teardown/Samsung-Galaxy-S-II-Teardown/5861/1#.T80Yk-vUOGk	[http://galaxynexusreview.org/specifications/]	http://www.ifixit.com/Teardown/HTC-Evo-4G-Teardown/2979/2	http://www.isuppli.com/Teardowns/Pages/ISuppli-Takes-on-iPhone-4.aspx	http://www.chipworks.com/en/technical-competitive-analysis/resources/recent-teardowns/2011/10/iphone-4s-image-sensor-and-touch-screen-controllers-identified/	http://www.chipworks.com/en/technical-competitive-analysis/resources/recent-teardowns/page/2/
				http://www.chipworks.com/en/technical-competitive-analysis/resources/recent-teardowns/2011/12/inside-the-samsung-galaxy-nexus-gt-i9250/	http://www.chipworks.com/en/technical-competitive-analysis/resources/recent-teardowns/2010/06/teardown-of-the-htc-evo-4g-smart-phone/		http://www.chipworks.com/en/technical-competitive-analysis/resources/recent-teardowns/2011/10/iphone-4s-teardown-a-closer-look-at-the-chips-inside/	http://www.engadget.com/2011/11/08/behind-the-glass-a-detailed-tour-inside-the-motorola-droid-razr/

Figure 16: Sensors found in some different cell phones during the thesis.

B Functionality Documents

Funktionsbeskrivning

Item	Falldetektion
Prioritet	1
ID	FID1
Ansvarig	David Byström
Typ	Algoritm
Hårdvara	Accelerometer, gyroskop

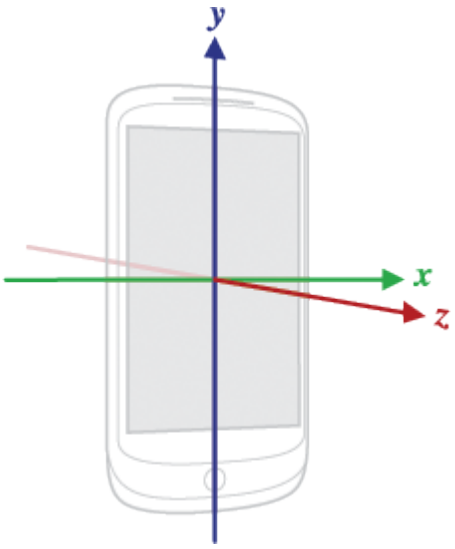
Version	Datum	Upphov	Ändringar
1	2012-03-16	David Byström	Original

---

1. Beskrivning

Algoritm för att detektera när en förare fallit. Algoritmen behandlar samplad accelerometer- och gyrodata för att avgöra detta i realtid. Första versionen av algoritmen kommer i mån av tid att förbättrats. Mjukvaran bör utformas så att man kan starta/stoppa algoritmen när som helst. Om algoritmen kan stängas av kan processorkraft/batteri sparas, man kan dessutom välja att endast sampla och logga sensordata vilket kan ses som en feature.

Teorin bakom algoritmen är att ett fall antas karaktäriseras av onormalt höga vinkelhastigheter på överkroppen följda utav statiska eller långsamma rörelser. Algoritmen kräver att mobilen är monterad med y-axeln i riktning uppåt.



Figur 1 - Koordinataxlar i Android.

Algoritmen går i lösa drag till på följande vis:

- om antalet vinkelhastigheter runt x- eller z-axeln (se figur 1) i följd överstiger ett visst värde - flagga.
- uppmäts därefter relativt statiska accelerationer utmed y-axeln under en längre tid är risken mycket stor att föraren ramlat och blivit liggande, följaktligen registreras då ett fall.

2. Beteende

Som användare ska jag, via en knapp av typen “Aktivera säkerhetsläge”, kunna aktivera falldetektionen. (OBS ej ännu implementerad i GUIt)
När ett fall registrerats ska falldetektionen upphöra och endast aktiveras igen när användaren själv avblåser alarmet eller när användaren återigen aktiverar säkerhetsläget.
När ett fall registrerats vidtar funktionaliteten beskriven i FID8 - “Skicka varning-/larmdata”.

3. Implementation

- Algoritmen kräver att mobilen är monterad med y-axeln i riktning uppåt.
- Algoritmen verkar på y-axeln på accelerometern.
- Algoritmen verkar på x- och z-axlarna på gyrot.
- Tre glidande medelvärdesfilter verkar på sensordatat.
- Algoritmen förutsätter att samplingen av accelerometer och gyro är aktiverad, se FID4 - “Loggning av sensordata”. Detta antyder att mjukvaran för FID4 och FID1 bör vara intimt sammankopplade.
- Förslagsvis bör implementationen vara state driven.
- Följande variabler bör finnas med

Namn	Beskrivning	Typ	Exempel
gyro_moving_avg	glidande medlevärdet på gyroaxel x	uint	41

gyro_moving_avg	glidande medlevärdet på gyroaxel z	uint	41
acc_moving_avg	glidande medlevärdet på accelerometer y	uint	33

4. Övrigt

Version 1 av algoritmen kommer i mån av tid att förbättras.

Ett diskret lågpassfilter kan implementeras och då låta en väldigt enkel logik verka på de filtrerade variablerna. Ett möjligt problem med detta är att det är ovisst huruvida man kan styra samplingsfrekvensen i Android, en fix samplingsfrekvens behövs för att filtret ska vara stabilt. Processorbelastningen kan dessutom tänkas öka vid implementation av ett LP-filter.

Funktionsbeskrivning

Item	Loggning av positionsdata
Prioritet	1
ID	FID3
Ansvarig	David Byström
Typ	Sampling, loggning
Hårdvara	GPS, minne

Version	Datum	Upphov	Ändringar
1	2012-03-16	David Byström	Original

---

1. Beskrivning

Sampling och loggning av positionsdata från GPS till en fil på telefonminnet. Aktivering av sampling/ loggning bör hållas på en lämplig abstraktionsnivå. I funktionaliteten ingår även att räkna ut förarens medelhastighet sedan start samt hans tillryggalagda sträcka.

2. Beteende

Som användare skall jag inte kunna välja när samplingen ska starta/stoppas, ingen “på/av”-funktion.
Som användare skall jag kunna se min nuvarande position. (inte implementerat i GUIt, finns stöd i mjukvaran)
Som användare skall jag kunna se min tillryggalagda sträcka. (inte implementerat i GUIt, finns stöd i mjukvaran)
Som användare skall jag kunna se min medelhastighet. (inte implementerat i GUIt, finns stöd i mjukvaran)
Som användare skall jag kunna välja att ta bort sparad data. (inte implementerat i GUIt, finns stöd i mjukvaran)
Sampling och loggning av positionsdata ska startas när falldetektionen startas.
Varje körning/träning/tävling som samplas skall sparas i olika filer, som användare skall man då kunna identifiera de olika filerna bara genom att läsa filnamnet.

3. Implementation

- Samplingsfrekvensen skall sättas relativt låg (~1Hz) för att minska belastningen på processorn till förmån för annan funktionalitet.
- Prioriteten för rutinen kan dessutom sättas låg eftersom osäkerheten i positionsangivelserna kan anses vara hög.
- Samplingsrutinen skall starta “så sent som möjligt” för att spara batteriet.
- Om implementationen av FID2 - “Kollisionsdetektion” använder positionsdata bör mjukvaran för FID2 skrivas med funktionalitet för FID3 i åtanke.
- Samplad rådata sparas på heapen i datastrukturer som över en viss storlek konverteras och sparas i en fil varefter datastrukturen töms och börjar fyllas med GPS-data igen.
- När minnet håller på att ta slut ska endast de mest aktuella värdena behållas.
- Datastrukturen för positionsdata skall innehålla följande:

Namn	Beskrivning	Typ	Exempel
GPS_ID	Globalt ID för GPS-data	String	“GPS”
position_data	Vektor innehållandes positionsdata på formatet lat, long, accuracy, timestamp, distance travelled	Objekt Java String Vector	position_data.add(Location)

- Den sparade filen skall ha följande struktur:
    - <Header med GPS_ID>
    - varje *n*:te rad efter headern innehåller data på formatet:
        - position_data[*n*]
- Den sparade filen skall ha ett filnamn som gör det lätt för användaren att identifiera när den blev sparad, tex “positionsdata20120316.txt”.
- Tillkommer gör dessutom två variabler som visar förarens medelhastighet sedan start samt den tillryggalagda sträckan.

Namn	Beskrivning	Typ	Exempel
vel_avg	Förarens medelhastighet	int	30 km/h
distance	Förarens tillryggalagda sträcka	int	2100 m

4. Övrigt

Funktionsbeskrivning

Item	Loggning av sensordata
Prioritet	1
ID	FID4
Ansvarig	David Byström
Typ	Sampling, loggning
Hårdvara	Sensorer, minne

Version	Datum	Upphov	Ändringar
1	2012-03-16	David Byström	Original

---

1. Beskrivning

Sampling och loggning av sensordata till lokala filer på telefonen. Mjukvaran bör vara modulärt utformad så att loggningen av en sensor kan startas av ett anrop med abstraktionsnivån: startSensorSampling(Sensor Accelerometer, x_axis true, y_axis true, z_axis false, logging true, filename accelerometer20120316.txt). Motsvarande avstängning stopSensorSampling (Sensor Accelerometer). Det skall bara finnas en instans av varje sensor i mjukvaran.

2. Beteende

Som användare skall jag inte kunna välja när samplingen ska starta/stoppas, ingen “på/av”-funktion.
Som användare skall jag inte explicit kunna välja vilka sensorer som ska samplas.
Samplade värden skrivs ej ut till användaren.
Som användare skall jag kunna välja att ta bort sparad data.
Varje körning/träning/tävling som samplas skall sparas i olika filer, som användare skall man då kunna identifiera de olika filerna bara genom att läsa filnamnet.

3. Implementation

- Endast sensorer som är intressanta för övrig funktionalitet såsom accelerometer, gyro etc. skall samplas.
- Samplingsrutinen skall starta “så sent som möjligt” för att spara batteriet.
- Samplingsrutinen skall i största möjliga mån sträva efter en hög, fix samplingsfrekvens (SensorManager.registerListener(..., int **rate, ...**)).
- Samplad rådata sparas på heapen i datastrukturer, en datastruktur för varje sensor som över en viss storlek konverteras och sparas i en fil varefter datastrukturen töms och börjar fyllas med sensordata igen.
- När minnet håller på att ta slut ska endast de mest aktuella värdena behållas.
- En sensors datastruktur skall innehålla följande:

Namn	Beskrivning	Typ	Exempel
Sensor	Sensorns ID	String	“Accelerometer”
x_axis	Vektor innehållande samplad data från sensorns x-axel	Objekt Java Vector	x_axis.add(1.152)
y_axis	Vektor innehållande samplad data från sensorns y-axel	Objekt Java Vector	y_axis.add(1.152)
z_axis	Vektor innehållande samplad data från sensorns z-axel	Objekt Java Vector	z_axis.add(1.152)
time_start	Timestamp för när samplingen startade. Erhålls från java.lang.system.currentTimeMillis()	Long64 millisekunder	360066455255
time_elapsed	Timestamp för samplingen. Erhålls från differensen java.lang.system.currentTimeMillis() - time_start. Tiden för samplingen mellan de olika axlarna antas vara konstant och känd.	Int32 millisekunder	36001223

- Kombinationen time_start&time_elapsed sparar minne i jämförelse med om varje timestamp skulle vara deklarerad som Long.
- Sparad fil är uppbyggd som:
    - <Header med sensorns ID, time_start>
    - varje *n*:te rad efter headern innehåller data på formatet:
      x_axis[*n*], y_axis[*n*], z_axis[*n*], time_elapsed[*n*]
    - Data separeras med komman “,”

4. Övrigt

x_axis[*n*], y_axis[*n*], z_axis[*n*] är samplade vid samma tid time_elapsed[*n*].

Mjukvaran bör vara intimt sammankopplad med mjukvaran för FID1 - “Falldetektion” då falldetektionen kräver att samplingen av gyro och accelerometer är igång.

Funktionsbeskrivning

Item	Upprätta anslutning till server och övriga klienter (FID7), Hantera anslutningsförfrågan (FID 11)
Prioritet	1
ID	FID7, FID11
Ansvarig	
Typ	Kommunikation , anslutning
Hårdvara	Kommunikationshårdavara, WLAN, BlueTooth alt. GPS

Version	Datum	Upphov	Ändringar
1	2012-03-16	Niclas Kempe	Original

---

1. Beskrivning

Upprätta och hantera anslutning mellan klientnoder och servernoden. Lista tillgängliga grupper att ansluta till, möjliggöra anslutning. Visa/lista förare i samma grupp.

2. Beteende

FID7:
“Som användare skall jag kunna lista och ansluta till aktiva deltagargrupper”
“Som användare skall jag kunna se övriga förare i gruppens anslutningsstatus och position”
“Om fel lösenord anges vid anslutning till en grupp skrivs in skall det uppmärksamma användaren”

FID11:
“Som användare skall jag kunna skapa deltagargrupper”

“Som administratör för deltagargruppen skall jag kunna begränsa åtkomst genom lösenord/login“

- Listning av tillgängliga förargrupper/event som en serverlista
- Möjlighet att skapa grupp genom Menyval.
- Anslutning genom att klicka på önskad grupp, ange loginuppgifter (Om nödvändigt)
- När anslutningen är upprättad skall samtliga förare listas

- En deltagar/-förargrupp skall innehålla:

Namn	Beskrivning	Typ	Exempel
GruppID	Tävling-/Eventnamn	string	Novemberkåsan
FörarID	Namn på föraren	uint	198806141234
Maxhastighet	Förarens maxhastighet		
Medelhastighet	Förarens medelhastighet, se FID3 - “Loggning av positionsdata“	string, XML eller motsv för kompatibilitet mot Google Maps API	-
Status	Visa övriga i samma grupp status på anslutningen (aktiv, inaktiv, frånkopplad) genom färgkodad bakgrund av FörarID-texten	string, enum	• Förare A (Aktiv) • Förare B (Inaktiv) • Förare C (Frånkopplad)

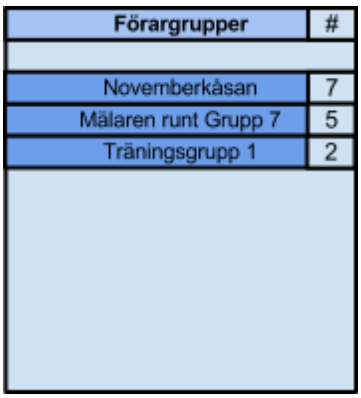
- Genom att trycka på en deltagare i gruppen kan följande information utläsas:

Namn	Beskrivning	Typ	Exempel
Medelhastighet	Förarens medelhastighet, se FID3	godt	XX km/h
Maxhastighet	Förarens maxhastighet, se FID3	godt	XX km/h
Distans	Visar förarens tillryggalaggda sträcka, se FID3	godt	X.Y km

Åktid	Visar förarens totala åktid	godt	hh:mm:ss

3. (GUI skiss + andra skisser) vid behov

Principiell GUI-utforming för listning av förargrupper och antalet deltagare i grupperna, anslutning genom att trycka på gruppen sedan välja anslut via Androids meny-system (FID7). Skapa grupp genom Androids meny-system (FID11)



4. Implementation

Serverimplementation genom Google App Engine eller motsv. för lagring av data och anslutning mot central server. Kommunikation till mobiltelefoner genom HTTP.

http://code.google.com/intl/sv/appengine/

5. Övrigt

-

Funktionsbeskrivning

Item	Skicka varning-/larmdata
Prioritet	1
ID	FID8
Ansvarig	
Typ	Logik, UI
Hårdvara	3G/GSM etc. WLAN, BT.

Version	Datum	Upphov	Ändringar
1	2012-03-16	David Byström	Original

---

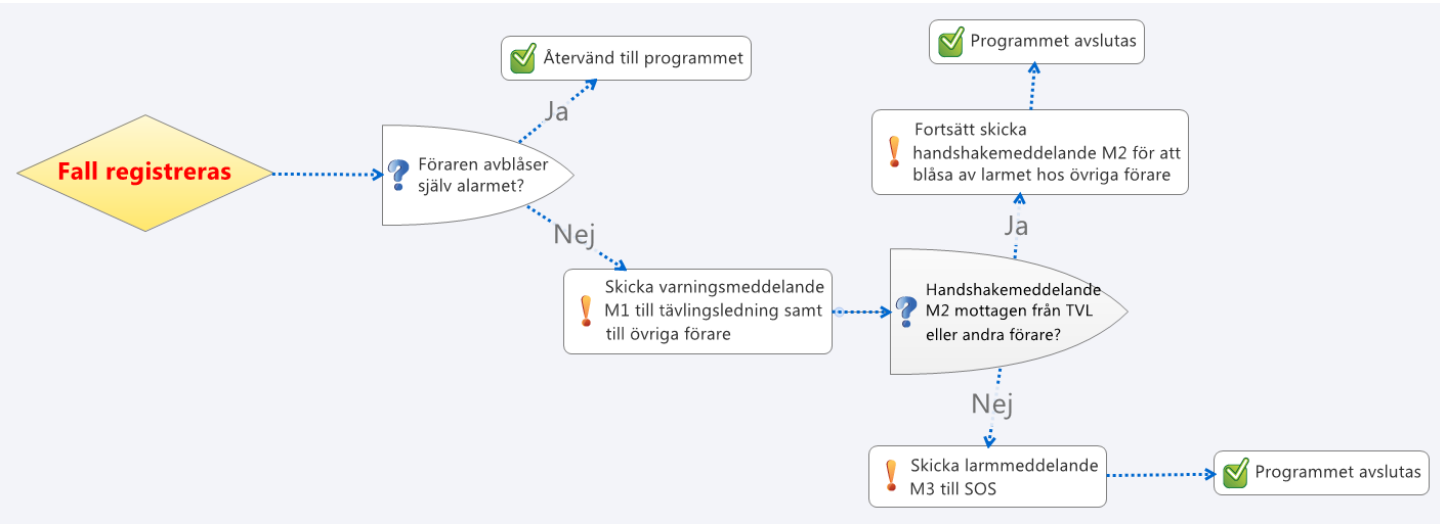
1. Beskrivning

Riktlinjer för hur varnings- och larmmeddelanden ska skickas till anhörig/tävlingsledning/SOS när ett fall har registrerats.

2. Beteende

Jag som användare skall själv när som helst kunna blåsa av ett larm.
Jag som användare skall själv “med-ett-knapptryck” kunna skicka iväg ett varnings- eller larmmeddelande i händelse av fall om falldetektionen fallerat.
Jag som användare skall kunna ange telefonnummer till anhöriga till vilka det skickas ut varningsmeddelanden när jag ramlat, sparas i fil som kallas för helmet_card.
Utomstående aktörer skall ha en chans att blåsa av ett ev. falsklarm, se nedan. (OBS ej implementerat!)

3. Flowchart vid tävling



OBS! Varnings- och handshakeförfarandet skiljer sig något mellan tävling och träning. Under träning skickas dessutom varningsmeddelande ut till anhöriga som även dom kan blåsa av ett larm genom att ringa upp föraren. Funktionalitet hos android.provider.CallLog.Calls tillåter att vi kan komma åt telefonloggen över missade/mottagna samtal. Har ett samtal mottagits från ett nummer som tillhör anhörig inom en viss tidsrymd avblåses alarmet. Har ett samtal missats från ett nummer som tillhör anhörig fortskrider alarmet. (OBS ej implementerat)

4. Implementation

- Om inte föraren själv avblåser larmet fortskrider detta.
- Varningsmeddelande, se M1 - “Varningsmeddelande: Fall” skickas till TVL/tränare och Förarmeddelande, se M2 - “Förarmeddelande: Kollisionsvarning” skickas till övriga förare med lämplig teknik, se FID7 - “Upprätta anslutning till server och övriga klienter” samt FID18-19 - “Skicka/ta emot/parse’a SMS”.
- Vilka anhöriga som ska motta ett SMS bestäms av vilka telefonnummer föraren har uppgett i filen helmet_card.txt.

5. Övrigt

Funktionsbeskrivning

Item	Ta emot meddelanden och lagra klientdata
Prioritet	1
ID	FID12
Ansvarig	Niclas Kempe
Typ	Kommunikation
Hårdvara	

Version	Datum	Upphov	Ändringar
1	2012-03-20	Niclas Kempe	Original

---

1. Beskrivning

Detta dokument beskriver hur meddelanden tas emot, hur det lagras i databasen.

2. Beteende

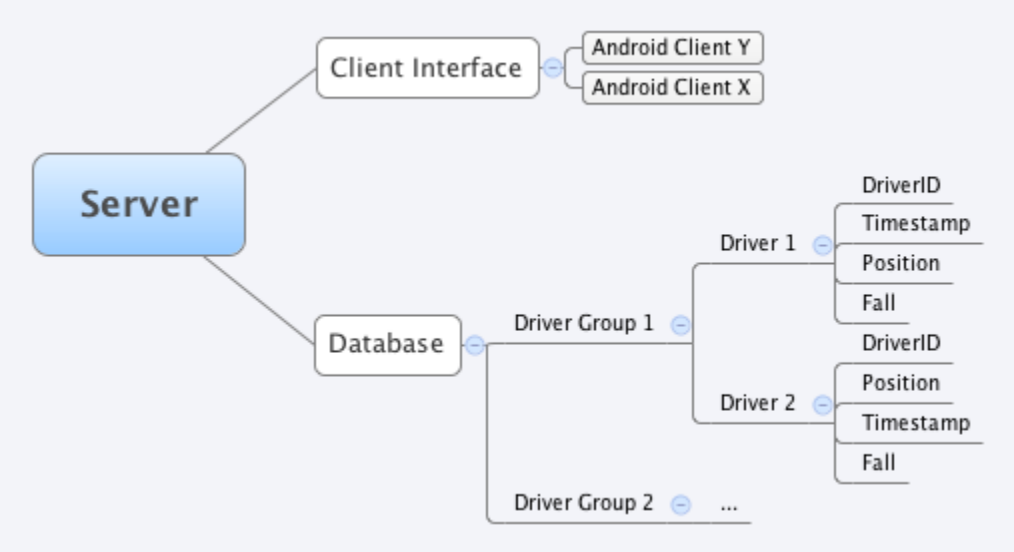
“Tar emot UP meddelande om position för respektive förar-ID och lagrar i databasen”
“Tar emot meddelande om fall (M1) och vidarebefodrar (genom push) till övriga klienter“
“Tar emot clear meddelande (M2) och rensar varningen på aktuell förare”

3. (GUI skiss + andra skisser) vid behov

Inget GUI, endast serverimplementation

4. Implementation

- Meddelandet tas emot på samma sätt som hos en klient, meddelandena sammanställs i en databas ordnad efter Förargrupp och FörarID med information om tid, position och om föraren ramlat.



5. Övrigt

allt annat av intresse som man kommer på

Funktionsbeskrivning

Item	Visa position för larm samt ID
Prioritet	1
ID	FID14
Ansvarig	Niclas Kempe
Typ	
Hårdvara	

Version	Datum	Upphov	Ändringar
1	2012-03-20	Niclas Kempe	-

---

1. Beskrivning

2. Beteende

“Som användare skall jag kunna se position och namn på föraren som larmat”

3. (GUI skiss + andra skisser)

N/A

4. Implementation

- En tab används för att visa informationen
- Tab-sidan visar iförarnamn och position.
- Förarens position skall kunna visas på Google Maps

Namn	Beskrivning	Typ	Exempel
Driver Id	Förarens ID	String	john.doe@gmail.com
Position	Förarens Position	String	59.6323425, 18.2345236
	Visa på karta	Activity-call	Button:“Show on map”

5. Övrigt

N/A

Funktionsbeskrivning

Item	Sortera/spara information om klienter under klientens ID
Prioritet	1
ID	FID15
Ansvarig	
Typ	Servermjukvara
Hårdvara	-

Version	Datum	Upphov	Ändringar
1	2012-03-20	David Byström	Original

---

1. Beskrivning

Mjukvara som för varje unik förare som är med och tävlar/tränar sorterar samt sparar inkomna meddelanden, data samt statusuppdateringar baserat på förarens ID.

2. Beteende

Som förare skall jag inte ha något inflytande överhuvudtaget på hur datat separeras eller sparas.
Som användare av servern skall jag bli varnad när minnet håller på att ta slut.
Som användare av servern skall jag kunna ta bort sparad data.
När minnet håller på att ta slut skall endast de mest aktuella värdena sparas.
Som användare av servern skall jag kunna exportera sparad data som beskrivet i FID5 - “Export av sensor och positionsdata”.
Som användare av servern skall jag genom att bara läsa filnamnen på sparade filer kunna utröna vem och vad filen avser.

3. Filuppdelning

När parse'ningen av mottaget meddelande är klar skickas datat vidare till SaveClass. SaveClass sparar datat till rätt filer i rätt Driver container beroende på vem som skickat meddelandet och vad meddelandet innehåller. Saknas filen som SaveClass försöker spara till ska den skapas. Är något data NULL sparas det inte. Om minnet håller på att ta slut skall endast de senast mottagna värdena i varje fil behållas. Tiderna som angivs i positions- samt alarmloggen utvärderas på mottagarsidan och om ett annat tidsformat önskas måste mjukvara för detta implementeras.

4. Implementation

- SaveClass bör vara singleton och innehålla följande metoder

Namn	Beskrivning	Typ	Exempel
logDriverPosition(Driver object, position, time)	Sparar positionsdata,	-	logDriverPosition(ID198806141234, pos_data, timestamp)
logDriverSensorData(Driver object, MAKRO_SENSORTYP, fil med sensordata)	Sparar sensordata	-	-
logDriverAlarm(Driver object, MAKRO_ALARM_ELLER_OK, time)	Loggar alarmdata	-	logDriverAlarm(ID198806141234, true, timestamp)

- Containerklassen “Driver” skapas så fort servern märker att en ny förare ansluter till gruppen/ tävlingen/träningen.
- Det skall vara endast en instans av “Driver” förknippad med varje förare.
- Containerklassen “Driver” bör innehålla följande variabler

Namn	Beskrivning	Typ	Exempel
driver_ID	Förarens ID, tex personnummer	uint	198806141234
pos_logg	Pekare till filström motsv. för positionsloggningen	-	-
sensor_logg	Array av pekare till olika filströmmar motsv. En pekare för varje sensor. Kan implementeras som 3-4-5-6 lokala pekare istället.	Vector med pekare till filströmmar motsv.	-
alarm_log	Pekare till filström motv. för alarmloggningen.	-	-

5. Övrigt

Funktionsbeskrivning

Item	Skicka/ta emot SMS
Prioritet	1
ID	FID18
Ansvarig	David Byström
Typ	Kommunikation, mjukvara
Hårdvara	

Version	Datum	Upphov	Ändringar
1	2012-03-19	David Byström	Original

---

1. Beskrivning

Mjukvara som tillåter att automatiska SMS kan skickas samt mottas. Ett automatiskt sms ska kunna skickas och tas emot i bakgrunden utan input från föraren. Ett automatiskt SMS kan skickas iväg till TVL/ tränare/anhörig när föraren fallit, se FID8 - “Skicka varning-/larmdata”.

2. Beteende

Som användare skall jag kunna skicka iväg ett automatgenererat SMS via en knapp i händelse av att falldetektionen fallerat.

Som användare skall jag kunna definiera ett personligt meddelande, t.ex. “Hej det är jag. Nu har jag ramlat igen, ring så fort du ser detta” som ska skickas med det automatiska SMS’et.

3. Implementation

- Det personliga meddelandet skall endast bestå av en sträng.
- Det personliga meddelandet skall sparas i telefonminnet och skall således kunna återanvändas.
- När ett SMS tas emot skall det direkt behandlas av parsern beskriven i FID19 - “Parse’a meddelanden”.

4. Övrigt

5. Frågor och svar

Q/A	Datum	Upphov	Fråga/Svar

Funktionsbeskrivning

Item	Meddelanden
Prioritet	-
ID	M1, M2, M3, UP, Sx
Ansvarig	
Typ	Meddelanden
Hårdvara	-

Version	Datum	Upphov	Ändringar
1	2012-03-14	David Byström	Original

---

1. Beskrivning

Mallar för de meddelanden som kan skickas vid fall. Meddelandetyperna skiljer sig beroende på vem mottagaren är. Meddelandetyperna särskiljs av deras ID'n.

De olika meddelandetyperna är:

- M1
  - Varningsmeddelande:Fall

    Skickas vid falldetektion till övriga förare, tävlingsledning/tränare, anhörig eller någon kombination av dessa.
- M2
  - Handshakemeddelande: OK

    Skickas av utomstående aktör, tex TVL, övriga förare eller anhörig, för att blåsa av ett larm. (Ej implementerat)
- M3
  - Larmmeddelande: Fall

    Förinspelat röstmeddelande som kan spelas upp när SOS ringts upp

automatiskt. (Ej implementerat)

- UP
    - Uppdatering av förares position

      Meddelande innehållande endast en förares nuvarande position, skickas till serverenhet om sådan finns. (Ej implementerat)
- Sx
    -

2. Implementation

Gemensamt för samtliga meddelandeframes är att de ska innehålla:

Namn	Beskrivning	Typ	Exempel
msg_ID	Meddelande-ID som används för att avgöra vilken parser som skall sköta ärendet.	String	M1, M2, M3, UP
data_length	Längd i byte på datafältet	int	128
data	Datafält innehållande intressant data	String, container, textfil etc.	

- Samtliga meddelanden är uppbyggda enligt mallen:
    - header <msg_ID, data_length>
    - data field <data>
- UPDATE (2012-04-10): The message structure follows the HTTP-message standard.

Datafältet <data> för meddelanden av typen M1, M3 skall innehålla:

Namn	Beskrivning	Typ	Exempel
sender_ID	Avsändarens ID, tex förarens personnummer	int	198806141234
sender_pos	Avsändarens position på formatet lat, long	String	15.1252, 56.12511

msg_send_time	Timestamp för när meddelandet skickades. Erhålls från java.lang.system.currentTimeMillis()	Long64	47849841536001248
pos_timestamp	Timestamp när positionen lästes in	Long64	

- Datafältet <data> för meddelanden av typen M1, M3 skall vara angivet på formatet <sender_ID, sender_pos, msg_send_time>.

Datafältet <data> för meddelanden av typen M2 skall innehålla:

Namn	Beskrivning	Typ	Exempel
sender_ID	Avsändarens ID, tex förarens personnummer	int	198806141234
sender_pos	Avsändarens position på formatet lat, long	String	16.236, 56.14616
msg_send_time	Timestamp för när meddelandet skickades. Erhålls från java.lang.system.currentTimeMillis()	Long64	47849841536001248
fallen_driver_ID	ID't på föraren som ramlat och som därmed kvitteras OK med detta meddelande	int	198806141234

- Datafältet <data> för meddelanden av typen M2 skall vara angivet på formatet <sender_ID, sender_pos, msg_send_time, fallen_driver_ID>.

- På mottagarsidan ska meddelandetyperna M1, M2 samt M3 dessutom förknippas med ett meddelandenamn för att underlätta identifieringen av meddelandet för personer som mottar och läser det, tex en anhörig, se FID19 - "Parse'a meddelanden".

Namn	Beskrivning	Typ	Exempel
msg_name	Används när tex en anhörig skall motta ett larmmeddelande via SMS, implementeras när meddelandet parse'as på mottagarsidan.	String	"Varningsmeddelande: Fall"

Datafältet <data> för meddelanden av typen UP skall innehålla:

Namn	Beskrivning	Typ	Exempel
sender_ID	Avsändarens ID, tex förarens personnummer	int	198806141234

sender_pos	Avsändarens position på formatet lat, long	String	15.12525,56.12351

- Datafältet <data> för meddelanden av typen UP skall vara angivet på formatet <sender_ID, sender_pos>.

3. Övrigt

4. Frågor och svar

Q/A	Datum	Upphov	Fråga/Svar